

Copies of this document may be purchased from:
Global Engineering, 15 Inverness Way East,
Englewood, CO 80112-5704

pANS X3.xxx-200x
NCITS/Project 1331-D

FIBRE CHANNEL

FRAMING AND SIGNALING (FC-FS)

REV 0.50

Phone: (800) 854-7179 or (303) 792-2181 Fax: (303) 792-2192

NCITS working draft proposed American National Standard for Information Technology

September 21, 2000

Secretariat:
Information Technology Industry Council

ABSTRACT: This standard consolidates the relevant clauses (framing and signaling) from FC-PH, FC-PH-2, and FC-PH-3 and associated errata, annexes and amendments into a single encompassing document. The Physical Interfaces clauses from the above documents are in a separate standard - Fibre Channel-Physical Interfaces (FC-PI). The project may also involve the deletion or obsoleting of outdated functions and features from those documents, the inclusion of additional link services in support of new functions defined by the FC family of documents, the inclusion of improvements and clarifications to the definitions of existing services as dictated by experience with existing implementations, and other capabilities which will improve the performance of existing FC products and fit those products for new applications.

NOTE: This is a working draft American National Standard of Accredited Standards Committee NCITS. As such this is not a completed standard. The T11 Technical Committee may modify this document as a result of comments received anytime, or during a future public review and its eventual approval as a Standard. Use of the information contained herein is at your own risk. Permission is granted to members of NCITS, its technical committees, and their associated task groups to reproduce this document for the purposes of NCITS standardization activities without further permission, provided this notice is included. All other rights are reserved. Any duplication of this document for commercial or for-profit use is strictly prohibited.

POINTS OF CONTACT:

Kumar Malavalli (T11 Chairman)
Brocade Communications
1901 Guadalupe Parkway
San Jose, CA 95131
408-487-8156 Fax: 408-524-8601
E-Mail: kumar@brocade.com

Edward L. Grivna (T11 Vice Chairman)
Cypress Semiconductor
2401 East 86th Street
Bloomington, MN 55425
612-851-5046 Fax: 612-851-5087
E-Mail: elg@cypress.com

Jeff Stai (T11.3 Chairman)
QLogic
26600 Laguna Hills Drive
Aliso Viejo, Ca 92656
949-389-6238 Fax: 949-389-6121
E-Mail: j_stai@qlc.com

John Scheible (FC-FS Facilitator)
IBM
Bldg 902 M/S 9263
11400 Burnet Road
Austin, Tx 78758
512-823-8208 Fax: 512-838-3822
E-Mail: jpscheible@aol.com

James Nelson (FC-FS Editor)
Northrop Grumman ISS
1 Hornet Way
Dept. 8F10/W8
El Segundo, Ca 90245
310-332-6540 Fax: 310-332-0750
E-Mail: nelsoja5@mail.northgrum.com

ANSI®
dpANS NCITS.xxx-200x

American National Standard for information Technology - Fibre Channel - Framing and Signaling (FC-FS)

Secretariat

Information Technology Industry Council

Approved ,200x

American National Standards Institute, Inc.

ABSTRACT

This standard consolidates the relevant clauses (framing and signaling) from FC-PH, FC-PH-2, and FC-PH-3 and associated errata, annexes and amendments into a single encompassing document. The Physical Interfaces clauses from the above documents are in a separate standard - Fibre Channel-Physical Interfaces (FC-PI). The project may also involve the deletion or obsoleting of outdated functions and features from those documents, the inclusion of additional link services in support of new functions defined by the FC family of documents, the inclusion of improvements and clarifications to the definitions of existing services as dictated by experience with existing implementations, and other capabilities which will improve the performance of existing FC products and fit those products for new applications.

**American
National
Standard**

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

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

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

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

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

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard. As of the date of publication of this standard and following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. No further patent search is conducted by the developer or publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Published by

American National Standards Institute, Inc.

11 West 42nd Street, New York, NY 10036

Copyright © 2000 by Information Technology Industry Council (ITI)

All rights reserved.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without prior written permission of ITI, 1250 Eye Street NW, Washington, DC 20005.

Printed in the United States of America

Update History

Version 0.50 (September 21, 2000)

- 1) Deleted Subclause 6.5, Automatic Link Speed Negotiation as it is superseded by Clause 33
- 2) Made 3 corrections to 15.5.33.2

Version 0.40 (September 19, 2000)

- 3) Incorporated comments 001-E to 015E, JS001-T, JS003-T, JS004-T, JS005-T, JS008-T, GC002-T, RW001-T, RW002-T
- 4) Reformatted per the IEC template
- 5) Incorporated item 10 of 00-284v4
- 6) Spelled out all instances of SCR where it means Stacked connect-request and defined the acronym SCR to mean State Change Registration
- 7) Completed the index
- 8) Incorporated Clause 15 (Link Services) edits from Bob Kembel (00-563v0)
- 9) Incorporated FC-FS Preemption edits from Mike Foster (00-564v0)
- 10) Made corrections to Figure 33 – ACK Precedence per direction at the September 12th meeting
- 11) Incorporated 00-538v1, LIRR proposal
- 12) Incorporated 00-532v1, Proposal for event qualifiers for RLIR common payload
- 13) Incorporated 00-451v1, FLOGI payload inconsistencies
- 14) Incorporated 00-544v2, Speed Negotiation Proposal
- 15) Stated the timeout for a Basic Link Service to be 2xR_A_TOV per 99-189v0.
- 16) Incorporated action item responses from Mike O'Donnell on RTIN (00-566v0), resolves comment
- 17) Deleted REC per 00-284v5, # 8
- 18) Made changes to the parameter field per 00-284v5, #9
- 19) Corrected references in clauses 3.1.93, 17.6.4 and 17.6.5

Version 0.30 (July 20, 2000)

- 1) Added the American National Standards page (page iii)
- 2) Edited the Title Page, Forward, Introduction and clauses 1-4
- 3) Deleted RNC per 00-205v1
- 4) Corrected the typos Bob Petersen found
- 5) Incorporated the Clock Sync ELS method per 00-341v0
- 6) Incorporated the Clock Sync ELS Informative Annex per 00-038v0
- 7) Incorporated the Primitive Clock Sync Method per 00-083v0
- 8) Incorporated Login Control per 00-056v2
- 9) Incorporated the approved changes in 00-284v4 (FCP-2 items that need to be included in FC-FS) (items 1, 2 and 6)
- 10) Added Read Port Status ELS per 00-022v2

- 11) Added Request Node FC-4 Types per 00-282v2
- 12) Updated the ELS Summary Table (Table 194)
- 13) Updated Clause 16 to make Class 4 more consistent
- 14) Made corrections to table 2 per errors that Bob Nixon found
- 15) Incorporated the BB_Credit Recovery proposal per 00-402v1
- 16) Lots of editorial changes (cross references, spelling, formatting, etc)

Version 0.20 (May 24, 2000)

- 1) Added an index (partial)
- 2) Deleted entries in Clause 3 that pertain to FC-PI
- 3) Removed changes effected by 99-094v0 as MCM is no longer part of AL-3
- 4) Made changes to clause 15 per 99-548v0 including a partial renumbering
- 5) Made changes per those approved in 99-179v0, except 2.4 (see 23 below)
- 6) Added the SL ELSs - BALLOC and DALLOC
- 7) Added RSCN and SCR ELSs per 99-304v0 and 98-327v0
- 8) Made changes per FC-VI (00-092v0), Annex B including adding the FARP ELSs
- 9) Added the RTIN ELS per 99-594v2
- 10) Made changes to the ABTX Basic link service and added the REC ELS per FCP-2, Annex B
- 11) Added the SRR ELS from FCP-2, clause 8.1
- 12) Added the RNID ELS per 99-593v0 and 99-707v0
- 13) Added the RLIR and LIRR ELSs per 99-603v1
- 14) Made changes to TPRLO per 99-569v0
- 15) Obsoleted Operation Associators per 99-433v0
- 16) Obsoleted Data Compression per 99-435v0
- 17) Removed the FutureBus type code
- 18) Changed the default values for E_D_TOV and R_A_TOV per 00-055v0
- 19) Deleted Annex L - Report Node Capabilities and added the words "New implementations shall not rely on the RNC fields" to the end of the first paragraph in clause 15.5.18 per the April 00 meeting direction
- 20) Added Bit-Error-Rate Thresholding text per 00-175v0 as clause 23.2.7
- 21) Obsoleted Data Encryption per 00-079v0
- 22) Added the Automatic Link Speed Negotiation per 99-649v1
- 23) Made changes per 99-189v0 except those from FC-Tape clause 4.2 (not approved) and that listed in 1.1 (don't understand what to change - I can't get any guidance from FC-Tape, also conflicts with 99-179v0, 2.4)
- 24) Added new type codes per 98-577v0
- 25) Made changes to the type codes per 99-549v1
- 26) Made changes to describe class 6 in a consistent manner per 00-337v0 (Don't believe this is complete)
- 27) Made changes to class 6 multicast per 00-216v0
- 28) Corrected typos in Clause 17 per 00-338v0

- 29) Deleted ACK_N (00-339v0)
- 30) Made changes to 12.4 per 00-221v0
- 31) Changed PH to FS where appropriate

Version 0.10 (July 28, 1999)

- 1) Section 28.3 - changed the numbering to be continuous (99-075v2, item 22)
- 2) Section 17.7, Table 52 - changed the designation for Hunt Group from H to G and Dynamic Half Duplex from H to Y (99-075v2, item 29)
- 3) Removed Annex X, Hunt Group Models (99-075v2, item 33)
- 4) Section 13.6.4 – changed the first sentence per 98-569v0
- 5) Made changes per 98-570v0, Preference Control bit in CS_CTL
- 6) Made changes per 99-365v0, Proposal to move the Preemption/Priority field from the OX_ID field to the CS_CTL field (this supersedes 99-011v0)
- 7) Made changes per 99-094v0, MCM capable login bit
- 8) Made changes per 99-307v0, Resolution of Stacked Connect-request issues
- 9) Made relevant changes per 99-169v0, PH-X rev 4.00 errata (items 2 and 6 are PI issues and 7 has been superseded by 6 above)
- 10) Made changes per 99-362v0, World Wide IDs
- 11) Made changes per 99-071v2, Usage of the IEEE-Extended format
- 12) Made changes per 99-311v0, items 1, 4, 5, 7 and 9.
- 13) Made changes per 99-305, request for new TYPE codes for FC-SB-2

Version 0.00 (March 2, 1999)

- 1) Based off of FC/C FC-PH-x rev 4.00 (T11/98-126v0).
- 2) Remove FC-PI Clauses 5-10, Annexes A-J
- 3) References to deleted clauses/Annexes were changed to "see FC-PI".
- 4) Annex numbering remained K-AD to make it easier to review (same Annexes as PH-x)

Table of Contents

1	Scope.....	1
2	Normative references	1
2.1	Approved references	1
2.2	References under development.....	2
3	Definitions and conventions	3
3.1	Definitions.....	3
3.2	Editorial conventions.....	11
3.3	Abbreviations, acronyms, and symbols	12
3.3.1	Data rate abbreviations	12
3.3.2	Synonyms	12
3.3.3	Acronyms and other abbreviations.....	12
3.3.4	Symbols	15
4	Structure and Concepts	16
4.1	FC-1 general description	17
4.2	FC-2 general description	18
4.3	FC-FS physical model	18
4.3.1	Node and N_Port identifiers.....	20
4.3.2	Link_Control_Facility (LCF)	20
4.4	Communication models	20
4.4.1	Hunt Group.....	20
4.4.2	Dedicated Simplex	20
4.4.3	Fractional Bandwidth.....	20
4.5	Bandwidth	21
4.6	Topology	21
4.6.1	Point-to-point topology	21
4.6.2	Fabric topology	22
4.6.3	Arbitrated Loop topology.....	22
4.7	Classes of service.....	23
4.7.1	Class 1 service - Dedicated Connection.....	23
4.7.2	Class 2 service - Multiplex.....	23
4.7.3	Class 3 service - Datagram.....	24
4.7.4	Class 4 service –	24
4.7.5	Class 6 – Multicast Connection	24
4.8	Intermix	24
4.9	General Fabric model.....	25
4.9.1	Fabric Ports (F_Ports)	27
4.9.2	Connection based Sub-Fabric	27
4.9.3	Connectionless Sub-Fabric	28
4.10	Fibre Channel services.....	28
4.11	Building Blocks.....	28
4.11.1	Building block hierarchy	28
4.11.2	Frame	29

4.11.3	Sequence	30
4.11.3.1	Sequence_Identifier (SEQ_ID).....	30
4.11.3.2	Sequence Status Blocks.....	30
4.11.4	Exchange	30
4.11.4.1	Exchange_Identifiers (OX_ID and RX_ID)	31
4.11.4.2	Association_Header	31
4.11.4.3	Exchange Status Blocks.....	31
4.11.5	Protocol	31
4.12	Segmentation and reassembly	32
4.12.1	Application data mapping	32
4.12.2	Sending end mapping.....	32
4.12.3	Relative Offset	32
4.12.4	Capability	32
4.12.5	FC-2 mapping.....	33
4.12.6	Segmentation	34
4.12.7	Reassembly.....	35
4.13	Error detection and recovery.....	35
4.14	FC-4 Regions	36
5	FC-1 8B/10B Transmission Code	39
5.1	Notation conventions.....	39
5.2	Character encoding and decoding	40
5.2.1	Transmission order	40
5.2.2	Valid and invalid Transmission Characters.....	40
5.2.2.1	Use of the tables for generating Transmission Characters	41
5.2.2.2	Use of the tables for checking the validity of received Transmission Characters.....	42
5.3	Word encoding and decoding.....	46
5.4	Ordered Sets	47
6	FC-1 receiver and transmitter description.....	49
6.1	Receiver state description	49
6.1.1	Receiver states.....	49
6.1.1.1	Operational states	49
6.1.1.2	Not Operational state.....	49
6.1.2	Entry into Synchronization-Acquired state.....	49
6.1.2.1	Bit Synchronization.....	49
6.1.2.2	Transmission-Word synchronization.....	50
6.1.3	Entry into Loss-Of-Synchronization state	50
6.1.3.1	Loss-of-Synchronization procedure	51
6.1.3.2	Transition to power on	53
6.1.3.3	Exit from receiver reset condition	53
6.1.3.4	Detection of loss of signal	53
6.1.4	Entry into Reset state.....	54
6.2	Receiver state diagram	54
6.3	Transmitter state description.....	54
6.3.1	Transmitter states	55

6.3.1.1	Operational states	55
6.3.1.2	Not Operational state	55
6.3.2	Entry into Working state	55
6.3.3	Entry into Not-Enabled state	56
6.3.4	Entry into Open-Fibre state	56
6.3.5	Entry into Failure state	56
6.4	Transmitter state diagram	56
7	Loopback mode	58
7.1	Receiver considerations	58
7.2	Transmitter considerations	58
8	Diagnostic mode	59
9	Transmitter safety	60
10	Ordered Sets	61
10.1	Introduction	61
10.2	Frame delimiters	61
10.2.1	Start_of_Frame (SOF)	61
10.2.2	End_of_Frame (EOF)	61
10.3	Primitive Signals	61
10.3.1	Idle	62
10.3.2	Receiver_Ready (R_RDY)	62
10.3.3	BB_SCs	62
10.3.4	BB_SCr	62
10.4	Primitive Sequences	62
10.4.1	Primitive Sequence Recognition	63
10.4.2	Not_Operational (NOS)	63
10.4.3	Offline (OLS)	63
10.4.4	Link Reset (LR)	63
10.4.5	Link Reset Response (LRR)	63
10.5	Port states	63
10.5.1	Active State (AC)	64
10.5.2	Link Recovery State	64
10.5.2.1	LR Transmit State (LR1)	64
10.5.2.2	LR Receive State (LR2)	65
10.5.2.3	LRR Receive State (LR3)	66
10.5.3	Link Failure State	66
10.5.3.1	NOS Transmit State (LF2)	66
10.5.3.2	NOS Receive State (LF1)	67
10.5.4	Offline State	67
10.5.4.1	OLS Transmit State (OL1)	68
10.5.4.2	OLS Receive State (OL2)	68
10.5.4.3	Wait for OLS State (OL3)	68
10.6	Primitive Sequence Protocols	69
10.6.1	Primitive Sequence meanings	69
10.6.2	Link Initialization	69

10.6.3	Online to Offline	70
10.6.4	Link Failure	70
10.6.5	Link Reset	71
11	Frame formats	72
11.1	Frame transmission	72
11.2	Start_of_Frame (SOF)	72
11.2.1	Start_of_Frame Connect Class 1 or 6 (SOFc1)	72
11.2.2	Start_of_Frame Initiate (SOFix)	72
11.2.2.1	Start_of_Frame Initiate Class 1 or 6 (SOFi1)	73
11.2.2.2	Start_of_Frame Initiate Class 2 (SOFi2)	73
11.2.2.3	Start_of_Frame Initiate Class 3 (SOFi3)	73
11.2.3	Start_of_Frame Normal (SOFnx)	73
11.2.3.1	Start_of_Frame Normal Class 1 or 6 (SOFn1)	73
11.2.3.2	Start_of_Frame Normal Class 2 (SOFn2)	73
11.2.3.3	Start_of_Frame Normal Class 3 (SOFn3)	73
11.2.4	Start_of_Frame Fabric (SOFf)	73
11.3	Frame_Header	73
11.4	Data Field	73
11.5	CRC	74
11.6	End_of_Frame (EOF)	74
11.6.1	Valid frame content	74
11.6.1.1	End_of_Frame Terminate (EOFt)	75
11.6.1.2	End_of_Frame Disconnect_Terminate (EOFdt)	75
11.6.1.3	End_of_Frame Normal (EOFn)	75
11.6.2	Invalid frame content	75
11.6.2.1	End_of_Frame Disconnect_Terminate_Invalid (EOFdti)	75
11.6.2.2	End_of_Frame Invalid (EOFni)	76
11.6.3	End_of_Frame Abort (EOFa)	76
11.7	Frame field order	76
11.8	Frame reception	77
11.8.1	Frame validity	77
11.8.2	Invalid frame processing	78
11.9	Fabric_Frame	78
12	Frame Header	79
12.1	Introduction	79
12.1.1	Frame identification	79
12.1.2	Sequence identification	79
12.2	Routing Control (R_CTL)	80
12.3	Address identifiers (D_ID, S_ID)	82
12.3.1	Destination_ID (D_ID)	83
12.3.2	Source_ID (S_ID)	83
12.4	Word 1, bits 31-24	83
12.4.1	Class Specific Control (CS_CTL)	84
12.4.1.1	Class 1	84
12.4.1.2	Class 2	84

12.4.1.3 Class 3	84
12.4.1.4 Class 4	85
12.4.1.5 Class 6	85
12.4.2 Priority	86
12.5 Data structure type (TYPE)	86
12.6 Frame Control (F_CTL)	88
12.6.1 F_CTL Summary	95
12.6.1.1 F_CTL bits on Data frames	95
12.6.1.2 F_CTL bits on Link_Control	96
12.7 Sequence_ID (SEQ_ID)	97
12.8 DF_CTL	98
12.9 Sequence count (SEQ_CNT)	99
12.10 Originator Exchange_ID (OX_ID)	99
12.11 Responder Exchange_ID (RX_ID)	99
12.12 Parameter	100
13 Optional headers	101
13.1 Introduction	101
13.2 Expiration_Security_Header	102
13.3 Network_Header	102
13.3.1 D_NAA or S_NAA	103
13.3.2 Network_Destination_ID or Network_Source_ID	103
13.3.2.1 IEEE 48-bit address	103
13.3.2.2 IEEE extended	104
13.3.2.3 Locally assigned	104
13.3.2.4 32-bit IP address	104
13.3.2.5 IEEE registered	105
13.3.2.6 IEEE registered extended	105
13.3.3 Application summary	106
13.4 Association_Header	107
13.4.1 Process_Associators	109
13.4.1.1 Originator and Responder Process_Associators	109
13.4.1.2 Multicast Process_Associator	109
13.4.2 Operation_Associators	109
13.5 Device_Header	109
13.6 Optional header usage	110
13.6.1 Network_Header	110
13.6.2 Association_Header	110
13.6.3 Device_Header	110
13.6.4 Link Service	110
13.6.5 Summary	110
14 Data frames and responses	111
14.1 Introduction	111
14.2 Data frames	111
14.2.1 R_RDY response	114
14.2.2 Data frame responses	114

14.2.2.1	ACK frames	114
14.2.2.2	Link_Response frames.....	114
14.2.3	Transfer mechanisms	115
14.3	Link_Control	115
14.3.1	R_RDY response.....	117
14.3.2	Link_Continue function	117
14.3.2.1	Receiver Ready (R_RDY).....	118
14.3.2.2	Acknowledge (ACK)	118
14.3.3	Link_Response	120
14.3.3.1	Fabric Busy (F_BSY)	121
14.3.3.2	N_Port Busy (P_BSY)	122
14.3.3.3	Reject (P_RJT, F_RJT).....	125
14.3.4	Link_Control commands	131
14.3.4.1	Link Credit Reset (LCR)	131
14.3.4.2	Notify (NTY).....	132
14.3.4.3	End (END)	132
14.4	Data frame summary	133
14.5	ACK generation assistance	134
14.5.1	N_Port_Login.....	134
14.5.1.1	Capability Indicator.....	134
14.5.1.2	Precedence requirements.....	134
14.5.2	Terms	134
14.5.3	Applicability.....	134
14.5.4	F_CTL bits	135
14.5.5	Login rules	135
14.5.6	ACK_Form errors	135
15	Link Services	136
15.1	Introduction	136
15.1.1	Default Login values	136
15.1.2	Sequence and Exchange management	136
15.2	Basic Link Service commands	137
15.2.1	Abort Sequence (ABTS)	138
15.2.1.1	Aborting Sequences using ABTS	139
15.2.1.2	Aborting an Exchange using ABTS.....	142
15.2.2	Basic Accept (BA_ACC)	145
15.2.3	Basic Reject (BA_RJT).....	145
15.2.4	No Operation (NOP).....	147
15.2.5	Remove Connection (RMC)	147
15.2.6	Dedicated Connection Preempted (PRMT)	148
15.3	Basic Link Service summary	148
15.4	Extended Link Services	149
15.4.1	Extended Link Service command codes.....	150
15.5	Extended Link Service requests	152
15.5.1	Link Service request collisions	153
15.5.2	Abort Exchange (ABTX).....	154

15.5.3 Advise Credit (ADVC).....	155
15.5.4 Echo (ECHO).....	157
15.5.5 Estimate Credit (ESTC).....	158
15.5.6 Establish Streaming (ESTS)	159
15.5.7 Login (FLOGI/PLOGI).....	160
15.5.8 Logout (LOGO)	162
15.5.9 Read Connection Status (RCS)	163
15.5.10 Read Exchange Status Block (RES).....	164
15.5.11 Read Link Error Status Block (RLS)	165
15.5.12 Read Sequence Status Block (RSS).....	166
15.5.13 Read Timeout Value (RTV)	168
15.5.14 Reinstate Recovery Qualifier (RRQ)	169
15.5.15 Request Sequence Initiative (RSI)	170
15.5.16 Test (TEST)	172
15.5.17 Report Node Capability Information (RNC)	172
15.5.18 Fabric Address Notification (FAN).....	173
15.5.19 Loop Initialize (LINIT)	173
15.5.20 Loop Port Control (LPC).....	175
15.5.21 Loop Status (LSTS)	177
15.5.22 Registered State Change Notification (RSCN)	180
15.5.22.1 RSCN issued by the Fabric Controller.....	180
15.5.22.2 RSCN issued by the Affected N_Port or NL_Port	180
15.5.22.3 RSCN initiative	181
15.5.22.4 RSCN registration	181
15.5.23 State Change Registration (SCR)	184
15.5.24 Process Extended link services	185
15.5.25 PRLI/PRLO command codes	185
15.5.26 Process login	186
15.5.26.1 PRLI/PRLO Relationships	193
15.5.26.2 Process login, mode of operation	194
15.5.26.3 Process login, protocol	194
15.5.27 Process logout	195
15.5.27.1 PRLO Operation	200
15.5.28 State change notification	200
15.5.28.1 SCN issued by a F_Port	201
15.5.28.2 SCN issued by a N_Port	201
15.5.28.3 SCN initiative	201
15.5.28.4 Interest registration	202
15.5.29 Test process login state	203
15.5.30 Fibre Channel Address Resolution Protocol (FARP)	206
15.5.30.1 Fibre Channel Address Resolution Protocol Request (FARP-REQ)	206
15.5.30.2 Fibre Channel Address Resolution Protocol Reply (FARP- REPLY)	210
15.5.31 Topology-Discovery Extended Link Service	212

15.5.31.1	Request Topology Information (RTIN).....	212
15.5.32	Self-Description Extended Link Service	219
15.5.32.1	Request Node Identification Data (RNID)	219
15.5.33	Link-Incident-Reporting Extended Link Services	226
15.5.33.1	Description of link incident reporting Extended Link Services	226
15.5.33.2	Registered Link Incident Record (RLIR)	226
15.5.33.3	Link Incident Record Registration (LIRR)	233
15.5.34	Fabric Link Services for Alias	236
15.5.34.1	Get Alias_ID (GAID)	236
15.5.34.2	Fabric Activate Alias_ID (FACT)	237
15.5.34.3	Fabric Deactivate Alias_ID (FDACT)	238
15.5.35	N_Port Link Services for Alias	239
15.5.35.1	N_Port Activate Alias_ID (NACT)	239
15.5.35.2	N_Port Deactivate Alias_ID (NDACT)	241
15.5.36	Class 4 Extended Link Services.....	241
15.5.36.1	Quality of Service Request (QoSR)	241
15.5.36.2	Read Virtual Circuit Status (RVCS)	247
15.5.37	Enhanced Login/Logout Link Service	248
15.5.37.1	Discover N_Port/F_Port Service Parameters (PDISC/FDISC)	248
15.5.37.2	Discover Address (ADISC)	249
15.5.37.3	Third Party Process Logout (TPRLO).....	250
15.5.38	Extended Link Services for Bandwidth Allocation.....	252
15.5.38.1	Bandwidth Allocation (BALLOC)	252
15.5.38.2	Bandwidth Deallocation (DALLOC)	253
15.5.39	Extended Link Services for Clock Synchronization.....	255
15.5.39.1	Clock Synchronization Request (CSR).....	255
15.5.39.2	Clock Synchronization Update (CSU).....	256
15.5.40	Login Control List Management	256
15.5.40.1	Login Control List Management (LCLM)	257
15.5.40.2	LCLM Operation.....	260
15.5.41	Read Port Status Block (RPS)	262
15.5.42	Read Port List (RPL).....	265
15.5.43	Report Node FC-4 Types (RNFT).....	267
15.5.44	Extended Link Service summary.....	269
15.6	Extended Link Service Reply Sequences	272
15.6.1	Accept (ACC).....	273
15.6.2	Link Service Reject (LS_RJT)	273
15.7	FC-4 Link Service	276
16	Classes of service	278
16.1	Class 1 - Dedicated Connection	278
16.1.1	Class 1 function	278
16.1.2	Class 1 rules.....	279
16.1.3	Class 1 delimiters	280

16.1.4	Class 1 frame size	281
16.1.5	Class 1 flow control.....	281
16.1.6	Stacked connect-requests.....	281
16.2	Class 2 - Multiplex	281
16.2.1	Class 2 function	281
16.2.2	Class 2 rules.....	281
16.2.3	Class 2 delimiters	283
16.2.4	Class 2 frame size	283
16.2.5	Class 2 flow control.....	283
16.3	Class 3 - Datagram	283
16.3.1	Class 3 function	283
16.3.2	Class 3 rules.....	283
16.3.3	Class 3 delimiters	285
16.3.4	Class 3 frame size	285
16.3.5	Class 3 flow control.....	285
16.3.6	Class 3 Sequence integrity	285
16.4	Intermix	285
16.4.1	Fabric management	285
16.4.2	Intermix rules	286
16.4.3	Intermix delimiters	287
16.4.4	Intermix frame size	287
16.4.5	Intermix flow control.....	287
16.5	Class 4.....	287
16.5.1	Class 4 functions	287
16.5.2	Class 4 rules.....	289
16.5.3	Class 4 intermix.....	290
16.5.4	Class 4 delimiters	291
16.5.5	Class 4 frame size	291
16.5.6	Class 4 flow control.....	291
16.6	Class 6 Multicast Connection.....	291
16.6.1	Class 6 function	292
16.6.2	Class 6 rules.....	292
16.6.3	Class 6 delimiters	294
16.6.4	Class 6 frame size	294
16.6.5	Class 6 flow control.....	294
16.6.6	Stacked Connect-requests	294
17	Login and Service Parameters.....	295
17.1	Introduction	295
17.2	Applicability.....	296
17.3	Fabric Login	296
17.3.1	Explicit Fabric Login	297
17.3.2	Responses to Fabric Login.....	297
17.3.2.1	FLOGI with S_ID = 0	298
17.3.2.2	FLOGI with SID = X	299
17.3.3	SOF delimiters.....	300

17.3.4	Frequency	301
17.3.4.1	Login completion - Originator	301
17.3.4.2	Login completion - Responder	301
17.4	N_Port_Login.....	301
17.4.1	Address Identifiers	302
17.4.2	Explicit N_Port_Login.....	302
17.4.2.1	Fabric present.....	303
17.4.2.2	Responses to N_Port_Login (Fabric).....	303
17.4.2.3	No Fabric present (point-to-point)	304
17.4.2.4	Responses to N_Port_Login (point-to-point).....	305
17.4.3	SOF delimiters.....	306
17.4.4	Frequency	306
17.4.4.1	Login completion - Originator	306
17.4.4.2	Login completion - Responder	306
17.4.5	N_Port_Login frame flow	307
17.5	Logout.....	307
17.5.1	Introduction	307
17.5.2	Explicit N_Port Logout.....	307
17.5.3	Implicit Logout.....	307
17.6	N_Port Service Parameters	308
17.6.1	N_Port Common Service Parameters	308
17.6.2	N_Port Common Service Parameters - Fabric Login	309
17.6.2.1	FC-PH/FS version	310
17.6.2.2	Buffer-to-buffer Credit	311
17.6.2.3	Common features	311
17.6.2.4	BB_SC Number.....	312
17.6.2.5	Buffer-to-buffer Data_Field size.....	312
17.6.3	N_Port Common Service Parameters - N_Port_Login	312
17.6.3.1	FC-PH/FS version	313
17.6.3.2	Buffer-to-buffer Credit	313
17.6.3.3	Common features	314
17.6.3.4	Buffer-to-buffer Data_Field size.....	316
17.6.3.5	Total Concurrent Sequences	316
17.6.3.6	Relative Offset by category	316
17.6.3.7	Point-to-point E_D_TOV value.....	316
17.6.3.8	BB_SC Number.....	316
17.6.4	N_Port_Name	317
17.6.5	Node_Name.....	317
17.6.6	N_Port Class Service Parameters	317
17.6.7	N_Port Class Service Parameters - Fabric Login	319
17.6.7.1	Class validity (V)	319
17.6.7.2	Service options	320
17.6.7.3	Initiator control.....	323
17.6.7.4	Recipient control	323
17.6.7.5	Receive Data_Field Size.....	323

17.6.7.6	Concurrent Sequences	323
17.6.7.7	N_Port End-to-end Credit	323
17.6.7.8	Open Sequences per Exchange	324
17.6.8	N_Port Class Service Parameters - N_Port_Login	324
17.6.8.1	Class validity	324
17.6.8.2	Service options	324
17.6.8.3	Initiator control	326
17.6.8.4	Recipient control	328
17.6.8.5	Receive Data_Field Size	331
17.6.8.6	Concurrent Sequences	331
17.6.8.7	End-to-end Credit	332
17.6.8.8	Open Sequences per Exchange	332
17.6.8.9	Class 6 Multicast RX_ID	333
17.6.9	Vendor Version Level	333
17.6.10	Services Availability	333
17.7	F_Port Service Parameters	333
17.7.1	F_Port Common Service Parameters	333
17.7.1.1	FC-PH/FS version	334
17.7.1.2	Buffer-to-buffer (F_Port) Credit	334
17.7.1.3	Common features	335
17.7.1.4	BB_SC Number	336
17.7.1.5	Buffer-to-buffer Data_Field size	336
17.7.1.6	E_D_TOV	336
17.7.1.7	R_A_TOV	336
17.7.2	F_Port_Name	337
17.7.3	Fabric_Name	337
17.7.4	F_Port Class Service Parameters	337
17.7.4.1	Class validity (V)	337
17.7.4.2	Service options	338
17.7.4.3	Initiator control	342
17.7.4.4	Recipient control	342
17.7.4.5	Receive Data_Field Size	342
17.7.4.6	Concurrent Sequences	342
17.7.4.7	N_Port End-to-end Credit	342
17.7.4.8	Open Sequences per Exchange	342
17.7.4.9	CR_TOV	343
17.7.5	Vendor Version Level	343
17.7.6	Services Availability	343
17.8	Procedure to estimate end-to-end Credit	344
17.8.1	Introduction	344
17.8.2	Procedure steps	344
17.8.2.1	Establish Streaming Sequence	344
17.8.2.2	Estimate Credit Sequence	346
17.8.2.3	Advise Credit Sequence	346
18	Exchange, Sequence, and sequence count management	348

18.1	Introduction	348
18.2	Applicability.....	350
18.3	Summary rules	351
18.3.1	Exchange management	351
18.3.2	Exchange origination.....	351
18.3.3	Sequence delimiters.....	352
18.3.4	Sequence initiation	352
18.3.5	Sequence management	353
18.3.6	Sequence count	354
18.3.7	Normal ACK processing.....	354
18.3.8	Normal Sequence completion	355
18.3.9	Detection of missing frames.....	357
18.3.10	Sequence errors - Class 1 and 2.....	357
18.3.10.1	Rules common to all Discard policies	357
18.3.10.2	Discard multiple Sequences Error Policy.....	358
18.3.10.3	Discard a single Sequence Error Policy	359
18.3.10.4	Process with infinite buffers Error Policy.....	360
18.3.11	Sequence errors - Class 3.....	360
18.3.11.1	Rules common to all discard policies	360
18.3.11.2	Process with infinite buffers Error Policy.....	360
18.3.12	Sequence Status Rules.....	361
18.3.13	Exchange termination	361
18.3.14	Exchange Status Rules	361
18.4	Exchange management	362
18.5	Exchange origination.....	362
18.5.1	Exchange Originator	363
18.5.2	Exchange Responder	363
18.5.3	X_ID assignment.....	364
18.5.4	X_ID interlock.....	364
18.6	Sequence management	364
18.6.1	Active and Open Sequence.....	364
18.6.2	Sequence_Qualifier management.....	364
18.6.3	Sequence initiative and termination.....	365
18.6.4	Transfer of Sequence Initiative	365
18.6.5	Sequence termination.....	365
18.7	Exchange termination.....	366
18.7.1	Normal termination	366
18.7.2	Abnormal termination	367
18.8	Status blocks	367
18.8.1	Exchange Status Block.....	367
18.8.2	Sequence Status Block.....	369
19	Association_Header management and usage.....	371
19.1	Introduction	371
19.2	Establishing the final Association_Header.....	371
19.2.1	Exchange Origination	371

19.2.2	First transfer rules.....	372
19.2.3	Final transfer rules.....	372
19.3	Extended Link Services.....	372
19.4	Error Recovery and other effects	373
20	Flow control management	374
20.1	Introduction	374
20.2	Physical flow control model	374
20.3	Credit and Credit_Count.....	376
20.4	End-to-end flow control	377
20.4.1	End-to-end management rules summary.....	377
20.4.2	Sequence Initiator.....	377
20.4.3	Sequence Recipient	377
20.4.3.1	ACK_0.....	378
20.4.3.2	ACK_1.....	378
20.4.3.3	Last ACK timeout	378
20.4.3.4	Streamed Sequences.....	378
20.4.4	EE_Credit.....	379
20.4.5	EE_Credit_Count	379
20.4.6	EE_Credit management.....	379
20.4.7	End-to-end flow control model.....	379
20.4.8	End-to-end Class dependency.....	381
20.4.9	EE_Credit recovery	381
20.5	Buffer-to-buffer flow control.....	381
20.5.1	Buffer-to-buffer management rules summary	382
20.5.2	BB_Credit.....	382
20.5.3	BB_Credit_Count	382
20.5.4	BB_Credit management.....	382
20.5.5	Buffer-to-buffer flow control model.....	382
20.5.6	Buffer-to-buffer Class dependency	383
20.5.7	Class dependent frame flow.....	383
20.5.8	R_RDY.....	389
20.5.9	BB_Credit_Count reset.....	389
20.5.10	Alternate buffer-to-buffer Credit management.....	389
20.5.11	Alternate BB_Credit management rules summary	389
20.5.12	BB_Credit Recovery	390
20.6	BSY / RJT in flow control.....	391
20.7	LCR in flow control.....	391
20.8	Integrated Class 2 flow control.....	393
20.9	Intermix	394
20.10	Point-to-point topology	394
21	Segmentation and reassembly.....	397
21.1	Introduction	397
21.2	Sending end	397
21.2.1	Relative Offset space.....	397
21.2.2	Block	397

21.2.3 Subblock	397
21.2.4 Sequence	397
21.2.5 Relationship between Sequences	398
21.3 FC-2	398
21.4 Login	398
21.5 Segmentation rules summary	399
21.6 Reassembly rules summary	399
22 Connection management	402
22.1 Introduction	402
22.2 Applicability	403
22.3 Topology models	404
22.3.1 Fabric model	404
22.3.2 Point to Point model	405
22.4 Connect / disconnect rules	405
22.4.1 Connect-request rules	405
22.4.1.1 Source of connect-request	405
22.4.1.2 Destination of connect-request	406
22.4.2 Connection Rules	407
22.4.3 Remove Connection rules	407
22.5 Establishing a Connection	408
22.5.1 Connection Initiator	408
22.5.2 Stacked connect-requests	410
22.5.3 Unidirectional Dedicated Connection	411
22.5.4 Destination of connect-request	411
22.6 Connected	412
22.7 Removing a Connection	412
22.7.1 When to remove a Connection	412
22.7.2 End_Connection bit	412
22.7.3 EOFdt transmission	413
22.8 Connection Recovery	413
22.8.1 Link timeout	414
22.8.2 Corrupted connect-request	414
22.9 Connection Preemption	414
22.9.1 Applicability	414
22.9.2 Topology Model	414
22.9.3 Rules for Preemption	414
22.9.3.1 Preemptor (P)	414
22.9.3.2 Preempted Source (PS)	415
22.9.3.3 Preempted Destination(s)(PD)	415
22.9.3.4 Preemption Destination(s) (PnD)	416
22.9.4 Connection Rules	416
22.9.5 Remove Connection Rules	416
22.10 Establishing a Connection Using Preemption	416
22.10.1 Connection Initiator	416
22.10.2 Preemption Destination	418

23	Error detection/recovery	419
23.1	Introduction	419
23.2	Timeouts	419
23.2.1	Timeout periods	419
23.2.1.1	R_T_TOV	419
23.2.1.2	E_D_TOV	419
23.2.1.3	R_A_TOV	420
23.2.2	Link Failure timeouts	420
23.2.3	Link timeout	420
23.2.4	Sequence timeout	421
23.2.4.1	Class 1 and 2	421
23.2.4.2	Class 3	421
23.2.4.3	End-to-end Class 2 Credit loss	422
23.2.5	OLS transmit timeout	422
23.2.6	Timer Enhancements	422
23.2.6.1	Terms	423
23.2.6.2	Applicability	423
23.2.6.3	Login	423
23.2.6.4	Value	423
23.2.6.5	Stacked Connect-request	423
23.2.6.6	Rules	423
23.2.7	Bit-Error-Rate Thresholding	424
23.2.7.1	Types of Link Errors Caused by Bit Errors	424
23.2.7.2	Error Bursts	424
23.2.7.3	Bit-Error-Rate-Thresholding Measurement	425
23.3	Link error detection	425
23.3.1	Link Failure	425
23.3.2	Code violations	425
23.3.3	Primitive Sequence protocol error	425
23.4	Link error recovery	425
23.5	Link recovery - secondary effects	427
23.6	Exchange Integrity	427
23.6.1	Exchange management	428
23.6.1.1	Exchange Error Policies	428
23.6.2	Sequence integrity	429
23.6.3	Sequence error detection	429
23.6.4	X_ID processing	429
23.7	Sequence recovery	429
23.7.1	Abnormal Sequence termination	430
23.7.1.1	Abort Sequence Protocol	430
23.7.1.2	Class 1 Sequence retransmission	431
23.7.1.3	Recipient abnormal termination	433
23.7.1.4	End_Sequence	433
23.7.2	Stop Sequence Protocol	433
23.7.3	End-to-end Credit loss	434

23.8	Link Error Status Block.....	434
23.9	Detailed error detection / actions	435
23.9.1	Errors detected	435
23.9.2	Actions by Initiator or Recipient	437
24	Hunt Group	440
24.1	Introduction	440
24.2	Function	440
24.3	Communication Model	440
24.4	Applicability	440
24.4.1	Classes	440
24.4.2	Class 4 Addressing	440
24.5	Formation	441
24.5.1	Registration / deregistration	441
24.5.2	Inquiry	442
24.5.3	Hunt Group ID Removal	442
24.6	Destination N_Port_Login	442
24.7	Addressing Protocol	443
24.7.1	Originator	443
24.7.2	Responder	443
24.7.3	Address Resolution	443
24.7.4	Class	444
24.7.5	Class 4	444
24.8	Resource management	444
24.8.1	HG_ID use	444
24.8.2	N_Port ID use	445
24.9	Rotary Group	445
24.9.1	Introduction	445
24.9.2	Function	445
25	Multicast	447
25.1	Introduction	447
25.2	Class 6 Multicast	447
25.2.1	Class 6 Multicast Routing	447
25.2.2	Class 6 Multicast Rules	448
25.2.3	Class 6 Multicast Server	449
25.2.4	Class 6 Multicast recovery	449
25.3	Class 3 Multicast	449
25.3.1	Registration and De-registration	449
25.3.2	Multicast Routing	449
25.3.3	Class 3 Multicast rules	450
25.4	Broadcast	451
25.5	Moviecast	451
25.6	Other	451
26	Aliases	452
26.1	Introduction	452

26.2	Alias Server	452
26.3	Alias Service protocol	452
26.4	Alias Routing	452
26.5	Function Flow	453
26.6	IPA Considerations	453
26.6.1	Hunt Groups	453
26.6.2	Multicast Groups	453
26.6.3	Broadcast	454
27	Dedicated Simplex	455
27.1	Introduction	455
27.2	Terms	455
27.3	Communications Model	455
27.4	Applicability	457
27.4.1	Permitted Functions	457
27.4.2	Restricted Functions	457
27.4.3	Topology	458
27.5	Requirements	458
27.5.1	Connection_Initiator N_Port	458
27.5.2	Connection_Recipient N_Port	458
27.5.3	Fabric	458
27.6	Connection Management	458
27.7	Login	458
27.7.1	N_Port Class Service Parameters	458
27.7.2	F_Port Class Service Parameters	458
27.8	Delimiters	459
27.8.1	Start of Frame	459
27.8.2	End of Frame	459
27.9	Connections	459
27.9.1	Establishment	460
27.9.2	Removal	460
27.10	Rules	461
27.10.1	Restriction	461
27.10.2	Connection_Initiator N_Port	461
27.10.3	Connection_Recipient N_Port	462
27.10.4	Fabric	462
27.10.5	F_Port	463
27.10.6	ACK Processing and Routing	463
27.11	Error Recovery	463
27.11.1	Connection_Initiator	463
27.11.2	Connection_Recipient	463
28	Class 4 – Fractional	465
28.1	Introduction	465
28.1.1	Topology	466
28.2	Communication Model	466
28.2.1	In-order delivery	467

28.2.2	Guaranteed delivery.....	467
28.2.3	Fractional bandwidth management	467
28.2.4	Class 4 circuit.....	468
28.2.4.1	Class 4 circuit – Setup	469
28.2.4.2	Class 4 circuit – Activation	472
28.2.4.3	Class 4 circuit – Deactivation	473
28.2.4.4	Class 4 circuit – Removal	475
28.3	Class 4 circuit – Deactivation, removal contention and error recovery	476
28.4	Ordered Sets for Class 4	478
28.4.1	Start Of Frame (SOF).....	478
28.4.1.1	SOF – Circuit Activate Class 4 (SOFc4).....	478
28.4.1.2	SOF – Initiate Class 4 (SOFi4)	479
28.4.1.3	SOF – Normal (SOFn4).....	480
28.4.2	End_of_Frame (EOF)	480
28.4.2.1	EOF – Deactivate Terminate (EOFdt)	480
28.4.2.2	EOF – Deactivate Terminate Invalid (EOFdti).....	480
28.4.2.3	EOF – Remove Terminate (EOFrt).....	480
28.4.2.4	EOF – Remove Terminate Invalid (EOFrti).....	481
28.4.3	Virtual Circuit Ready (VC_RDY)	481
28.5	Primitive Sequences	481
28.5.1	LR Transmit state (LR1)	481
28.5.2	LR Receive state (LR2)	481
28.5.3	LRR Receive state (LR3).....	482
28.5.4	NOS Receive state (LF1).....	482
28.5.5	NOS Transmit state (LF2).....	482
28.5.6	OLS Transmit state (OL1).....	482
28.5.7	OLS Receive state (OL2).....	483
28.5.8	Wait for OLS state (OL3).....	483
28.6	Data frames and responses	483
28.6.1	Class 4 frame size	483
28.6.2	QoSF Well-known Address identifier	483
28.6.3	Class 4 – CS_CTL field usage.....	483
28.6.4	Frame control (F_CTL)	484
28.6.5	Data Frames.....	484
28.6.6	Link_Continue function	485
28.6.6.1	VC_RDY Primitive Signal.....	485
28.6.6.2	ACK frames	485
28.6.7	Link_Response frames	486
28.6.7.1	Reject (P_RJT, F_RJT).....	486
28.6.7.2	Fabric Busy (F_BSY)	487
28.6.7.3	N_Port Busy (P_BSY)	487
28.6.8	Link Credit Reset (LCR) Link_Control command.....	487
28.6.9	END Link_Control command	487
28.7	Login and Service Parameters	487
28.8	Link Services.....	488

28.8.1	Basic Link Services	488
28.8.2	Extended Link Services	488
28.9	Class 4 procedures	488
28.9.1	Class 4 login.....	489
28.9.2	Class 4 rules - summary	489
28.9.3	Class 4 intermix.....	491
28.9.4	Class 4 flow control.....	491
28.9.4.1	End-to-end flow control	491
28.9.4.2	Buffer-to-buffer flow control	491
28.10	Quality of Service	492
29	Camp-On	494
29.1	Introduction	494
29.2	FC-2 Mechanisms	494
29.3	Applicability.....	494
29.4	Communications Model	494
29.5	Requirements	495
29.5.1	Connection Initiator.....	495
29.5.2	Connection Recipient	495
29.5.3	Fabric	495
29.6	Login	495
29.6.1	F_Port Class Service Parameters.....	495
29.7	Camp-On Invoked.....	495
29.8	Rules	496
29.8.1	Connection Initiator.....	496
29.8.2	Connection Recipient	496
29.8.3	Fabric	496
29.8.4	Multiple requests.....	497
30	Stacked Connect-request	498
30.1	Introduction	498
30.2	FC-2 Mechanisms	498
30.3	Applicability.....	498
30.4	Communications Model	498
30.5	Requirements	498
30.5.1	Connection Initiator.....	498
30.5.2	Connection Recipient	498
30.5.3	Fabric	498
30.6	Login	499
30.7	Stacked Connect-request Invoked	499
30.8	Timer	499
30.8.1	CR_TOV.....	499
30.8.2	E_D_TOV	499
30.9	Rules	499
30.9.1	Connection Initiator.....	499
30.9.2	Connection Recipient	499
30.9.3	Fabric	499

30.9.4 Multiple requests.....	500
31 Buffered Class 1 Service	501
31.1 Fabric Login	501
31.2 Applicability.....	501
31.2.1 Topology	501
31.3 Fabric guaranteed bandwidth	501
31.4 Delimiters.....	501
31.5 CS_CTL bit.....	502
31.6 Rules	502
31.6.1 Connection_Initiator N_Port.....	502
31.6.2 Connection_Recipient N_Port	502
31.6.3 F_Port.....	502
31.6.4 BB_Credit rules for Buffered Class 1	503
31.6.5 Buffered Class 1 frame size.....	503
31.6.6 Intermix.....	503
32 Clock synchronization service	505
32.1 Introduction	505
32.1.1 References	505
32.1.2 Applicability.....	505
32.1.3 Function	505
32.2 ELS Command Service	505
32.2.1 Assumptions	505
32.2.2 Fabric Topology	505
32.2.2.1 Server Rules	506
32.2.2.2 Server Options	506
32.2.2.3 Fabric Rules.....	507
32.2.2.4 Client Rules	507
32.2.2.5 Client Options	507
32.2.3 Loop Topology	507
32.2.3.1 L_Port Server Rules	508
32.2.3.2 L_Port Server Options	508
32.2.3.3 L_Port Client Rules.....	509
32.2.3.4 L_Port Client Options.....	509
32.2.4 Use of FC-FS Constructs.....	509
32.2.4.1 Login/Logout	509
32.2.4.2 Common Required FC Parameters.....	509
32.2.4.3 Clock Synchronization Request (CSR) ELS command	509
32.2.4.4 Clock Synchronization Update (CSU) ELS command.....	511
32.3 Primitive Signal Service	511
32.3.1 Communication Model	512
32.3.2 Requirements	512
32.3.2.1 Server Rules	514
32.3.2.2 Fabric Rules.....	515
32.3.2.3 Client Rules	515
33 Link Speed Negotiation	516

33.1	Speed Negotiation overview	516
33.2	Link physical architecture and requirements.....	516
33.3	Speed Negotiation requirements on L_Ports.....	517
33.4	Speed Negotiation algorithm.....	517
33.4.1	State machines	518
33.4.2	Primitives	518
33.4.3	Algorithm overview	518
33.4.4	Stage 1 - Wait_for_signal	519
33.4.5	Stage 2 - Negotiate_master (& Negotiate_fail)	520
33.4.6	Stage 3 - Negotiate_follow.....	523
33.4.7	Stage 4 - Normal operation.....	524
33.4.8	Timing requirements	524
Annex A	(informative) FC-1 service interface	526
A.1	FC-1_Receiver_State.Indication.....	528
A.1.1	Function	528
A.1.2	Semantics	528
A.1.3	When generated	528
A.1.4	Effect on receipt	529
A.1.5	Additional comments	529
A.2	FC-1_Transmitter_State.Indication	529
A.2.1	Function	529
A.2.2	Semantics	529
A.2.3	When generated	529
A.2.4	Effect on receipt	530
A.2.5	Additional comments	530
A.3	FC-1_Data_Path_Width.Request.....	530
A.3.1	Function	530
A.3.2	Semantics	530
A.3.3	When generated	531
A.3.4	Effect on receipt	531
A.3.5	Additional comments	531
A.4	FC-1_Data_Path_Width.Confirm	531
A.4.1	Function	531
A.4.2	Semantics	531
A.4.3	When generated	531
A.4.4	Effect on receipt	531
A.4.5	Additional comments	532
A.5	FC-1_Transmitter_Clock.Indication	532
A.5.1	Function	532
A.5.2	Semantics	532
A.5.3	When generated	532
A.5.4	Effect on receipt	533
A.5.5	Additional comments	533
A.6	FC-1_Data.Request.....	533
A.6.1	Function	533

A.6.2	Semantics	533
A.6.3	When generated	534
A.6.4	Effect on receipt	534
A.6.5	Additional comments	535
A.7	FC-1_Data.Indication	535
A.7.1	Function	535
A.7.2	Semantics	535
A.7.3	When generated	536
A.7.4	Effect on receipt	538
A.7.5	Additional comments	538
A.8	FC-1_Receiver_Control.Request	538
A.8.1	Function	538
A.8.2	Semantics	539
A.8.3	When generated	539
A.8.4	Effect on receipt	539
A.8.5	Additional comments	539
A.9	FC-1_Transmitter_Control.Request	539
A.9.1	Function	539
A.9.2	Semantics	539
A.9.3	When generated	539
A.9.4	Effect on receipt	539
A.9.5	Additional comments	540
A.10	FC-1_Loopback_Control.Request	540
A.10.1	Function	540
A.10.2	Semantics	540
A.10.3	When generated	540
A.10.4	Effect on receipt	540
A.10.5	Additional comments	541
A.11	FC-1_Loopback_Control.Confirm	541
A.11.1	Function	541
A.11.2	Semantics	541
A.11.3	When generated	541
A.11.4	Additional comments	541
Annex B (informative)	Communication models	542
B.1	Model 1 example	542
B.2	Model 2 example	542
B.3	Model 3 example	544
Annex C (informative)	CRC generation and checking	545
C.1	Extract from FDDI	545
C.2	CRC generation	546
C.3	Transmit order of a word	547
C.4	CRC generation example for ACK_1 frame	548
Annex D (informative)	Data transfer protocols and examples	550
D.1	Frame level protocol	550

D.1.1	Class 1 frame level protocol	550
D.1.2	Class 2 frame level protocol	551
D.2	Sequence level protocol example.....	555
D.3	Class 1 frame level protocol example.....	557
D.4	Class 2 frame level protocol example.....	558
D.5	Class 3 frame level protocol example.....	559
Annex E	(informative) Connection management applications.....	561
E.1	Example cases	561
E.1.1	Case 1	561
E.1.2	Case 2	561
E.1.3	Case 3	561
E.1.4	Case 4	562
E.2	Ending sequence and Connection.....	562
E.3	Port states.....	563
E.4	State change table	565
Annex F	(informative) FC-FS service interface.....	567
F.1	FC_FS to ULP Data Services.....	567
F.1.1	FC_FS_SEQUENCE.request.....	568
F.1.1.1	Semantics of the Primitive	568
F.1.1.2	When Generated	570
F.1.1.3	Effect of Receipt	570
F.1.2	FC_FS_SEQUENCE_TAG.indication	571
F.1.2.1	Semantics of the Primitive	571
F.1.2.2	When Generated	571
F.1.2.3	Effect of Receipt	571
F.1.3	FC_FS_SEQUENCE.indication	571
F.1.3.1	Semantics of the Primitive	571
F.1.3.2	When Generated	573
F.1.3.3	Effect of Receipt	573
F.1.4	FC_FS_SEQUENCE.confirmation	573
F.1.4.1	Semantics of the Primitive	573
F.1.4.2	When Generated	574
F.1.4.3	Effect of Receipt	574
F.2	Login Services.....	574
F.2.1	Fabric Login Primitive Flows.....	575
F.2.2	Fabric Login Service Parameters	576
F.2.3	FABRIC_LOGIN.request.....	577
F.2.3.1	Semantics of the Primitive	577
F.2.3.2	When Generated	577
F.2.3.3	Effect of Receipt	577
F.2.4	FABRIC_LOGIN.indication.....	577
F.2.4.1	Semantics of the Primitive	577
F.2.4.2	When Generated	577
F.2.4.3	Effect of Receipt	577
F.2.5	FABRIC_LOGIN.confirmation.....	577

F.2.5.1	Semantics of the Primitive	577
F.2.5.2	When Generated	578
F.2.5.3	Effect of Receipt	578
F.2.6	IMPLICIT_FABRIC_LOGIN.request	578
F.2.6.1	Semantics of the Primitive	578
F.2.6.2	When Generated	579
F.2.6.3	Effect of Receipt	579
F.2.7	N_Port_Login Primitive Flows	579
F.2.8	N_Port_Login Service Parameters	580
F.2.9	N_PORT_LOGIN.request	582
F.2.9.1	Semantics of the Primitive	582
F.2.9.2	When Generated	582
F.2.9.3	Effect of Receipt	582
F.2.10	N_PORT_LOGIN.indication	582
F.2.10.1	Semantics of the Primitive	582
F.2.10.2	When Generated	582
F.2.10.3	Effect of Receipt	582
F.2.11	N_PORT_LOGIN.response	583
F.2.11.1	Semantics of the Primitive	583
F.2.11.2	When Generated	583
F.2.11.3	Effect of Receipt	583
F.2.12	N_PORT_LOGIN.confirmation	583
F.2.12.1	Semantics of the Primitive	583
F.2.12.2	When Generated	584
F.2.12.3	Effect of Receipt	584
F.2.13	IMPLICIT_N_PORT_LOGIN.request	584
F.2.13.1	Semantics of the Primitive	584
F.2.13.2	When Generated	584
F.2.13.3	Effect of Receipt	584
F.2.14	N_PORT_LOGOUT.request	584
F.2.14.1	Semantics of the Primitive	584
F.2.14.2	When Generated	585
F.2.14.3	Effect of Receipt	585
F.2.15	N_PORT_LOGOUT.indication	585
F.2.15.1	Semantics of the Primitive	585
F.2.15.2	When Generated	585
F.2.15.3	Effect of Receipt	585
F.2.16	N_PORT_LOGOUT.confirmation	585
F.2.16.1	Semantics of the Primitive	585
F.2.16.2	When Generated	586
F.2.16.3	Effect of Receipt	586
F.3	Link Services	586
F.3.1	LS_CONTROL.request	586
F.3.1.1	Semantics of the Primitive	587
F.3.1.2	When Generated	588

F.3.1.3	Effect of Receipt	588
F.3.2	LS_CONTROL.confirmation	588
F.3.2.1	Semantics of the Primitive	588
F.3.2.2	When Generated	588
F.3.2.3	Effect of Receipt	589
Annex G (informative)	Out of order characteristics	590
G.1	Out of order Data frame delivery.....	590
G.2	Out of order ACK transmission.....	591
Annex H (informative)	Link Error Status Block.....	592
H.1	Link failure counters.....	592
H.2	Invalid Transmission Word	592
H.3	Invalid CRC Count	592
Annex I (informative)	Available BB_Credit management example.....	594
Annex J (informative)	48-bit Global Identifier Format.....	597
J.1	General.....	597
J.2	Format	597
Annex K (informative)	Fibre Channel use of IEEE Global Identifiers	598
Annex L (informative)	Class 4 Examples.....	600
L.1	Setup, activate and deactivate.....	600
L.2	Activation collision and removal.....	603
Annex M (Informative)	Priority and Preemption	606
M.1	Overview.....	606
M.2	Preemption Process	606
M.3	Link Reset Protocol.....	608
Annex N (Informative)	Tutorial for IEEE Registered.....	610
N.1	Introduction	610
N.2	Formats of FC-FS identifiers.....	611
N.3	Availability of references	615
Annex O (informative)	Clock Synchronization.....	616
O.1	Introduction	616
O.2	Discussion.....	616
O.2.1	A Model of an NL_Port	616
O.2.2	Hardware-Assisted Clock Synchronization.....	617
O.2.3	A Point-to-Point System	620
O.2.3.1	Discussion of Errors.....	620
O.2.3.2	Techniques for Reducing Deterministic Errors	626
O.2.3.3	Dealing With Non-Deterministic Error	630
O.2.3.4	Dealing With Non-Monotonicity.....	630
O.2.4	Fabric Considerations.....	631
O.2.4.1	Discussion of Errors.....	632
O.2.4.2	Fixes for Fabric Errors	634
O.2.5	Loop Considerations.....	635
O.2.5.1	Discussion of Errors.....	635
O.2.5.2	Fixes for Loop Errors	637

O.3 An Example	637
Annex P Speed negotiation details (informative)	640
P.1 Basic assumptions	640
P.2 Supported configurations	641
P.3 Derivation of timing requirements	641
P.4 Ports using separate PMD components	642
P.5 Implementation notes	644

Figures

Figure 1 - Document relationship	xlv
Figure 2 - Fibre Channel structure	16
Figure 3 - Node functional configuration	17
Figure 4 - FC-FS physical model	19
Figure 5 - Point-to-point topology	21
Figure 6 - Fabric topology	22
Figure 7 - Examples of the Arbitrated Loop Topology	23
Figure 8 - Informative general Fabric model	26
Figure 9 - FC-2 building block hierarchy	29
Figure 10 - Mapping multiple blocks to a single Sequence	33
Figure 11 - Mapping subblocks to a single Sequence	34
Figure 12 - Segmentation and re-assembly	35
Figure 13 - Fabric Regions	37
Figure 14 - FC-4 Regions	38
Figure 15 - Bit designations	39
Figure 16 - Conversion example	40
Figure 17 - Delayed Code Violation example	42
Figure 18 - Loss-of-Synchronization procedure state diagram	53
Figure 19 - Receiver state diagram	54
Figure 20 - Transmitter state diagram	57
Figure 21 - FC-2 general frame format	72
Figure 22 - Frame byte order	76
Figure 23 - Frame_Header	79
Figure 24 - Optional headers order	101
Figure 25 - Network_Header	102
Figure 26 - IEEE 48-bit address format	104
Figure 27 - 32-bit IP address format	105
Figure 28 - IEEE Registered	105
Figure 29 - IEEE registered format	106
Figure 30 - Association_Header	108

Figure 31 - ACK precedence	118
Figure 32 - F_BSY TYPE field.....	122
Figure 33 - P_BSY code format.....	123
Figure 34 - Reject code format.....	126
Figure 35 - BA_RJT format	146
Figure 36 - Images/groups of related processes.....	187
Figure 37 - Image pairs.....	187
Figure 38 - Acquisition Flags.....	218
Figure 39 - Topology Discovery Flags	225
Figure 40 – Common Link-incident Descriptor	230
Figure 41 - QoS SR Sequence flow.....	242
Figure 42 - LS_RJT format.....	274
Figure 43 - N_Port Common Service Parameters - Fabric Login	310
Figure 44 - N_Port Common Service Parameters - N_Port_Login	313
Figure 45 - N_Port Class Service Parameters - Fabric Login	318
Figure 46 - N_Port Class Service Parameters - N_Port_Login	324
Figure 47 - F_Port Common Service Parameters - Fabric Login	334
Figure 48 - F_Port Class Service Parameters - Fabric Login	337
Figure 49 - Procedure to estimate end-to-end Credit.....	345
Figure 50 - Exchange - Sequence relationship.....	349
Figure 51 - Exchange origination	363
Figure 52 - Physical flow control model.....	375
Figure 53 - End-to-end flow control model.....	380
Figure 54 - Buffer-to-buffer flow control model	384
Figure 55 - Class1/SOFc1 frame flow with delivery or non-delivery to the fabric	385
Figure 56 - Class1/SOFc1 frame flow with delivery or non-delivery to a N_Port	386
Figure 57 - Buffer-to-buffer - Class2 frame flow with delivery or non-delivery to a Fabric	387
Figure 58 - Buffer-to-buffer - Class2 frame flow with delivery or non-delivery to a N_Port.....	388
Figure 59 - Buffer-to-buffer - Class3 frame flow.....	389
Figure 60 - LCR frame flow and possible responses.....	392
Figure 61 - LCR flow control model.....	393
Figure 62 - Integrated Class 2 flow control	394
Figure 63 - Frame Flow Timers	423
Figure 64 - Link recovery hierarchy	426
Figure 65 - Link Error Status Block format for RLS command.....	435
Figure 66 - Relationship of a Non-Participating to a Participating N_Port and switch	446
Figure 67 - Class 6 Multicast Routing	448
Figure 68 - Class 3 Multicast Routing	450
Figure 69 - Function Flow	453
Figure 70 - Dedicated Simplex	456

Figure 71 - Dedicated Simplex Fabric Circuit and Paths.....	456
Figure 72 - Dedicated Simplex Frame Flow.....	460
Figure 73 - Class 4 Circuit hierarchy.....	465
Figure 74 - Basic Class 4 circuit.....	466
Figure 75 - Class 4 circuits – Example.....	467
Figure 76 - Class 4 circuit - Port level state diagram.....	469
Figure 77 - Class 4 circuit – Setup.....	470
Figure 78 - Class 4 circuit – Activation.....	472
Figure 79 - Class 4 - Example of frame flow.....	473
Figure 80 - Class 4 circuit - CTI or CTR initiated	474
Figure 81 - Class 4 circuit - Fabric initiated deactivation	474
Figure 82 - Class 4 circuit - CTI or CTR initiated removal	475
Figure 83 - Class 4 circuit - Fabric initiated removal	476
Figure 84 - Class 4 circuit - Data frame responses.....	479
Figure 85 - CIVC_ID and CRVC_ID management.....	485
Figure 86 - Class 4 circuit – Management	486
Figure 87 - Buffered and unbuffered Class 1 frame and R_RDY flow.....	504
Figure 88 ELS Clock Sync Model – Fabric.....	506
Figure 89 ELS Clock Sync Model – Loop.....	508
Figure 90 - Clock Synchronization Data Distribution.....	512
Figure 91 - Synchronization Primitive Substitution for IDLE Primitives in Inter-Frame Interval.....	513
Figure 92 - Clock Synchronization Primitive Ordered Sets.....	513
Figure 93 - Physical architecture of the Speed Negotiating Link.....	517
Figure 94 - Overview of the Speed Negotiation algorithm.....	519
Figure 95 - Wait_for_signal flowchart	520
Figure 96 - Negotiate_master and Negotiate_fail flowchart.....	521
Figure 97 - Negotiate_follow flowchart.....	523
Figure 98 - Normal operation flowchart	524

Equations

Equation 1 - Average frame size related to Responder.....	244
Equation 2 - CTI maximum delay.....	244
Equation 3 - Average frame size related to Originator.....	244
Equation 4 - CTR maximum delay	244
Equation 5 - Available Credit	390
Equation 6 - Information category.....	398
Equation C.1 - FCS sequence equation 1	546
Equation C.2 - FCS sequence equation 2	546

Equation C.3 - FCS sequence equation 3	546
Equation C.4 - Received checking	546
Equation C.5 - Checking remainder	546

Tables

Table 1 - Bandwidth examples.....	21
Table 2 - Valid Data Characters	42
Table 3 - Valid Special Characters	46
Table 4 - Frame delimiters	47
Table 5 - Primitive Signals.....	48
Table 6 - Primitive Sequences	48
Table 7 - Primitive Sequence summary.....	69
Table 8 - R_CTL - Type Code summary	81
Table 9 - Information Categories	81
Table 10 - Data Descriptor Payload	82
Table 11 - Well-known address identifiers	83
Table 12 - Priority Preemption Enabled/Not Enabled.....	83
Table 13 - CS_CTL field - Class 1	84
Table 14 - CS_CTL field - Class 2	84
Table 15 - CS_CTL field - Class 3	85
Table 16 - CS_CTL field - Class 4	85
Table 17 - CS_CTL field – Class 6.....	85
Table 18 - TYPE codes - N_Port/F_Port Link Service.....	86
Table 19 - TYPE codes - Video_Data.....	87
Table 20 - TYPE codes - FC-4 (Device_Data and Link_Data)	87
Table 21 - Exchange/Sequence Control (F_CTL)	88
Table 22 - ACK_Form.....	92
Table 23 - F_CTL bit interactions on Data frames	96
Table 24 - F_CTL bit interactions on ACK, BSY or RJT	97
Table 25 - DF_CTL bit definition.....	98
Table 26 - NAA identifiers.....	103
Table 27 - Network addresses	106
Table 28 - Fibre Channel user identifiers	106
Table 29 - Name_Identifier formats	107
Table 30 - Association_Header Validity bits (Word 0, Bits 31 - 24).....	108
Table 31 - Optional header usage summary	110
Table 32 - Frame formats	112
Table 33 - Data frame delimiters	113
Table 34 - Link_Control codes.....	116

xxxv

Table 35 - Link_Control frame delimiters.....	117
Table 36 - F_BSY reason codes	122
Table 37 - P_BSY action codes	124
Table 38 - P_BSY reason codes	124
Table 39 - Reject action codes	126
Table 40 - Reject reason codes	126
Table 41 - Data frame summary	134
Table 42 - N_Port Class Service Parameters - ACK support	134
Table 43 - Basic Link Service Command Codes.....	138
Table 44 - ABTS Parameter Field Definitions	138
Table 45 - ABTS Basic Accept Payload.....	144
Table 46 - BA_RJT reason codes	146
Table 47 - BA_RJT reason code explanation	146
Table 48 - Basic Link Service Payload	148
Table 49 - LS_Command codes.....	151
Table 50 - ABTX Payload.....	155
Table 51 - ABTX Accept Payload	155
Table 52 - ADVC Payload	156
Table 53 - ADVC Accept Payload	156
Table 54 - ECHO Payload	157
Table 55 - ECHO Accept Payload	158
Table 56 - ESTC Payload.....	158
Table 57 - ESTS Payload.....	159
Table 58 - ESTS Accept Payload.....	159
Table 59 - LOGI Payload	160
Table 60 - LOGI Accept Payload	161
Table 61 - LOGO Payload.....	162
Table 62 - LOGO Accept Payload.....	163
Table 63 - RCS Payload	163
Table 64 - RCS Accept Payload	164
Table 65 - RES Payload.....	165
Table 66 - RES Accept Payload.....	165
Table 67 - RLS Payload	166
Table 68 - RLS Accept Payload	166
Table 69 - RSS Payload.....	167
Table 70 - RSS Accept Payload.....	167
Table 71 - RTV Payload.....	168
Table 72 - RTV Accept Payload.....	168
Table 73 - RRQ Payload	170
Table 74 - RRQ Accept Payload	170

xxxvi

Table 75 - RSI Payload.....	171
Table 76 - RSI Accept Payload.....	172
Table 77 - TEST Payload.....	172
Table 78 - FAN Payload.....	173
Table 79 - LINIT Payload	174
Table 80 - Initialization Function	174
Table 81 - LINIT Accept Payload	175
Table 82 - LINIT Status.....	175
Table 83 - LPC Payload.....	176
Table 84 - Port Control Function.....	176
Table 85 - LPC Accept Payload	177
Table 86 - LPC Status	177
Table 87 - LSTS Payload	178
Table 88 - LSTS Accept Payload	178
Table 89 - FC-FLA Compliance Level	179
Table 90 - Loop State	179
Table 91 - RSCN Payload	181
Table 92 - Generic Affected N_Port ID page.....	182
Table 93 - Address Format.....	182
Table 94 - Port Address Format - Affected N_Port ID page.....	182
Table 95 - Area Address Format - Affected N_Port ID page.....	183
Table 96 - Domain Address Format - Affected N_Port ID page	183
Table 97 - Fabric Address Format Affected N_Port ID page.....	183
Table 98 - RSCN Accept Payload	183
Table 99 - SCR Payload	184
Table 100 - Registration Function.....	184
Table 101 -SCR Accept Payload.....	185
Table 102 - LS_Command codes	185
Table 103 - Extended link service information units	185
Table 104 - PRLI accept response code	188
Table 105 - PRLI payload	189
Table 106 - PRLI service parameter page format.....	190
Table 107 - PRLI accept payload	191
Table 108 - PRLI accept service parameter response page format.....	192
Table 109 - PRLO payload	196
Table 110 - PRLO logout parameter page format.....	197
Table 111 - PRLO accept payload	198
Table 112 - PRLO accept logout parameter response page format.....	199
Table 113 - PRLO accept response code.....	199
Table 114 - SCN payload.....	202

Table 115 - SCN affected N_Port ID page format	202
Table 116 - TPLS payload.....	203
Table 117 - TPLS image pair ID page format.....	204
Table 118 - TPLS accept payload	205
Table 119 - TPLS response page format	205
Table 120 - TPLS accept response code	206
Table 121 - FARP-REQ Request Payload Item	207
Table 122 – FARP-REQ Match Address Code Points	207
Table 123 - Responder Action	209
Table 124 - FARP-REQ Address Fields.....	209
Table 125 - FARP-REPLY Request Payload	210
Table 126 - FARP-REPLY Address Fields	211
Table 127 - FARP-REPLY Accept Payload	212
Table 128 - RTIN Payload.....	213
Table 129 - RTIN Accept Payload.....	214
Table 130 - Connectivity Unit Header Record	215
Table 131 - Domain Record	215
Table 132 - Topology Information Record.....	216
Table 133 - Entity-node Node Identification Data Record	216
Table 134 - Attached Node Port Information Record	217
Table 135 - Attached-node Node Identification Data Record	217
Table 136 - RNID Payload	220
Table 137 - Node Identification Data Format	220
Table 138 - RNID Accept Payload	221
Table 139 - Node Identification Data Format	221
Table 140 - Common Node Identification Data.....	222
Table 141 - Topology Discovery Specific Node Identification Data	224
Table 142 - Topology Discovery Unit Type	224
Table 143 - IP Version	225
Table 144 - RLIR Payload	227
Table 145 - Link Incident Record Format.....	228
Table 146 - Common Link Incident Record Data	228
Table 147 - RLIR Accept Payload	232
Table 148 - LIRR Payload	234
Table 149 - Registration Function.....	234
Table 150 - Link Incident Record-Registration Format.....	235
Table 151 - LIRR Accept Payload	235
Table 152 - Get Alias_ID Payload.....	236
Table 153 - Get Alias_ID Accept Payload.....	237
Table 154 - Fabric Activate Alias_ID Payload	238

xxxviii

Table 155 - Activate Alias_ID Accept Payload	238
Table 156 - Fabric Deactivate Alias_ID Payload	239
Table 157 - Fabric Deactivate Alias_ID Accept Payload	239
Table 158 - N_Port Activate Alias_ID Payload	240
Table 159 - N_Port Activate Alias_ID Accept Payload	240
Table 160 - N_Port Deactivate Alias_ID Payload	241
Table 161 - N_Port Deactivate Alias_ID Accept Payload	241
Table 162 - QoS SR Payload	243
Table 163 - QoS SR Accept Payload	246
Table 164 - RVCS Payload	247
Table 165 - RVCS Accept Payload	248
Table 166 - Class 4 VC status block entry	248
Table 167 - Response summary to FDISC/PDISC	249
Table 168 - ADISC Payload	249
Table 169 - ADISC ACC Payload	250
Table 170 - TPRLO Payload	251
Table 171 - TPRLO logout parameter page format	251
Table 172 - TPRLO ACC Payload	251
Table 173 - BALLOC Payload Format	252
Table 174 - BALLOC Accept Payload format	253
Table 175 - DALLOC Payload	254
Table 176 - DALLOC Accept Payload	255
Table 177 - CSR Payload	255
Table 178 - CSR Accept Payload	256
Table 179 - CSU Payload	256
Table 180 - LCLM payload	258
Table 181 - Global capabilities	258
Table 182 - LCLM accept payload	259
Table 183 - Global capabilities	260
Table 184 - Port/Node_Name List	260
Table 185 - RPS Payload	262
Table 186 - RPS Accept Payload	263
Table 187 - L_Port Extension	263
Table 188 - RPL Payload	265
Table 189 - RPL ACC Payload	266
Table 190 - Port Number Block	266
Table 191 - RNFT Payload	267
Table 192 - RNFT ACC Payload	268
Table 193 - RNFT FC-4 Entry	268
Table 194 - Extended Link Services Summary	269

Table 195 - LS_RJT reason codes	274
Table 196 - LS_RJT reason code explanation.....	275
Table 197 - Responses to FLOGI sequence (S_ID = 0 or 0000yy) - Fabric Login.....	298
Table 198 - Responses to FLOGI sequence (S_ID = X) - Fabric Login.....	299
Table 199 - Responses to PLOGI frame - N_Port_Login (Fabric present).....	303
Table 200 - Responses to PLOGI frame - N_Port_Login (No Fabric, point to point)	305
Table 201 - FLOGI, PLOGI, or ACC payload	308
Table 202 - N_Port Common Service Parameter applicability	308
Table 203 - FC-PH/FS Version - N_Port.....	310
Table 204 - N_Port Class Service Parameter Applicability.....	318
Table 205 - Buffered Class 1 Login Service parameters	322
Table 206 – Priority/Preemption bit definition	322
Table 207 – ACK_0 Support Conditions	327
Table 208 - Stacked Connect-request support Login bit.....	339
Table 209 - Exchange Status Block	367
Table 210 - Sequence Status Block	369
Table 211 - Flow control applicability.....	375
Table 212 - Buffer participation	376
Table 213 - End-to-end flow control management	394
Table 214 - Buffer-to-buffer flow control management.....	395
Table 215 - Integrated Class 2 flow control management.....	395
Table 216 - Segmentation and reassembly rules summary	400
Table 217 - Responses to connect-request (SOFc1).....	408
Table 218 - Responses to Preemption Requests	417
Table 219 - Detailed errors and actions.....	435
Table 220 - Address Resolution.....	443
Table 221 - Bandwidth Allocation Example.....	468
Table 222 - Frame delimiter usage	478
Table 223 - N_Port – Default Class 4 service parameters.....	488
Table 224 - Camp-On vs. Stacked Connect-request	495
Table 225 - CSR ELS Payload.....	509
Table 226 - CSR ELS Accept Payload.....	510
Table 227 - CSR ELS Reject Payload	510
Table 228 - CSU ELS Payload.....	511
Table 229 - Neutral Disparity Character Values	513
Table 230 - Timing requirements	525

FORWARD (This Foreword is not part of NCITS.xxx-200x.) (Editors note – need to get the proper words to say that this is the next generation along with PI – talk to Jeff Stai)

This standard consolidates the relevant clauses (framing and signaling) from FC-PH, FC-PH-2, and FC-PH-3 and associated errata, annexes and amendments into a single encompassing document. The Physical Interfaces clauses from the above documents are in a separate standard - Fibre Channel-Physical Interfaces (FC-PI). The project may also involve the deletion or obsoleting of outdated functions and features from those documents, the inclusion of additional link services in support of new functions defined by the FC family of documents, the inclusion of improvements and clarifications to the definitions of existing services as dictated by experience with existing implementations, and other capabilities which will improve the performance of existing FC products and fit those products for new applications.

TASK GROUP T11 of Accredited Standards Committee NCITS developed this standard during 1999-2000. The standards approval process started in 2000. This document includes annexes, which are informative, and are not considered part of the standard.

Requests for interpretation, suggestions for improvement or addenda, or defect reports are welcome. They should be sent to the National Committee for Information Technology (NCITS), 1250 Eye Street, NW, Suite 200, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by NCITS. Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, NCITS had the following members:

(to be filled in by NCITS)

Technical Committee T11 on Lower Level Interfaces, which reviewed this standard, had the following members:

Kumar Malavalli, Chair

Edward Grivna, Vice-Chair

Neil Wanamaker, Secretary

(Membership P&A list to be added for final draft prior to T11 approval)

Rick Allison	Roger Ronald	Thom Hall (alt.)
Damian Bannon	Kenneth Samarra	Ken Ham (alt.)
David Baldwin	John Scheible	Brian Herzing (alt.)
Joe Breher	John Schroeder	Ken Hirata (alt.)
Paul Boulay	Pak Seto	Skip Jones (alt.)
Charles Brill	Predrag Spasic	Jack E. Keller (alt.)
Craig Carlson	Jeffrey Stai	Julie Ann Kembel (alt.)
Scott Carlson	Gary R. Stephens	Scott Kipp (alt.)
Edward S. Chang	Arlan Stone	Allen N. Kramer (alt.)
James Coomes	Craig Stuber	Michael Lamatsch (alt.)
Zane Daggett	Etsuji Sugita	Alan Land (alt.)
Jan V. Dedek	Tad Szostak	Alan Langerman (alt.)
David Deming	Rich Taborek	Robert Liu (alt.)
Mike Dudek	Luis Torres	John Lohmeyer (alt.)
Mike Fitzpatrick	Horst Truestedt	Bill Lynn (alt.)
David Ford	Schelto Van Doorn	Bill Mable (alt.)
Michael S. Foster	Fred Van Roessel	Chuck Mcknett (alt.)
Edward M. Frymoyer	Neil Wanamaker	Stephan Meyer (alt.)
Edward A. Gardner	Gary Warden	Gene Milligan (alt.)
Edward Grivna	Kevin White	Taisei Miyake (alt.)
Laura Hagins	Michael Wingard	Kevin Monroe (alt.)
Norman Harris	Jeff Young	Mike Morandi (alt.)
David E. Instone	Leonard Young	Bob Nixon (alt.)
Michael J. Karg	Carl Zeitler	Harry V. Paul (alt.)
Terence Kelleher	Greg Alvey (alt.)	Robert Pearson (alt.)
Albert F. Kelley	Tatsua Arai (alt.)	George Penokie (alt.)
Robert W. Kembel	Bob Atkinson (alt.)	Ken Peters (alt.)
Bret Ketchum	Charles Binford (alt.)	David Peterson (alt.)
Karl Kwiat	Ronald Bossard (alt.)	Bart Raudebaugh (alt.)
Tom Lindsay	Richard Bradley (alt.)	Bob Reynolds (alt.)
Kumar Malavalli	Andy Chen (alt.)	Wayne Rickard (alt.)
Roland Marbot	Terry Cobb (alt.)	Suresh Roy (alt.)
William R. Martin	Bill Collette (alt.)	Frank Samela (alt.)
Ken Moe	Jeff Connell (alt.)	Steve Sletten (alt.)
Charles Monia	Robert Cornelius (alt.)	Brad Solomon (alt.)
Dennis P. Moore	Mark DeWilde (alt.)	Steven E. Swanson (alt.)
Francis Mottini	Mike Dorsett (alt.)	Phillip Swart (alt.)
Hari Naidu	Stephen F. Dreyer (alt.)	Lloyd E. Thorsbakken (alt.)
Jay H. Neer	Bob Dugan (alt.)	Jeanne K. Truestedt (alt.)
James Nelson	Peter Dunlap (alt.)	Rich Turner (alt.)
Clifford Oberholtzer	Neil Edmunds (alt.)	Ralph Weber (alt.)
Michael O'Donnell	Jie Fan (alt.)	Steven L. Wilson (alt.)
Robert K. Pedersen	Marc Farley (alt.)	Ray Yang (alt.)
Craig Prunty	Steve Farmer (alt.)	Paula Zoller (alt.)
Curtis A. Ridgeway	Ren Franse (alt.)	
Gary S. Robinson	Ali Ghiasi (alt.)	
Elizabeth G. Rodriguez	Joe Golio (alt.)	

Task Group T11.3 on Fibre Channel Protocols, which developed and reviewed this standard, had the following members:

Jeff Stai, Chair

George Penokie, Vice-Chair

Craig Carlson, Secretary

(Membership P&A list to be added for final draft prior to T11.3 approval)

Rick Allison	Fred Van Roessel	Bob Nixon (alt.)
David Baldwin	Matt Wakeley	Charles Nogales (alt.)
Damian Bannon	Neil Wanamaker	Michael O'Donnell (alt.)
Doug Berry	Gary Warden	Harry V. Paul (alt.)
Paul Boulay	Ralph Weber	George Penokie (alt.)
Joe Breher	Jeff Young	David Peterson (alt.)
Craig Carlson	Carl Zeitler	Gary Porter (alt.)
Scott Carlson	Greg Alvey (alt.)	Michael Pugh (alt.)
James Coomes	David Allen (alt.)	Colin Schaffer (alt.)
Jan V. Dedek	Charles Binford (alt.)	Brian R. Smith (alt.)
Don Deel	Richard L. Bradley (alt.)	Brad Solomon (alt.)
David Deming	Andy Chen (alt.)	Robert N. Snively (alt.)
Neil Edmunds	Greg Chesson (alt.)	Gary R. Stephens (alt.)
Mike Fitzpatrick	Terry Cobb (alt.)	Paul Suhler (alt.)
David Ford	Bill Collette (alt.)	Terry Sutherland (alt.)
Michael S. Foster	Jeff Connell (alt.)	Lloyd E. Thorsbakken (alt.)
Kenneth J. Fredericks	Robert Cornelius (alt.)	Jeanne K. Truestedt (alt.)
Edward M. Frymoyer	Robert Dahlgren (alt.)	Naoki Watanabe (alt.)
Edward A. Gardner	Jerry D'Alessandr (alt.)	Steven L. Wilson (alt.)
Ken Hirata	Ketan Desai (alt.)	Stewart Wyatt (alt.)
Sid Jones	Mike Dorsett (alt.)	Ruben Yonitoubian (alt.)
Terence Kelleher	Peter Dunlap (alt.)	Paula Zoller (alt.)
Robert Kembel	Robert Dugan (alt.)	
Bret Ketchum	Matt Gaffney (alt.)	
Kumar Malavalli	Ali Ghiasi (alt.)	
Ken Moe	Mark Hamel (alt.)	
Charles Monia	Michael Hoard (alt.)	
James Nelson	David E. Instone (alt.)	
John J. Nutter	Larry Jones (alt.)	
Clifford Oberholtzer	Skip Jones (alt.)	
Stephen O'Neil	Jack E. Keller (alt.)	
Robert K. Pedersen	Julie Ann Kembel (alt.)	
Murali Rajagopal	Ernest Kohlwey (alt.)	
Curtis A. Ridgeway	Marjorie Krueger (alt.)	
Gary S. Robinson	Michael Lamatsch (alt.)	
Elizabeth G. Rodriguez	Alan Land (alt.)	
John Scheible	Alan Langerman (alt.)	
John Schroeder	Robert Liu (alt.)	
Pak Seto	John Lohmeyer (alt.)	
Steve Sletten	Tan Lu (alt.)	
Predrag Spasic	Bill Lynn (alt.)	
Jeffrey Stai	Paul Manka (alt.)	
Arlan Stone	William R. Martin (alt.)	
Craig Stuber	Gene Milligan (alt.)	
Rich Taborek	Mike Morandi (alt.)	
Ron Taulton	Douglas Nast (alt.)	
Horst L. Truestedt	David Neil (alt.)	

Introduction

The Fibre Channel provides a general transport vehicle for Upper Level Protocols (ULPs) such as Intelligent Peripheral Interface (IPI) and Small Computer System Interface (SCSI) command sets, the High-Performance Parallel Interface (HIPPI) data framing, IP (Internet Protocol), IEEE 802.2, and others. Proprietary and other command sets may also use and share the Fibre Channel, but such use is not defined as part of the Fibre Channel standard. Other usages such as local area network protocols and backbone configuration have been considered.

The Fibre Channel standard is organized in the following levels (see Figure 1):

- FC-0 defines the physical portions of the Fibre Channel including the fibre, connectors, and optical and electrical parameters for a variety of data rates and physical media. Coax and twisted pair versions are defined for limited distance applications. FC-0 provides the point-to-point physical portion of the Fibre Channel. A variety of physical media is supported to address variations in cable plants. FC-0 enhancements define higher data rates in the 2 to 4 Gigabaud range, and new variants for the copper media.
- FC-1 defines the transmission protocol that includes the serial encoding, decoding, and error control.
- FC-2 defines the signaling protocol that includes the frame structure and byte sequences. FC-2 enhancements include alias address definition, protocols for Hunt Group, Multicast, Dedicated Simplex, Camp-On, Stacked Connect-request, Fractional Bandwidth allocation management (Class 4), Data Compression, E_D_TOV resolution enhancements, Process error policy extensions (especially for Class 3), Time distribution, Clock synchronization, new Extended Link Service commands, and others.
- FC-3 defines a set of services that are common across multiple ports of a node.
- FC-4 is the highest level in the Fibre Channel standards set. It defines the mapping, between the lower levels of the Fibre Channel and the IPI and SCSI command sets, the HIPPI data framing, IP, and other Upper Level Protocols (ULPs).

Of these levels, FC-1 and FC-2 are integrated into this document. The Fibre Channel protocol provides a range of implementation possibilities extending from minimum cost to maximum performance. The transmission medium is isolated from the control protocol so that each implementation may use a technology best suited to the environment of use.

(Editor's note – the following paragraph needs to reflect Figure 1 better)

Figure 1 shows the relationship of this document (the highlighted rectangle) with other Fibre Channel standards and draft proposed standards. FC-FS combines the framing and signaling content of FC-PH, its Amendments 1 and 2 and the enhanced functions specified by FC-PH-2 and FC-PH-3. FC-PI combines the FC-0 content of these documents. FC-FG and FC-SW specify fabric requirements. FC-AL specifies the arbitrated loop topology. (FC-AL-2 specifies enhancements to FC-AL.) FC-GS specifies Fibre Channel Generic Services requirements. FC-SB, FC-FP, FC-LE, IPI-3 Disk revision, IPI-3 Tape revision; and FCP are FC-4 standards.

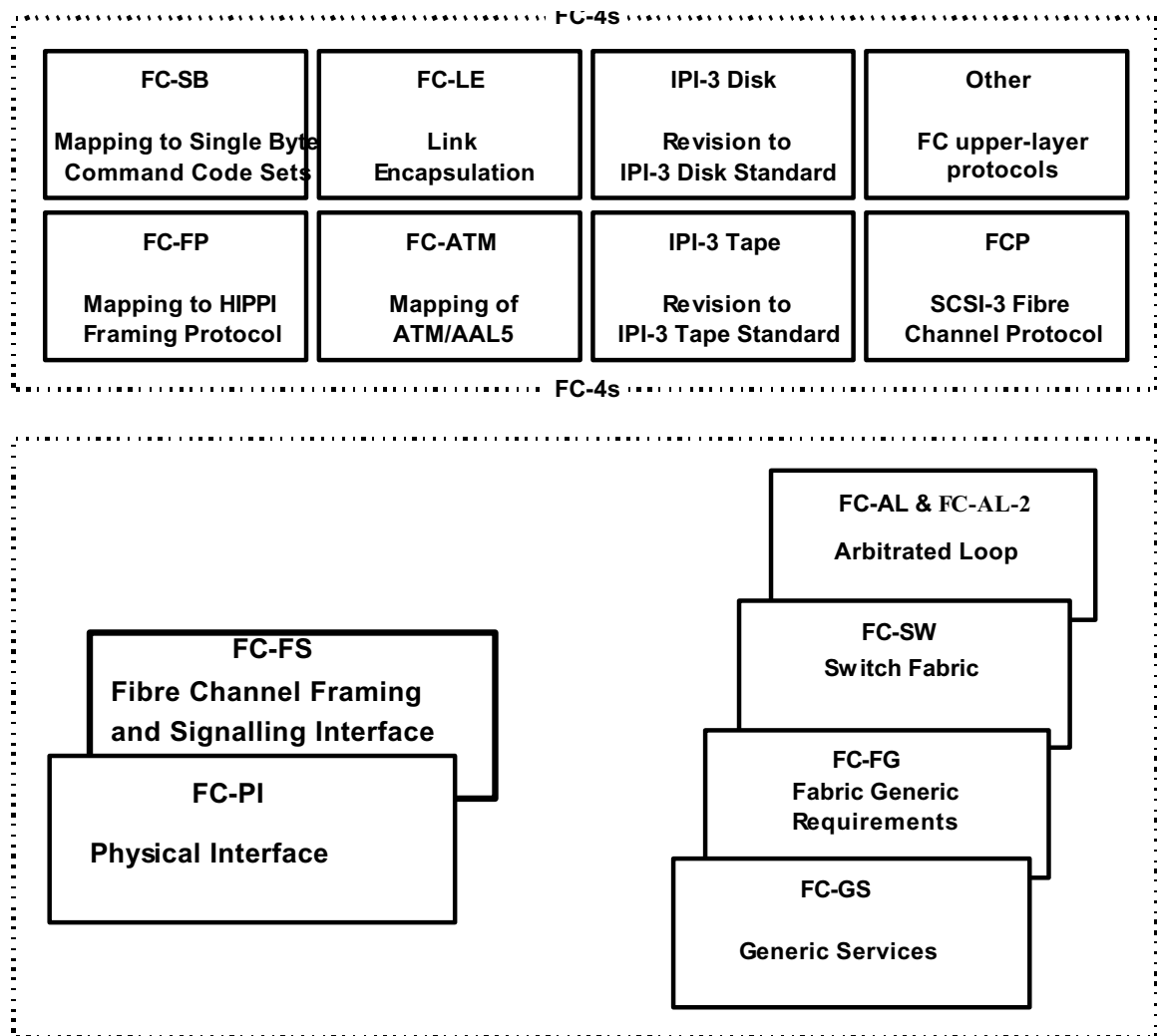


Figure 1 - Document relationship

1 Scope

This international standards describes the physical and signaling interface of a high performance serial link for support of the Upper Level Protocols (ULPs) associated with HIPPI, IPI, SCSI, IP and others.

FC-FS is the combination of FC-PH, its amendments 1 and 2, FC-PH-2 and FC-PH-3 standards but does not replace those standards. The standard also involves the deletion or obsoleting of outdated functions and features from those documents, the inclusion of additional link services in support of new functions defined by the FC family of documents, the inclusion of improvements and clarifications to the definitions of existing services as dictated by experience with existing implementations, and other capabilities which will improve the performance of existing FC products and fit those products for new applications.

2 Normative references

The references listed in 2.1 and 2.2 contain provisions that, through reference in this text, constitute provisions of this document. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this unofficial compilation of standards are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

Copies of the following documents can be obtained from ANSI: Approved ANSI standards, approved and draft international and regional standards (ISO, IEC, CEN/CENELEC, ITUT), and approved foreign standards (including BSI, JIS, and DIN). For further information, contact ANSI Customer Service Department at 212-642-4900 (phone), 212-302-1286 (fax) or via the World Wide Web at <http://www.ansi.org>.

2.1 Approved references

ANSI X3.230-1994, *Fibre Channel Physical and Signaling Interface (FC-PH)*

ANSI X3.230-1994/AM 1:1996, *Fibre Channel Physical and Signaling Interface (FC-PH) Amendment 1*

ANSI X3.230-1994/AM 2:1999, *Fibre Channel Physical and Signaling Interface (FC-PH) Amendment 2*

ANSI X3.297-1996, *Fibre Channel Physical and Signaling Protocol - 2 (FC-PH-2)*

ANSI X3.303-1997, *Fibre Channel Physical and Signaling Protocol - 3 (FC-PH-3)*

ANSI X3.289-1996, *Fibre Channel-Fabric Requirements (FC-FG)*

ANSI X3.272-1996, *Fibre Channel-Arbitrated Loop (FC-AL)*

ANSI NCITS 332-1999, *Fibre Channel-Arbitrated Loop-2 (FC-AL-2)*

ANSI X3.271-1996, *Fibre Channel-SBCC Mapping Protocol (FC-SB)*

ANSI X3.287-1996, *Fibre Channel-Link Encapsulation (FC-LE)*

ANSI NCITS.321-1998, *Fibre Channel-Switch Topology (FC-SW)*

ANSI X3.269:1996, *Fibre Channel Protocol (FC-FCP)*

ISO/IEC 9314-2:1989, *Fiber Distributed Data Interface - Media Access Control (FDDI-MAC)*

ANSI NCITS TR-20:1998 *Fibre Channel - Fabric Loop Attach (FC-FLA)*

ANSI X3.254-1994 *Fibre Channel-Mapping – to HIPPI-FC (FC-FP)*

IPv6

FC-EP (ref in 15.5.27.1)

2.2 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the documents, or regarding availability, contact the relevant standards body or other organization as indicated.¹

ANSI NCITS Project 1235-D, *Fibre Channel Physical Interfaces (FC-PI)*

ANSI NCITS Project 1305-D, *Fibre Channel-Switch Topology (FC-SW-2)*

ANSI NCITS Project 1356-D, *Fibre Channel-Generic Services (FC-GS-3)*

ANSI NCITS Project 1357-D, *Fibre Channel-Single Byte Command Code Sets-2 Mapping Protocol (FC-SB-2)*

ANSI NCITS Project 1144-D, *Fibre Channel Protocol-2 (FC-FCP-2)*

ANSI/IEEE Std 802-1990, Local and Metropolitan Area Networks: Overview and Architecture (formerly known as IEEE Std 802.1A, Project 802: Local and Metropolitan Area Networks Standard -- Overview and Architecture)

ANSI NCITS Project 1232-D, *Fibre Channel-Slotted Loop (FC-SL)*

¹

For information about obtaining copies of these documents or more information on the current status of these documents, contact NCITS at: <http://www.x3.org> or 202-626-5738.

3 Definitions and conventions

For the purposes of this unofficial compilation of standards, the following definitions, conventions, abbreviations, acronyms, and symbols apply.

3.1 Definitions

3.1.1 Acknowledged Class: Any class of service in which a transfer is acknowledged.

3.1.2 Active: The state of Sequence Initiator until all the Data frames for the Sequence have been transmitted. The state of Sequence Recipient until all the Data frames for the Sequence have been received. (see 18.6.1).

3.1.3 Active Virtual Circuit Credit limit: The maximum number of VC_Credits available for a Virtual Circuit in the Active state. It represents the maximum number of VC_Credits held by a N_Port on a given Virtual Circuit when the Circuit is in the active state.

3.1.4 address identifier: An address value used to identify source (S_ID) or destination (D_ID) of a frame.

3.1.5 Alias: Alias is a group address recognized by a N_Port if the N_Port has registered as a member of the group, with the Alias Server (see 26.1).

3.1.6 alias address identifier (alias): One or more address identifiers that may be recognized by a N_Port in addition to its N_Port Identifier. An alias address identifier is Fabric unique and may be common to multiple N_Ports.

3.1.7 Alias-Token: A 12 byte field to indicate the type of Alias (such as Multicast or Hunt Group) and certain properties associated with the Alias (such as FC-PH TYPE, Node_Name for the Common Controlling Entity).

3.1.8 Arbitrated Loop topology: A configuration that allows multiple ports to be connected serially (see FC-AL-2).

3.1.9 Available BB_Credit: The difference between the number of R_RDYs received and the number of frames sent in a single circuit.

3.1.10 baud: The encoded bit rate per second.

3.1.11 BB_buffer: The buffer associated with buffer-to-buffer flow control.

3.1.12 Beginning Running Disparity: The Running Disparity present at a transmitter when Encoding of the Special Code associated with an Ordered Set is initiated, or at a receiver when Decoding of the Special Character associated with an Ordered Set is initiated.

3.1.13 block: An upper level construct of application data related to a single Information Category and transferred within a single Sequence.

3.1.14 buffer: A logical construct which holds the contents of a single frame (i.e., contents of a frame $\leq$ one buffer) (see Credit and clause 11).

3.1.15 byte: An eight-bit entity with its least significant bit denoted as bit 0 and most significant bit as bit 7. The most significant bit is shown on the left side in FC-FS, unless specifically indicated otherwise.

3.1.16 character: Any Transmission Character associated by FC-1 transmission code with a FC-2 data byte or special code. Transmission characters are generated and interpreted only by FC-1.

3.1.17 circuit: A bi-directional path within the Fabric.

3.1.18 Classes of service: Different types of services provided by the Fabric and used by the communicating N_Ports (see 4.7 and 16).

3.1.19 Class 1 service: A service that establishes a dedicated connection between communicating N_Ports (see 4.7.1 and 16.1).

3.1.20 Class 2 service: A service that multiplexes frames at frame boundaries to or from one or more N_Ports with acknowledgement provided. (see 4.7.2 and 16.2).

3.1.21 Class 3 service: A service that multiplexes frames at frame boundaries to or from one or more N_Ports without acknowledgement (see 4.7.3 and 16.3).

3.1.22 Class 4 bi-directional circuits: A pair of unidirectional Virtual Circuits between two communicating N_Ports.

3.1.23 Class 4 end-to-end credit limit: The maximum amount of end-to-end credit available for a Virtual Circuit. It represents the maximum number of Class 4 end-to-end credits held by a N_Port on a given Virtual Circuit.

3.1.24 Class 4 service: A service that establishes Virtual Circuits to provide fractional bandwidth service between communicating N_Ports. The service multiplexes frames at frame boundaries to or from one or more N_Ports with acknowledgment provided. (see 16.5 and clause 28).

3.1.25 Class 4 Circuit Initiator: The N_Port that initiates the setup of a Class 4 Circuit.

3.1.26 Class 4 Circuit Recipient: The N_Port that accepts a Class 4 Circuit with the Originator N_Port.

3.1.27 Class 6 service: A service, which allows a N_Port to establish simultaneous Dedicated Connections with multiple N_Ports (see 4.7.516.6 and clause 25).

3.1.28 code bit: The smallest time period used by FC-0 for transmission on the media.

3.1.29 code violation: An error condition that occurs when a received Transmission Character cannot be decoded to a Valid Data Byte or Special Code using the validity checking rules specified by the Transmission Code.

3.1.30 comma: The seven-bit sequence 0011111 or 1100000 in an encoded stream.

3.1.31 comma character: A Special Character containing a comma.

3.1.32 concatenation: A logical operation that "joins together" strings of data and is represented with the symbol "||". Two or more fields are concatenated to provide a reference of uniqueness (e.g., S_ID||X_ID).

3.1.33 Connection: See Dedicated Connection.

3.1.34 Connection Initiator: The source N_Port that initiates a Class 1 Connection with a destination N_Port through a connect-request and also receives a valid response from the destination N_Port to complete the Connection establishment.

3.1.35 Connection-oriented frames: Frames sent in either a Class 1 Dedicated Connection or a Class 4 Circuit.

3.1.36 Connection Recipient: The destination N_Port that receives a Class 1 connect-request from the Connection Initiator and accepts establishment of the Connection by transmitting a valid response.

3.1.37 Connectionless buffers: Receive buffers participating in connectionless service and capable of receiving Connectionless frames (see clause 20).

3.1.38 Connectionless frames: Frames participating in Connectionless service (i.e., Class 1 frames with SOFc1, Class 2, and Class 3 frames referred to individually or collectively) (see clause 20).

3.1.39 Connectionless service: Communication between two N_Ports performed without a Dedicated Connection.

3.1.40 Continuously Increasing Relative Offset: the relationship specified between Relative Offset values contained in frame (n) and frame (n+1) of an Information Category within a single Sequence. Continuously Increasing Relative Offset (RO_i) for a given Information Category i is specified by the following:

$$RO_i(n+1) = RO_i(n) + \text{Length of Payload}_i(n)$$

where n is ≥ 0 and represents the consecutive frame count of frames for a given Information Category within a single Sequence.

RO_i(0) = Initial Relative Offset (IRO_i) for the Information Category i (see Relative Offset and Random Relative Offset).

3.1.41 Credit: The maximum number of receive buffers allocated to a transmitting N_Port or F_Port. It represents the maximum number of outstanding frames which can be transmitted by that N_Port or F_Port without causing a buffer overrun condition at the receiver (see 20.3).

3.1.42 current running disparity: The Running Disparity present at a transmitter when Encoding of a Valid Data Byte or Special Code is initiated, or at a receiver when Decoding of a Transmission Character is initiated.

3.1.43 Data Character: Any Transmission Character associated by the Transmission Code with a Valid Data Byte.

3.1.44 Data frame: An FC-4Device_Data frame, an FC-4 Video_Data frame, or a Link Data frame. (See clause 14.).

3.1.45 Decoding: Validity checking of received Transmission Characters and generation of Valid Data Bytes and Special Codes from those characters.

3.1.46 Dedicated Connection: A communicating circuit guaranteed and retained by the Fabric for two given N_Ports.

3.1.47 Dedicated Duplex: A synonym for Class 1 Dedicated Connection (see clause 27).

3.1.48 Dedicated Simplex: A unidirectional Class 1 connection with ACKs transmitted in Class 2 (see clause 27).

3.1.49 delimiter: An Ordered Set used to indicate a frame boundary.

3.1.50 Destination_Identifier (D_ID): The address identifier used to indicate the targeted destination of the transmitted frame.

3.1.51 destination N_Port: The N_Port to which a frame is targeted.

3.1.52 discard policy: An error handling policy where a N_Port is able to discard Data frames received following detection of a missing frame in a Sequence (see 23.6.1.1).

3.1.53 disconnection: The process of removing a Dedicated Connection between two N_Ports.

3.1.54 disparity: The difference between the number of ones and zeros in a Transmission Character.

3.1.55 EE_buffer: The buffer associated with end-to-end flow control.

3.1.56 encoding: Generation of Transmission Characters from Valid Data Bytes and Special Codes.

3.1.57 Exchange: The basic mechanism which transfers information consisting of one or more related non-concurrent Sequences which may flow in the same or opposite directions. An Exchange may span multiple Class 1 Dedicated Connections. The Exchange is identified by an Originator Exchange_ID (OX_ID) and a Responder Exchange_Identifier (RX_ID) (see clause 18).

3.1.58 Exchange_Identifier (X_ID): A generic reference to OX_ID and RX_ID (see Exchange).

3.1.59 Exchange Status Block: A logical construct that contains the state of an Exchange. An Originator N_Port has an Originator Exchange Status Block and the Responder N_Port has a Responder Exchange Status Block for each concurrently active Exchange (see 4.11.4.3 and 18.8.1).

3.1.60 Exclusive Connection: A Class 1 Dedicated Connection without Intermix (see Dedicated Connection and 16.4).

3.1.61 F_Port: The Link_Control_Facility within the Fabric that attaches to a N_Port through a link. A F_Port is addressable by the N_Port attached to it, with a common well-known address identifier (hex 'FF FF FE') (see 12.3, local F_Port, and remote F_Port).

3.1.62 Fabric: The entity which interconnects various N_Ports attached to it and is capable of routing frames by using only the D_ID information in a FC-2 frame header.

3.1.63 Fabric_Name: A Name_Identifier associated with a Fabric.

3.1.64 FC-4 Region: A set of N_Ports connected either point-to-point or to a common Fabric, such that any N_Port in the set can successfully complete the N_Port_Login procedure with all other N_Ports in the set and successfully maintain an Exchange for a particular FC-4.

3.1.65 FC device: An entity that contains the FC protocol functions and which has one or more of the connectors defined in this document.

3.1.66 fibre: A general term used to cover all transmission media specified in FC-FS (see clause 4).

3.1.67 Fibre Channel Name: A Name_Identifier that is Fibre Channel unique.

3.1.68 FL_Port: A F_Port that contains Arbitrated Loop functions associated with Arbitrated Loop topology (see FC-AL-2).

3.1.69 fractional bandwidth: A portion of the total bandwidth available on a path.

3.1.70 frame: An indivisible unit of information used by FC-2.

3.1.71 F_Port_Name: A Name_Identifier associated with a F_Port.

3.1.72 Frame Content: The information contained in a frame between its Start-of-Frame and End-of-Frame delimiters, excluding the delimiters.

3.1.73 Hunt Group: A set of N_Ports with a common alias address identifier managed by a single node or common controlling entity. However, FC-FS does not presently specify how a Hunt Group can be realized.

3.1.74 Identified Company: The company that has purchased and is identified by a particular value of the IEEE company_id.

3.1.75 Idle Word (Idle): An ordered set of four transmission characters (see clause 5) which are normally transmitted between frames. The Idle Word is also referred to as an Idle (see 5.4).

3.1.76 ignored: A field that is not interpreted by the receiver.

3.1.77 Internal connector: A connector, whose purpose is to carry the FC signals within an enclosure (may be shielded or unshielded).

3.1.78 Internal FC Device: A FC device whose FC device connector is contained within an enclosure.

3.1.79 infinite buffer: A terminology to indicate that at FC-2 level, the amount of buffer available at the Sequence Recipient is unlimited. The ULP chooses the amount of buffer per Sequence based on its MTU (maximum transfer unit).

3.1.80 Information Category: A frame header field indicating the category to which the frame payload belongs (e.g., Solicited Data, Unsolicited Data, Solicited Control and Unsolicited Control).

3.1.81 information transfer: Transfer of frames whose Payload has meaning to the cooperating FC-4s.

3.1.82 Information Unit: An organized collection of data specified by FC-4 to be transferred as a single Sequence by FC-2.

3.1.83 initialization: For FC-1 level the period beginning with power on and continuing until the transmitter and receiver of that level become Operational.

3.1.84 Initial Relative Offset: A Relative Offset value specified at the sending end by an upper level for a given block or subblock and used by the sending FC-2 in the first frame of that block or subblock (see subblock, block, and Relative Offset). Initial Relative Offset value may be zero or non-zero.

3.1.85 Intermix: A service which interleaves Class 2 and Class 3 frames on an established Class 1 Connection (see 4.8 and 16.4).

3.1.86 link:

1. Two unidirectional fibers transmitting in opposite directions and their associated transmitters and receivers.
2. The full-duplex FC-0 level association between FC-1 entities in directly attached Ports (see Port).

3.1.87 Link_Control_Facility: A link hardware facility that attaches to an end of a link and manages transmission and reception of data. It is contained within each N_Port and F_Port.

3.1.88 local F_Port: The F_Port to which a N_Port is directly attached by a link (see remote F_Port).

3.1.89 loopback: A mode of FC-1 operation in which the information passed to the FC-1 transmitter for transmission is shunted directly to the FC-1 receiver, overriding any signal detected by the receiver on its attached fibre.

3.1.90 L_Port: A N_Port or F_Port that contains Arbitrated Loop functions associated with Arbitrated Loop topology (see FC-AL-2).

3.1.91 mandatory: A function that is required to be supported by a compliant implementation of FC-FS.

3.1.92 meaningful: A control field or bit shall be applicable and shall be interpreted by the receiver, wherever it is specified as meaningful. Wherever it is specified as "not meaningful", it shall be ignored (see valid).

3.1.93 Name_Identifier: A 64 bit identifier, with a 60 bit value preceded with a 4 bit Network_Address_Authority Identifier, used to identify entities in Fibre Channel such as N_Port, Node, F_Port, or Fabric (see Table 28 and Table 29).

3.1.94 Network_Address_Authority (NAA): An organization such as CCITT or IEEE that administers network addresses (see 13.3).

3.1.95 Network_Address_Authority (NAA) identifier: A four-bit identifier defined in FC-FS to indicate a Network_Address_Authority (NAA) (see 13.3).

3.1.96 NL_Port: A N_Port that contains Arbitrated Loop functions associated with Arbitrated Loop topology (see FC-AL-2).

3.1.97 Node: A collection of one or more N_Ports controlled by a level above FC-2.

3.1.98 Node_Name: A Name_Identifier associated with a Node.

3.1.99 Non-repeating ordered set: An Ordered Set which, when issued by FC-2 to FC-1 for transmission, is to be transmitted once.

3.1.100 Not Operational: A receiver or transmitter that is not capable of receiving or transmitting an encoded bit stream respectively, based on the rules defined by FC-FS for error control. For example, FC-1 is Not Operational during Initialization.

3.1.101 N_Port: A hardware entity that includes a Link_Control_Facility. It may act as an Originator, a Responder, or both.

3.1.102 N_Port Identifier: A Fabric unique address identifier by which a N_Port is uniquely known. The identifier may be assigned by the Fabric during the initialization procedure. The identifier may also be assigned by other procedures not defined in FC-FS. The identifier is used in the S_ID and D_ID fields of a frame.

3.1.103 N_Port_Name: A Name_Identifier associated with a N_Port.

3.1.104 Open: The period of time starting when a Sequence (an Exchange) is initiated until that Sequence (the Exchange) is normally or abnormally terminated (see 18.6.1).

3.1.105 Operation: A construct that may be used by a level above FC-2 and is associated with one or more Exchanges.

3.1.106 Operational: The state of a receiver or transmitter that is capable of receiving or transmitting an encoded bit stream, respectively, based on the rules defined by FC-FS for error control. Those receivers capable of accepting signals from transmitters requiring laser safety procedures are not considered Operational after power on until a signal of a duration longer than that associated with laser safety procedures is present at the fibre attached to the receiver.

3.1.107 optional: Characteristics that are not required by FC-FS. However, if any optional characteristic is implemented, it shall be implemented as defined in FC-FS.

3.1.108 ordered set: A Transmission Word composed of a Special Character in its first (left-most) position and Data Characters in its remaining positions. An Ordered Set is represented by the combination of Special Codes and data bytes which, when encoded, result in the generation of the Transmission Characters specified for the Ordered Set.

3.1.109 Originator: The logical function associated with a N_Port responsible for originating an Exchange.

3.1.110 Originator Exchange Identifier (OX_ID): An identifier assigned by an Originator to identify an Exchange and meaningful only to the Originator (see Responder Exchange Identifier).

3.1.111 Payload: Contents of the Data Field of a frame, excluding Optional Headers and fill bytes, if present (see Figure 22, and clauses 11, 13, and 14.2).

3.1.112 Port: A generic reference to a N_Port or F_Port.

3.1.113 Port_Name: A Name_Identifier associated with a Port.

3.1.114 Primitive Sequence: An ordered set transmitted repeatedly and continuously until a specified response is received (see 10.4 and 18.8.1).

3.1.115 Primitive Signal: An ordered set designated to have a special meaning such as an Idle or Receiver_Ready (R_RDY) (see 10.3).

3.1.116 Process_Associator: A value used in the Association_Header to identify a process or a group of processes within a Node. Process_Associator is the mechanism by which a process is addressed by another communicating process. Process_Associator is a generic reference to Originator Process_Associator and Responder Process_Associator.

3.1.117 process policy: An error handling policy where a N_Port is able to continue processing Data frames received following detection of one or more missing frames in a Sequence (see 23.6.1.1).

3.1.118 Profile: An interoperability specification that provides implementation guidelines for systems manufacturers, system integrators, component manufacturers, and users seeking to design and select interoperable Fibre Channel peripherals, hosts, and components. A Profile specifies particular settings for various Fibre Channel physical, link-level, and upper-level protocol options to enhance interoperability.

3.1.119 Random Relative Offset: The relationship specified between Relative Offset values contained in frame (n) and frame (n+1) of an Information Category within a single Sequence. For a given Information Category i within a single Sequence, Random Relative Offset (RO_i) value for a frame (n+1) is unrelated to that of the previous frame (n). (see Initial Relative Offset and Continuously Increasing Relative Offset).

3.1.120 receiver: The portion of a Link_Control_Facility dedicated to receiving an encoded bit stream from a fibre, converting this bit stream into Transmission Characters, and Decoding these characters using the rules specified by FC-FS.

3.1.121 Relative Offset (Offset): The displacement, expressed in bytes, of the first byte of a Payload related to a upper level defined origin for a given Information Category (see Continuously Increasing and Random Relative Offset and 4.12.3).

3.1.122 Relative Offset space: A virtual address space defined by the sending upper level for a single Information Category. The address space starts from zero, representing the upper level defined-origin, and extends to its highest value.

3.1.123 remote F_Port: The F_Port to which the other communicating N_Port is directly attached (see local F_Port).

3.1.124 repeating ordered set: An Ordered Set which, when issued by FC-2 to FC-1 for transmission, is to be repetitively transmitted until a subsequent transmission request is issued by FC-2.

3.1.125 reserved: A field that is filled with binary zeros by the source N_Port and is ignored by the destination N_Port. Each bit in the reserved field is denoted by "r".

NOTE - Future enhancements to FC-FS may define usages for these reserved fields. The reserved fields should not be checked or interpreted. Any violation of this guideline may result in loss of upward compatibility with future implementations that comply with future enhancements to FC-FS.

3.1.126 Responder: The logical function in a N_Port responsible for supporting the Exchange initiated by the Originator in another N_Port.

3.1.127 Responder Exchange_Identifier (RX_ID): An identifier assigned by a Responder to identify an Exchange and meaningful only to the Responder.

3.1.128 run length: Number of consecutive identical bits in the transmitted signal e.g., the pattern 0011111010 has a run length of five (5).

3.1.129 running disparity: A binary parameter indicating the cumulative disparity (positive or negative) of all previously issued Transmission Characters.

3.1.130 Sequence: A set of one or more Data frames with a common Sequence_ID (SEQ_ID), transmitted unidirectionally from one N_Port to another N_Port with a corresponding response, if applicable, transmitted in response to each Data frame (see clause 18).

3.1.131 Sequence_ID (SEQ_ID): An identifier used to identify a Sequence.

3.1.132 Sequence Initiator: The N_Port that initiates a Sequence and transmits Data frames to the destination N_Port (see clause 18).

3.1.133 Sequence Recipient: The N_Port which receives Data frames from the Sequence Initiator and, if applicable, transmits responses (Link_Control frames) to the Sequence Initiator (see clause 18).

3.1.134 Sequence Status Block: A logical construct that tracks the state of a Sequence. Both the Sequence Initiator and the Sequence Recipient have a Sequence Status Block for each concurrently active Sequence (see 4.11.3.2 and 18.8.2).

3.1.135 Solicited Control: One of the Information Categories indicated in the frame header (see 12.2).

3.1.136 Solicited Data: One of the Information Categories indicated in the frame header (see 12.2).

3.1.137 Source_Identifier (S_ID): The address identifier used to indicate the source Port of the transmitted frame.

3.1.138 source N_Port: The N_Port from which a frame is transmitted.

3.1.139 Special Character: Any Transmission Character considered valid by the Transmission Code but not equated to a Valid Data Byte. Special Characters are provided by the Transmission Code for use in denoting special functions.

3.1.140 Special Code: A code which, when encoded using the rules specified by the Transmission Code, results in a Special Character. Special Codes are typically associated with control signals related to protocol management (e.g., K28.5).

3.1.141 streamed Sequence: A new sequence initiated by a Sequence Initiator in any Class of service for an Exchange while it already has Sequences Open for that Exchange (see 18.6.1).

3.1.142 subblock: An upper level construct which contains partial application data for a single Information Category (see block). A collection of subblocks for a given Information Category may be specified for transfer within a single Sequence.

3.1.143 synchronization: Receiver identification of a Transmission-Word boundary.

3.1.144 Transmission Character: Any encoded character (valid or invalid) transmitted across a physical interface specified by FC-0. Valid Transmission Characters are specified by the Transmission Code and include Data and Special Characters.

3.1.145 Transmission Code: A means of encoding data to enhance its Transmission Characteristics. The Transmission Code specified by FC-FS is byte-oriented, with (1) Valid Data Bytes and (2) Special Codes encoded into 10-bit Transmission Characters.

3.1.146 Transmission Word: A string of four contiguous Transmission Characters occurring on boundaries that are zero modulo 4 from a previously received or transmitted Special Character.

3.1.147 transmitter: The portion of a Link_Control_Facility dedicated to converting Valid Data Bytes and Special Codes into Transmission Characters using the rules specified by the Transmission Code, converting these Transmission Characters into a bit stream, and transmitting this bit stream onto the transmission medium (optical or electrical).

3.1.148 transceiver: A transmitter and receiver combined in one package.

3.1.149 TxRx connection: The complete signal path between a transmitter in one FC device and a receiver in another FC device.

3.1.150 Uncategorized Information Category: One of the Information Categories indicated in the frame header (see 12.2).

3.1.151 unrecognized ordered set: A Transmission Word containing a K28.5 in its first (left-most) position but not defined to have meaning by FC-FS.

3.1.152 Unsolicited Control: One of the Information Categories indicated in the frame header (see 12.2).

3.1.153 Unsolicited Data: One of the Information Categories indicated in the frame header (see 12.2).

3.1.154 upper level: A level above FC-2.

3.1.155 Upper Level Protocol (ULP): The protocol user of FC-4 (see clause 4).

3.1.156 valid: A validity control bit indicates if a field is valid, in which case, the value in the field shall be treated as valid. If a validity control bit indicates that a field is invalid, the value in the field shall be treated as invalid (see meaningful).

3.1.157 Valid Data Byte: A string of eight contiguous bits within FC-1 that represents a value with 0 to 255, inclusive.

3.1.158 Valid frame: A frame received with a valid Start_of_Frame (SOF), a valid End_of_Frame (EOF), valid Data Characters, and proper Cyclic Redundancy Check (CRC) of the Frame Header and Data Field (see 11).

3.1.159 vendor unique: Functions, code values, and bits not defined by FC-FS and set aside for private usage between parties using FC-FS. Caution: Different implementations of FC-FS may assign different meanings to these functions, code values, and bits.

3.1.160 Virtual Circuit (VC): A unidirectional path between two communicating N_Ports that permits Class 4 service to be used. Two Virtual Circuits are required to form a Class 4 Circuit.

3.1.161 Virtual Circuit Credit (VC_Credit): The number of receiver buffers allocated to a Virtual Circuit by a F_Port. It represents the maximum number of frames that a N_Port may transmit without causing a buffer overrun condition at the F_Port receiver.

3.1.162 Virtual Circuit Credit limit: The maximum number of VC_Credits available for a Virtual Circuit. It represents the maximum number of VC_Credits held by a N_Port on a given Virtual Circuit.

3.1.163 Virtual Circuit Identifier (VC_ID): An identifier associated with either the Originator (OVC_ID) or Responder (RVC_ID) for a Virtual Circuit.

3.1.164 virtual path: A fixed route through a Fabric in support of a Virtual Circuit.

3.1.165 well-known addresses: A set of address identifiers defined in FC-FS to access global server functions such as a name server (see 12.3).

3.1.166 word: A string of four contiguous bytes occurring on boundaries that are zero modulo 4 from a specified reference.

3.1.167 Worldwide_Name: A Name_Identifier which is worldwide unique, and represented by a 64 bit unsigned binary value.

3.2 Editorial conventions

In this unofficial compilation of standards, a number of conditions, mechanisms, sequences, parameters, events, states, or similar terms are printed with the first letter of each word in upper-case and the rest lower-case (e.g., Exchange, Class). Any lower case uses of these words have the normal technical English meanings.

Numbered items in this unofficial compilation of standards do not represent any priority. Any priority is explicitly indicated.

In case of any conflict between figures, tables, and text, the text takes precedence. Exceptions to this convention are indicated in the appropriate sections.

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 sections.

The term "shall" is used to indicate a mandatory rule. If such a rule is not followed, the results are unpredictable unless indicated otherwise.

The fields or control bits that are not applicable shall be reset to zero.

If a field or a control bit in a frame is specified as not meaningful, the entity that receives the frame shall not check that field or control bit.

In the various ladder diagrams that show a sequence of events, the vertical axis (i.e., up and down the page) shows time from top to bottom.

The ISO convention of numbering is used (i.e., the thousands and higher multiples are separated by a space). A comma is used as the decimal point. A comparison of the American and ISO conventions is shown below:

ISO	American
0,6	0.6
1000	1,000
1 323 462,9	1,323,462.9

Hexadecimal notation

Hexadecimal notation is used to represent fields. For example, a three byte D_ID field containing a binary value of 11111111 11111111 11111010 is denoted by hex 'FF FF FA'.

3.3 Abbreviations, acronyms, and symbols

Abbreviations, acronyms and symbols applicable to this unofficial compilation of standards are listed. Definitions of several of these items are included in 3.1. The index at the back of the document is an aid to help locate these terms in the body of the document.

3.3.1 Data rate abbreviations

Abbreviation	Data Rate
133 Mb/s	132,821 5 Mb/s
266 Mb/s	265,625 Mb/s
531 Mb/s	531,25 Mb/s
1063 Mb/s	1062,5 Mb/s
2125 Mb/s	2125 Mb/s
4250 Mb/s	4250 Mb/s

The exact data rates are used in the tables and the abbreviated form is used in text. Note that 1,062 5 Gbaud is the preferred ISO method and is used instead of 1062,5 Mbaud where it makes sense to do so.

3.3.2 Synonyms

Listed terms are synonyms:

ACC ≡ LS_ACC

3.3.3 Acronyms and other abbreviations

ABTS Abort Sequence

ABTX	Abort Exchange
ACC	Accept
ACK	Acknowledgement
ADVC	Advise Credit
alias	alias address identifier
BB_Credit	Buffer-to-buffer Credit
BB_Credit_CNT	Buffer-to-buffer Credit_Count
BB_SCs	Buffer-to-buffer State Change (SOF)
BB_SCr	Buffer-to-buffer State Change (R_RDY)
BB_CS_N	Buffer-to-buffer State Change Number
BSY	busy
CD	compact disk
Class 1/SOFc1	Class 1 frame with a SOFc1 delimiter
COR	Camp-On Request
Credit_CNT	Credit_Count
CTI	Circuit Initiator
CTR	Circuit Recipient
DF_CTL	Data_Field Control
D_ID	Destination_Identifier
ECL	Emitter Coupled Logic
E_D_TOV	Error_Detect_Timeout value
EE_Credit	End-to-end Credit
EE_Credit_CNT	End-to-end Credit_Count
EOF	End-of-Frame
ESB	Exchange Status Block
ESTC	Estimate Credit
ESTS	Establish Streaming
FACT	Fabric Activate Alias
F_BSY	Fabric_Port_Busy
F_BSY(Df)	F_BSY response to a Data frame
F_BSY(LC)	F_BSY response to any Link_Control except P_BSY
FC	Fibre Channel
FCS	Frame Check Sequence (see C.1)
F_CTL	Frame Control
FDACT	Fabric Deactivate Alias
FOTP	fiber optic test procedure
F_Port	Fabric_Port
F_RJT	Fabric_Port_Reject
FT-0	frame type 0
FT-1	frame type 1
GAID	Get Alias_ID
hex	hexadecimal notation
HG	Hunt Group
Hz	Hertz = 1 cycle per second
Idle	Idle Word
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
JNA	Join Alias Group
LA_RJT	Link Application Reject
LCF	Link_Control_Facility
LCR	Link Credit Reset
LOGI	Login
LOGO	Logout
LOL	loss of light

LR	Link Reset Primitive Sequence
LRR	Link Reset Response Primitive Sequence
LS_ACC	Link Service Accept
m	Metre
MAS	Master of link
Mb	Megabit
MB	MegaByte
MBd	MegaBaud
ms	millisecond
μs	microsecond
MM	multimode
NA	not applicable
NACT	N_Port Activate Alias
NAA	Network_Address_Authority
NDACT	N_Port Deactivate Alias
nm	nanometer
NOP	No Operation
NOS	Not_Operational Primitive Sequence
N_Port	Node_Port
ns	nanosecond
NTP	Network Time Protocol
OESB	Originator Exchange Status Block
OFC	open fibre control
Offset	Relative Offset
OLS	Offline Primitive Sequence
OX_ID	Originator Exchange_ID
OVC_ID	Originator VC_ID
P_AS	Process Associator
P_BSY	N_Port_Busy
PDISC	Discover N_Port Service Parameters
PLOGI	N_Port_Login
P_RJT	N_Port_Reject
PRLI	Process Login
PRLO	Process Logout
QoSF	Quality of Service Facilitator
QoSR	Quality of Service Request
R_A_TOV	Resource_Allocation_Timeout value
RCS	Read Connection Status
R_CTL	Routing Control
REQCS	Request Clock Synchronization
RES	Read Exchange Status Block
RESB	Responder Exchange Status Block
RFI	Radio Frequency Interference
RIN	relative intensity noise
RIIN	reflection induced intensity noise
RJT	reject
RMC	Remove Connection
RMS	root mean square
RNC	Report Node Capability
RO	Relative Offset
R_RDY	Receiver_Ready
RSS	Read Sequence Status Block
R_T_TOV	Receiver_Transmitter_Timeout value
RTV	Read Timeout Value
RVCS	Read Virtual Circuit Status

RVC_ID	Responder VC_ID
Rx	receiver
RX_ID	Responder Exchange_Identifier
SBCCS	Single Byte Command Code Sets
S	second
SEQ_CNT	Sequence_Count
SCN	State Change Notification
SCR	State Change Registration
SEQ_ID	Sequence_ID
S_ID	Source_Identifier
SISB	Sequence Initiator Status Block
S_Length	Security_Length
SM	single mode
SOF	Start-of-Frame
SRSB	Sequence Recipient Status Block
SSB	Sequence Status Block
S_Type	Security_Type
STP	shielded twisted pair
SW	short wavelength
TP	twisted pair
TPLS	Test Process Login Status
Tx	transmitter
TYPE	Data structure type
UI	unit interval = 1 bit period
ULP	Upper Level Protocol
VC	Virtual Circuit
VC_Credit	Virtual Circuit Credit
VC_ID	Virtual Circuit Identifier
VC_RDY	Virtual Circuit Ready
X_ID	Exchange_Identifier
WWN	Worldwide_Name

3.3.4 Symbols

Unless indicated otherwise, the following symbols have the listed meaning.

@	at
	concatenation
Ω	ohms
μ	micro (e.g., μm = micrometer)
λ	Wavelength
$\Leftarrow$	assignment
$\Uparrow$	increment
$\equiv$	synonym

4 Structure and Concepts

This clause provides an overview of the structure, concepts and mechanisms used in FC-FS and is intended for informational purposes only.

The Fibre Channel (FC) is logically a bi-directional, point-to-point, serial data channel, structured for high performance capability. Physically, the Fibre Channel can be an interconnection of multiple communication points, called N_Ports, interconnected by a switching network, called a Fabric, or a point-to-point link. Fibre is a general term used to cover all physical media types supported by the Fibre Channel - such as optical fibre, twisted pair, and coaxial cable.

Fibre Channel is structured as a set of hierarchical functions as illustrated in Figure 2. Fibre Channel Framing and Signaling interface (FC-FS) consists of related functions FC-0, FC-1, and FC-2. Each of these functions is described as a level. FC-FS does not restrict implementations to specific interfaces between these levels.

(editor's note – figure 2 – need to see if all those listed as FC4 and ULP are not withdrawn)

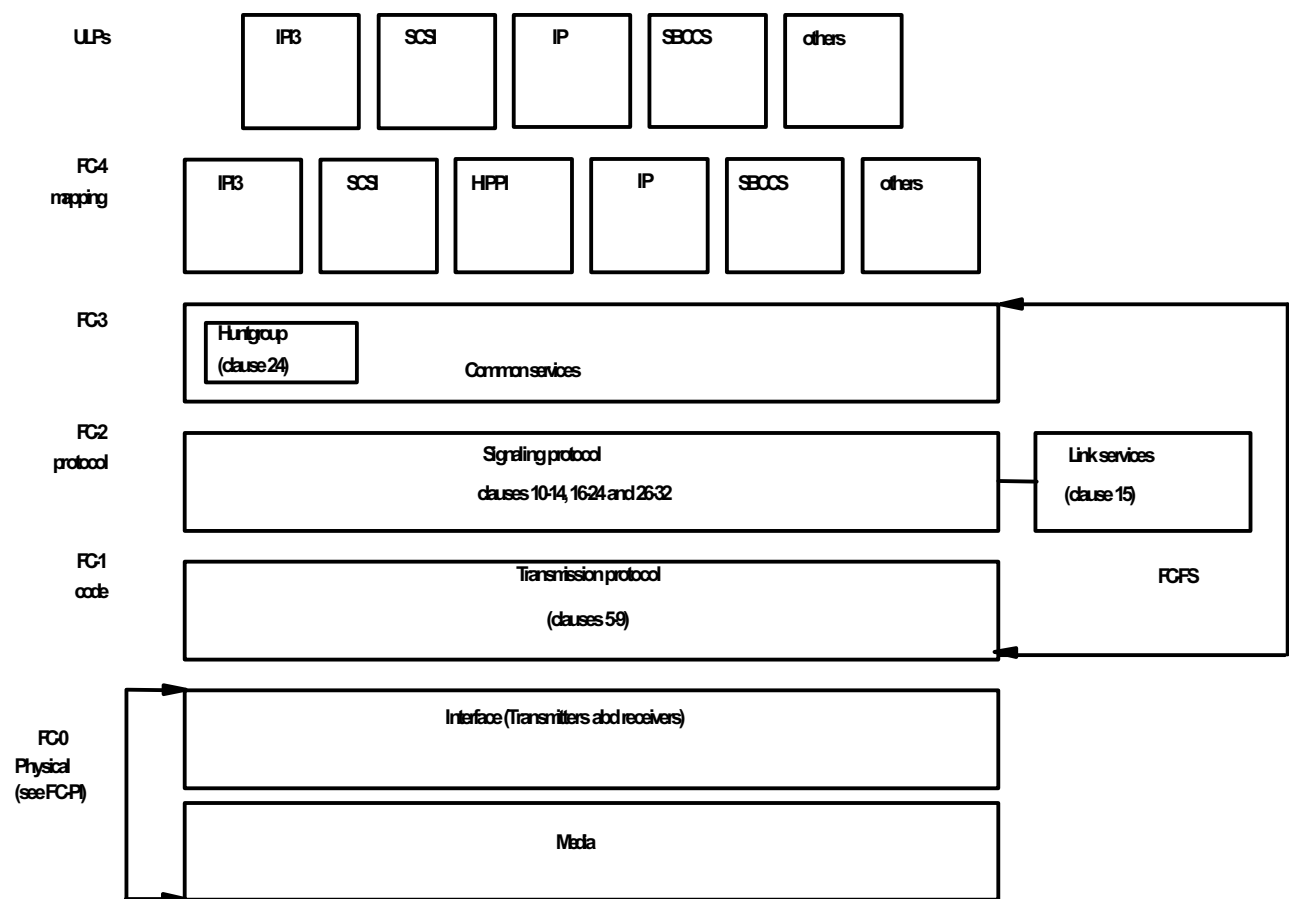


Figure 2 - Fibre Channel structure

The Physical interface (FC-0) consists of transmission media, transmitters, and receivers and their interfaces. The Physical interface specifies a variety of media, and associated drivers and receivers capable of operating at various speeds. See FC-P1 for the details. The transmission code (FC-1) used is 8B/10B and is specified in clause 5.

The signaling protocol (FC-2) specifies the rules, and provides mechanisms needed to transfer blocks of data end to end. FC-2 defines a suite of functions and facilities available for use by a FC-4. This suite of functions and facilities may be more than what a given FC-4 may require. Based on their needs, a FC-4 may choose only a subset of these functions and facilities.

FC-3 provides a set of services that are common across multiple N_Ports of a FC node. An Upper Level Protocol mapping to FC-FS constitutes a FC-4 that is the highest level in the Fibre Channel structure.

Fibre Channel provides a method for supporting a number of Upper Level Protocols (ULPs). The Link Services represent a mandatory function required by FC-2 (see clause 15).

A Fibre Channel Node is functionally configured as illustrated in Figure 3. A Node may support one or more N_Ports and one or more FC-4s. Each N_Port contains FC-0, FC-1 and FC-2 functions. FC-3 optionally provides the common services to multiple N_Ports and FC-4s.

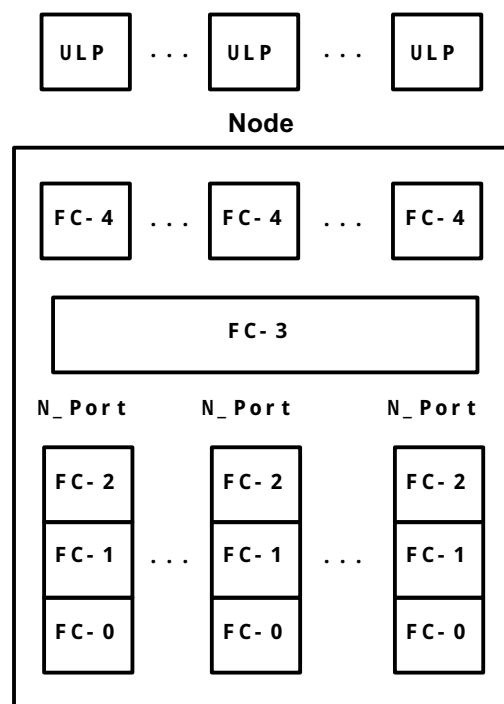


Figure 3 - Node functional configuration

4.1 FC-1 general description

The Fibre Channel transmits information using an adaptive 8B/10B code to bound the maximum run length of the code, maintain DC-balance, and provide word alignment. The encoding process results in the generation of Transmission Characters. Two types of Transmission Characters (Data and Special) are defined.

Certain combinations of Transmission Characters, referred to as Ordered Sets, are designated by FC-FS to have special meaning. Ordered Sets are used to identify frame boundaries, transmit primitive function requests, and maintain proper link transmission characteristics during periods of inactivity.

Transmitter and Receiver behavior is specified via a set of states and their interrelationships. These states are divided into Operational and Not Operational classes. Error monitoring capabilities and special operational modes are also defined for Operational Receivers and Transmitters.

4.2 FC-2 general description

The FC-2 level serves as the transport mechanism of the Fibre Channel. The transported data is transparent to FC-2 and visible to FC-3 and above.

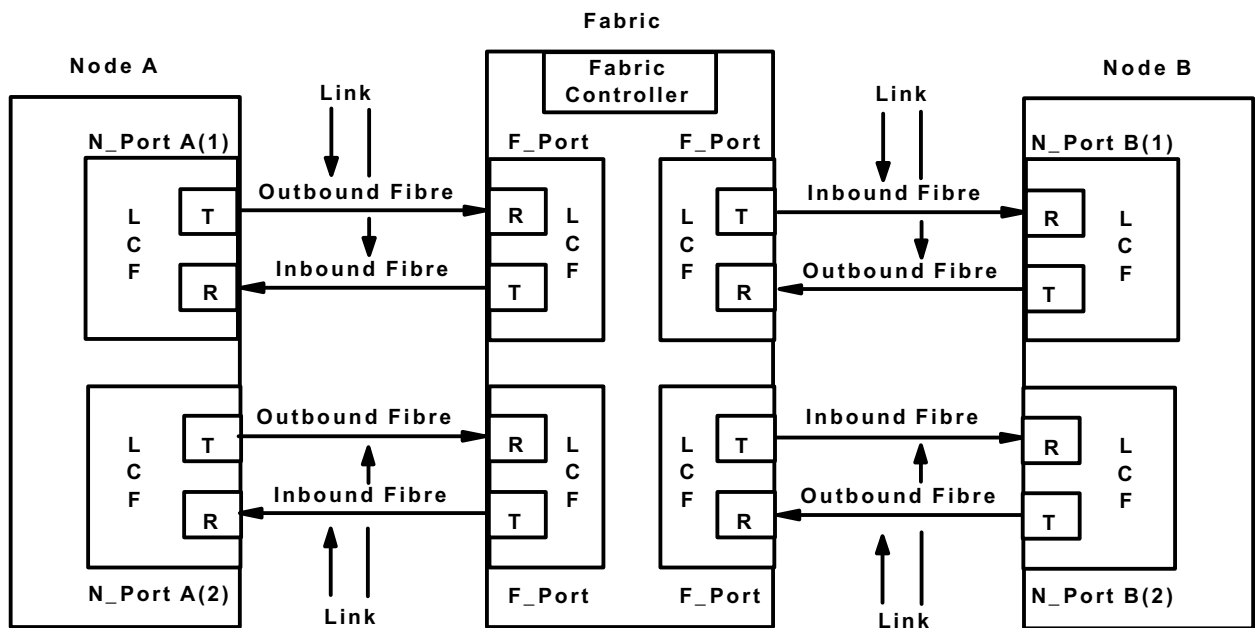
The following concepts are defined and described:

- Node and N_Port and their identifiers
- communication models.
- Topologies based on the presence or absence of a Fabric.
- Classes of service provided by the Fabric and the N_Ports.
- General Fabric model
- Building Blocks and their hierarchy
- Sequence and Sequence Identifier
- Exchange and Exchange Identifier
- Sequence Status Block model
- Exchange Status Block model
- Segmentation and reassembly

A Request-Reply protocol example is used in several clauses for illustration but other protocols are equally valid.

4.3 FC-FS physical model

Figure 4 depicts the FC-FS physical model and illustrates the FC-FS physical structure and components. The Fibre Channel (FC) physically consists of a minimum of two Nodes, each with a minimum of one N_Port interconnected by a pair of fibers - one outbound and the other inbound at each N_Port. This pair of unidirectional fibers transmitting in opposite directions with their associated transmitters and receivers is referred to in FC-FS as a link. The link is used by the interconnected N_Ports to perform data communication.



Legend:
 T - transmitter
 R - receiver
 Fibre - Any medium supported by Fibre Channel
 LCF - Link_Control_Facility
 A(1), A(2) - N_Port addresses within Node A
 B(1), B(2) - N_Port addresses within Node B

Figure 4 - FC-FS physical model

Physical equipment such as a processor, controller, or terminal can be interconnected to other physical equipment through these links. Attached physical equipment supports one or more Nodes and each Node contains one or more N_Ports, with each N_Port containing a transmitter and a receiver.

The physical model shown in Figure 4 is inherently capable of simultaneous, symmetrical bi-directional flow. A Fabric may be present between the N_Ports and some Fabrics may not support this type of flow. From the perspective of a given N_Port, for instance A(1) or B(1), its transmitter sends Data frames on the outbound fibre and its receiver receives the responses on the inbound fibre.

In Class 1 service, a N_Port logically performs a point-to-point communication with another N_Port at any given time. This statement is true regardless of the presence of Fabric. However, multiple N_Ports in a Node can simultaneously perform data transfers in parallel with single or multiple N_Ports contained in one or more Nodes (see 4.7.1 and clause 16 for Class of service description).

In Class 2 and Class 3 service, a N_Port may multiplex frames to or demultiplex frames from multiple N_Ports (see 4.7.2, 4.7.3, and clause 16 for Class of service description).

This structure provides the attached equipment flexible mechanisms to perform simultaneous data transfers in parallel, to or from, single or multiple equipments.

4.3.1 Node and N_Port identifiers

Each N_Port has an identifier to identify itself during communication. Referring to Figure 4, Node A consists of two N_Ports represented as A(1) and A(2) and Node B consists of two N_Ports represented as B(1) and B(2). Thus, single or multiple N_Ports contained within a Node are represented by a set of N_Port Identifiers. Each N_Port is represented as an element of the set, for example,

Node B as {B(1), B(2), ..., B(n)}.

4.3.2 Link_Control_Facility (LCF)

The Link_Control_Facility is a hardware facility that attaches to each end of a link and manages transmission and reception of data. It is contained within each N_Port or F_Port and includes a transmitter and a receiver. Each LCF provides the logical interface to the Originator and/or the Responder, as applicable to a communication model.

4.4 Communication models

A N_Port transmits Data frames as a result of information transfer requests from its associated FC-4s at its end, and transmits Link_Control responses to Data frames that it receives from other N_Ports. A N_Port receives Link_Control responses to Data frames that it transmitted, and it receives Data frames from other N_Ports.

A N_Port may operate according to the communication models described below:

- Model 1 operation is defined as a N_Port transferring Data frames in one direction only, with Link_Control frames flowing in the opposite direction.
- Model 2 operation is defined as a N_Port simultaneously transmitting and receiving Data frames, with Link_Control frames flowing in both directions as well.
- Model 3 operation is defined as a N_Port both transmitting and receiving data, but not simultaneously. Data frames and Link_Control frames flow in both directions, but the flow is limited, possibly by a Fabric, to a single direction at a time.

4.4.1 Hunt Group

A Hunt Group is an enhancement to the FC-FS structure of N_Ports. In FC-FS N_Ports are addressed using Port specific N_Port Identifiers. The Hunt Group is a collection of N_Ports that may be addressed by either a group identifier (Hunt Group ID) or Port specific N_Port Identifier. Grouping N_Ports and addressing them as a group makes it possible to increase bandwidth and decrease latency significantly.

Refer to clause 24 for details on Hunt Group.

4.4.2 Dedicated Simplex

Dedicated Simplex approach allows a N_Port to have two separate dedicated simplex connections with two other N_Ports, by using inbound and outbound fibers separately, thereby increasing the bandwidth.

Refer to clause 27 for details on Dedicated Simplex.

4.4.3 Fractional Bandwidth

Class 4 provides the capability to allocate and guarantee bandwidth for multiple N_Ports communicating with a single N_Port. Class 4 also provides a way to support service quality through quality of service parameters that may be specified by a N_Port.

Refer to clause 28 for details on Class 4.

4.5 Bandwidth

Effective transfer rate is a function of physical variants such as the communication model, Payload size, Fibre speed, and overhead specified by FC-FS. Examples of bandwidth for a selected set of variants are shown in Table 1.

Table 1 - Bandwidth examples			
Communication model	Speed	Payload size (bytes)	Bandwidth (MB/sec)
1	1,062 5 Gbaud	2 048	103,22 ¹⁾
2	1,062 5 Gbaud	2 048	100,37 ²⁾
3	1,062 5 Gbaud	2 048	100,37 ³⁾
Notes:			
1) Aggregate for both Fibers			
2) Per direction			
3) Aggregate for all communicating N_Ports			

4.6 Topology

Topologies are defined, based on the capability and the presence or absence of Fabric between the N_Ports, and they are:

- Point-to-point topology
- Fabric topology
- Arbitrated Loop topology

FC-FS protocols are topology independent. Attributes of a Fabric may restrict operation to certain communication models.

4.6.1 Point-to-point topology

The topology shown in Figure 5, in which communication between N_Ports occurs without the use of Fabric, is defined as point-to point.

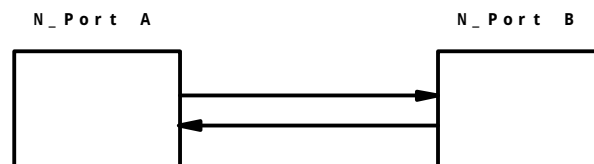


Figure 5 - Point-to-point topology

4.6.2 Fabric topology

This topology uses the Destination_Identifier (D_ID) embedded in the Frame_Header to route the frame through a Fabric to the desired destination N_Port. Figure 6 illustrates multiple N_Ports interconnected by a Fabric.

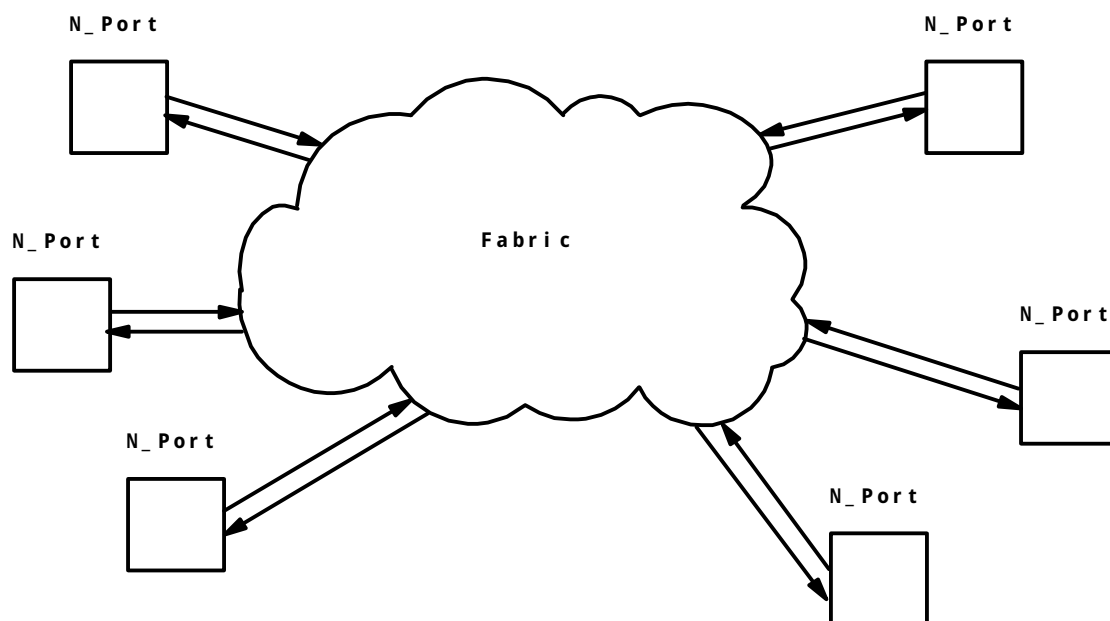


Figure 6 - Fabric topology

4.6.3 Arbitrated Loop topology

The Arbitrated Loop topology permits three or more L_Ports to communicate without the use of a Fabric, as in Fabric topology. The arbitrated loop supports a maximum of one point-to-point circuit at a time. When two L_Ports are communicating, the arbitrated loop topology supports simultaneous, symmetrical bi-directional flow.

Figure 7 illustrates two independent Arbitrated Loop configurations each with multiple L_Ports attached. Each line in the figure between L_Ports represents a single fibre. The first configuration shows an Arbitrated Loop composed only of NL_Ports. The second configuration shows an Arbitrated Loop composed of one FL_Port and three NL_Ports.

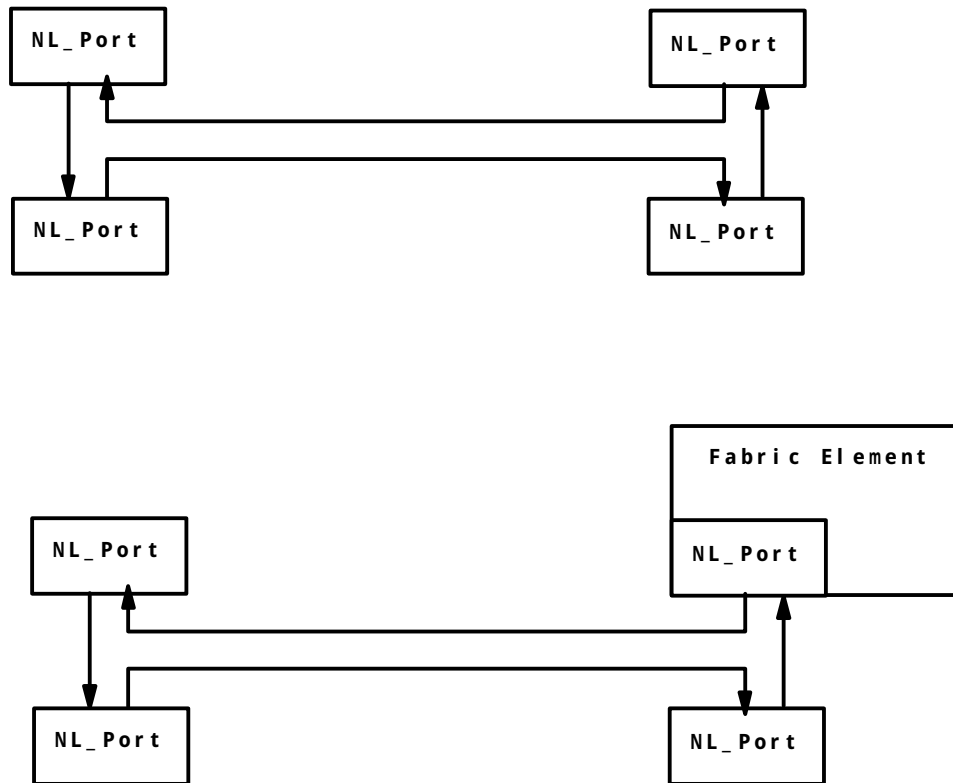


Figure 7 - Examples of the Arbitrated Loop Topology

4.7 Classes of service

Five Classes of service are defined. These Classes of service are distinguished primarily by the methodology with which the communication circuit is allocated and retained between the communicating N_Ports and the level of delivery integrity required for an application. Classes 1, 2, and 3 are topology independent. If the Fabric is not present, the service is provided as a special case of point-to-point. Classes 4 and 6 require functionality outside of the participating N_Ports. Fabrics and N_Ports are not required to support all Classes of service.

4.7.1 Class 1 service - Dedicated Connection

Class 1 is a service that establishes Dedicated Connections. Once established, a Dedicated Connection is retained and guaranteed by the Fabric. This service guarantees maximum bandwidth available between two N_Ports across the established connection.

In Class 1, frames are delivered to the destination N_Port by the Fabric in the same order as they are transmitted by the source N_Port. If the Fabric is not present, this service becomes a special case of point-to-point.

4.7.2 Class 2 service - Multiplex

Class 2 is a connectionless service with the Fabric multiplexing frames at frame boundaries. If a Fabric is not present, this service becomes a special case of point-to-point.

The transmitter transmits Class 2 Data frames in a sequential order within a given Sequence. However the Fabric may not guarantee the order of delivery and frames may be delivered out of order.

The Fabric guarantees notification of delivery or failure to deliver in the absence of link errors. In case of link errors, notification is not guaranteed since the Source_Identifier (S_ID) may not be error free.

4.7.3 Class 3 service - Datagram

Class 3 is a connectionless service with the Fabric multiplexing frames at frame boundaries, if a Fabric is present. If a Fabric is not present, this service becomes a special case of point-to-point.

Class 3 supports only unacknowledged delivery where the destination N_Port does not send any confirmation Link_Control frames on receipt of valid Data frames. Any acknowledgement of Class 3 service is up to and determined by ULPs.

The transmitter transmits Class 3 Data frames in sequential order within a given Sequence. However, the Fabric may not guarantee the order of delivery and frames may be delivered out of order.

In Class 3, the Fabric is expected to make a best effort to deliver the frame to the intended destination and does not issue a busy or reject frame to the source N_Port if unable to deliver the frame.

When a Class 3 frame is received in error, any error recovery or notification is performed at the ULP level.

4.7.4 Class 4 service –

4.7.5 Class 6 – Multicast Connection

Class 6 allows a N_Port to establish simultaneous Dedicated Connections with multiple N_Ports. Once established, these Dedicated Connections are retained and guaranteed by the Fabric. This service guarantees maximum bandwidth available from the source N_Port to each destination N_Port across the established connections. The effective bandwidth of any Class 6 connection is multiplied by the number of destination N_Ports.

Class 6 is inherently uni-directional. Data flows only from the source N_Port to the destination N_Ports. All destination N_Ports respond with the appropriate Link_Response frames to a Multicast Server. The Multicast Server collects the Link_Response frames and returns a single Link_Response frame to the source N_Port. Frames are delivered to the destination N_Ports by the Fabric in the same order as they are transmitted by the source N_Port. This service requires a Fabric.

4.8 Intermix

Intermix is an option of Class 1 service which allows interleaving of Class 2 and Class 3 frames during an established Class 1 Connection between two N_Ports. During a Class 1 Connection, a N_Port capable of Intermix may transmit and receive Class 2 and Class 3 frames interleaved with Class 1 frames. Class 2 and Class 3 frames may be interchanged between the two connected N_Ports or between either of the connected N_Ports and other N_Ports. Support for the Intermix option of Class 1 service is indicated during Login (see clause 17). A N_Port that supports Intermix is capable of both transmitting and receiving intermixed frames.

In a point-to-point topology, both interconnected N_Ports are required to support Intermix if Intermix is to be used. In the presence of a Fabric, both the N_Port and the F_Port to which it

is attached are required to support Intermix if Intermix is to be used. Fabric support for Intermix requires that the full Class 1 bandwidth during a Dedicated Connection be maintained. Intermix permits the potentially unused Class 1 bandwidth to be used for transmission of Class 2 and Class 3 frames.

4.9 General Fabric model

The primary function of the Fabric is to receive the frames from a source N_Port and route the frames to the destination N_Port whose address identifier is specified in the frames. Each N_Port is physically attached through a link to the Fabric. FC-2 specifies the protocol between the Fabric and the attached N_Ports. A Fabric is characterized by a single address space in which every N_Port has a unique N_Port identifier.

A Fabric specifies the Classes of service it supports in its Service Parameters (see 17.7). Fabrics are allowed to provide the Classes of service through equivalent mechanisms and/or functions. Refer to the Fabric requirements (FC-FG) document for these equivalent functions provided by some Fabrics.

Figure 8 illustrates the general Fabric model. The model is conceptual and may provide the following major functions:

- Bi-directional Fabric Ports (F_Ports)
- Receive buffer
- Connection Sub-Fabric
- Connectionless Sub-Fabric
- Receive buffer queue management

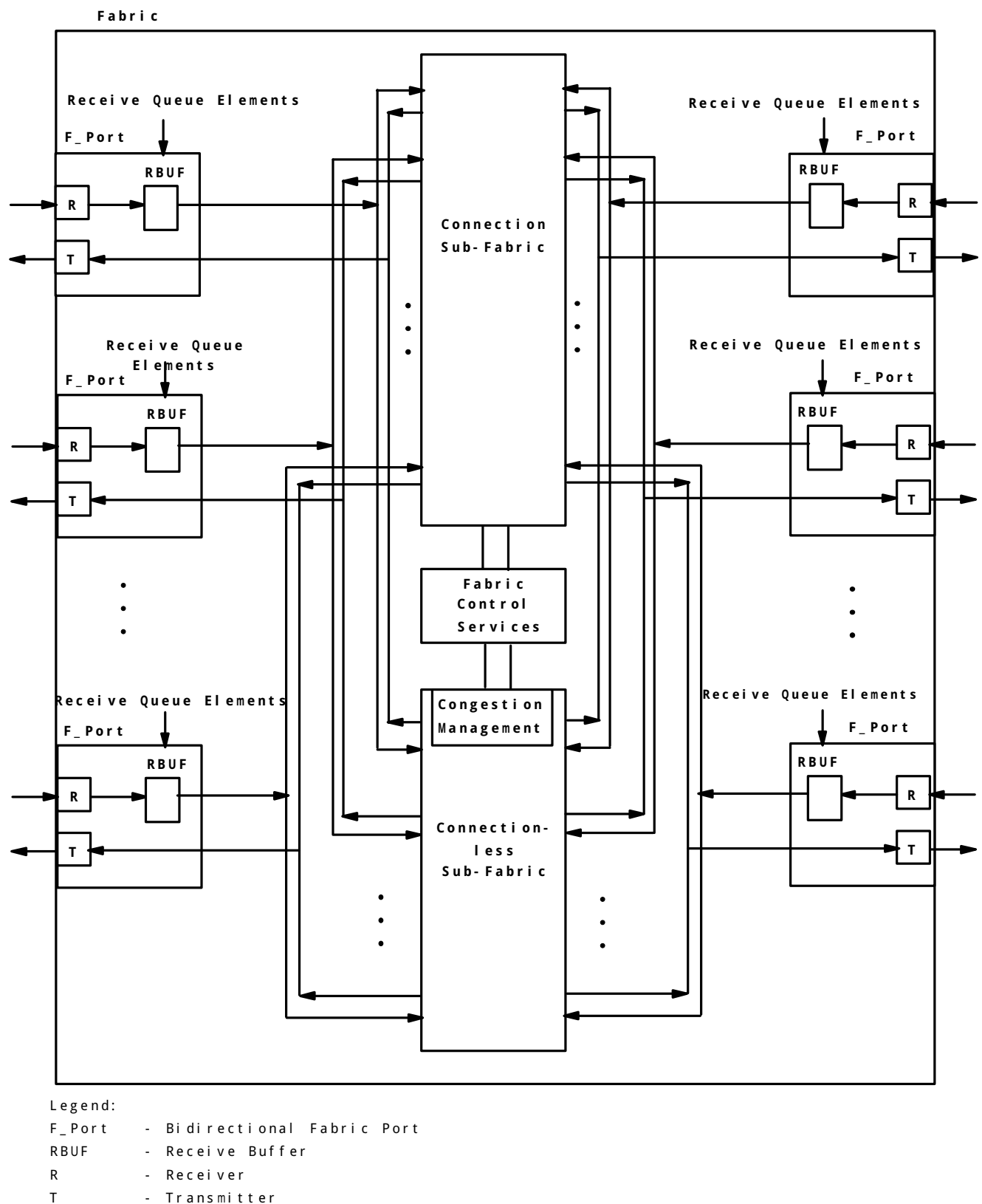


Figure 8 - Informative general Fabric model

4.9.1 Fabric Ports (F_Ports)

The Fabric model contains two or more F_Ports. Each F_Port can be attached to a N_Port through a link. Each F_Port is bi-directional and supports one or more communication models.

The receiving F_Port responds to the sending N_Port according to the FC-2 protocol. The Fabric may or may not verify the validity of the frame as it passes through the Fabric (see 11.6.2 and 11.8.1).

A F_Port may or may not contain receive buffers for the incoming frames. The maximum frame size that the Fabric can handle for Class 2 service, Class 3 service, and for the frame which requests Class 1 service is determined during Login. One of the Fabric Service Parameters indicates the maximum frame size for the entire Fabric.

The Fabric routes the frame to the F_Port directly attached to the destination N_Port based on the N_Port identifier (D_ID) embedded in the frame. The address translation and the routing mechanisms within the Fabric are transparent to N_Ports and are not specified in FC-FS.

4.9.2 Connection based Sub-Fabric

The Connection based Sub-Fabric provides Dedicated Connections between F_Ports and the N_Ports attached to these F_Ports. Such Dedicated Connections are bi-directional and support full transmission rate concurrently in both directions. A Dedicated Connection is retained until a removal request is received from one of the communicating N_Ports or an exception condition occurs which causes the Fabric to remove the Connection.

On receiving a request from a N_Port, the Fabric establishes a Dedicated Connection to the destination N_Port through the Connection Sub-Fabric. The mechanisms used by the Fabric to establish the connection are transparent to FC-2.

The Sub-Fabric is not involved in flow control, which is managed end-to-end by the N_Ports.

If the Fabric is unable to establish a Dedicated Connection, it returns a busy or reject frame with a reason code. Once the Dedicated Connection is established, the frames between the communicating N_Ports are routed through the same circuit and all responses are issued by N_Ports.

A Class 1 or Class 6 N_Port and the Fabric may support stacked connect-requests. This stacked connect-requests support allows a N_Port to request multiple Dedicated Connections to multiple destinations and allows the Fabric to service them in any order. While the N_Port is connected to one destination, another connect-request may be processed by the Fabric to minimize the connect latency.

Unless a Class 1 or Class 6 N_Port and the Fabric support stacked connect-requests, when two N_Ports are in Dedicated Connection, a Class 1 connect-request from another source for one of these N_Ports is busied regardless of Intermix support.

If a Class 2 frame destined to one of the N_Ports established in a Dedicated Connection is received, the Class 2 frame may be busied and the transmitting N_Port is notified. In the case of a Class 3 frame, the frame is discarded and no notification is sent. The destination F_Port may be able to hold the frame for a period of time before discarding the frame or returning a busy frame. If Intermix is supported and the Fabric receives a Class 2 or Class 3 frame destined to one of the N_Ports established in a Dedicated Connection, the Fabric may allow delivery with or without a delay if the delivery does not interfere with the transmission of Class 1 or Class 6 frames (see 16.4).

4.9.3 Connectionless Sub-Fabric

A Connectionless Sub-Fabric is characterized by the absence of Dedicated Connections. The Connectionless Sub-Fabric multiplexes frames at frame boundaries between a F_Port and any other F_Port and between the N_Ports attached to them.

The Fabric notifies the transmitting N_Port with a reason code embedded in a Link_Response frame, if it is unable to deliver a Class 2 frame. In the case of a Class 3 frame, the Fabric does not notify the transmitting N_Port.

A given frame flows through the Connectionless Sub-Fabric for the duration of the routing. After the frame is routed, the Connectionless Sub-Fabric is not required to have memory of source, routing, or destination of the frame.

When frames from multiple N_Ports are targeted for the same destination N_Port in Class 2 or Class 3, congestion of frames may occur within the Fabric. Management of this congestion is part of the Connectionless Sub-Fabric and buffer-to-buffer flow control.

If any buffer-to-buffer flow control error occurs and as a result causes overflow (see clause 20), the Fabric logs the error and may discard the overflow frame without notification. Error logging is vendor unique.

4.10 Fibre Channel services

Fibre Channel services (e.g., Directory Server) may be provided in a Fibre Channel configuration to meet the needs of the configuration. Each of these services is addressed with a well-known address for the N_Port providing the service (see 12.3). These well-known addresses are recognized and routed to by the Fabric. These services may be centralized or distributed.

4.11 Building Blocks

The set of building blocks defined in FC-2 are:

- Frame
- Sequence
- Exchange
- Protocol

4.11.1 Building block hierarchy

The FC-2 building blocks are used in a hierarchical fashion, as illustrated in Figure 9.

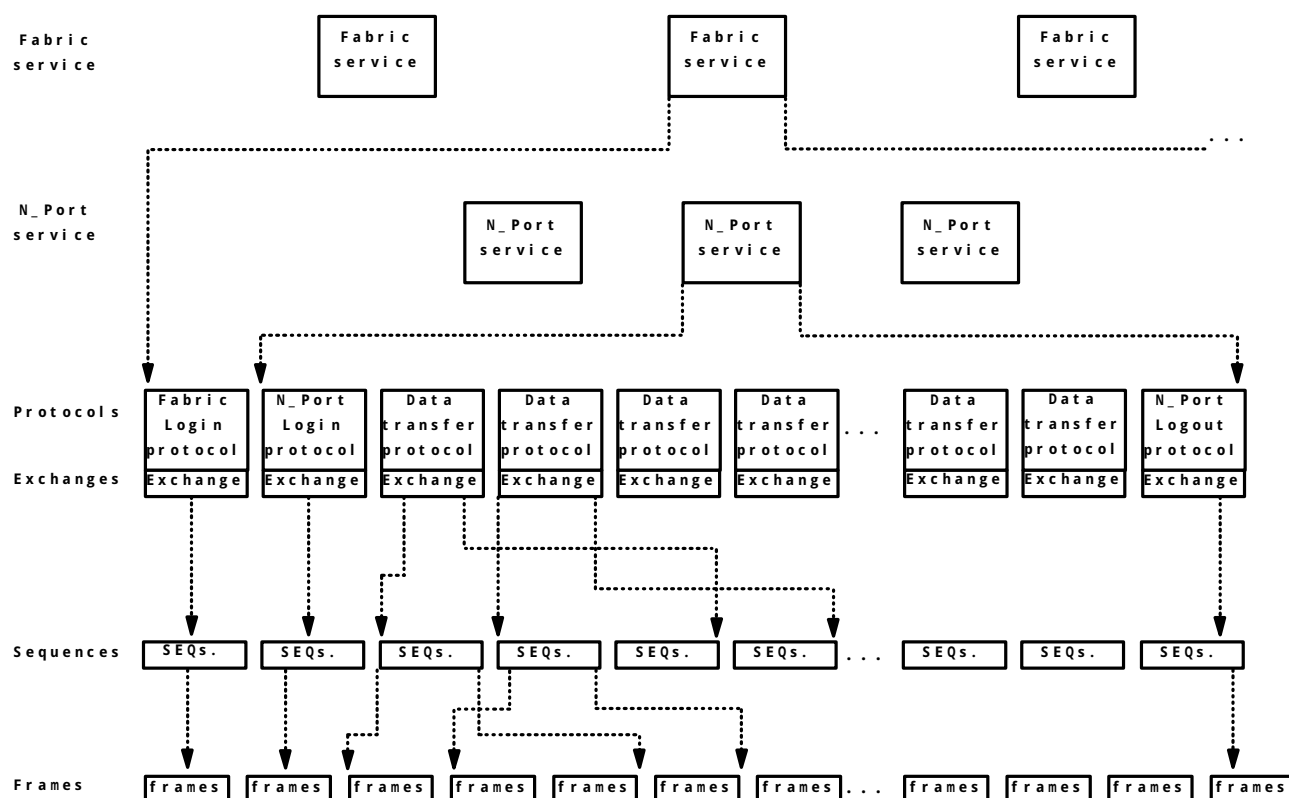


Figure 9 - FC-2 building block hierarchy

A Sequence is made up of one or more Data frames and if applicable, corresponding responses (see 18.1 and clause 14). An Exchange is made up of one or more Sequences flowing in a single direction from the Originator of the Exchange to the Responder or in both directions between the Originator and the Responder (see clause 18).

Prior to use by a ULP for its data transfer, the Fibre Channel has to be setup for the operating environment. The Fibre Channel operating environment is setup by performing two protocols named Fabric Login and N_Port_Login (see clause 17). Once these two Logins are performed, a ULP may start using the Fibre Channel until one or both of these Logins are invalidated. For example, after power down of a N_Port, the previous Fabric Login might have become invalid due to an event such as Fabric reconfiguration and the N_Port may need to repeat the Fabric Login.

Each Login uses an Exchange as the mechanism to accomplish the function. A ULP data transfer is performed using an Exchange as the mechanism (see Figure 9) with the related FC-4 translating the ULP protocol to FC-2 protocol.

4.11.2 Frame

Frames are based on a common frame format (see clause 11). Frames are broadly categorized as Data frames and Link_ Control frames (see Table 32). Data frames (see 14.2) are classified as

- Link_Data frames,
- Device_Data frames, and
- Video_Data frames.

Link_Control frames (see 14.3) are classified as

- Acknowledge (ACK) frames,
- Link_Response (Busy and Reject) frames, and
- Link_Command frames.

Selective retransmission of frames for error recovery is not supported in FC-FS (see clause 23). However, an individual frame may be busied in Class 2 and the sender retransmits the busied frame (see 14.3.3.1 and 14.3.3.2) up to the ability of the sender to retry (see clause 16). The number of times the sender may retry is not specified in FC-FS and may be application or vendor unique.

4.11.3 Sequence

A Sequence is a set of one or more related Data frames transmitted unidirectionally from one N_Port to another N_Port with corresponding Link_Control frames, if applicable, transmitted in response. A N_Port that transmits a Sequence is referred to as the Sequence Initiator and the N_Port that receives the Sequence is referred to as the Sequence Recipient. Sequence rules are specified in clause 18.

Error recovery is performed on the Sequence boundary at the discretion of a level higher than FC-2. If a frame is not transmitted error free, and the error policy requires error recovery, the Sequence to which the frame belongs may be retransmitted (see clause 23).

4.11.3.1 Sequence_Identifier (SEQ_ID)

The Sequence Initiator assigns the Sequence a Sequence_Identifier (SEQ_ID). The Sequence Recipient uses the same SEQ_ID in its response frames. The Sequence Initiator at each of the communicating N_Port pair assigns SEQ_ID independent of the other.

4.11.3.2 Sequence Status Blocks

A Sequence Status Block (SSB) is a logical construct representing the format of the Sequence status information (see 18.8). It is used to track the progress of a Sequence at a N_Port on a frame by frame basis. A Sequence Initiator SSB and a Sequence Recipient SSB are used by the respective N_Ports to track the status of a given Sequence.

- When a Sequence Initiator starts a Sequence, the Sequence Initiator allocates a SSB to be associated with the Sequence it has initiated.
- The Sequence Recipient subsequently allocates a SSB at its end, associated with the sequence the Sequence Initiator has initiated.
- Both the Sequence Initiator and Sequence Recipient N_Ports track the status of this Sequence respectively through the Sequence Initiator and the Sequence Recipient SSBs.

The maximum number of concurrent Sequences between two N_Ports is limited to the smaller of the number of SSBs available at these N_Ports. This value is established during N_Port_Login through the Service Parameters (see clause 17)

4.11.4 Exchange

An Exchange is composed of one or more non-concurrent Sequences (see clause 18). An Exchange may be unidirectional or bi-directional. A unidirectional Exchange results when the same N_Port initiates all the Sequences within the Exchange. A bi-directional Exchange results when the Sequences within the Exchange are initiated by both the N_Ports, but not concurrently. An Exchange may be performed within one Class 1 Connection or may be continued across multiple Class 1 Connections.

NOTE - FC-FS does not specify an Exchange performed concurrently across multiple N_Ports.

4.11.4.1 Exchange_Identifiers (OX_ID and RX_ID)

Exchange_Identifiers may be used by the Originator and Responder to uniquely identify an Exchange.

- The Originator assigns each new Exchange an Originator Exchange_ID (OX_ID) unique to the Originator or Originator-Responder pair and embeds it in all frames of the Exchange.
- The Responder assigns a Responder Exchange_Identifier (RX_ID) unique to the Responder or Responder-Originator pair and communicates it to the Originator before the end of the first Sequence of the Exchange in Classes 1 and 2 (see 18.5). The Responder embeds the RX_ID along with OX_ID in all subsequent frames of the Exchange.
- On receiving the RX_ID from the Responder, the Originator embeds both the RX_ID and OX_ID in all subsequent frames of the Exchange.

The Originator may initiate multiple concurrent Exchanges, but each uses a unique OX_ID.

4.11.4.2 Association_Header

In some system models, there is a need to associate an Exchange with higher-level system constructs (see clause 19). A specific process or group of processes in such system models is identified using the Association_Header (see 19.2). The format of the Association_Header is specified in 13.4.

The support requirements of these two functions are determined during N_Port_Login (see 17.6.8.3).

4.11.4.3 Exchange Status Blocks

An Exchange Status Block is a logical construct representing the format of the Exchange status information (see 18.8). It is used to track the progress of an Exchange on a Sequence by Sequence basis. An Originator Exchange Status Block and a Responder Exchange Status Block respectively are used by an Originator and a Responder to track the status of an Exchange.

- When an Originator initiates an Exchange, it assigns an Originator Exchange Status Block associated with the OX_ID assigned to the Exchange (see 18.8.1).
- The Responder assigns a Responder Exchange Status Block associated with the RX_ID the Responder assigned to the Exchange. The Responder references the Responder Exchange Status Block through its respective RX_ID (see 18.8.1).
- Both the Originator and the Responder track the status of the Exchange at their respective N_Ports.

4.11.5 Protocol

FC-FS provides and describes data transfer protocols and rules to be used by higher level protocols to accomplish a given function. FC-FS provides Login and Logout control functions to manage the operating environment to perform data transfers. FC-FS specifies Primitive Sequences which are low-level Ordered Sets providing word synchronization and a handshake mechanism and ensuring that both the transmitter and receiver are able to transmit and receive Primitive Signals (see 10.4).

- Primitive Sequence protocols: Primitive Sequence protocols are based on Primitive Sequences and specified for Link Failure, Link Initialization, Link Reset, and Online to Offline transition (see 10.6).

- Fabric Login protocol: A N_Port interchanges Service Parameters with the Fabric if present, by explicitly performing the Fabric Login protocol or implicitly through an equivalent method not defined in FC-FS (see 17.1 and 17.3).
- N_Port Login protocol: Before performing data transfer, the N_Port interchanges its Service Parameters (see 17.1, **Error! Reference source not found.**, and 17.6) with another N_Port explicitly through a N_Port Login protocol or implicitly through an equivalent method not defined in FC-FS.
- Data transfer protocol: The ULP data is transferred using Data transfer protocols.
- N_Port Logout protocol: A N_Port may request removal of its Service Parameters from the other N_Port by performing a N_Port Logout protocol (see 17.5). This request may be used to free up resources at the other N_Port.

4.12 Segmentation and reassembly

Segmentation and reassembly (see clause 21) are the FC-2 functions provided to subdivide application data to be transferred into Payloads, embed each Payload in an individual frame, transfer these frames over the link and reassemble the application data at the receiving end as sent by the sending end.

4.12.1 Application data mapping

Mapping application data to Upper Level Protocols (ULPs) is outside the scope of FC-FS. ULPs maintain the status of application data transferred.

4.12.2 Sending end mapping

Upper levels at the sending end specify to FC-2

- blocks or subblocks to be transferred within a Sequence,
- Information Category for each block or subblock (see 12.2),
- a Relative Offset space starting from zero, representing a ULP-defined origin, to the highest address, for each Information Category (see 21.2.1), and
- an Initial Relative Offset for each block or subblock to be transferred.

The Relative Offset relationship between the blocks to be transferred in multiple Sequences is defined by an upper level and is transparent to FC-2.

Examples of mapping multiple blocks or subblocks are illustrated in Figure 10 and Figure 11.

4.12.3 Relative Offset

Relative Offset is a FC-2 construct used to indicate the displacement of the first data byte of each frame's Payload with reference to a ULP-defined origin for a block or a collection of subblocks corresponding to an Information Category at the sending end.

If Relative Offset is not supported (see clause 17), SEQ_CNT is used to perform the segmentation and reassembly (see clause 21).

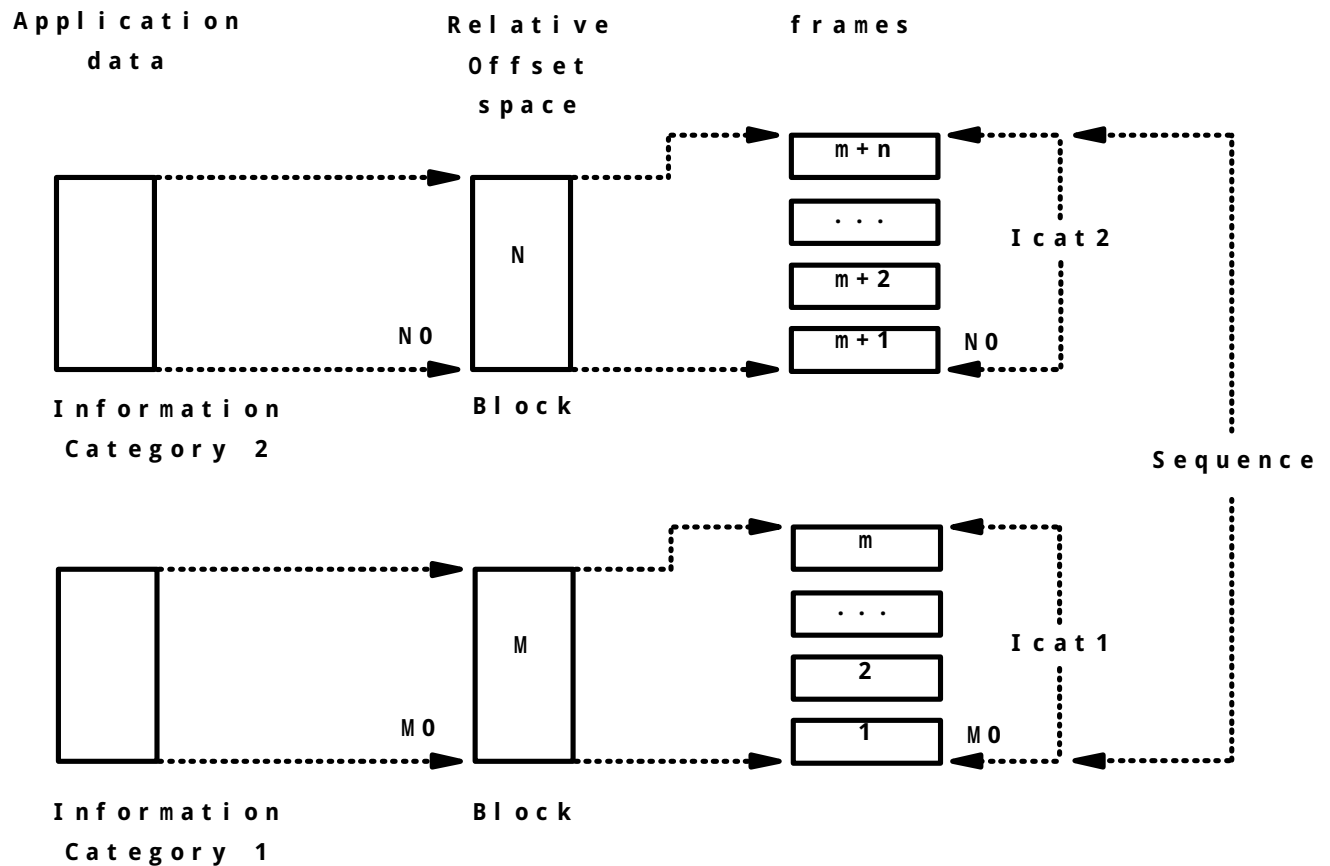
4.12.4 Capability

The Sequence Recipient indicates during Login its capability to support Continuously Increasing or Random Relative Offset (see clause 17). If only the former is supported, each Information Category transferred within a Sequence is treated as a block by upper levels. If Random Relative Offset is supported, an Information Category may be specified as subblocks by upper levels and subblocks may be transmitted in a random order.

4.12.5 FC-2 mapping

The FC-2 maps each block as a single Information Category within the Relative Offset space specified with Continuously Increasing Relative Offset. The FC-2 maps one or more Information Categories into a single Sequence as specified.

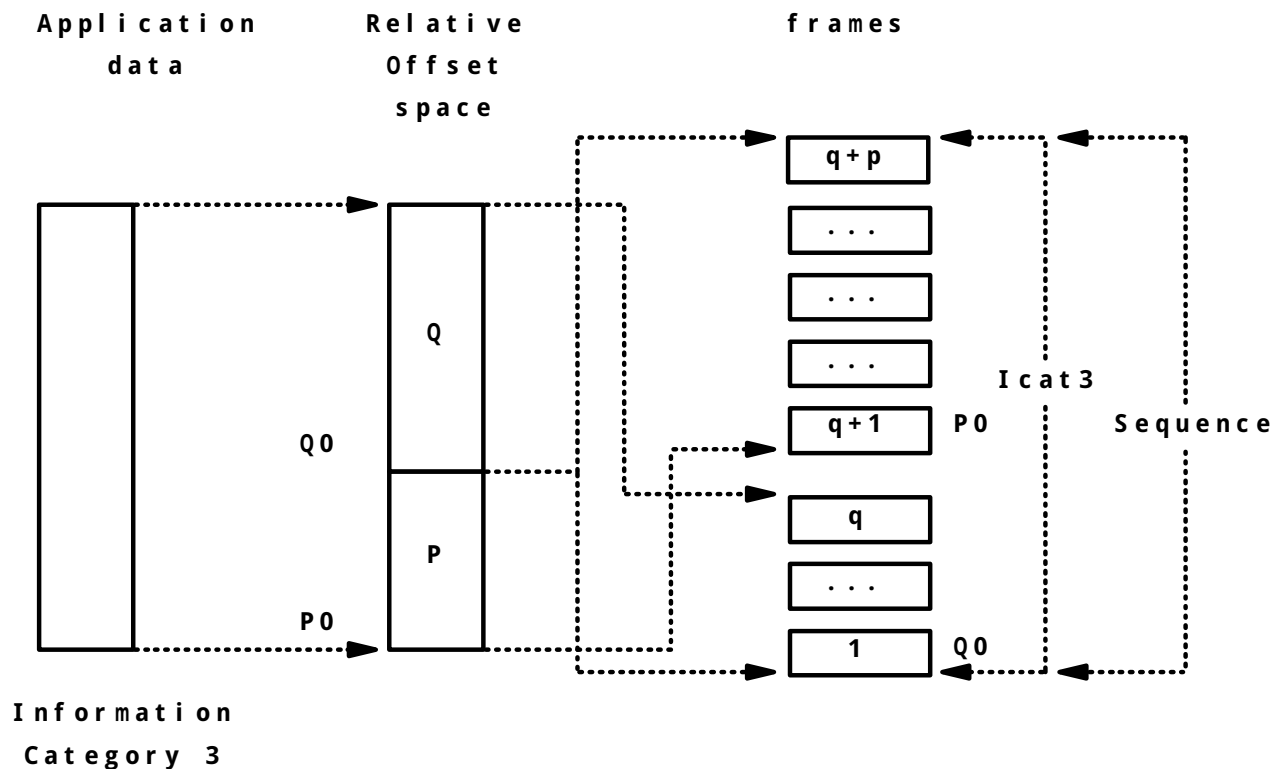
The FC-2 maps a collection of subblocks corresponding to an Information Category into a single Sequence within the Relative Offset space as specified with Random Offset.



Legend:

- m and n - Number of frames for M and N
- M0 and N0 - Initial Relative Offset for M and N
- Icat1 - Information Category 1
- Icat2 - Information Category 2

Figure 10 - Mapping multiple blocks to a single Sequence



Legend:

P and Q - Sub-blocks

p and q - Number of frames for P and Q

P0 and Q0 - Initial Relative Offset for P and Q

Icat3 - Information Category 3

Figure 11 - Mapping subblocks to a single Sequence

4.12.6 Segmentation

Segmentation is the process by which the FC-2 subdivides a block or subblock to be transferred into frames with Payloads of equal or varied sizes and transmits them over the link. During segmentation, the FC-2 determines the size of Payload for each frame in the Sequence. It embeds the data in the Payload of the frame along with the Relative Offset of the Payload.

Relative Offset for the first frame of the block equals the Initial Relative Offset value specified to the FC-2. The Initial Relative Offset specified may be zero or non-zero. Relative Offset for each succeeding frame of the block or subblock is computed as the sum of Relative Offset and the Payload of the current frame.

From the FC-2 perspective, the Relative Offset management is very similar for a block or subblock. Since subblocks may be transferred in a random order, the FC-2 is requested to transfer subblocks, only if the recipient supports Random Relative Offset as specified during Login.

If Relative Offset is Continually Increasing, a block contains only one subblock. If Relative Offset is Random, a block consists of more than one subblock and the upper level specifies the

Initial Relative Offset of each subblock; the Initial Relative Offsets of successive blocks are allowed to vary as required by the upper level.

4.12.7 Reassembly

Reassembly is the FC-2 process which reassembles each block or subblock by placing the Payload from each frame at the Relative Offset specified in the frame or by using the SEQ_CNT if Relative Offset is not supported (see clause 21). Reassembly is performed on an Information Category basis.

The FC-2 segmentation and reassembly processes are illustrated in Figure 12.

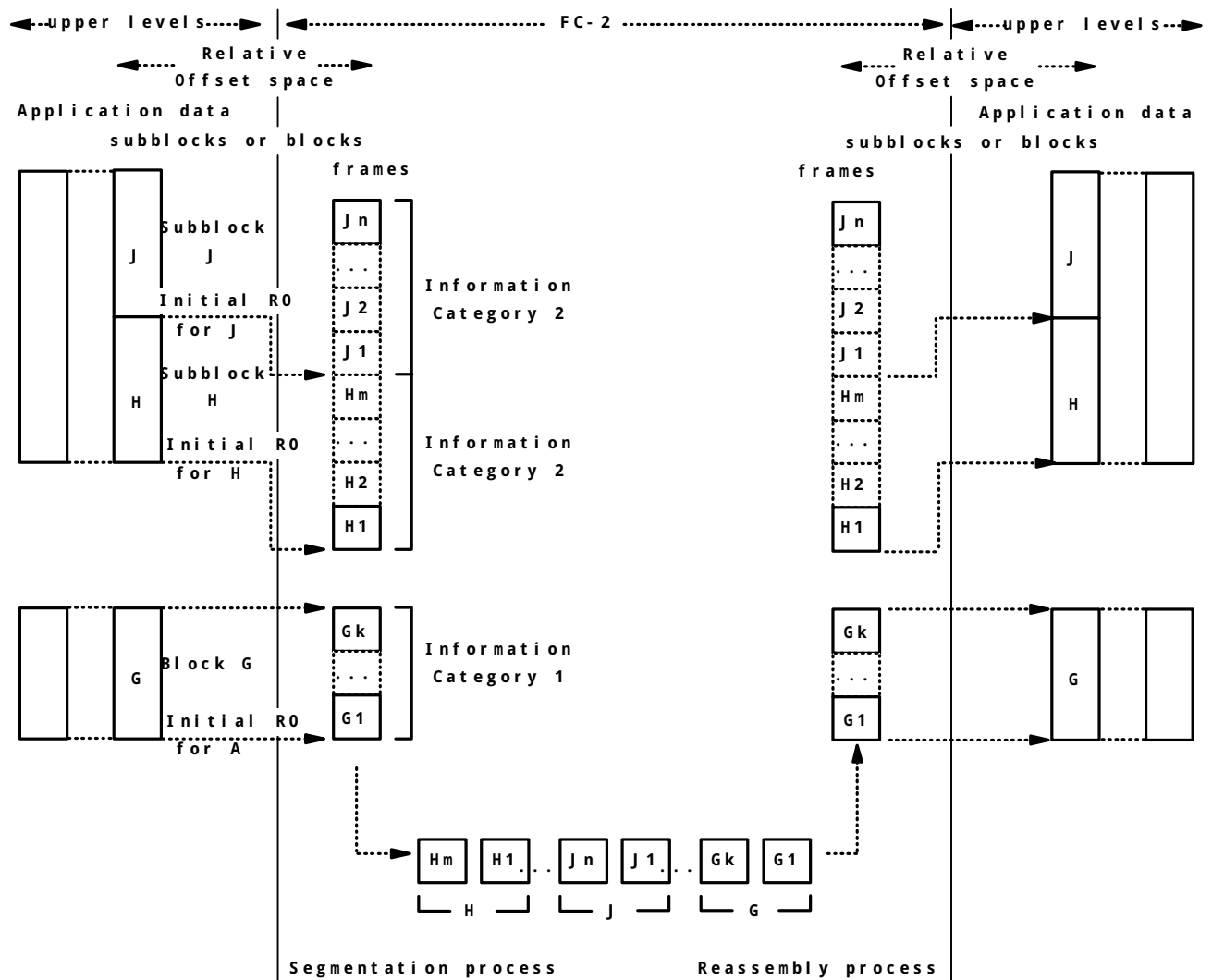


Figure 12 - Segmentation and re-assembly

4.13 Error detection and recovery

In general, detected errors fall into two broad categories: frame errors and link-level errors. Frame errors result from missing or corrupted frames. Corrupted frames are discarded and the

resulting error is detected at the Sequence level. At the Sequence level, a missing frame is detected or the Sequence times out due to one or more missing Data frames or Acknowledgments. If the discard policy is used, the Sequence is aborted at the Sequence level once an error is detected. Sequence errors may also cause Exchange errors that may also cause the Exchange to be aborted. Error recovery may be performed on the failing Sequence or Exchange with the involvement of the sending ULP. Other properly performing Sequences are unaffected.

Link-level errors result from errors detected at a lower level of granularity than frames, where the basic signal characteristics are in question. Link-level errors include such errors as loss of signal, loss of synchronization and several link timeout errors that indicate no frame activity. Link-level errors may be isolated to a portion of the link. Transmission and reception of Primitive Sequences accomplish recovery from link-level errors. Recovery at the link-level disturbs normal frame flow and may introduce Sequence errors that must be resolved after recovery at the link-level.

Error recovery hierarchy

The recovery of errors may be described by the following hierarchy:

```

Abort Exchange
  (Abort Exchange Protocol-frames)
    Abort Sequence
      (Abort Sequence Protocol-frames)
        Link Reset
          (Link Reset Protocol-primitives)
            Link Failure
              (Link Failure Protocol-primitives)
                Link Initialization
                  (Link Init Protocol-primitives)

```

4.14 FC-4 Regions

FC-FG defines a *Region* as a section of a Sub-Fabric such that all the N_Ports in that Region operate at the same data rate and the same Class of Service. Additionally, the N_Ports within the Region share the same Regional Service Parameters e.g., address assignment mode and in-order frame delivery. Essentially, this means that all N_Ports within a Region are able to successfully perform the PLOGI procedure with each other. Figure 13 illustrates a set of N_Ports across 2 Regions: N_Ports A, B, C, and D are in Region 1 because they all support Class 1 at 1 Gbit/second. N_Ports D, E, and F are in Region 2 because they all support Class 2 at 1 Gbit/second. Note that N_Port D is in both Regions.

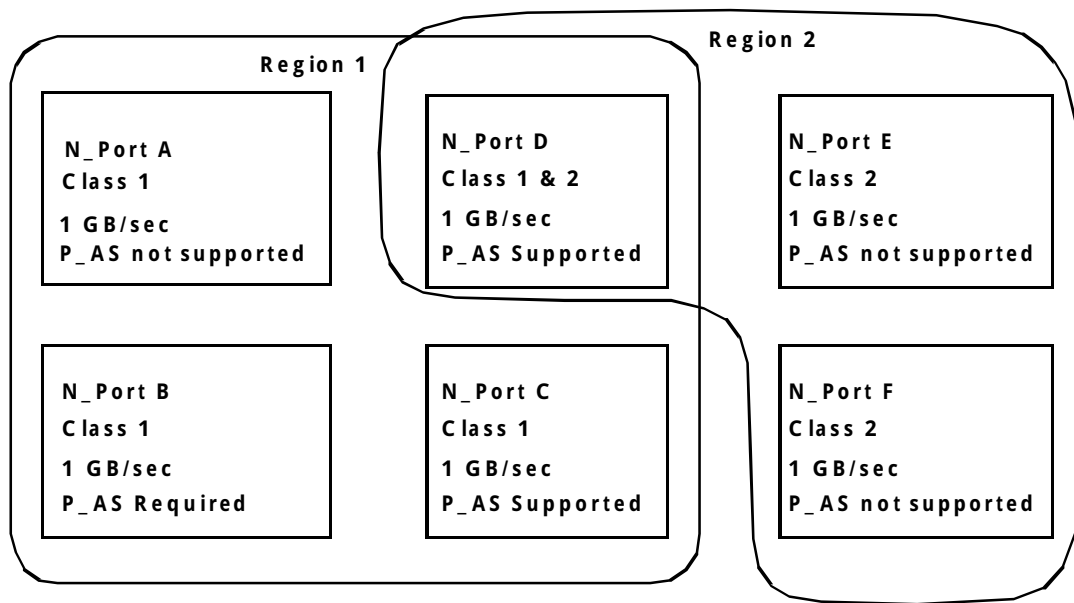


Figure 13 - Fabric Regions

However, just because a pair of N_Ports can execute a PLOGI-ACC with each other does not guarantee that they can successfully perform a FC-4 Exchange. For example, N_Port B that requires an Initial Process_Associator is not capable of communicating with N_Port A that does not support the Initial Process_Associator.

Therefore, the term *FC-4 Region* is used to define a set of N_Ports within a Fabric Region which can successfully communicate via FC-4 Exchanges. That is, all their PLOGI Service Parameters are such that they can successfully originate an Exchange for the purpose of supporting a FC-4 protocol. Figure 14 illustrates the same set of N_Ports, now in 3 FC-4 Regions: N_Ports A, C, and D are in Region 1 because they all support Class 1 at 1 Gbit/second and they all will not use the Initial Process Associator, as determined by PLOGI. N_Ports D, E, and F are in Region 2 because they all support Class 2 at 1 Gbit/second and they all will not use the Initial Process Associator, as determined by PLOGI. N_Ports B, C, and D are in Region 3 because they all support Class 1 at 1 Gbit/second and they all will use the Initial Process Associator, as determined by PLOGI.

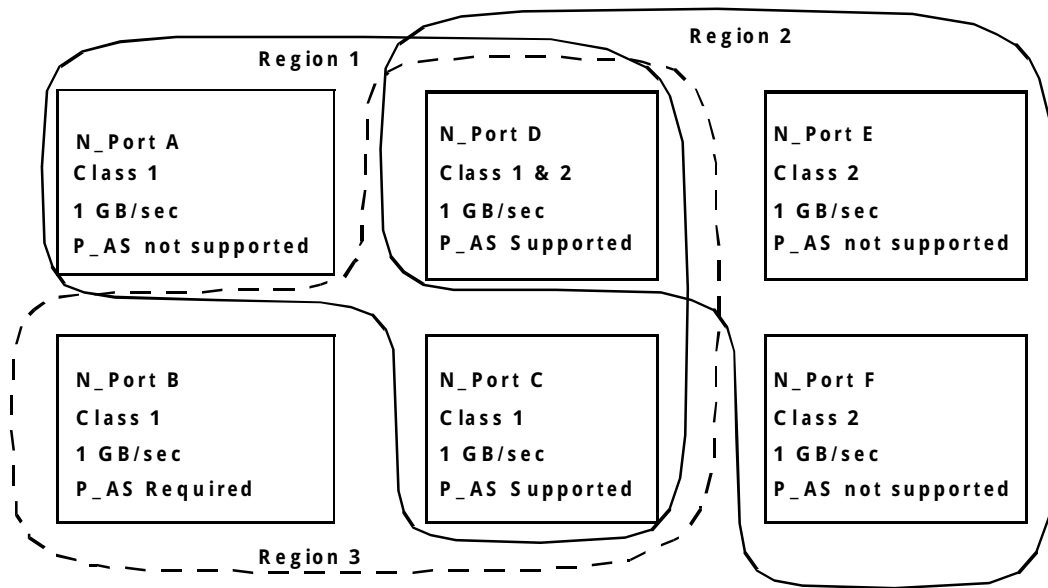


Figure 14 - FC-4 Regions

5 FC-1 8B/10B Transmission Code

Information to be transmitted over a Fibre shall be encoded eight bits at a time into a 10-bit Transmission Character and then sent serially by bit. Information received over a Fibre shall be collected ten bits at a time, and those Transmission Characters that are used for data, called Data Characters, shall be decoded into the correct eight-bit codes. The 10-bit Transmission Code supports all 256 eight-bit combinations. Some of the remaining Transmission Characters, referred to as Special Characters, are used for functions, which are to be distinguishable from the contents of a frame.

- NOTE - The following definition of the 10-bit Transmission Code is based on (and is in basic agreement with) Widmer, 1983 ⁽¹⁾ and Franasak, 1984 ⁽²⁾. Both of these references describe the same 10-bit transmission code.
- NOTE - The primary rationale for use of a Transmission Code is to improve the transmission characteristics of information to be transferred across a Fibre. The encodings defined by the Transmission Code ensure that sufficient transitions are present in the serial bit stream to make clock recovery possible at the receiver. Such encoding also greatly increases the likelihood of detecting any single or multiple bit errors that may occur during transmission and reception of information. In addition, some of the Special Characters of the Transmission Code contain a distinct and easily recognizable bit pattern (a Comma) which assists a receiver in achieving word alignment on the incoming bit stream.

5.1 Notation conventions

FC-1 uses letter notation for describing information bits and control variables. Such notation differs from the bit notation specified by the remainder of this standard (see 3.2 The following text describes the translation process between these notations and provides a translation example. It also describes the conventions used to name valid Transmission Characters. This text is provided for the purposes of terminology clarification only and is not intended to restrict the implementation of FC-1 functions in any way.

An unencoded FC-1 information byte is composed of eight information bits A,B,C,D,E,F,G,H and the control variable Z. This information is encoded by FC-1 into the bits a,b,c,d,e,i,f,g,h,j of a 10-bit Transmission Character.

An information bit contains either a binary zero or a binary one. A control variable has either the value D or the value K. When the control variable associated with an unencoded FC-1 information byte contains the value D, that byte is referred to as a Valid Data Byte. When the control variable associated with an unencoded FC-1 information byte contains the value K, that byte is referred to as a Special Code.

The information bit labeled A corresponds to bit 0 in the numbering scheme of the FC-2 specification, B corresponds to bit 1, and so on, as shown in Figure 15. The control variable is typically not specified by FC-2. When the control variable is not specified by FC-2, FC-1 assumes its value to be D (data).

FC-2 bit									Control
notation:	7	6	5	4	3	2	1	0	variable
FC-1 unencoded									
bit notation:	H	G	F	E	D	C	B	A	Z

Figure 15 - Bit designations

Each valid Transmission Character has been given a name using the following convention: Zxx.y, where Z is the control variable of the unencoded FC-1 information byte, xx is the decimal value of the binary number composed of the bits E, D, C, B, and A of the unencoded FC-1 information byte in that order, and y is the decimal value of the binary number composed of the bits H, G, and F of the unencoded FC-1 information byte in that order. The value of Z is used

to indicate whether the Transmission Character is a Data Character (Z = D) or a Special Character (Z = K).

Figure 16 shows the conversion from FC-2 byte notation to the FC-1 Transmission Character naming convention described above.

FC-2 byte notation:	hex 'BC' -- Special
	Code
FC-2 bit notation:	<u>7654</u> <u>3210</u> <u>Control</u>
	1011 1100 K
FC-1 unencoded	<u>HGF</u> <u>EDCBA</u> <u>Z</u>
bit notation:	101 11100 K
FC-1 unencoded bit notation	<u>Z</u> <u>EDCBA</u> <u>HGF</u>
reordered to conform with	K 11100 101
Zxx.y naming convention:	
FC-1 Transmission	
Character name:	K 28 .5

Figure 16 - Conversion example

NOTE - Most Kxx.y combinations do not result in valid Transmission Characters within the 8B/10B Transmission Code. Only those combinations which result in Special Characters as specified by Table 3 are considered valid.

5.2 Character encoding and decoding

The following information describes how the tables shall be used for both generating valid Transmission Characters (encoding) and checking the validity of received Transmission Characters (decoding). It also specifies the ordering rules to be followed when transmitting the bits within a character and the characters within the higher-level constructs specified by the document (i.e., Ordered Sets and frames).

5.2.1 Transmission order

Within the definition of the 8B/10B Transmission Code, the bit positions of the Transmission Characters are labeled a,b,c,d,e,i,f,g,h, and j. Bit "a" shall be transmitted first, followed by bits "b," "c," "d," "e," "i," "f," "g," "h," and "j," in that order (note that bit "i" shall be transmitted between bit "e" and bit "f," rather than in the order that would be indicated by the letters of the alphabet).

Characters within Ordered Sets (as specified by 5.4) shall be transmitted sequentially beginning with the Special Character used to distinguish the Ordered Set (e.g., K28.5) and proceeding character by character from left to right within the definition of the Ordered Set until all characters of the Ordered Set are transmitted.

The contents of a frame (as specified in clause 11) shall be transmitted sequentially beginning with the Ordered Set used to denote the start of frame (the SOF delimiter) and proceeding character by character from left to right within the definition of the frame until the Ordered Set used to denote the end of frame (the EOF delimiter) is transmitted.

5.2.2 Valid and invalid Transmission Characters

Table 2 and Table 3 define the valid Data Characters and valid Special Characters (K characters), respectively. These tables shall be used for both generating valid Transmission Characters (encoding) and checking the validity of received Transmission Characters (decoding). In the tables, each Valid-Data-Byte or Special-Code entry has two columns that represent two (not

necessarily different) Transmission Characters. The two columns correspond to the current value of the Running Disparity ("Current RD -" or "Current RD +"). Running Disparity is a binary parameter with either the value negative (-) or the value positive (+).

After powering on or exiting diagnostic mode, the transmitter shall assume the negative value for its initial Running Disparity. Upon transmission of any Transmission Character, the transmitter shall calculate a new value for its Running Disparity based on the contents of the transmitted character.

After powering on or exiting diagnostic mode, the receiver should assume either the positive or negative value for its initial Running Disparity. Upon reception of any Transmission Character, the receiver shall determine whether the Transmission Character is valid or invalid according to the following rules and tables and shall calculate a new value for its Running Disparity based on the contents of the received character.

The following rules for Running Disparity shall be used to calculate the new Running Disparity value for Transmission Characters that have been transmitted (transmitter's Running Disparity) and that have been received (receiver's Running Disparity).

Running Disparity for a Transmission Character shall be calculated on the basis of sub-blocks, where the first six bits (abcdei) form one sub-block (six-bit sub-block) and the second four bits (fghj) form the other sub-block (four-bit sub-block). Running Disparity at the beginning of the six-bit sub-block is the Running Disparity at the end of the last Transmission Character. Running Disparity at the beginning of the four-bit subblock is the Running Disparity at the end of the six-bit sub-block. Running Disparity at the end of the Transmission Character is the Running Disparity at the end of the four-bit sub-block.

Running Disparity for the sub-blocks shall be calculated as follows:

- Running Disparity at the end of any sub-block is positive if the sub-block contains more ones than zeros. It is also positive at the end of the six-bit sub-block if the six-bit sub-block is 000111, and it is positive at the end of the four-bit sub-block if the four-bit sub-block is 0011.
- Running Disparity at the end of any sub-block is negative if the sub-block contains more zeros than ones. It is also negative at the end of the six-bit sub-block if the six-bit sub-block is 111000, and it is negative at the end of the four-bit sub-block if the four-bit sub-block is 1100.
- Otherwise, Running Disparity at the end of the sub-block is the same as at the beginning of the sub-block.

NOTE - All sub-blocks with equal numbers of zeros and ones are disparity neutral. In order to limit the run length of 0s or 1s between sub-blocks, the 8B/10B transmission code rules specify that sub-blocks encoded as 000111 or 0011 are generated only when the Running Disparity at the beginning of the sub-block is positive; thus, Running Disparity at the end of these subblocks will also be positive. Likewise, sub-blocks containing 111000 or 1100 are generated only when the Running Disparity at the beginning of the sub-block is negative; thus, Running Disparity at the end of these sub-blocks will also be negative.

5.2.2.1 Use of the tables for generating Transmission Characters

The appropriate entry in the table shall be found for the Valid Data Byte or Special Code for which a Transmission Character is to be generated (encoded). The current value of the transmitter's Running Disparity shall be used to select the Transmission Character from its corresponding column. For each Transmission Character transmitted, a new value of the Running Disparity shall be calculated. This new value shall be used as the transmitter's Current Running Disparity for the next Valid Data Byte or Special Code to be encoded and transmitted.

5.2.2.2 Use of the tables for checking the validity of received Transmission Characters

The column corresponding to the current value of the receiver's Running Disparity shall be searched for the received Transmission Character. If the received Transmission Character is found in the proper column, then the Transmission Character shall be considered valid and the associated data byte or Special Code determined (decoded). If the received Transmission Character is not found in that column, then the Transmission Character shall be considered invalid and a Code Violation detected and reported to its associated port. Independent of the Transmission Character's validity, the received Transmission Character shall be used to calculate a new value of Running Disparity. This new value shall be used as the receiver's Current Running Disparity for the next received Transmission Character.

NOTE - Detection of a Code Violation does not necessarily indicate that the Transmission Character in which the Code Violation was detected is in error. Code Violations may result from a prior error which altered the Running Disparity of the bit stream but which did not result in a detectable error at the Transmission Character in which the error occurred. The example shown in Figure 17 exhibits this behavior:

	RD	Character	RD	Character	RD	Character	RD
Transmitted character stream	-	D21.1	-	D10.2	-	D23.5	+
Transmitted bit stream	-	101010 1001	-	010101 0101	-	111010 1010	+
Bit stream after error	-	101010 1011	+	010101 0101	+	111010 1010	+
Decoded character stream	-	D21.0	+	D10.2	+	Code violation	+

Figure 17 - Delayed Code Violation example

Table 2 - Valid Data Characters							
Data Byte Name	Bits HGF EDCBA	Current RD - Abcdei fghj	Current RD+ abcdei fghj	Data Byte Name	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD+ abcdei fghj
D0.0	000 00000	100111 0100	011000 1011	D16.1	001 10000	011011 1001	100100 1001
D1.0	000 00001	011101 0100	100010 1011	D17.1	001 10001	100011 1001	100011 1001
D2.0	000 00010	101101 0100	010010 1011	D18.1	001 10010	010011 1001	010011 1001
D3.0	000 00011	110001 1011	110001 0100	D19.1	001 10011	110010 1001	110010 1001
D4.0	000 00100	110101 0100	001010 1011	D20.1	001 10100	001011 1001	001011 1001
D5.0	000 00101	101001 1011	101001 0100	D21.1	001 10101	101010 1001	101010 1001
D6.0	000 00110	011001 1011	011001 0100	D22.1	001 10110	011010 1001	011010 1001
D7.0	000 00111	111000 1011	000111 0100	D23.1	001 10111	111010 1001	000101 1001
D8.0	000 01000	111001 0100	000110 1011	D24.1	001 11000	110011 1001	001100 1001
D9.0	000 01001	100101 1011	100101 0100	D25.1	001 11001	100110 1001	100110 1001
D10.0	000 01010	010101 1011	010101 0100	D26.1	001 11010	010110 1001	010110 1001
D11.0	000 01011	110100 1011	110100 0100	D27.1	001 11011	110110 1001	001001 1001
D12.0	000 01100	001101 1011	001101 0100	D28.1	001 11100	001110 1001	001110 1001
D13.0	000 01101	101100 1011	101100 0100	D29.1	001 11101	101110 1001	010001 1001
D14.0	000 01110	011100 1011	011100 0100	D30.1	001 11110	011110 1001	100001 1001

Table 2 - Valid Data Characters							
Data Byte Name	Bits HGF EDCBA	Current RD - Abcdei fghj	Current RD+ abcdei fghj	Data Byte Name	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD+ abcdei fghj
D15.0	000 01111	010111 0100	101000 1011	D31.1	001 11111	101011 1001	010100 1001
D16.0	000 10000	011011 0100	100100 1011	D0.2	010 00000	100111 0101	011000 0101
D17.0	000 10001	100011 1011	100011 0100	D1.2	010 00001	011101 0101	100010 0101
D18.0	000 10010	010011 1011	010011 0100	D2.2	010 00010	101101 0101	010010 0101
D19.0	000 10011	110010 1011	110010 0100	D3.2	010 00011	110001 0101	110001 0101
D20.0	000 10100	001011 1011	001011 0100	D4.2	010 00100	110101 0101	001010 0101
D21.0	000 10101	101010 1011	101010 0100	D5.2	010 00101	101001 0101	101001 0101
D22.0	000 10110	011010 1011	011010 0100	D6.2	010 00110	011001 0101	011001 0101
D23.0	000 10111	111010 0100	000101 1011	D7.2	010 00111	111000 0101	000111 0101
D24.0	000 11000	110011 0100	001100 1011	D8.2	010 01000	111001 0101	000110 0101
D25.0	000 11001	100110 1011	100110 0100	D9.2	010 01001	100101 0101	100101 0101
D26.0	000 11010	010110 1011	010110 0100	D10.2	010 01010	010101 0101	010101 0101
D27.0	000 11011	110110 0100	001001 1011	D11.2	010 01011	110100 0101	110100 0101
D28.0	000 11100	001110 1011	001110 0100	D12.2	010 01100	001101 0101	001101 0101
D29.0	000 11101	101110 0100	010001 1011	D13.2	010 01101	101100 0101	101100 0101
D30.0	000 11110	011110 0100	100001 1011	D14.2	010 01110	011100 0101	011100 0101
D31.0	000 11111	101011 0100	010100 1011	D15.2	010 01111	010111 0101	101000 0101
D0.1	001 00000	100111 1001	011000 1001	D16.2	010 10000	011011 0101	100100 0101
D1.1	001 00001	011101 1001	100010 1001	D17.2	010 10001	100011 0101	100011 0101
D2.1	001 00010	101101 1001	010010 1001	D18.2	010 10010	010011 0101	010011 0101
D3.1	001 00011	110001 1001	110001 1001	D19.2	010 10011	110010 0101	110010 0101
D4.1	001 00100	110101 1001	001010 1001	D20.2	010 10100	001011 0101	001011 0101
D5.1	001 00101	101001 1001	101001 1001	D21.2	010 10101	101010 0101	101010 0101
D6.1	001 00110	011001 1001	011001 1001	D22.2	010 10110	011010 0101	011010 0101
D7.1	001 00111	111000 1001	000111 1001	D23.2	010 10111	111010 0101	000101 0101
D8.1	001 01000	111001 1001	000110 1001	D24.2	010 11000	110011 0101	001100 0101
D9.1	001 01001	100101 1001	100101 1001	D25.2	010 11001	100110 0101	100110 0101
D10.1	001 01010	010101 1001	010101 1001	D26.2	010 11010	010110 0101	010110 0101
D11.1	001 01011	110100 1001	110100 1001	D27.2	010 11011	110110 0101	001001 0101
D12.1	001 01100	001101 1001	001101 1001	D28.2	010 11100	001110 0101	001110 0101
D13.1	001 01101	101100 1001	101100 1001	D29.2	010 11101	101110 0101	010001 0101
D14.1	001 01110	011100 1001	011100 1001	D30.2	010 11110	011110 0101	100001 0101
D15.1	001 01111	010111 1001	101000 1001	D31.2	010 11111	101011 0101	010100 0101
D0.3	011 00000	100111 0011	011000 1100	D16.4	100 10000	011011 0010	100100 1101
D1.3	011 00001	011101 0011	100010 1100	D17.4	100 10001	100011 1101	100011 0010
D2.3	011 00010	101101 0011	010010 1100	D18.4	100 10010	010011 1101	010011 0010

Table 2 - Valid Data Characters							
Data Byte Name	Bits HGF EDCBA	Current RD - Abcdei fghj	Current RD+ abcdei fghj	Data Byte Name	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD+ abcdei fghj
D3.3	011 00011	110001 1100	110001 0011	D19.4	100 10011	110010 1101	110010 0010
D4.3	011 00100	110101 0011	001010 1100	D20.4	100 10100	001011 1101	001011 0010
D5.3	011 00101	101001 1100	101001 0011	D21.4	100 10101	101010 1101	101010 0010
D6.3	011 00110	011001 1100	011001 0011	D22.4	100 10110	011010 1101	011010 0010
D7.3	011 00111	111000 1100	000111 0011	D23.4	100 10111	111010 0010	000101 1101
D8.3	011 01000	111001 0011	000110 1100	D24.4	100 11000	110011 0010	001100 1101
D9.3	011 01001	100101 1100	100101 0011	D25.4	100 11001	100110 1101	100110 0010
D10.3	011 01010	010101 1100	010101 0011	D26.4	100 11010	010110 1101	010110 0010
D11.3	011 01011	110100 1100	110100 0011	D27.4	100 11011	110110 0010	001001 1101
D12.3	011 01100	001101 1100	001101 0011	D28.4	100 11100	001110 1101	001110 0010
D13.3	011 01101	101100 1100	101100 0011	D29.4	100 11101	101110 0010	010001 1101
D14.3	011 01110	011100 1100	011100 0011	D30.4	100 11110	011110 0010	100001 1101
D15.3	011 01111	010111 0011	101000 1100	D31.4	100 11111	101011 0010	010100 1101
D16.3	011 10000	011011 0011	100100 1100	D0.5	101 00000	100111 1010	011000 1010
D17.3	011 10001	100011 1100	100011 0011	D1.5	101 00001	011101 1010	100010 1010
D18.3	011 10010	010011 1100	010011 0011	D2.5	101 00010	101101 1010	010010 1010
D19.3	011 10011	110010 1100	110010 0011	D3.5	101 00011	110001 1010	110001 1010
D20.3	011 10100	001011 1100	001011 0011	D4.5	101 00100	110101 1010	001010 1010
D21.3	011 10101	101010 1100	101010 0011	D5.5	101 00101	101001 1010	101001 1010
D22.3	011 10110	011010 1100	011010 0011	D6.5	101 00110	011001 1010	011001 1010
D23.3	011 10111	111010 0011	000101 1100	D7.5	101 00111	111000 1010	000111 1010
D24.3	011 11000	110011 0011	001100 1100	D8.5	101 01000	111001 1010	000110 1010
D25.3	011 11001	100110 1100	100110 0011	D9.5	101 01001	100101 1010	100101 1010
D26.3	011 11010	010110 1100	010110 0011	D10.5	101 01010	010101 1010	010101 1010
D27.3	011 11011	110110 0011	001001 1100	D11.5	101 01011	110100 1010	110100 1010
D28.3	011 11100	001110 1100	001110 0011	D12.5	101 01100	001101 1010	001101 1010
D29.3	011 11101	101110 0011	010001 1100	D13.5	101 01101	101100 1010	101100 1010
D30.3	011 11110	011110 0011	100001 1100	D14.5	101 01110	011100 1010	011100 1010
D31.3	011 11111	101011 0011	010100 1101	D15.5	101 01111	010111 1010	101000 1010
D0.4	100 00000	100111 0010	011000 1101	D16.5	101 10000	011011 1010	100100 1010
D1.4	100 00001	011101 0010	100010 1101	D17.5	101 10001	100011 1010	100011 1010
D2.4	100 00010	101101 0010	010010 1101	D18.5	101 10010	010011 1010	010011 1010
D3.4	100 00011	110001 1101	110001 0010	D19.5	101 10011	110010 1010	110010 1010
D4.4	100 00100	110101 0010	001010 1101	D20.5	101 10100	001011 1010	001011 1010
D5.4	100 00101	101001 1101	101001 0010	D21.5	101 10101	101010 1010	101010 1010
D6.4	100 00110	011001 1101	011001 0010	D22.5	101 10110	011010 1010	011010 1010

Table 2 - Valid Data Characters							
Data Byte Name	Bits HGF EDCBA	Current RD - Abcdei fghj	Current RD+ abcdei fghj	Data Byte Name	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD+ abcdei fghj
D7.4	100 00111	111000 1101	000111 0010	D23.5	101 10111	111010 1010	000101 1010
D8.4	100 01000	111001 0010	000110 1101	D24.5	101 11000	110011 1010	001100 1010
D9.4	100 01001	100101 1101	100101 0010	D25.5	101 11001	100110 1010	100110 1010
D10.4	100 01010	010101 1101	010101 0010	D26.5	101 11010	010110 1010	010110 1010
D11.4	100 01011	110100 1101	110100 0010	D27.5	101 11011	110110 1010	001001 1010
D12.4	100 01100	001101 1101	001101 0010	D28.5	101 11100	001110 1010	001110 1010
D13.4	100 01101	101100 1101	101100 0010	D29.5	101 11101	101110 1010	010001 1010
D14.4	100 01110	011100 1101	011100 0010	D30.5	101 11110	011110 1010	100001 1010
D15.4	100 01111	010111 0010	101000 1101	D31.5	101 11111	101011 1010	010100 1010
D0.6	110 00000	100111 0110	011000 0110	D0.7	111 00000	100111 0001	011000 1110
D1.6	110 00001	011101 0110	100010 0110	D1.7	111 00001	011101 0001	100010 1110
D2.6	110 00010	101101 0110	010010 0110	D2.7	111 00010	101101 0001	010010 1110
D3.6	110 00011	110001 0110	110001 0110	D3.7	111 00011	110001 1110	110001 0001
D4.6	110 00100	110101 0110	001010 0110	D4.7	111 00100	110101 0001	001010 1110
D5.6	110 00101	101001 0110	101001 0110	D5.7	111 00101	101001 1110	101001 0001
D6.6	110 00110	011001 0110	011001 0110	D6.7	111 00110	011010 1110	011010 0001
D7.6	110 00111	111000 0110	000111 0110	D7.7	111 00111	111000 1110	000111 0001
D8.6	110 01000	111001 0110	000110 0110	D8.7	111 01000	111001 0001	000110 1110
D9.6	110 01001	100101 0110	100101 0110	D9.7	111 01001	100101 1110	100101 0001
D10.6	110 01010	010101 0110	010101 0110	D10.7	111 01010	010101 1110	010101 0001
D11.6	110 01011	110100 0110	110100 0110	D11.7	111 01011	110100 1110	110100 1000
D12.6	110 01100	001101 0110	001101 0110	D12.7	111 01100	001101 1110	001101 0001
D13.6	110 01101	101100 0110	101100 0110	D13.7	111 01101	101100 1110	101100 1000
D14.6	110 01110	011100 0110	011100 0110	D14.7	111 01110	011100 1110	011100 1000
D15.6	110 01111	010111 0110	101000 0110	D15.7	111 01111	100111 0001	101000 1110
D16.6	110 10000	011011 0110	100100 0110	D16.7	111 10000	011011 0001	100100 1110
D17.6	110 10001	100011 0110	100011 0110	D17.7	111 10001	100011 0111	100011 0001
D18.6	110 10010	010011 0110	010011 0110	D18.7	111 10010	010011 0111	010011 0001
D19.6	110 10011	110010 0110	110010 0110	D19.7	111 10011	110010 1110	110010 0001
D20.6	110 10100	001011 0110	001011 0110	D20.7	111 10100	001011 0111	001011 0001
D21.6	110 10101	101010 0110	101010 0110	D21.7	111 10101	101010 1110	101010 0001
D22.6	110 10110	011010 0110	011010 0110	D22.7	111 10110	011010 1110	011010 0001
D23.6	110 10111	111010 0110	000101 0110	D23.7	111 10111	111010 0001	000101 1110
D24.6	110 11000	110011 0110	001100 0110	D24.7	111 11000	110011 0001	001100 1110
D25.6	110 11001	100110 0110	100110 0110	D25.7	111 11001	100110 1110	100110 0001
D26.6	110 11010	010110 0110	010110 0110	D26.7	111 11010	010110 1110	010110 0001

Table 2 - Valid Data Characters							
Data Byte Name	Bits HGF EDCBA	Current RD - Abcdei fghj	Current RD+ abcdei fghj	Data Byte Name	Bits HGF EDCBA	Current RD - abcdei fghj	Current RD+ abcdei fghj
D27.6	110 11011	110110 0110	001001 0110	D27.7	111 11011	110110 0001	001001 1110
D28.6	110 11100	001110 0110	001110 0110	D28.7	111 11100	001110 1110	001110 0001
D29.6	110 11101	101110 0110	010001 0110	D29.7	111 11101	101110 0001	010001 1110
D30.6	110 11110	011110 0110	100001 0110	D30.7	111 11110	011110 0001	100001 1110
D31.6	110 11111	101011 0110	010100 0110	D31.7	111 11111	101011 0001	010100 1110

Table 3 - Valid Special Characters		
Special Code Name	Current RD - abcdei fghj	Current RD + abcdei fghj
K28.0	001111 0100	110000 1011
K28.1	001111 1001	110000 0110
K28.2	001111 0101	110000 1010
K28.3	001111 0011	110000 1100
K28.4	001111 0010	110000 1101
K28.5	001111 1010	110000 0101
K28.6	001111 0110	110000 1001
K28.7	001111 1000	110000 0111
K23.7	111010 1000	000101 0111
K27.7	110110 1000	001001 0111
K29.7	101110 1000	010001 0111
K30.7	011110 1000	100001 0111

Unless a transmitter is operating in diagnostic mode, the K28.7 Special Character shall not be followed by any of the following Special or Data Characters: K28.x, D3.x, D11.x, D12.x, D19.x, D20.x, or D28.x, where x is a value in the range 0 to 7, inclusive (see clause 8 for a discussion of diagnostic mode).

NOTE - The above restriction simplifies receiver word synchronization hardware.

5.3 Word encoding and decoding

A Transmission Word is composed of four contiguous Transmission Characters treated as a unit. A word prior to transmission and after reception is composed of a combination of four Valid Data Bytes and Special Codes treated as a unit. These Valid Data Bytes and Special Codes shall be encoded according to the rules specified by 5.2 to create a Transmission Word. Likewise, the Transmission Characters of a Transmission Word shall be decoded according to the rules specified by 5.2 to create Valid Data Bytes and Special Codes.

5.4 Ordered Sets

Table 4, Table 5, and Table 6 specify the Ordered Sets (composed of Special and Data Characters) which are defined for use by this standard. The following Ordered Set types are defined in clause 10:

- Frame Delimiters
- Primitive Signals
- Primitive Sequences

Table 4 - Frame delimiters			
Abbr.	Delimiter Function	Beginning RD	Ordered Set
SOFc1	SOF Connect Class 1	Negative	K28.5 - D21.5 - D23.0 - D23.0
SOFi1	SOF Initiate Class 1	Negative	K28.5 - D21.5 - D23.2 - D23.2
SOFi1	SOF Normal Class 1	Negative	K28.5 - D21.5 - D23.1 - D23.1
SOFi2	SOF Initiate Class 2	Negative	K28.5 - D21.5 - D21.2 - D21.2
SOFn2	SOF Normal Class 2	Negative	K28.5 - D21.5 - D21.1 - D21.1
SOFi3	SOF Initiate Class 3	Negative	K28.5 - D21.5 - D22.2 - D22.2
SOFn3	SOF Normal Class 3	Negative	K28.5- D21.5 - D22.1 - D22.1
SOFc4	SOF Activate Class 4	Negative	K28.5 - D21.5 - D25.0 - D25.0
SOFi4	SOF Initiate Class 4	Negative	K28.5 - D21.5 - D25.2 - D25.2
SOFn4	SOF Normal Class 4	Negative	K28.5 - D21.5 - D25.1 - D25.1
SOFf	SOF Fabric	Negative	K28.5 - D21.5 - D24.2 - D24.2
EOFt	EOF Terminate	Negative	K28.5 - D21.5 - D21.3 - D21.3
		Positive	K28.5 - D21.4 - D21.3 - D21.3
EOFdt	EOF Disconnect-Terminate-Class 1 EOF Deactivate-Terminate-Class 4	Negative	K28.5 - D21.4 - D21.4 - D21.4
		Positive	K28.5 - D21.5 - D21.4 - D21.4
EOFa	EOF Abort	Negative	K28.5 - D21.4 - D21.7 - D21.7
		Positive	K28.5 - D21.5 - D21.7 - D21.7
EOFn	EOF Normal	Negative	K28.5 - D21.4 - D21.6 - D21.6
		Positive	K28.5 - D21.5 - D21.6 - D21.6
EOFni	EOF Normal-Invalid	Negative	K28.5 - D10.4 - D21.6 - D21.6
		Positive	K28.5 - D10.5 - D21.6 - D21.6
EOFdti	EOF Disconnect-Terminate-Invalid Class 1 EOF Disconnect-Deactivate-Invalid Class 4	Negative	K28.5 - D10.4 - D21.4 - D21.4
		Positive	K28.5 - D10.5 - D21.4 - D21.4
EOFr	EOF Remove-Terminate Class 4	Negative	K28.5 - D21.4 - D25.4 - D25.4
		Positive	K28.5 - D21.5 - D25.4 - D25.4
EOFr	EOF Remove-Terminate Invalid Class 4	Negative	K28.5 - D10.4 - D25.4 - D25.4
		Positive	K28.5 - D10.5 - D25.4 - D25.4
Explanation: SOF = Start-of-frame delimiter EOF = End-of-frame delimiter			

Table 5 - Primitive Signals			
Abbr.	Primitive Signal	Beginning RD	Ordered Set
Idle	Idle	Negative	K28.5 - D21.4 - D21.5 - D21.5
R_RDY	Receiver_Ready	Negative	K28.5 - D21.4 - D10.2 - D10.2
VC_RDY	Virtual Circuit Ready	Negative	K28.5 - D21.7 - VC_ID - VC_ID
BB_SCs	Buffer-to-buffer State Change (SOF)	Negative	K28.5 - D21.4 - D22.4 - D22.4
BB_SCR	Buffer-to-buffer State Change (R_RDY)	Negative	K28.5 - D21.4 - D22.6 - D22.6
NOTE - For a definition of VC_ID see clause 28.			

Table 6 - Primitive Sequences		
Primitive Sequence	Beginning RD	Ordered Set
Offline (OLS)	Negative	K28.5 D21.1 D10.4 D21.2
Not_Operational (NOS)	Negative	K28.5 D21.2 D31.5 D5.2
Link_Reset (LR)	Negative	K28.5 D9.2 D31.5 D9.2
Link_Reset_Response (LRR)	Negative	K28.5 D21.1 D31.5 D9.2

NOTES

- Each EOF-delimiter Ordered Set is defined such that negative Current Running Disparity will result after processing of the final (rightmost) character of the Ordered Set. This, in combination with the Running Disparity Initialization rules specified in 5.2.2, ensures that the first Ordered Set following an EOF delimiter, transmitter power on, or transmitter exit from diagnostic mode will always be transmitted with negative Beginning Running Disparity. The Ordered Sets defined for the Primitive Signals and Primitive Sequences preserve this negative Disparity, ensuring that the Ordered Sets associated with SOF Delimiters, Primitive Signals, and Primitive Sequences will also always be transmitted with negative Beginning Running Disparity. As a result, Primitive Signal, Primitive Sequence, and SOF Delimiter Ordered Sets are defined for the negative Beginning Running Disparity case only. The primary benefit of such a definition is that it allows idle words to be removed and added from an encoded bit stream one word at a time without altering the Beginning Running Disparity associated with the Transmission Word subsequent to the removed idle word.
- The K28.5 Special Character is chosen as the first character of all Ordered Sets for the following reasons:
 - Bits abcdeif make up a Comma; this is a singular bit pattern which in the absence of transmission errors cannot appear in any other location of a Transmission Character and cannot be generated across the boundaries of any two adjacent Transmission Characters. The Comma can be used to easily find and verify character and word boundaries of the received bit stream.
 - Bits ghj of the encoded character present the maximum number of transitions, simplifying receiver acquisition of bit synchronization.
- The second character of the Ordered Sets used to represent EOF Delimiters differentiates between normal and invalid frames. It also ensures that the Running Disparity resulting after processing of an EOF Ordered Set is negative independent of the value of Beginning Running Disparity. Link_Reset (LR) and Link_Reset_Response (LRR) Ordered Sets are also differentiated through the use of their second characters. In all other Ordered Sets, it serves only as a placeholder, which provides a high number of bit transitions.
- The third and fourth characters of the Delimiter functions, Receiver_Ready, and the idle word are repeated to ensure that an error affecting a single character will not result in the recognition of an Ordered Set other than the one transmitted. The third and fourth characters of the other Ordered Sets defined by the standard have been selected to provide distinct patterns which provide a large number of transitions and good spectral characteristics.
- The categories of Delimiter, Primitive Signal, and Primitive Sequence have meaning to the port and are defined by clause 10. Transmitters and receivers recognize the character combinations defined in this section as Ordered Sets only.

6 FC-1 receiver and transmitter description

6.1 Receiver state description

Whenever a signal (as defined in FC-PI) is detected on a fibre, and the receiver is not in Loopback mode, the receiver attached to that fibre shall attempt to achieve synchronization on both bit and Transmission-Word boundaries of the received encoded bit stream (see clause 7 for a description of Loopback mode). A definition of Bit Synchronization is provided in clause 3. Transmission-Word synchronization is defined in 6.1.2.2. Synchronization failures on either bit or Transmission-Word boundaries are not separately identifiable and cause loss-of-synchronization errors.

6.1.1 Receiver states

The receiver state diagram is shown in Figure 19.

6.1.1.1 Operational states

Synchronization to Transmission-Word boundaries, hereafter referred to as Synchronization, is achieved when the receiver identifies the same Transmission-Word boundary on the received bit stream as that established by the transmitter at the other end of the fibre. The procedure used to achieve this condition is described in 6.1.2.2. When this condition is achieved, the receiver shall enter the Synchronization-Acquired state. A receiver in the Synchronization-Acquired state shall provide information that has been received from its attached fibre and decoded.

When the Transmission-Word boundary identified on the received bit stream by the receiver no longer matches the boundary established by the transmitter to which it is connected, the receiver shall enter the Loss-Of-Synchronization state. Such a receiver shall remain Operational but shall cease to provide received and decoded information. When the receiver is in the Loss-Of-Synchronization state, the procedure described in 6.1.2 shall be used to regain Synchronization.

When the receiver is Operational, it is always in one of the two states described above. The determination of an Operational receiver's state is based on the conditions described in 6.1.2 and 6.1.3.

6.1.1.2 Not Operational state

A receiver shall become Not Operational and shall enter the Reset state when a reset condition is imposed upon it, either internally or externally.

6.1.2 Entry into Synchronization-Acquired state

A receiver shall enter the Synchronization-Acquired state when it has achieved both bit and Transmission-Word synchronization.

6.1.2.1 Bit Synchronization

An Operational receiver that is in the Loss-Of-Synchronization state shall first acquire Bit Synchronization before attempting to acquire Transmission-Word synchronization. A definition of Bit Synchronization is provided in clause 3. After achieving Bit Synchronization, the receiver shall remain in the Loss-Of-Synchronization state until it achieves Transmission-Word synchronization.

6.1.2.2 Transmission-Word synchronization

An Operational receiver that is in the Loss-Of-Synchronization state and that has acquired Bit Synchronization shall attempt to acquire Transmission-Word synchronization. Transmission-Word synchronization is acquired by the detection of three Ordered Sets containing Commas in their leftmost bit positions without an intervening invalid Transmission Word, as specified by "Invalid Transmission Word rules." Upon acquisition of Transmission-Word synchronization, the receiver shall enter the Synchronization-Acquired state. The third detected Ordered Set shall be considered valid information and shall be decoded and provided by the receiver to its port. Ordered Set detection shall include both detection of the individual characters that make up an Ordered Set and proper alignment of those characters (i.e., the Special Character used to designate an Ordered Set shall be aligned in the leading (leftmost) character position of the received Transmission Word).

Word alignment of the received bit stream shall be achieved via the detection of a Comma or K28.5 Transmission Character in that stream. In the absence of bit errors, placement of a detected Comma or K28.5 Transmission Character at the leftmost position of a received Transmission Word ensures proper alignment of that word and of subsequently received Transmission Words.

While attempting to achieve word synchronization, an Operational receiver shall be required to detect only those Ordered Sets that contain a Comma in their leftmost bit positions.

The method used by the receiver to implement the word alignment function and to detect Ordered Sets is not defined by this unofficial compilation of standards.

NOTES

1. The Comma contained within the K28.1, K28.5, and K28.7 Special Characters is a singular bit pattern which in the absence of transmission errors cannot appear in any other location of a Transmission Character and cannot be generated across the boundaries of any two adjacent Transmission Characters. This bit pattern is sufficient to identify the word alignment of the received bit stream. Some implementations (e.g., those which choose to implement the word alignment function in continuously-enabled mode) may choose to align on the full K28.5 Ordered Set to decrease the likelihood of false alignment when bit errors are present in the received bit stream.
2. The word alignment function may be enabled in either of two modes:
 - Continuously-enabled word alignment (continuous-mode alignment)
 - Explicitly-enabled word alignment (explicit-mode alignment)

Continuous-mode alignment allows the receiver to reestablish word alignment at any point in the incoming bit stream while the receiver is Operational. Such realignment is likely (but not guaranteed) to result in Code Violations and subsequent loss of Synchronization. Under certain conditions, it may be possible to realign an incoming bit stream without loss of Synchronization. If such a realignment occurs within a received frame, detection of the resulting error condition is dependent upon higher-level function (e.g., invalid CRC, missing EOF Delimiter).

Explicit-mode alignment allows the receiver to reestablish word alignment under controlled circumstances (e.g., while in the Loss-Of-Synchronization state). Once Synchronization has been acquired, the word alignment function of the receiver is disabled.

6.1.3 Entry into Loss-Of-Synchronization state

The following four conditions shall cause an Operational receiver to enter the Loss-Of-Synchronization state:

- Completion of the Loss-Of-Synchronization procedure
- Transition to power on
- Exit from receiver reset condition

- Detection of loss of signal

NOTE - While in the Loss-Of-Synchronization state, the receiver may attempt to reacquire bit resynchronization. In some instances, this may allow the receiver to regain Synchronization and enter the Synchronization-Acquired state when it otherwise would not be possible. However, initiation of bit resynchronization may also delay the resynchronization process by forcing the receiver to re-establish a clock reference when such reestablishment is otherwise unnecessary. See FC-PI for a detailed discussion of Bit Synchronization.

6.1.3.1 Loss-of-Synchronization procedure

The loss-of-Synchronization procedure defines the method by which the receiver changes from the Synchronization-Acquired state to the Loss-Of-Synchronization state. The procedure tests each received Transmission Word according to the rules defined in "Invalid Transmission Word rules" to determine the validity of the Transmission Word.

Starting in the Synchronization-Acquired state, the receiver shall check each received Transmission Word according to the rules of "Invalid Transmission Word rules" to determine if the word is valid. The receiver shall continue to check the Transmission Words and shall remain in the Synchronization-Acquired state until the loss-of-Synchronization procedure, as described by "Loss-of-Synchronization-procedure states," is completed.

Invalid Transmission Word rules

An invalid Transmission Word shall be recognized by the receiver when one of the following conditions is detected:

- A Code Violation on a character basis, as specified by Table 2 and Table 3, is detected within a Transmission Word. This is referred to as a Code Violation condition.
- Any valid Special Character is detected in the second, third, or fourth character position of a Transmission Word. This is referred to as an invalid Special Code alignment condition.
- A defined Ordered Set (as listed in 5.4) is received with improper Beginning Running Disparity (e.g., a SOF delimiter is received with positive Beginning Running Disparity, an EOF delimiter specified for positive Beginning Running Disparity is received when Beginning Running Disparity for that Transmission Word is negative). This is referred to as an invalid Beginning Running Disparity condition.

Loss-of-Synchronization-procedure states

The following five detection states are defined as part of the loss-of-Synchronization procedure:

- a) No invalid Transmission Word has been detected (the No Invalid-Transmission-Word detection state).
- b) The first invalid Transmission Word is detected (the First Invalid-Transmission-Word detection state).
- c) The second invalid Transmission Word is detected (the Second Invalid-Transmission-Word detection state).
- d) The third invalid Transmission Word is detected (the Third Invalid-Transmission-Word detection state).
- e) The fourth invalid Transmission Word is detected (the Fourth Invalid-Transmission-Word detection state).

A receiver in the Synchronization-Acquired state may be in any of the first four detection states listed above. A receiver in the fifth detection state listed above (the Fourth-Invalid Transmission-Word detection state) shall enter the Loss-Of-Synchronization state.

When the procedure is in detection state a, checking for an invalid Transmission Word shall be performed. After each invalid Transmission Word is detected, one of the detection states b through e shall be entered. When the procedure is in detection state b, c, or d, checking for additional invalid Transmission Words shall be performed in groups of two consecutive Transmission Words. If two consecutive valid Transmission Words are received, the count of previously detected invalid Transmission Words shall be reduced by one, and the previous detection state shall be entered. The loss-of-Synchronization procedure is completed when detection state e is entered.

The No-Invalid-Transmission-Word detection state shall be entered on a transition to the Synchronization-Acquired state or after the First-Invalid-Transmission-Word detection state is reset.

The First-Invalid-Transmission-Word detection state shall be entered after the first invalid Transmission Word is detected or after the previous detection state (detection of the second invalid Transmission Word) is reset. Subsequent Transmission Words received shall be checked to determine whether an additional invalid Transmission Word is detected within the next consecutive two or fewer Transmission Words received. If an additional invalid Transmission Word is detected within the next consecutive two or fewer Transmission Words received, the Second Invalid-Transmission-Word detection state shall be entered. If two consecutive valid Transmission Words are received, the No Invalid-Transmission-Word detection state shall be entered.

The Second-Invalid-Transmission-Word detection state shall be entered after the second invalid Transmission Word is detected or after the previous detection state (detection of the third invalid Transmission Word) is reset. Subsequent Transmission Words received shall be checked to determine whether an additional invalid Transmission Word is detected within the next consecutive two or fewer Transmission Words received. If an additional invalid Transmission Word is detected within the next consecutive two or fewer Transmission Words received, the Third Invalid-Transmission-Word detection state shall be entered. If two consecutive valid Transmission Words are received, the First Invalid-Transmission-Word detection state shall be entered.

The Third-Invalid-Transmission-Word detection state shall be entered after the third invalid Transmission Word is detected. Subsequent Transmission Words received shall be checked to determine whether an additional invalid Transmission Word is detected within the next consecutive two or fewer Transmission Words received. If an additional invalid Transmission Word is detected within the next consecutive two or fewer Transmission Words received, the Fourth-Invalid-Transmission-Word detection state shall be entered and the receiver shall immediately enter the Loss-Of-Synchronization state. If two consecutive valid Transmission Words are received, the Second-Invalid-Transmission-Word detection state shall be entered.

The Fourth-Invalid-Transmission-Word detection state shall be entered after the fourth invalid Transmission Word is detected. Upon entering this detection state, the receiver shall immediately enter the Loss-Of-Synchronization state. Subsequent Transmission Words received shall not be checked as the loss-of-Synchronization procedure is complete. The receiver shall remain in the Loss-Of-Synchronization state until one of the following occurs:

- The receiver regains Synchronization
- The receiver is reset

Figure 18 graphically portrays the Loss-of-Synchronization procedure. States a through e are keyed to the detection states described by the ordered list at the beginning of this section. State a is the initial detection state entered by a receiver upon acquisition of Synchronization. States a through d are detection states which are possible only when the receiver has achieved Synchronization. Entry into State e results in loss of receiver Synchronization.

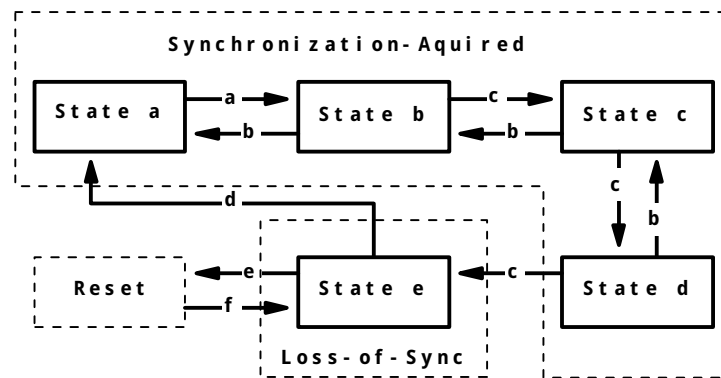


Figure 18 - Loss-of-Synchronization procedure state diagram

State Transition Conditions:

- The first invalid Transmission Word is detected
- An additional invalid Transmission Word is not detected in the next two or fewer consecutive Transmission Words
- An additional invalid Transmission Word is detected in the next two or fewer consecutive Transmission Words
- The receiver regains Synchronization
- The receiver is Reset
- The receiver exits a previously established Reset condition

NOTE - The rationale for the loss-of-Synchronization procedure is to reduce the likelihood that a single error will result in a loss of Synchronization. For example, a single two-bit error positioned to overlap two Transmission Words could result in the detection of three invalid Transmission Words; the two Transmission Words directly affected by the error and a subsequent Transmission Word which was affected by a disparity change resulting from the error. The procedure described above would maintain Synchronization in such a case.

6.1.3.2 Transition to power on

When the receiver is Operational after being powered on, the receiver shall enter the Loss-Of-Synchronization state.

6.1.3.3 Exit from receiver reset condition

When the receiver is Operational after exiting from a receiver reset condition imposed upon it, either externally or internally, the receiver shall enter the Loss-Of-Synchronization state.

NOTE - The conditions required for a receiver in the Reset state to exit that state are not defined by this unofficial compilation of standards. Such conditions may be based on explicit indications. They may also be time-dependent in nature.

6.1.3.4 Detection of loss of signal

When an Operational receiver recognizes a loss-of-signal condition, the Loss-Of-Synchronization state shall be entered (if the receiver is not presently in that state). The receiver shall remain in this state until one of the following conditions occur:

- The loss-of-signal condition is corrected and Synchronization is regained
- The receiver is reset

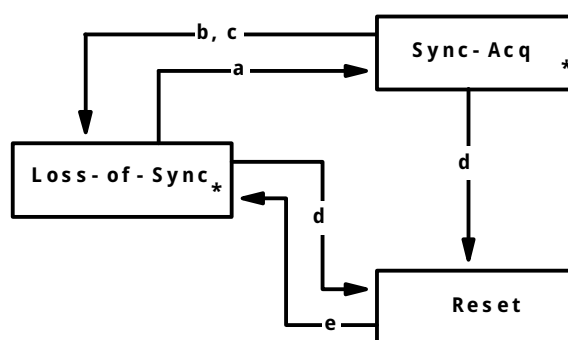
6.1.4 Entry into Reset state

When a receiver reset condition is imposed on a receiver, either internally or externally, the receiver shall enter the Reset state. Once the Reset state is entered, the receiver shall become Not Operational and shall remain in the Reset state until it is subsequently made Operational by exiting the receiver reset condition.

NOTES

1. A typical use of receiver reset is to force a receiver in the Loss-Of-Synchronization state to attempt reacquisition of Bit Synchronization (note that entry into this state does not necessarily indicate loss of Bit Synchronization).
2. The conditions required for a receiver in the Reset state to exit that state are not defined by this unofficial compilation of standards. Such conditions may be based on explicit indications. They may also be time-dependent in nature.

6.2 Receiver state diagram



* See note 3 on page 54.

Figure 19 - Receiver state diagram

State Transition Conditions:

- a) Acquisition of Synchronization (see 6.1.2)
- b) Completion of loss-of-Synchronization procedure (see 6.1.3.1)
- c) Detection of a loss of signal condition (see 6.1.3.4)
- d) Reset condition imposed on the receiver (see 6.1.4)
- e) Exiting of receiver reset condition (see 6.1.3.3)

NOTES

1. The Loss-Of-Synchronization state is the default state upon completion of receiver Initialization.
2. The data path width of a variable-path-width receiver must be established before that receiver is capable of entering the Synchronization-Acquired state.
3. The receiver may be in either Loopback or normal mode when in states denoted by * (see clause 7 for a discussion of Loopback mode).

6.3 Transmitter state description

While Operational and not disabled, a transmitter attached to a Fibre shall constantly attempt to transmit an encoded bit stream onto that Fibre. Some transmitters are capable of monitor-

ing this transmitted signal and verifying its validity. Such transmitters may become Not Operational as a result of this monitoring. A transmitter may also be placed in a disabled state under certain internal or external conditions.

The transmitter state diagram is shown in Figure 20.

6.3.1 Transmitter states

6.3.1.1 Operational states

A transmitter actively attempting to transmit an encoded bit stream onto its attached fibre is defined to be in the *Working* state.

Under certain conditions, it may be necessary or desirable to cease the transmission of signals by a transmitter onto its attached Fibre. When such an action is initiated by a request from the port associated with a transmitter or as the result of an external event, a transmitter shall enter the Not-Enabled state. When such an action results from detection of a laser safety condition, a transmitter shall enter the Open-Fibre state. A transmitter in a Not-Enabled or Open-Fibre state shall remain Operational.

NOTE - A transmitter in a Not-Enabled or Open-Fibre state is free to transmit those signals associated with laser safety procedures. See clause 9 and FC-PI for additional information related to laser safety.

6.3.1.2 Not Operational state

A transmitter shall become Not Operational and shall enter the Failure state when the transmitter detects a failure condition. Detectable transmitter failure conditions are defined by vendor unique signal monitoring rules. The presence or absence of such rules in an implementation shall not affect compliance with the standard.

A transmitter-failure condition shall be recognized and reported by the transmitter to its associated port when the transmitter becomes Not Operational and enters the Failure state.

6.3.2 Entry into Working state

An Operational transmitter shall enter the Working state when its transmitter becomes enabled. A transmitter shall become enabled under either of the following conditions:

- Processing of a port request to enter the enabled state when the transmitter was not previously placed in the Open-Fibre state as a result of laser safety procedures
- Removal of a previously-detected laser safety condition (i.e., a condition which requires that the transmitter cease transmission) while no transmitter disable request remains outstanding

While in the Working state, a transmitter shall actively attempt to transmit valid requests for information transfer from its associated port. Valid requests for information transfer conform to the following rules:

- Requests for Ordered Set transmission are aligned with the word alignment established by the transmitter upon entry into the Working state.
- Requests for Valid Data Byte transmission are aligned with the byte alignment established by the transmitter upon entry into the Working state. Each byte of a four-byte word is aligned based on the established word alignment.

The following requests for information transfer are inconsistent with the established transmission rules and cannot occur:

- Transmission requests improperly aligned with transmission (word and byte) boundaries

- Transmission requests not consistent with the current transmission boundary
- Requests for Ordered Set transfers on a non-word boundary
- Requests for data transfers overlapping an Ordered Set transfer currently in progress
- Lack of transmission requests (i.e., additional bytes required to complete a Transmission Word) when a Transmission Word has been partially transmitted as a result of prior transmission requests

6.3.3 Entry into Not-Enabled state

Entry into the Not-Enabled state may result from external error conditions detected by the Link_Control Facility, or it may result from internally generated signals. Specific conditions under which the transmitter shall enter the Not-Enabled state are as follows:

- Completion of transmitter Initialization
- Processing of a port request to enter the Not-Enabled state when a laser safety condition is not currently present
- Removal of a previously-detected laser safety condition while a transmitter disable request remains outstanding

The transmitter shall remain in the Not-Enabled state until the conditions causing it to cease transmission are removed.

6.3.4 Entry into Open-Fibre state

Entry into the Open-Fibre state results from external error conditions detected by the Link_Control Facility. Specifically, the transmitter enters the Open-Fibre state upon determination by FC-0 OFC that a laser safety condition exists.

The transmitter shall remain in the Open-Fibre state until the laser safety condition is removed. A state change shall not result from receipt of a transmitter disable or enable request while in the Open-Fibre state. However, such receipt may affect the subsequent state change that results upon removal of the previously detected laser safety condition (see 6.3.2 and 6.3.3).

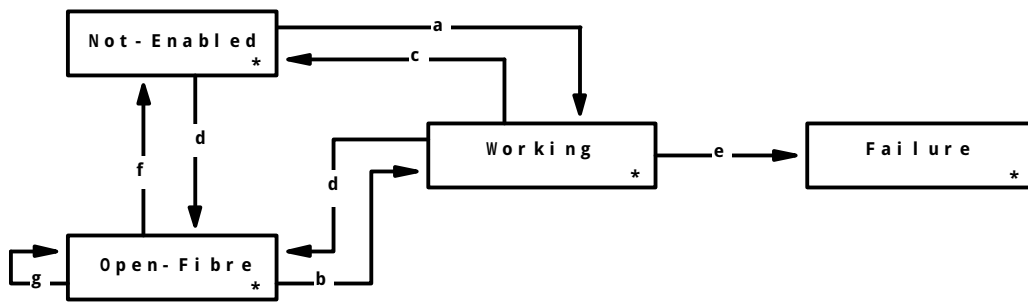
6.3.5 Entry into Failure state

A transmitter shall enter into the Failure state upon recognition (via signal monitoring) of a transmitter-failure condition while in the Working state. Once the Failure state is entered, the transmitter shall become Not Operational and shall remain in the Failure state until it is subsequently made Operational through external intervention.

A transmitter shall not be required to provide a signal monitoring function. Each transmitter capable of monitoring its transmitted signal shall define a method by which this monitoring is to take place. It shall also define the conditions necessary to cause the transmitter to recognize a transmitter-failure condition and change from the Working state to the Failure state.

NOTE - Monitoring of average light levels is a typical method of transmitter monitoring. Typically, an error condition is detected when the transmitted light level of a transmitter falls outside of the range of allowed values specified for its transmitter class (see FC-PI for a discussion of signal monitoring.).

6.4 Transmitter state diagram



* See note 3 on page below.

Figure 20 - Transmitter state diagram

State Transition Conditions:

- Receipt of transmitter enable request from the port when a laser safety condition is not currently present (see 6.3.2)
- Determination that a laser safety condition no longer exists and no transmitter disable request remains outstanding (see 6.3.2)
- Receipt of transmitter disable request from the port when a laser safety condition is not currently present (see 6.3.3)
- Determination that a laser safety condition exists (see 6.3.4)
- Detection of a transmitter failure condition (see 6.3.5)
- Determination that a laser safety condition no longer exists when an outstanding transmitter disable request exists (see 6.3.3)
- Receipt of transmitter disable or enable request (see 6.3.4)

NOTES

- The Not_Enabled state is the default state upon completion of transmitter Initialization.
- The data path width of a variable-path-width transmitter must be established before that transmitter may enter the Working state.
- The transmitter may be in either Loopback or normal mode when in states denoted by * (see clause 7 for a discussion of Loopback mode).

7 Loopback mode

Loopback mode shall be provided by the transmitter and receiver of a port as a diagnostic function to the port. When Loopback mode is selected, transmission requests passed to the transmitter shall be shunted directly to the receiver, overriding any signal detected by the receiver on its attached Fibre. A Link_Control Facility shall be explicitly placed in Loopback mode (i.e., Loopback mode is not the normal mode of operation of a Link_Control Facility). The method of implementing Loopback mode is not defined by this standard.

NOTE - Loopback mode may be implemented either in the parallel or the serial circuitry of a Link_Control Facility.

7.1 Receiver considerations

A receiver in the Loss-Of-Synchronization or Synchronization-Acquired state may be placed in Loopback mode. A receiver in any other state shall not be placed in Loopback mode.

Entry into or exit from Loopback mode may result in a temporary loss of Synchronization. Under such conditions, decoded information shall not be presented by the receiver to the port until Synchronization has been reestablished.

7.2 Transmitter considerations

A transmitter in the Working, Open-Fibre, or Not-Enabled state may be placed in Loopback mode. A transmitter in any other state shall not be placed in Loopback mode. The external behavior of a transmitter (i.e., the activity of a transmitter with respect to its attached Fibre) simultaneously in the Working state and in Loopback mode is unpredictable.

8 Diagnostic mode

A limited set of diagnostic functions may be provided as an implementation option for testing of the transmitter function of the FC-0 portion of the Fibre Channel (see FC-PI).

Diagnostic functions

Some diagnostic functions, which are not defined by this standard, may be provided by certain implementations. The provision or exclusion of such functions shall not affect compliance with the standard by an implementation.

NOTE - A typical diagnostic function is the ability to transmit invalid Transmission Characters within an otherwise valid bit stream. Certain invalid bit streams may cause a receiver to lose word and/or bit synchronization. See FC-PI for a more detailed discussion of receiver and transmitter behavior under various diagnostic conditions.

9 Transmitter safety

Short wave (780 nm) laser transmitters (as defined in FC-PI) require special procedures to ensure that Class 1 laser safety standards are met. These procedures have the following effect on the transmitter definition:

- A special transmitter state (the Open-Fibre state) is defined.
- When FC-0 OFC determines that a laser safety condition (i.e., a condition that requires that the transmitter cease transmission) exists, the transmitter shall enter the Open-Fibre state.
- When FC-0 OFC determines that a laser safety condition no longer exists, the transmitter shall exit the Open-Fibre state.
- Port requests are incapable of forcing a transmitter which enters the Open-Fibre state as a result of FC-0 OFC to exit that state.

For a detailed description of FC-0 OFC, see FC-PI.

10 Ordered Sets

10.1 Introduction

An Ordered Set is a four-character combination of data and special transmission characters. FC-FS designates a number of Ordered Sets to have special meaning. Ordered Sets provide the ability to obtain bit and word synchronization, which also establishes word boundary alignment. See 5.4 for additional information on Ordered Sets and 6.1.2.2 on rules for synchronization. The following types of Ordered Sets are defined:

- Start_of_Frame delimiters
- End_of_Frame delimiters
 - Primitive Signals
 - Idle
- Receiver_Ready (R_RDY)
- Primitive Sequences
 - Not_Operational (NOS)
 - Offline (OLS)
 - Link_Reset (LR)
 - Link_Reset_Response (LRR).

If an unrecognized Ordered Set is detected, it shall be treated as an Idle Primitive Signal.

NOTE - Treating unrecognized Ordered Sets as Idles allows future introduction of Ordered Sets for additional features and functions beyond the scope of FC-FS.

10.2 Frame delimiters

A frame delimiter is an Ordered Set that immediately precedes or follows the contents of a frame. Separate and distinct delimiters shall identify the start of a frame and the end of a frame. Frame delimiters shall be recognized when a single Ordered Set is detected. See 5.4 for details on Ordered Sets and Table 4 for the bit encodings for delimiters.

10.2.1 Start_of_Frame (SOF)

The Start_of_Frame (SOF) delimiter is an Ordered Set that immediately precedes the frame content. There are multiple SOF delimiters defined for the Fabric and for N_Port Sequence control. The SOF delimiter shall be transmitted on a word boundary. See 11.1 for more information on frame transmission.

10.2.2 End_of_Frame (EOF)

The End_of_Frame (EOF) delimiter is an Ordered Set that shall immediately follow the CRC and shall be transmitted on a word boundary. The EOF delimiter shall designate the end of the frame content and shall be followed by Idles. There are multiple EOF delimiters defined for the Fabric and for N_Port Sequence control. See 11.1 for more information on frame transmission.

10.3 Primitive Signals

A Primitive Signal is an Ordered Set designated by FC-FS to have special meaning. Primitive Signals shall be recognized when a single Ordered Set is detected. See 5.4 for details on Ordered Sets and Table 5 for the bit encodings for Primitive Signals.

10.3.1 Idle

An Idle is a Primitive Signal transmitted on the link to indicate an operational Port facility ready for frame transmission and reception. Idles shall be transmitted on the link during those periods of time when frames, R_RDY, or Primitive Sequences are not being transmitted. See 11.1 regarding frame transmission and 11.8 regarding frame reception.

10.3.2 Receiver_Ready (R_RDY)

The R_RDY Primitive Signal shall indicate that a single Class 1 connect-request (SOF_{c1}), Class 2, or Class 3 frame was received and that the interface buffer which received the frame is available for further frame reception. Validity of the frame content shall not be required. Transmission of the R_RDY Primitive Signal shall be preceded and followed by a minimum of two Idles and at the N_Port transmitter, there shall be a minimum of six Primitive Signals (R_RDY and Idles) between frames. The R_RDY Primitive Signal shall be received by the F_Port but shall not be passed through the Fabric, if present. Since R_RDY is not passed through the Fabric, there is no requirement for additional Idles for clock skew management (see 11.1). See 14.3.2.1 for a discussion regarding the use of R_RDY in flow control in conjunction with ACK frames.

10.3.3 BB_SCs

The BB_SCs Primitive Signal shall indicate that a predetermined number ($2^{BB_SC_N}$) of Class 1 connect-request (SOF_{c1}), Class 2, or Class 3 frames were sent since the previous BB_SCs was sent. See 17.6 and 17.7 for a discussion regarding the process of determining BB_SC_N. See 20.5.12 for a discussion regarding how BB_SCs is used.

10.3.4 BB_SCr

The BB_SCr Primitive Signal shall indicate that a predetermined number ($2^{BB_SC_N}$) of R_RDY Primitive Signals were sent since the previous BB_SCr was sent. See 17.6 and 17.7 for a discussion regarding the process of determining BB_SC_N. See 20.5.12 for a discussion regarding how BB_SCr is used.

10.4 Primitive Sequences

A Primitive Sequence is an Ordered Set that is transmitted repeatedly and continuously. Primitive Sequences are transmitted to indicate specific conditions within a Port (N_Port or F_Port) or conditions encountered by the Receiver logic of a Port. See Table 6 for bit encodings of Primitive Sequences.

Primitive Sequences shall be transmitted continuously while the condition exists. A detailed description of Port state changes relative to Primitive Sequence reception and transmission is shown in Annex E. When a Primitive Sequence is received and recognized, depending on the State of the Port, a corresponding Primitive Sequence or Idles shall be transmitted in response. The following Primitive Sequence Protocols are described:

- Link Initialization (see 10.6.2)
- Online to Offline (see 10.6.3)
- Link Failure (see 10.6.4)
- Link Reset (see 10.6.5).

Primitive Sequences transmitted from a N_Port to its locally attached F_Port of a Fabric shall remove a pending or existing Dedicated Connection. The locally attached F_Port shall respond appropriately to the Primitive Sequence received and shall notify the F_Port attached to the other connected N_Port which shall transmit the Link Reset Primitive to the other connected N_Port (i.e., initiate Link Reset Protocol).

If a Dedicated Connection does not exist, a Primitive Sequence transmitted by a N_Port shall be received and recognized by the locally attached F_Port, but not transmitted through the Fabric.

10.4.1 Primitive Sequence Recognition

Recognition of a Primitive Sequence shall require consecutive detection of three instances of the same Ordered Set without any intervening data indications from the Receiver logic (FC-1).

10.4.2 Not_Operational (NOS)

The Not_Operational Primitive Sequence shall be transmitted to indicate that the Port transmitting this Sequence has detected a Link Failure condition or is Offline, waiting for OLS to be received.

Link failure conditions shall be defined as:

- Loss of Synchronization for more than a timeout period (R_T_TOV) while not in the Offline State (see 6.1.1.2)
- Loss of Signal while not in the Offline State (see 6.1.1.2)
- Timeout (R_T_TOV) during the Link Reset Protocol (see 10.6.5)

NOTE - Loss of Signal is an optional indication related to FC-0 implementations, which have the ability to detect the presence of an input (see FC-PI for Loss of Light and for FC-0 signal_detect.indication).

10.4.3 Offline (OLS)

The Offline Primitive Sequence shall be transmitted to indicate that the Port transmitting this Sequence is:

- initiating the Link Initialization Protocol
- receiving and recognizing NOS
- entering the Offline State

A Port shall transmit the OLS Primitive Sequence for a minimum period of 5 ms before further actions are taken. A Port shall enter the Offline State in order to perform diagnostics or powerdown.

NOTE - The value of 5 ms is based on distances up to 10 km. If longer distances are employed in the future, the time should be increased accordingly.

10.4.4 Link Reset (LR)

The Link Reset Primitive Sequence shall be transmitted by a Port to initiate the Link Reset Protocol (see 10.6.5) or to recover from a Link Timeout (see 23.2.3). A N_Port supporting Class 1 Service may also transmit the LR Primitive Sequence when it is unable to determine its Connection status, a procedure known as Connection Recovery (see 22.8).

10.4.5 Link Reset Response (LRR)

The Link Reset Response Primitive Sequence shall be transmitted by a Port to indicate that it is receiving and recognizes the LR Primitive Sequence.

10.5 Port states

This section defines the possible states for a Port. For conditions which are not explicitly listed to cause state changes to occur, the Port shall remain within the current state.

10.5.1 Active State (AC)

A Port shall enter the Active State when it completes the Link Initialization Protocol (see 10.6.2) or the Link Reset Protocol (see 10.6.5). When a Port is in the Active State, it is able to transmit and receive frames and Primitive Signals (Idle and R_RDY). When a Primitive Sequence is received, the Port exits the Active State and shall enter another state based on the Primitive Sequence received:

- If LR is received and recognized, enter the LR Receive State (see 10.5.2.2)
- If LRR is received and recognized, enter the LRR Receive State (see 10.5.2.3). A Primitive Sequence protocol error is detected and counted in the LESB (see 23.8)
- If NOS is received and recognized, enter the NOS Receive State (see 10.5.3.2)
- If OLS is received and recognized, enter the OLS Receive State (see 10.5.4.2)
- If Idles are received and recognized, remain in the Active State.
- If R_RDY is received and recognized, remain in the Active State.
- If a frame is received, remain in the Active State.

If certain conditions are detected within the Port, the Port may exit the Active State and perform a Primitive Sequence Protocol:

- Link Initialization (see 10.6.2)
- Online to Offline (see 10.6.3)
- Link Failure (see 10.6.4)
- Link Reset (see 10.6.5)

10.5.2 Link Recovery State

When a Port is in Link Recovery State, it is in one of the following three substates.

10.5.2.1 LR Transmit State (LR1)

A Port shall enter the LR Transmit State to initiate the Link Reset Protocol. A N_Port supporting Class 1 or Class 6 may also enter the LR Transmit State when it is unable to determine its connection status. While in the LR Transmit State, the Port shall transmit the LR Primitive Sequence.

- While in the LR Transmit State, the Port shall respond as follows:
 - If LR is received and recognized, enter the LR Receive State (see 10.5.2.2).
 - If LRR is received and recognized, enter the LRR Receive State (see 10.5.2.3).
 - If NOS is received and recognized, enter the NOS Receive State (see 10.5.3.2).
 - If OLS is received and recognized, enter the OLS Receive State (see 10.5.4.2).
 - If Idle or R_RDY is received and recognized, remain in the LR Transmit State.
- If a Port remains in the LR Transmit State for a period of time greater than a timeout period (R_T_TOV), a Link Reset Protocol Timeout shall be detected which results in a Link Failure condition (enter the NOS Transmit State, see 10.5.3.1).
- If a Loss of Signal is detected, the Port shall enter the NOS Transmit State (see 10.5.3.1). Loss of synchronization need not be checked since item b will occur first.

Transmission of the Link Reset Primitive Sequence has different consequences based on Class:

Class 1

In Class 1, a pending or existing Dedicated Connection shall be removed and the end-to-end Credit associated with the connected N_Ports shall be reset to the Login value. The locally attached F_Port shall enter the LR Receive State and shall notify the F_Port attached to the other connected N_Port which shall transmit the Link Reset Primitive Sequence to the other connected N_Port (i.e., initiate Link Reset Protocol).

All Class 1 frame Sequences which are Active or Open in a N_Port when LR is received and recognized, or transmitted shall be abnormally terminated. When a N_Port is transmitting the Link Reset Primitive, all Class 1 frames received shall be discarded. See 22.8 for more discussion on Connection Recovery.

Class 2 and 3

Buffer-to-buffer Credit (associated with Class 1 (SOF_{c1}), Class 6 (SOF_{c1}), Class 2, and Class 3) within the N_Port or F_Port shall be reset to its Login value and a F_Port shall process or discard any Class 1 connect-request, Class 6 connect-request, Class 2, or Class 3 frames currently held in the receive buffer associated with the outbound fibre of the attached N_Port. Class 2 end-to-end Credit shall not be affected.

Class 6

In Class 6, a pending or existing Multicast Connection shall be removed and the end-to-end Credit associated with the connected N_Ports shall be reset to the Login value. The locally attached F_Port shall enter the LR Receive State and shall notify all other F_Ports attached to the other connected N_Ports which shall transmit the Link Reset Primitive Sequence to the other connected N_Ports (i.e., initiate Link Reset Protocol).

All Class 6 frame Sequences which are Active or Open in a N_Port when LR is received and recognized, or transmitted shall be abnormally terminated. When a N_Port is transmitting the Link Reset Primitive, all Class 1 frames received shall be discarded. See 22.8 for more discussion on Connection Recovery.

10.5.2.2 LR Receive State (LR2)

A Port shall enter the LR Receive State when it receives and recognizes the LR Primitive Sequence while it is not in the Wait for OLS or NOS Transmit State. While in the LR Receive State, the Port shall transmit the LRR Primitive Sequence.

- While in the LR Receive State, the Port shall respond as follows:
 - If LR is received and recognized, remain in the LR Receive State.
 - If LRR is received and recognized, enter the LRR Receive State (see 10.5.2.3)
 - If NOS is received and recognized, enter the NOS Receive State (see 10.5.3.2)
 - If OLS is received and recognized, enter the OLS Receive State (see 10.5.4.2)
 - If Idles are received and recognized, enter the Active State (see 10.5.1)
- If a Port remains in the LR Receive State for a period of time greater than a timeout period (R_T_TOV), a Link Reset Protocol Timeout shall be detected which results in a Link Failure condition (enter the NOS Transmit State, see 10.5.3.1)
- If a Loss of Signal is detected, the Port shall enter the NOS Transmit State (see 10.5.3.1). Loss of Synchronization need not be checked since item b will occur first.

Reception of the Link Reset Primitive Sequence has different consequences based on Class:

Class 1

In Class 1, a pending or existing Dedicated Connection shall be removed and the end-to-end Credit associated with the connected N_Ports shall be reset to the Login value. All Class 1 frame Sequences which are Active or Open in a N_Port when LR is received and recognized shall be abnormally terminated.

Class 2 and 3

A Port which receives and recognizes the Link Reset Primitive Sequence shall process or discard Class 2, or Class 3 frames currently held in its receive buffers. Buffer-to-buffer Credit (associated with Class 1 (SOFc1), Class 6 (SOFc1), Class 2, and Class 3) within the N_Port or F_Port shall be reset to its Login value.

Class 6

In Class 6, a pending or existing Multicast Connection shall be removed and the end-to-end Credit associated with the connected N_Ports shall be reset to the Login value. All Class 6 frame Sequences which are Active or Open in a N_Port when LR is received and recognized shall be abnormally terminated.

10.5.2.3 LRR Receive State (LR3)

A Port shall enter the LRR Receive State when it receives and recognizes the LRR Primitive Sequence while it is in the Active, LR Transmit, LR Receive, or OLS Receive State. While in the LRR Receive State, the Port shall transmit Idles.

- While in the LRR Receive State, the Port shall respond as follows:
 - If LR is received and recognized, enter the LR Receive State (see 10.5.2.2)
 - If LRR is received and recognized, remain in the LRR Receive State.
 - If NOS is received and recognized, enter the NOS Receive State (see 10.5.3.2).
 - If OLS is received and recognized, enter the OLS Receive State (see 10.5.4.2).
 - If Idles are received and recognized, the Port shall exit the LRR Receive State and enter the Active State (see 10.5.1).
- If a Port remains in the LRR Receive State for a period of time greater than a timeout period (R_T_TOV), a Link Reset Protocol Timeout shall be detected which results in a Link Failure condition (enter the NOS Transmit State, see 10.5.3.1).
- If a Loss of Signal is detected, the Port shall enter the NOS Transmit State (see 10.5.3.1). Loss of Synchronization need not be checked since item b will occur first.

NOTES

1. If a Port is in the NOS Transmit, NOS Receive, or OLS Transmit State, it ignores the LRR Primitive Sequence if it is received and recognized.
2. If a Port is in the Wait for OLS State, it enters the NOS Transmit State if the LRR Primitive Sequence is received and recognized.

10.5.3 Link Failure State

When a Port is in Link Failure State, it is one of the following substates.

10.5.3.1 NOS Transmit State (LF2)

A Port shall enter the NOS Transmit State when a Link Failure condition is detected. Upon entry into the NOS Transmit State, the Port shall update the appropriate error counter in the Link Error Status Block (see 23.8). Only one error per Link Failure event shall be recorded.

The Port shall remain in the NOS Transmit State while the condition, which caused the Link Failure, exists. While in the NOS Transmit State, the Port shall transmit the NOS Primitive Sequence.

When the Link Failure condition is no longer detected, the Port shall respond as follows:

- If LR is received and recognized, remain in the NOS Transmit State
- If LRR is received and recognized, remain in the NOS Transmit State
- If NOS is received and recognized, enter the NOS Receive State (see 10.5.3.2)
- If OLS is received and recognized, enter the OLS Receive State (see 10.5.4.2)
- If Idle is received and recognized, remain in the NOS Transmit State

Transmission of NOS by a N_Port to its locally attached F_Port of a Fabric shall remove a pending or existing Dedicated Connection. The locally attached F_Port shall respond by entering the NOS Receive State and shall notify the F_Port attached to the other connected N_Port which shall transmit the Link Reset Primitive to the other connected N_Port (i.e., initiate Link Reset Protocol).

If a Dedicated Connection does not exist, NOS transmission by a N_Port shall be received and recognized by the locally attached F_Port, but not transmitted through the Fabric. The F_Port shall respond by entering the NOS Receive State.

10.5.3.2 NOS Receive State (LF1)

A Port shall enter the NOS Receive State when it receives and recognizes the NOS Primitive Sequence. Upon entry into the NOS Receive State, the Port shall update the appropriate error counter in the Link Error Status Block (see 23.8). Only one error per Link Failure event is recorded. While in the NOS Receive State, the Port shall transmit the OLS Primitive Sequence.

- While in the NOS Receive State, the Port shall respond as follows:
 - If LR is received and recognized, enter the LR Receive State (see 10.5.2.2)
 - If LRR is received and recognized, remain in the NOS Receive State
 - If NOS is received and recognized, remain in the NOS Receive State
 - If OLS is received and recognized, enter the OLS Receive State (see 10.5.4.2)
 - If Idle is received and recognized, remain in the NOS Receive State.
- If a Link Failure condition is detected due to Loss of Signal or Synchronization, the Port shall enter the NOS Transmit State (see 10.5.3.1)
- A timeout (R_T_TOV) period is started when NOS is no longer recognized and no other events occur which cause a transition out of the NOS Receive State. If the timeout period expires, the Port shall enter the NOS Transmit State (see 10.5.3.1).

10.5.4 Offline State

While in the Offline State, a Port shall not record receiver errors such as loss of synchronization. A Port shall enter the Offline State under the following conditions:

- after power-up, or internal reset, before the Link Initialization Protocol is complete
- after transmission of the first OLS Ordered Set
- after reception and recognition of the OLS Primitive Sequence

When a Port is in the Offline state, it is in one of the following substates.

10.5.4.1 OLS Transmit State (OL1)

A Port shall enter the OLS Transmit State in order to:

- perform Link Initialization using the Link Initialization Protocol (see 10.6.2) in order to exit the Offline State
- transition from Online to Offline using the Online to Offline Protocol (see 10.6.3)

When the Port enters the OLS Transmit State, it shall transmit OLS for a minimum time of 5 ms. After that period of time has elapsed, it shall proceed according to the steps defined by the Link Initialization Protocol or the Online to Offline Protocol based on the reason the Port entered the OLS Transmit State. While a Port is attempting to go Online, if no Primitive Sequence is received or event detected which causes the Port to exit the OLS Transmit State after a timeout period (R_T_TOV), the Port shall enter the Wait for OLS State.

Transmission of OLS by a N_Port to its locally attached F_Port of a Fabric shall remove a pending or existing Dedicated Connection. The locally attached F_Port shall respond by entering the OLS Receive State and shall notify the F_Port attached to the other connected N_Port which shall transmit the Link Reset Primitive Sequence to the other connected N_Port (i.e., initiate Link Reset Protocol).

If a Dedicated Connection does not exist, OLS transmission by a N_Port shall be received and recognized by the locally attached F_Port, but not transmitted through the Fabric. The F_Port shall respond by entering the OLS Receive State.

10.5.4.2 OLS Receive State (OL2)

A Port shall enter the OLS Receive State when it receives and recognizes the OLS Primitive Sequence. While in the OLS Receive State, the Port shall transmit the LR Primitive Sequence.

- While in the OLS Receive State, the Port shall respond as follows:
 - If LR is received and recognized, enter the LR Receive State (see 10.5.2.2)
 - If LRR is received and recognized, enter the LRR Receive State (see 10.5.2.3)
 - If NOS is received and recognized, enter the NOS Receive State (see 10.5.3.2)
 - If OLS is received and recognized, remain in the OLS Receive State.
 - If Idle is received and recognized, remain in the OLS Receive State.
- If Loss of Synchronization is detected for more than a timeout period (R_T_TOV), or Loss of Signal is detected, the Port shall enter the Wait for OLS State (see 10.5.4.3). Detection of Loss of Signal or Loss of Synchronization shall not be counted as a Link Failure event in the Link Error Status Block.

10.5.4.3 Wait for OLS State (OL3)

A Port shall enter the Wait for OLS State when it detects Loss of Signal or Loss of Synchronization for more than a timeout period (R_T_TOV) while it is in the OLS Receive State, or while it is in the OLS Transmit State at an appropriate time in the Link Initialization Protocol (see 10.6.2).

Upon entry into the Wait for OLS State, the Port shall transmit the NOS Primitive Sequence. While in the Wait for OLS State, the Port shall respond as follows:

- If LR is received and recognized, enter the NOS Transmit State (see 10.5.3.1)
- If LRR is received and recognized, enter the NOS Transmit State (see 10.5.3.1)
- If NOS is received and recognized, enter the NOS Receive State (see 10.5.3.2)

- If OLS is received and recognized, enter the OLS Receive State (see 10.5.4.2)
- If Idle is received and recognized, remain in the Wait for OLS State

10.6 Primitive Sequence Protocols

The interchange of frames is basically asynchronous in nature. The Data protocols defined in FC-FS allow multiple frames in transit in each direction at the same time. When conditions cause state changes to occur within a N_Port or F_Port, Primitive Sequence Protocols provide a fully interlocked and acknowledged mechanism to recover from those conditions.

The Link recovery hierarchy is shown in Figure 65.

10.6.1 Primitive Sequence meanings

Table 7 provides a summary of the defined Primitive Sequences, their meaning, and the corresponding response when received.

Table 7 - Primitive Sequence summary		
Trans	Meaning	Response
NOS	Not-operational - Link Failure	OLS
OLS	Offline Sequence - Internal failure in Port - Transmitter may power-down, perform diagnostics, or perform initialization. - Receiver shall ignore Link errors or Link failure	LR
LR	Link Reset - Remove Class 1 Connection, - Reset F_Port, or - OLS recognized	LBR
LRR	Link Reset Response - Link Reset recognized	Idles
Idles	Operational Link	Idles

10.6.2 Link Initialization

Link initialization is required after a Port has been powered-on, has been internally reset, or has been Offline. The Ports involved may be a N_Port and a F_Port, or two N_Ports.

The following series of events defines the Link Initialization Protocol from the perspective of the Port that initiates the Protocol. While in the Offline State, NOS Reception or Link Failure conditions which are detected shall not be recorded as Link Failure events in the Link Error Status Block. The following series of events defines the Link Initialization Protocol.

- Enter the OLS Transmit State and transmit OLS for a minimum period of 5 ms. After the period has elapsed, the Port shall respond as follows:
 - If LR is received and recognized, enter the LR Receive State (see 10.5.2.2)
 - If LRR is received and recognized, remain in the OLS Transmit State

- If OLS is received and recognized, enter the OLS Receive State (see 10.5.4.2)
 - If NOS is received and recognized, enter the NOS Receive State (see 10.5.3.2)
 - If Idle is received and recognized, remain in the OLS Transmit State
 - If Loss of Synchronization is detected for more than a timeout period (R_T_TOV), or Loss of Signal is detected, enter the Wait for OLS State (see 10.5.4.3)
- If no Primitive Sequence is received or event detected which causes the Port to exit the OLS Transmit State after a timeout period (R_T_TOV), the Port shall enter the Wait for OLS State

At the end of the Link Initialization Protocol both Ports shall be transmitting and receiving Idles (Active State).

10.6.3 Online to Offline

A Port is Online when it is in the Active State. A Port shall perform the Online to Offline Protocol to enter the Offline State. The Offline State shall be entered prior to power-down (or at power-down), or performing diagnostics. To exit the Offline State, a Port shall perform the Link Initialization Protocol.

The following series of events shall define the Online to Offline Protocol from the perspective of the Port that intends to go Offline:

- a) The Port shall enter the OLS Transmit State.
- b) The Port shall transmit OLS for a minimum time of 5 ms.
- c) After transmitting OLS for 5 ms, the Port shall be Offline and may perform diagnostic procedures, turn off its transmitter, or transmit any signal (excluding Primitive Sequences) without errors being detected by the other attached Port. While in the Offline State, a Port may also power-down.

NOTE - After entering the OLS Transmit State and transmitting OLS for a minimum of 5 ms, the Port may then do anything which will not cause the attached Port to leave the OLS Receive State or Wait for OLS State unless intended to do so.

To exit the Offline State, a Port shall perform the Link Initialization Protocol.

10.6.4 Link Failure

The Link Failure Protocol shall be performed after a Port has detected either a Loss of Synchronization for a period of time greater than a timeout period (R_T_TOV), or Loss of Signal while not in the Offline State.

The Link Failure Protocol shall also be performed after a Link Reset Protocol timeout error is detected (see 10.6.5). The following series of events defines the Link Failure Protocol:

- a) The Port shall transmit NOS and enter the NOS Transmit State while the Link Failure condition in the receiver or Port exists. While in the NOS Transmit State, after the Receiver is operational, the Port shall respond to Primitive Sequences received.
 - If NOS is received and recognized, enter the NOS Receive State.
 - If OLS is received and recognized, enter the OLS Receive State.
- b) In the OLS Receive State, transmit LR.
 - If LR is received and recognized, enter the LR Receive State.
 - If LRR is received and recognized, enter the LRR Receive State.
- c) In the NOS Receive State, transmit OLS.

- If LR is received and recognized, enter the LR Receive State.
 - If OLS is received and recognized, enter the OLS Receive State.
- d) In the LR Receive State, transmit LRR.
- If Idles are received and recognized, exit the LR Receive State and enter the Active State.
- e) In the LRR Receive State, transmit Idles.
- If Idles are received and recognized, exit the LRR Receive State and enter the Active State.
- f) At the end of the Link Failure Protocol both Ports shall be transmitting and receiving Idles (Active State).

10.6.5 Link Reset

The Link Reset Protocol shall be performed following a Link timeout. If a Port performs step a, b, or c without receiving an appropriate response within the timeout period (R_T_TOV), a Link Failure shall be detected and the Link Failure Protocol shall be performed.

The Link Reset Protocol shall comply with the following series of events described from the perspective of the Port initiating the Protocol.

- a) The Port shall transmit LR and enter the LR Transmit State.
- If LRR is received and recognized, enter the LRR Receive State.
 - If LR is received and recognized, enter the LR Receive State.
- b) In the LR Receive State, transmit LRR.
- If Idles are received and recognized, exit the LR Receive State and enter the Active State.
- c) In the LRR Receive State, transmit Idles.
- If Idles are received and recognized, exit the LRR Receive State and enter the Active State.
- d) If other conditions or Primitive Sequences are recognized, the Port shall respond according to the definitions associated with its current State as specified in 10.4.
- e) At the end of the Link Reset Protocol both Ports shall be transmitting and receiving Idles (Active State).

11 Frame formats

All FC-2 frames follow the general frame format as shown in Figure 21. A FC-2 frame is composed of a Start_of_Frame delimiter, frame content, and an End_of_Frame delimiter. The frame content is composed of a Frame_Header, Data_Field, and CRC. Unless otherwise specified, the term frame refers to a FC-2 frame within FC-FS.

11.1 Frame transmission

Frame transmission shall be performed by inserting a frame immediately following a sequence of Idles. Idles shall be transmitted immediately upon completion of the frame.

At the N_Port transmitter there shall be a minimum of six Primitive Signals (Idles and R_RDY) between frames. A minimum of two Idles shall be guaranteed to precede the start (SOF) of each frame received by a destination N_Port. The Fabric may insert or remove Idles as long as the destination receives at least two Idles preceding each frame. See 11.8 for a discussion on frame reception.

11.2 Start_of_Frame (SOF)

The Start_of_Frame delimiter is an Ordered Set that immediately precedes the frame content. There are multiple SOF delimiters defined for the Fabric and for N_Port Sequence control. The SOF delimiter shall be transmitted on a word boundary. Table 33 and Table 35 specify allowable delimiters by Class for Data and Link_Control frames. The bit encodings for the SOF delimiters are defined in Table 4.

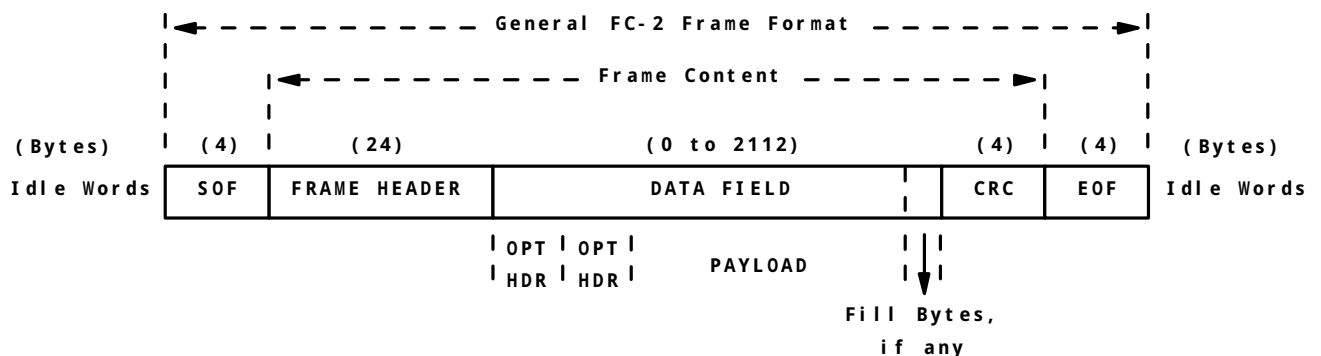


Figure 21 - FC-2 general frame format

11.2.1 Start_of_Frame Connect Class 1 or 6 (SOF_{c1})

SOF_{c1} shall be used to request a Class 1 or Class 6 Dedicated Connection. The frame Data Field size is limited to the maximum size specified by the destination N_Port or by the Fabric, if present, whichever size is smaller. The maximum Data Field size is determined during the Login procedure. This delimiter also identifies the start of the first Sequence (i.e., implicit SOF_{i1}).

11.2.2 Start_of_Frame Initiate (SOF_{ix})

A Sequence shall be initiated and identified by using the SOF_{ix} delimiter in the first frame. SOF_{ix} is used to indicate SOF_{i1}, SOF_{i2}, or SOF_{i3}. The following three delimiters identify the start of a Sequence based on Class of Service.

11.2.2.1 Start_of_Frame Initiate Class 1 or 6 (SOF_{i1})

The first Sequence of a Dedicated Connection is initiated with SOF_{c1}. After a Class 1 or Class 6 Connection is established, all subsequent Sequences within that Dedicated Connection shall be initiated with SOF_{i1}.

11.2.2.2 Start_of_Frame Initiate Class 2 (SOF_{i2})

The SOF_{i2} shall be used on the first frame to initiate a Sequence for Class 2 Service.

11.2.2.3 Start_of_Frame Initiate Class 3 (SOF_{i3})

The SOF_{i3} shall be used on the first frame to initiate a Sequence for Class 3 Service.

11.2.3 Start_of_Frame Normal (SOF_{nx})

The following three delimiters identify the start of all frames other than the first frame of a Sequence based on Class of Service. SOF_{nx} is used to indicate SOF_{n1}, SOF_{n2}, or SOF_{n3}.

11.2.3.1 Start_of_Frame Normal Class 1 or 6 (SOF_{n1})

The SOF_{n1} shall be used for all frames except the first frame of a Sequence for Class 1 or Class 6 Service.

11.2.3.2 Start_of_Frame Normal Class 2 (SOF_{n2})

The SOF_{n2} shall be used for all frames except the first frame of a Sequence for Class 2 Service.

11.2.3.3 Start_of_Frame Normal Class 3 (SOF_{n3})

The SOF_{n3} shall be used for all frames except the first frame of a Sequence for Class 3 Service.

11.2.4 Start_of_Frame Fabric (SOF_f)

If a N_Port receives a Fabric_Frame, indicated by SOF_f, the Fabric_Frame shall be discarded and ignored. See 11.9 for a description of a Fabric_Frame.

11.3 Frame_Header

The Frame_Header shall be the first field of the frame content and shall immediately follow the SOF for all frames. The Frame_Header shall be transmitted on a word boundary. The Frame_Header is used by the Link_Control facility to control link operations, control device protocol transfers, and detect missing or out of order frames. The Frame_Header is described in detail in clause 12.

11.4 Data Field

The Data Field shall follow the Frame_Header. Data Fields shall be aligned on word boundaries and shall be equal to a multiple of four bytes in length (including zero length). Two Frame_Types are defined based on the value of bits 31-28 in the R_CTL field of the Frame_Header.

The two Frame_Types are

- FT_0 (Data Field = 0 bytes)

When the R_CTL field bits 31-28 are set to 1100b, a FT_0 frame (Link_Control) shall be specified which has a Data Field length of zero.

- FT_1 (Data Field = 0 to 2 112 bytes)

When the R_CTL field bits 31-28 are set to values other than 1100b, a FT_1 frame (Data frame) shall be specified whose Data Field size is a multiple of four bytes and ranges in size from 0 to 2 112 bytes. If the ULP supplies a Payload which is not divisible by 4, the remaining bytes (up to 3) shall contain fill bytes as specified in the F_CTL field in Table 21. However, fill bytes shall only be meaningful, and recognized as present, on the last Data frame of a series of consecutive Data frames of a single Information Category within a single Sequence.

The Data Field in FT_1 frames may contain optional headers, as defined by the DF_CTL field, as well as the Payload. Optional Data Field headers are described in clause 13.

A discussion of FT_0 (Link_Control) and FT_1 (Data) frames is found in clause 14.

11.5 CRC

The Cyclic Redundancy Check (CRC) is a four byte field that shall immediately follow the Data Field and shall be used to verify the data integrity of the Frame_Header and Data Field. SOF and EOF delimiters shall not be included in the CRC verification. The CRC field shall be calculated on the Frame_Header and Data Field prior to encoding for transmission and after decoding upon reception. The CRC field shall be aligned on a word boundary.

For the purpose of CRC computation, the bit of the word-aligned, four-byte field that corresponds to the first bit transmitted is the highest order bit. See Figure C.1. Bit 24 of the first word of the Frame_Header is the first bit of the transmission word transmitted by the transmitter (see 5.2.1).

Fiber Distributed Data Interface (FDDI) - Media Access Control (MAC) specifies the FC-FS CRC. See Annex C for further discussions of the CRC. The CRC shall use the following 32-bit polynomial:

$$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

11.6 End_of_Frame (EOF)

The End_of_Frame (EOF) delimiter is an Ordered Set that immediately follows the CRC. The EOF delimiter shall designate the end of the frame content and shall be followed by Idles. Table 33 and Table 35 specify allowable delimiters by Class for Data and Link_Control frames. There are three categories of EOF delimiters. One category of delimiter shall indicate that the frame is valid from the sender's perspective and potentially valid from the receiver's perspective.

The second category shall indicate that the frame content is invalid. This category shall only be used by a F_Port, which receives a complete frame and decodes it before forwarding that frame on to another destination.

The third category shall indicate the frame content is corrupted and the frame was truncated during transmission. The third category shall be used by both N_Ports and F_Ports to indicate an internal malfunction, such as a transmitter failure, which does not allow the entire frame to be transmitted normally.

The bit encodings for the EOF delimiters are defined in Table 4.

11.6.1 Valid frame content

Three types of End_of_Frame delimiter are defined to indicate that the frame content is valid from the sender's perspective and potentially valid from the receiver's perspective. Table 33 and Table 35 specify allowable delimiters by Class.

11.6.1.1 End_of_Frame Terminate (EOFt)

The EOFt shall indicate that the Sequence associated with this SEQ_ID is complete. EOFt or EOFdt shall be used to properly close a Sequence without error.

11.6.1.2 End_of_Frame Disconnect_Terminate (EOFdt)

The EOFdt shall remove a Dedicated Connection through a Fabric, if present. The EOFdt shall also identify the last ACK of a Sequence and indicate that all Class 1 or Class 6 Sequences associated with this S_ID are terminated. Open Class 1 or Class 6 Sequences, other than the SEQ_ID specified in the frame containing EOFdt, shall be abnormally terminated and may require Sequence recovery on a ULP protocol-dependent basis.

11.6.1.3 End_of_Frame Normal (EOFn)

The EOFn shall identify the end of frame when one of the other EOF delimiters indicating valid frame content is not required.

11.6.2 Invalid frame content

There are two EOF delimiters, which indicate that the frame content is invalid. If a frame is received by a facility internal to a Fabric and an error is detected within the frame content, the frame may be passed to the destination N_Port with a modified EOF to indicate that an error was previously detected. Error detection in the Frame Content by the Fabric is optional.

Errors such as code violation or CRC errors are examples of detectable frame errors. For example, if the Fabric decodes a frame into bytes and one or more transmission characters cannot be validly decoded into bytes, these bytes shall be filled with valid data bytes, a new CRC shall not be generated, and the appropriate EOF delimiter shall replace the original EOF delimiter.

When a frame is received with an EOF delimiter, which indicates that the frame content is invalid, the receiver shall not report the detection of an invalid frame. The invalid frame condition shall be reported by the entity, which replaces the EOF delimiter that indicates invalid frame content. The receiving N_Port, at its discretion, may report the event of receiving a frame with one of these delimiters.

NOTES

1. Errors are counted at the point at which they are detected. Events may also be reported at the point at which they are recognized.
2. Due to the importance of the EOFdt delimiter in Class 1 and Class 6, it is very desirable for the Fabric to forward a frame, which has either EOFdt or EOFdti.

The following EOF delimiters shall indicate that the frame contents are invalid (see 11.8.2) regarding reception of invalid frames).

11.6.2.1 End_of_Frame Disconnect_Terminate_Invalid (EOFdti)

The EOFdti shall replace a recognized EOFdt delimiter on a frame with invalid frame content. The EOFdti shall remove a Dedicated Connection through a Fabric, if present. It shall also indicate that all Class 1 Open Sequences associated with the Connected N_Port are abnormally terminated and may require Sequence recovery on a ULP protocol-dependent basis.

The frame containing the EOFdti shall be processed by the receiver in the following manner:

- no response frame shall be transmitted
- the Data Field may be used at the receiver's discretion (see 11.8.2)
- the Dedicated Connection to the receiver is removed

11.6.2.2 End_of_Frame Invalid (EOF_{ni})

The EOF_{ni} shall replace an EOF_n or EOF_t, indicating that the frame content is invalid.

The receiver shall process the frame containing the EOF_{ni} in the following manner:

- no response frame shall be transmitted
- the Data Field may be used at the receiver's discretion (see 11.8.2)

11.6.3 End_of_Frame Abort (EOF_a)

The EOF_a shall terminate a partial frame due to a malfunction in a link facility during transmission. The frame shall end on a word boundary and shall be discarded by the receiver without transmitting a reply. The transmitter shall retransmit the aborted frame with the same sequence count (SEQ_CNT) up to its ability to retry, which may be zero.

An invalid EOF (EOF_{dti} or EOF_{ni}) delimiter may be changed to an EOF_a delimiter under the conditions specified for EOF_a.

EOF_a delimiters shall not be changed to an invalid EOF delimiter under any conditions.

11.7 Frame field order

The frame content shall be transmitted sequentially following the SOF delimiter as an ordered byte stream within the definition of the frame as specified in Figure 21, Figure 23, and Figure 24 until the EOF delimiter is transmitted.

Figure 22 relates the frame format to the ordered byte stream transferred from the Upper Level Protocol (or FC-4) and transmitted by FC-FS.

		Bits							
Word		31	24	23	16	15	08	07	00
SOF		K28.5		Dxx.x		Dxx.x		Dxx.x	
		A0		A1		A2		A3	
0		R_CNTL		D_ID					
		B0		B1		B2		B3	
1		rrrr	rrrr	S_ID					
		B4		B5		B6		B7	
2		TYPE		F_CTL					
		B8		B9		B10		B11	
3		SEQ_ID		DF_CTL		SEQ_CNT			
		B12		B13		B14		B15	
4		OX_ID				RX_ID			
		B16		B17		B18		B19	
5		Parameter							
		B20		B21		B22		B23	
6		Payload							
		B24		B25		B26		B27	
7		Payload							
		B28		B29		B30		B31	
8		CRC							
		B32		B33		B34		B35	
EOF		K28.5		Dxx.x		Dxx.x		Dxx.x	
		C0		C1		C2		C3	

Figure 22 - Frame byte order

A frame shall be transmitted in the following byte order:

A0, A1, A2, A3, B0, B1, B2, B3, B4 ...

B32, B33, B34, B35, C0, C1, C2, C3.

If there were one byte of fill in the Payload of this frame, the fill byte is B31. With no optional header present, the Relative Offset (Parameter Field) shall be specified as follows:

Relative Offset + 0 specifies B24

Relative Offset + 3 specifies B27

Relative Offset + 4 specifies B28

For a Relative Offset of decimal 1024 (hex '00 00 04 00') the Parameter Field shall be specified as:

B20, B21, B22, B23 = hex '00 00 04 00'.

See Figure C.1 and Figure C.2.

11.8 Frame reception

The following list specifies frame reception rules.

- a) Data bytes received outside the scope of a delimiter pair (SOF and EOF) shall be discarded.
- b) Frame reception shall be started by recognition of an SOF delimiter.
- c) Recognition of SOFc1, SOFx2, or SOFx3, shall be responded to by transmission of the R_RDY Primitive Signal when the inter face buffer is available.
- d) Detection of a code violation after frame reception is started but before frame reception is terminated shall be identified as an Invalid Transmission Word within the frame.
- e) Frame reception shall continue until an Ordered Set, or a Link Failure is detected.
- f) If the Frame_Content between the SOF and EOF delimiters exceeds the maximum allowable Data Field size as specified by Login, a N_Port may consider the frame invalid and discard Data Field bytes as received. However, an Ordered Set or Link Failure shall still terminate frame reception. A N_Port is also allowed to receive the frame and if valid, it shall respond with a P_RJT with a length error indicated in the reason code.
- g) In either process or discard policy, if an EOFa terminates frame reception, the entire frame shall be discarded, including the Frame_Header and Data Field.
- h) If an EOFdt or EOFdti terminates frame reception, a pending or existing Dedicated Connection shall be recognized to be removed without regard to frame validity.

11.8.1 Frame validity

The following list of items determines whether a frame is considered valid or invalid.

- a) If the Ordered Set terminating frame reception is not recognized as an EOF delimiter, the frame shall be considered invalid.
- b) If the EOF delimiter is EOFni, EOFdti, or EOFa, the frame shall be considered invalid.
- c) If the Frame_Content between the SOF and EOF delimiters is not a multiple of four bytes, the frame shall be considered invalid.
- d) If the EOF delimiter is EOFn, EOFt, or EOFdt and no Invalid Transmission Words have been detected during frame reception for a frame with a multiple of four byte words, the CRC field shall be used. If the CRC is valid, the frame shall be considered valid and nor-

mal processing for valid frames shall be performed. If the CRC is invalid, the frame shall be considered invalid.

During normal processing of valid frames, errors may be detected which are rejectable in Class 1, 2 and 6 using the P_RJT Link_Response frame (see 14.3.3.3). P_RJT frames shall not be transmitted for invalid frames. If a rejectable error condition or a busy condition is detected for a valid Class 3 frame, the frame shall be discarded.

When errors such as Invalid Transmission Word and Invalid CRC are detected, the event count in the Link Error Status Block shall be updated (see 23.8). If delimiter usage does not follow allowable delimiters by class as specified in Table 33 and Table 35, a valid frame shall be considered rejectable.

11.8.2 Invalid frame processing

If a N_Port is able to determine that an invalid frame is associated with an Exchange which is designated as operating under Process policy, the N_Port may process and use the Data Field at its discretion, otherwise, the entire invalid frame shall be discarded.

NOTE - When a frame is corrupted, it is not known if the Frame_Header is correct. The XIDs, SEQ_ID, SEQ_CNT, and Parameter fields may not contain reliable information. The error may cause a misrouted frame to have a D_ID, which appears to be correct. Such a frame may be used under very restricted conditions.

11.9 Fabric_Frame

A Fabric_Frame may be used by the Fabric for intrafabric communication. A Fabric_Frame is composed of a SOF_f delimiter, frame header and content, and an EOF_n delimiter. The Fabric_Frame header and content is not defined within FC-FS.

12 Frame Header

12.1 Introduction

The Frame_Header shall be subdivided into fields as shown in Figure 23.

Bits	
Word	3124231615080700
0	R_CTLD_ID
1	CS_CTLS_ID
2	TYPEF_CTL
3	SEQ_IDDF_CTLSEQ_CNT
4	OX_IDRX_ID
5	Parameter

Figure 23 - Frame_Header

The Frame_Header shall be the first field of the frame content and immediately follow the SOF delimiter. The Frame_Header shall be transmitted on a word boundary. The Frame_Header is used to control link operations, control device protocol transfers, and detect missing or out of order frames.

12.1.1 Frame identification

A single frame is uniquely identified by S_ID, D_ID, SEQ_ID, SEQ_CNT, and Sequence Context. The OX_ID and RX_ID fields (generally referred to as X_ID) may be used by a N_Port to provide a locally assigned value which may be used in place of S_ID, D_ID, and SEQ_ID to identify frames in a non-streamed Sequence or when only one Sequence is Open. When Sequences are streamed, or more than one Sequence is Open, the X_ID field may be used in place of the S_ID and D_ID to identify the N_Port pair associated with a specific frame. The X_ID field may also be used in conjunction with S_ID, D_ID, or SEQ_ID to relate one or more Sequences to actions initiated by Upper Level Protocols.

When one or more Sequences are aborted using the Abort Sequence Protocol (see 23.7.1.1), a Recovery_Qualifier range is identified by the Sequence Recipient which consists of S_ID, D_ID, OX_ID, RX_ID in combination with a range of SEQ_CNT values (low and high). In Class 2 and 3, the Recovery_Qualifier range shall be used by the Sequence Initiator to discard ACK and Link_Response frames and by the Sequence Recipient to discard Data frames. If a Recovery_Qualifier is used in Class 2 or 3, a Reinstate Recovery Qualifier Link Service request shall be performed after an R_A_TOV timeout period has expired (see 15.2.1).

12.1.2 Sequence identification

The set of IDs described previously (S_ID, D_ID, OX_ID, RX_ID, and SEQ_ID) is referred to as the Sequence_Qualifier throughout the remainder of FC-FS. A N_Port implementation makes use of these IDs in an implementation-dependent manner to uniquely identify Active and Open Sequences (see 18.6.1).

NOTE - A N_Port's freedom to assign a SEQ_ID is based on Sequence Context (Initiator or Recipient). This may affect the means by which a N_Port implementation chooses to uniquely identify Sequences.

A given N_Port Originator may choose to provide frame tracking outside of the signaling protocol of FC-FS (FC-2). This is indicated by setting the OX_ID to hex 'FF FF'. This implies that the Exchange Originator shall only have one Exchange Active with a given destination N_Port. If a N_Port chooses an alternative frame tracking mechanism outside the scope of FC-2, it is still responsible for providing proper SEQ_ID and SEQ_CNT values. In addition, it shall return the RX_ID assigned by the Responder.

A given N_Port Responder may choose to provide frame tracking outside of the signaling protocol of FC-FS (FC-2). This is indicated by setting the RX_ID to hex 'FF FF'. If a N_Port chooses an alternative frame tracking mechanism outside the scope of FC-2, it is still responsible for providing proper SEQ_ID and SEQ_CNT values. In addition, it shall return the OX_ID assigned by the Originator.

12.2 Routing Control (R_CTL)

Routing Control (Word 0, Bits 31-24) is a one-byte field that contains routing bits and information bits to categorize the frame function.

When the R_CTL field is used in combination with the TYPE field (Word 2, bits 31-24), it provides a N_Port with assistance in frame routing, data routing, or addressing as summarized in Table 8.

- **Bits 31-28 - Routing Bits**

- 0000 = FC-4 Device_Data frame
- 0010 = Extended Link_Data frame
- 0011 = FC-4 Link_Data frame
- 0100 = Video_Data frame
- 1000 = Basic Link_Data frame
- 1100 = Link_Control frame
- Others = Reserved

Routing bits differentiate frames based on function or service within a N_Port or F_Port. FC-4 Device_Data frames contain Payload information related to a specific Upper Level Protocol. Video_Data frames contain Payload information directed to a video buffer without transferring the Data to main store.

Link_Data frames contain request and reply commands that are directed to the N_Port. There are three kinds of Link_Data frames:

- Basic Link_Data which contain commands in the information bits of R_CTL (bits 27-24) which are part of a Sequence of frames
- Extended Link_Data frames which are directed to the N_Port in order to provide N_Port Extended Link Services, such as Login, which are common to multiple FC-4
- FC-4 Link_Data frames which are directed to the N_Port in order to provide FC-4 Services to assist in the processing of FC-4 Device_Data frames

See clause 14 for specification of Data (FC-4 Device_Data and Video_Data) and Link_Control frames. See clause 15 for specification on Basic Link Services, Extended Link Services, and FC-4 Link Services.

- **Bits 27-24 - Information field**

The interpretation of the Information field is dependent on the Routing (bits 31-28) field value. For Routing = 1000 and 1100, the Information field contains a command. For all other R_CTL values, the Information field specifies the Common Information Categories

specified in Table 9. When the Routing bits indicate a Link_Control frame, the command code is specified in the Information field (bits 27-24) as shown in Table 34. Link_Control frames are described in 14.3.

Table 8 - R_CTL - Type Code summary				
R_CTL Word 0, bits 31-24		TYPE Word 2, bits 31-24	Payload	Comments
Bits 31-28 Routing	Bits 27-24 Information			
0000	0000-0111 (see Table 9)	FC-4 Device_Data (see Table 20)	Information Categories 0 to 15	Device_Data frames
0010	0010 0011 (see Table 9)	Extended Link_Data (see Table 18)	Request Reply	Extended Link Services
0011	0010 0011 (see Table 9)	FC-4 Link_Data (see Table 20)	Request Reply	FC-4 Link_Data
0100	0000-0111 (see Table 9)	Video_Data (see Table 19)		Video_Data
1000	Command (see Table 43)	Basic Link_Data (see Table 18)	12 byte max	Basic Link Service
1100	Command (see Table 34)	Reserved/ Reason Code	none	Link_Control frame

When the Routing bits indicate a Basic Link_Data frame, the command code shall be specified in the Information field (bits 27-24) as shown in Table 43.

When the Routing bits indicate Extended Link_Data or FC-4 Link_Data, the Information Category shall indicate Unsolicited Control for a request Sequence and Solicited Control for a reply Sequence as encoded in Table 9.

When the Routing bits indicate FC-4 Device_Data or Video_Data, the Information Category bits are available to each FC-4 TYPE to indicate data type or data control information relating to the information within the Payload portion of the Data Field as shown in Table 9. When the Routing bits indicate Device_Data, it indicates that the Payload of this frame may be processed by a level above the N_Port that received the frame. The current Types for FC-4 Device_Data and FC-4 Link_Data frames are shown in Table 20.

Table 9 - Information Categories	
Bit Value 27-24	Description
0000	Uncategorized information
0001	Solicited Data
0010	Unsolicited Control
0011	Solicited Control
0100	Unsolicited Data
0101	Data Descriptor

0110	Unsolicited Command
0111	Command Status
Others	Unspecified

Category bits shall specify an Information Category for the Payload of a single frame. A series of consecutive frames, which specify the same Information Category within a single Sequence, shall be considered a single instance of the Information Category. The fill bits (F_CTL bits 1-0) shall only be meaningful on the last frame of a single instance of an Information Category (see 12.6). Only one instance of an Information Category shall be allowed in a single Sequence. Multiple Information Categories are allowed in the same Sequence as a Login option (see 17.6.8.4). The format of the Payload for Data_Descriptor category is shown in Table 10. The format of the Unsolicited Command Payload is FC-4 dependent. The format of the Payload for Command Status is FC-4 dependent. The format of the Payload of other categories is FC-4 specific.

Table 10 - Data Descriptor Payload	
Item	Size-Bytes
Offset of Data being transferred	4
Length of Data being transferred	4
Reserved	4
Other optional information (FC-4 dependent)	max

12.3 Address identifiers (D_ID, S_ID)

Each N_Port shall have a native N_Port Identifier, which is unique within the address domain of a Fabric. In addition, a N_Port may optionally have one or more alias address identifiers, which may be shared across multiple N_Ports. For example, alias addressing may be used to implement a Hunt Group.

A N_Port Identifier of binary zeros indicates that a N_Port is unidentified. When a N_Port transitions from the Offline State to the Online State, its N_Port Identifier shall be unidentified. While a N_Port is unidentified, it shall

- promiscuously accept frames with any Destination Identifier value
- not Reject (P_RJT) a frame with a reason code of Invalid D_ID
- Reject (P_RJT) frames other than Basic and Extended Link Service with a reason code of Login Required.

A N_Port determines its native N_Port address Identifier by performing the Login Procedure as specified in **Error! Reference source not found.** During the Login Procedure a N_Port may be assigned a native N_Port Identifier or it may determine its own native N_Port Identifier.

Address identifiers in the range of hex 'FF FF F0' to 'FF FF FE' are well-known and reserved for use as shown in Table 11. The address identifier of hex 'FF FF FF' is reserved as a broadcast address.

Table 11 - Well-known address identifiers	
Address Value (hex)	Description
FFFFFF0 to FFFFF4	Reserved
FFFFFF5	Multicast Server (see 25.2.3)
FFFFFF6	Clock Synchronization Server (see clause 32)
FFFFFF7	Security Key Distribution Server (see FC-GS-2)
FFFFFF8	Alias Server (see 26.2)
FFFFFF9	Quality of Service Facilitator - Class 4 (QoS) (see 28.2.4.1)
FFFFFFA	Management Server
FFFFFFB	Time Server
FFFFFFC	Directory Server
FFFFFFD	Fabric Controller
FFFFFFE	Fabric F_Port
FFFFFFF	Broadcast Alias ID (see 25.4)

12.3.1 Destination_ID (D_ID)

The Destination Identifier (D_ID) is a three-byte field (Word 0, Bits 23-0) that shall contain the address identifier of a N_Port or F_Port within the destination entity.

12.3.2 Source_ID (S_ID)

The Source Identifier (S_ID) is a three-byte field (Word 1, Bits 23-0) that shall contain the address identifier of a N_Port or F_Port within the source entity.

12.4 Word 1, bits 31-24

The meaning of these bits is controlled by bit 17 of the F_CTL field. When bit 17 of F_CTL is set to zero, word 1, bits 31-24 shall be interpreted to be class specific control (CS_CTL) information as described in 12.4.1. When bit 17 of F_CTL is set to one, word 1, bits 31-25 shall be interpreted to be the Priority field; and word 1, bit 24 shall be interpreted to be the preemption request flag. For Class 4, if bit 17 is set to one, Word 1 bits 31-24 of the Frame_Header shall be interpreted both as defined for Class 4 in 12.4.1.4, and as the Priority field. The Priority field shall be interpreted as described in sub-clause 12.4.2. See Table 12 for illustrations of this field.

Table 12 - Priority Preemption Enabled/Not Enabled		
	Word 1, bit(s)	Meaning
Priority/Preemption Enabled	31-25	Priority
	24	0 = No Preemption 1 = Preemption
Priority/Preemption Not Enabled	31-24	CS_CTL

12.4.1 Class Specific Control (CS_CTL)

Word 1, bits 31-24 of the Frame_Header is defined as the Class Specific Control (CS_CTL) field; it contains management information for the class of service identified by the Start of Frame delimiter. The definition of the CS_CTL field is class dependent.

12.4.1.1 Class 1

The CS_CTL field is defined in Table 13 for Class 1:

- Simplex shall be meaningful only in the connect-request frame.
- Stacked Connect-request shall be meaningful only in the connect-request frame.
- COR shall be meaningful only in the connect-request frame.
- BCR shall be meaningful in all Class 1 frames.

Table 13 - CS_CTL field - Class 1		
Bit	Abbr.	Meaning
31	Simplex	0 = Duplex 1 = Simplex
30	Stacked Connect-request	0 = Stacked Connect-request not requested 1 = Stacked Connect-request requested
29	COR	0 = COR Not requested 1 = COR requested
28	BCR	0 = Unbuffered Class1 requested 1 = Buffered Class 1 requested
27-24		Reserved

12.4.1.2 Class 2

The CS_CTL field is defined in Table 14 for Class 2.

PREF shall be meaningful in all Class 2 frames.

Table 14 - CS_CTL field - Class 2		
Bit	Abbr.	Meaning
31	PREF	0 = Frame is delivered with no Preference 1 = Frame may be delivered with Preference
30-29		Reserved for additional Preference function
28-24		Reserved

12.4.1.3 Class 3

The CS_CTL field is defined in Table 15 for Class 3.

PREF shall be meaningful in all Class 3 frames.

Table 15 - CS_CTL field - Class 3		
Bit	Abbr.	Meaning
31	PREF	0 = Frame is delivered with no Preference 1 = Frame may be delivered with Preference
30-29		Reserved for additional Preference function
28-24		Reserved

12.4.1.4 Class 4

The CS_CTL field is defined in Table 16 for Class 4.

VC_ID shall be meaningful only on the connect-request frame.

Table 16 - CS_CTL field - Class 4		
Bit	Abbr.	Meaning
31-24	VC_ID	Virtual Circuit Identifier

12.4.1.5 Class 6

The CS_CTL field is defined in Table 17 for Class 6:

- Stacked Connect-request shall be meaningful only on the connect-request frame.
- Camp-On shall be meaningful only on the connect-request frame
- BCR shall be meaningful in all Class 6 frames.

Table 17 - CS_CTL field – Class 6		
Bit	Abbr.	Meaning
31		Reserved
30	Stacked Connect-request	0 = Stacked Connect-request not requested 1 = Stacked Connect-request requested
29	COR	0 = COR Not requested 1 = COR requested
28	BCR	0 = Unbuffered Class 1 requested 1 = Buffered Class 1 requested
27-24		Reserved

12.4.2 Priority

If bit 17 of the F_CTL field is set to one, the following rules shall apply to the use of the Priority field:

- A value of hex '00' shall indicate that no Priority has been assigned to the frame. The remaining values shall indicate, in ascending order, the relative priority of the frame. That is, a Priority of hex '23' shall be considered to have a lower priority than a Priority of hex '57'. The Fabric may use the Priority to resolve resource contention or to determine the order in which to deliver frames.
- In Class 1 and Class 6, Priority is meaningful for a dedicated connection. The Priority for a dedicated connection shall be established by the priority value provided by the SOFc1 frame.

NOTE - Changing Priority in a subsequent frame of a Dedicated Connection will not affect a Fabric.

- In Class 2 and Class 3, Priority is meaningful for an Exchange. The Exchange Originator on the first frame of the Exchange shall set priority. Priority shall only be modified on a SOFi2 or SOFi3 frame.
- In Class 4, Word 1, bits 31-24 on the QoS ELS frame is defined as the VC_ID. The VC_ID may be considered a priority value by a fabric.

NOTE - Once a Virtual Circuit has been set up, Priority is meaningless.

12.5 Data structure type (TYPE)

The data structure type (TYPE) is a one-byte field (Word 2, Bits 31-24) that shall identify the protocol of the frame content for Data frames. The TYPE field specified identifies the TYPE at the Sequence Recipient for Data frames.

When the Routing bits in R_CTL indicate a Link_Control frame other than F_BSY, the TYPE field is reserved. F_BSY frames use the TYPE field to indicate a reason code for the F_BSY in TYPE field bits 31-28. When the F_BSY is in response to a Link_Control frame, R_CTL bits 27-24 of the busied frame are copied by the Fabric into the TYPE bits 27-24. The bit encodings are shown in Table 34. Duplication of the Link_Control command code allows a N_Port to easily retransmit the frame if it is busied by the Fabric (see 14.3.3.1).

When the Routing bits in R_CTL indicate Basic or Extended Link_Data, TYPE codes are decoded as shown in Table 18.

Table 18 - TYPE codes - N_Port/F_Port Link Service	
Encoded Value Wd 2, bits 31-24	Description
0000 0000	Basic Link Service
0000 0001	Extended Link Service
0000 0010 to 1100 1111	Reserved
1101 0000 to 1111 1111	Vendor Unique

When the Routing bits in R_CTL indicate Video_Data, the TYPE codes are decoded as shown in Table 19.

Table 19 - TYPE codes - Video_Data	
Encoded Value Wd 2, bits 31-24	Description
0000 0010 to 1100 1111	Reserved
1101 0000 to 1111 1111	Vendor Unique

When the Routing bits in R_CTL indicate FC-4 Device_Data or FC-4 Link_Data TYPE codes are decoded as shown in Table 20.

Table 20 - TYPE codes - FC-4 (Device_Data and Link_Data)		
Encoded Value Wd 2, bits 31-24	Hex	Description
0000 0000 to 0000 0011	00 to 01	Reserved
0000 0100	04	ISO/IEC 8802-2 LLC/SNAP (in order)
0000 0101	05	IP over FC, RFC 2625
0000 0110 to 0000 0111	06 to 07	Reserved
0000 1000	08	SCSI - FCP
0000 1001	09	SCSI - GPP
0000 1010 to 0000 1111	0A to 0F	Reserved - SCSI
0001 0000	10	Reserved - IPI-3
0001 0001	11	IPI-3 Master
0001 0010	12	IPI-3 Slave
0001 0011	13	IPI-3 Peer
0001 0100 to 0001 0111	14 to 17	Reserved
0001 1000	18	Reserved - SBCCS
0001 1001	19	SBCCS - Channel
0001 1010	1A	SBCCS - Control Unit
0001 1011	1B	FC-SB-2 Channel to Control Unit
0001 1100	1C	FC-SB-2 Control Unit to Channel
0001 1101 to 0001 1111	1D to 1F	Reserved – FC-SB-2
0010 0000	20	Fibre Channel Common Transport (FC-GS)
0010 0001	21	Reserved for FC-FG
0010 0010	22	Switch Fabric Internal Link Services (FC-SW)
0010 0011	23	FC-AL
0010 0100	24	SNMP
0010 0101 to 0010 1111	25 to 27	Reserved - Fabric Services
0010 1000 to 0010 1111	28 to 2F	Reserved
0011 0000 to 0011 0011	30 to 33	Reserved - Scalable Coherent Interface
0011 0100 to 0011 1111	34 to 3F	Reserved

Table 20 - TYPE codes - FC-4 (Device_Data and Link_Data)		
Encoded Value Wd 2, bits 31-24	Hex	Description
0100 0000	40	HIPPI-FP
0100 0001 to 0100 0111	41 to 47	Reserved - HIPPI
0100 1000 to 0100 1111	41 to 4F	Reserved for FC-AE
0101 0000	50	BBL Control
0101 0001	51	BBL FDDI Encapsulated LAN PDU
0101 0002	52	BBL 802.3 Encapsulated LAN PDU
0101 0003 to 0101 0007	53 to 57	Reserved for future use by FC-BB
0101 1000	58	Virtual Interface (FC-VI)
0101 1001 to 1101 1111	59 to 5F	Reserved
0110 0000	60	FC-AV Container
0110 0001 to 0110 0011	61 to 63	Reserved for future use by FC-AV
0110 0100 to 1101 1111	64 to 67	Reserved
0110 1000	68	Slotted Loop
0110 1001 to 1111 1111	69 to FF	Vendor Unique

12.6 Frame Control (F_CTL)

The Frame Control (F_CTL) field (Word 2, Bits 23-0) is a three-byte field that contains control information relating to the frame content. The following subclause describes the valid uses of F_CTL bits. If an error in bit usage is detected, a reject frame (P_RJT) shall be transmitted in response with an appropriate reason code (see 14.3.3.3) for Class 1, 2 and 6. The format of the F_CTL field is defined in Table 21.

Table 21 - Exchange/Sequence Control (F_CTL)			
Control Field	Word 2, Bits	Description	Reference
Exchange Context	23	0 = Originator of Exchange 1 = Responder of Exchange	see 18.4
Sequence Context	22	0 = Sequence Initiator 1 = Sequence Recipient	see 18.4
First_Sequence	21	0 = Sequence other than first of Exchange 1 = first Sequence of Exchange	see 18.4
Last_Sequence	20	0 = Sequence other than last of Sequence 1 = last Sequence of Exchange	see 18.7
End_Sequence	19	0 = Data frame other than last of Sequence 1 = last Data frame of Sequence	see 22.7.2
End_Connection (Class 1 or 6) or Deactivate Class 4 circuit	18	0 = Connection active 1 = End of Connection Pending (Class 1 or 6) End of live Class circuit	see 22.7.2 see 28

Priority Enable	17	0 = Word 1, Bits 31-24 = CS_CTL 1 = Word 1, Bits 31-24 = Priority/Preemption	see 12.4.1 see 12.4.2
Sequence Initiative	16	0 = hold Sequence initiative 1 = transfer Sequence initiative	see 18.6.3
X_ID reassigned - Obsolete	15		
Invalidate X_ID - Obsolete	14		
ACK_Form	13-12	00 = No assistance provided 01 = Ack_1 Required 10 = reserved 11 = Ack_0 Required	see 12.6
Data Compression – Obsolete	11		
Data Encryption - Obsolete	10		
Retransmitted Sequence	9	0 = Original Sequence transmission 1 = Sequence retransmission	see 23.7.1.2
Unidirectional Transmit (Class 1) or Remove Class 4 circuit (Class 4)	8	0 = Bi-directional transmission (Class 1) 1 = Unidirectional transmission (Class 1)	see 22.5.3
		0 = Retain or deactivate circuit (Class 4) 1 = Remove circuit (Class 4)	see clause 28
Continue Sequence Condition	7-6	Last Data frame - Sequence Initiator 0 0 = No information 0 1 = Sequence to follow-immediately 1 0 = Sequence to follow-soon 1 1 = Sequence to follow-delayed	see 18.6.5
Abort Sequence Condition	5-4	ACK frame - Sequence Recipient 0 0 = Continue sequence 0 1 = Abort Sequence, Perform ABTS 1 0 = Stop Sequence 1 1 = Immediate Sequence retransmission requested	see 18.6.5
		Data frame (1 st of Exchange) - Sequence Initiator 0 0 = Abort, Discard multiple Sequences 0 1 = Abort, Discard a single Sequence 1 0 = Process policy with infinite buffers 1 1 = Discard multiple Sequences with immediate retransmission	See 15.2.1
Relative Offset present	3	0 = Parameter field defined for some frames 1 = Parameter Field = Relative Offset	See 12.12
Exchange reassembly	2	Reserved for Exchange reassembly	
Fill Data Bytes	1-0	End of Data field - bytes of fill 0 0 = 0 Bytes if fill 0 1 = 1 Byte of fill (last byte of Data field) 1 0 = 2 Bytes of fill (last 2 bytes of Data field) 1 1 = 3 Bytes of fill (last 3 bytes of Data field)	

- **Bit 23 - Exchange Context**

An Exchange shall be started by the Originator facility within a N_Port. The destination N_Port of the Exchange shall be known as the Responder. Each frame for this Exchange indicates whether the S_ID is associated with the Originator (0) or Responder (1).

- **Bit 22 - Sequence Context**

A Sequence shall be started by a Sequence Initiator facility within a N_Port. The destination N_Port of the Sequence shall be known as the Sequence Recipient. Each frame of the Sequence indicates whether the S_ID is associated with the Sequence Initiator (0) or the Sequence Recipient (1).

NOTE - Ownership is required for proper handling of Link_Control frames received in response to Data frame transmission in Class 2. When a Busy frame is received, it may be in response to a Data frame (Sequence Initiator) or to an ACK frame (Sequence Recipient). This bit simplifies the necessary constructs to distinguish between the two cases.

- **Bit 21 - First_Sequence**

This bit shall be set to one on all frames in the First Sequence of an Exchange. It shall be set to zero for all other Sequences within an Exchange.

- **Bit 20 - Last_Sequence**

This bit shall be set to one on the last Data frame in the Last Sequence of an Exchange. This bit is permitted to be set to one on a Data frame prior to the last frame. Once it is set to one, it shall be set to one on all subsequent Data frames in the last

Sequence of an Exchange. It shall be set to zero for all other Sequences within an Exchange. This bit shall be set to the same value in the Link_Control frame as the Data frame to which it corresponds.

NOTE - The early transition of this bit, unlike other F_CTL bits, is permitted as a hardware assist by providing an advance indication that the Sequence is nearing completion.

- **Bit 19 - End_Sequence**

This bit shall be set to one on the last Data frame of a Sequence. In Class 1 or Class 6, if this bit is set to one in the ACK corresponding to the last Data frame, it confirms that the Sequence Recipient recognized it as the last Data frame of the Sequence. In Class 2, the final ACK with this bit set to one confirms the end of the Sequence, however, the SEQ_CNT shall match the last Data frame delivered, which may not be the last Data frame transmitted. This indication is used for Sequence termination by the two N_Ports involved in the Sequence in addition to EOF_t or EOF_{dt} (see 18.3.8). This bit shall be set to zero for other frames within a Sequence.

- **Bit 18 - End_Connection (E_C)**

In Class 1 or Class 6 the E_C bit shall be set to one in the last Data frame of a Sequence to indicate that the N_Port transmitting E_C is beginning the disconnect procedure. The N_Port transmitting E_C set to one on the last Data frame of a Sequence is requesting the receiving N_Port to transmit an ACK frame terminated by EOF_{dt} if the receiving N_Port has completed all Active Sequences. If the receiving N_Port is not able to transmit EOF_{dt}, E_C set to one requests that the receiving N_Port complete all Active Sequences and not initiate any new Sequences during the current Connection.

The E_C bit is only applicable to Class 1, 4 and 6 and is only meaningful on the last Data frame of a Sequence. The E_C bit shall be set to zero on a connect-request frame (SOF_{c1}) in order to avoid ambiguous error scenarios where the ACK (EOF_{dt}) is not properly returned to the Connection Initiator (see 22.7.2 for a discussion on removing Dedicated Connections (E_C bit)).

In Class 4 the E_C bit shall be set to one in the last Data frame of a Sequence to indicate that the N_Port transmitting E_C is beginning the deactivation or removal procedure. The N_Port transmitting E_C = 1 on the last Data frame of a Sequence is requesting the receiving N_Port to transmit an ACK frame terminated by EOFdt, in the case of deactivation or EOFrt in the case of removal if the receiving N_Port has completed all active Sequences. If bit R_C = 1 in addition to E_C = 1, then a Class 4 circuit removal is requested otherwise a deactivation is requested.

If the receiving N_Port is not able to deactivate or remove the Class 4 circuit, E_C = 1 requests that the receiving N_Port complete all active Sequences and not initiate any new Sequences during the current activation cycle (see 28.6.4).

- **Bit 17 – Priority Enable**

When bit 17 is set to zero, Word 1, bits 31-24 of the Frame_Header shall be interpreted to be the CS_CTL field as described in clause 12.4.1. When bit 17 is set to one, Word 1, bits 31-24 of the Frame_Header shall be interpreted to be the Priority field as described in clause 12.4.2. For Class 4, if bit 17 is asserted, Word 1 bits 31-24 of the Frame_Header shall be interpreted both as defined for Class 4 and as the Priority field.

- In Class 1 and Class 6, F_CTL bit 17 is meaningful only for the SOFc1 frame.

NOTE - Changing Priority in a subsequent frame of a Dedicated Connection shall not affect the Fabric.

- In Class 2 and Class 3, F_CTL bit 17 shall only be modified on a SOFi2 or SOFi3 frame. I.e., the value of bit 17 shall be the same for all frames of a given Sequence.

- **Bit 16 - Sequence Initiative**

The Originator of an Exchange shall initiate the first Sequence as the Sequence Initiator. If the Sequence Initiative bit is set to zero, the Sequence Initiator shall hold the initiative to continue transmitting Sequences for the duration of this Sequence Initiative. The Sequence Recipient gains the initiative to transmit a new Sequence for this Exchange after the Sequence Initiative has been transferred to the Recipient. This shall be accomplished by setting the Sequence Initiative bit to one in the last Data frame of a Sequence (End_Sequence = 1). In Class 1, 2 and 6, the Sequence Initiator shall consider Sequence Initiative transferred when the ACK to the corresponding Data frame is received with the Sequence Initiative bit = 1. Setting bit 16 = 1 is only meaningful when End_Sequence = 1.

- **Bit 15 - X_ID reassigned - Obsolete**

- **Bit 14 - Invalidate X_ID - Obsolete**

- **Bits 13 - 12 - ACK_Form**

The ACK_Form bits provide an optional assistance to the Sequence Recipient by translating the ACK capability bits in the N_Port Class Service Login Parameters into an F_CTL field accompanying the frame to be acknowledged. ACK_Form is meaningful on all Class 1, 2 or 6 Data Frames of a Sequence and on all connect-request frames. ACK_Form is not meaningful on Class 1, 2 or 6 Link_Control frames, or any Class 3 frames. The meaning of the ACK_Form bits is given below and summarized in Table 22.

- By resetting ACK_Form=00, the Sequence Initiator is indicating that it does not provide ACK generation assistance.
- By setting ACK_Form=01, the Sequence Initiator is indicating that the ACK_1 shall be used as determined during Login.
- By ACK_Form=11, the Sequence Initiator is indicating that ACK_0 shall be used as determined during Login.

Table 22 - ACK_Form	
Word 2, F_CTL Bits 13 and 12	
Encode	Meaning
00	No assistance provided
01	ACK_1 required
10	reserved
11	ACK_0 required

- **Bit 11 – Obsolete**
- **Bit 10 - Obsolete**
- **Bit 9 - Retransmitted Sequence**

Bit 9 is only meaningful in Class 1 and 6 and only if the Exchange Error Policy specified on the first Data frame of the Exchange by the Originator is specified as 1 1 (Discard multiple Sequences with immediate retransmission). Bit 9 shall be set = 1 if the Sequence Initiator has received an ACK with the Abort Sequence Condition bits (F_CTL bits 5-4) set = 1 1. Bit 9 shall be set = 1 in all frames of the retransmitted Sequence. If the Sequence Initiator is not able to determine that all Sequences prior to the Sequence identified in the ACK with bits 5-4 set = 1 1 (i.e., missing final ACK) are complete, the Sequence Initiator shall not retransmit any Sequences until the successful reception of all previous Sequences has been verified using a Read Exchange Status (RES) Extended Link Service request (see 23.7.1.2) or other such method.

- **Bit 8 - Unidirectional Transmit**

For Class 1 bit 8 is meaningful on a connect-request (SOFc1) in Class 1. If bit 8 is set = 0, the Dedicated Connection is bi-directional. If a Dedicated Connection is bi-directional, the Connection Recipient may initiate Sequences immediately after the Dedicated Connection is established. If bit 8 is set = 1, the Dedicated Connection is unidirectional and only the N_Port which transmitted the connect-request which establishes the Connection shall transmit Data frames.

For Class 1 other than the connect-request, bit 8 is meaningful on the first and last Data frames of a Sequence. After the connect-request with bit 8 set = 1, the Connection Initiator may reset bit 8 = 0 making the Connection bi-directional for the duration of the Connection on the first or last Data frames of a Sequence (i.e., the bit is not meaningful in other Data frames of the Sequence).

The Connection Recipient may request that a unidirectional Connection be changed to a bi-directional Connection by setting bit 8 = 0, in an ACK frame. Once set to zero in an ACK, all subsequent ACKs shall be transmitted with bit 8 = 0. The Connection Initiator is not required to honor the request to become bi-directional. See 22.5.3 for additional clarification.

For Class 4 bit 8 is only meaningful on the last Data frame of a Sequence and only if the E_C bit is 1, see the description for bit 18 and 28.6.4.

- **Bits 7-6 - Continue Sequence Condition**

The Continue Sequence Condition bits are information bits that may be set to indicate an estimated transmission time for the next consecutive Sequence of the current Exchange. The Continue Sequence Condition bits are meaningful on a Data frame when the End_Sequence bit is set to one and the Sequence Initiative bit is set to zero. The Continue Sequence Condition bits are not meaningful on a Data frame when the End_Sequence bit is set to one and the Sequence Initiative bit is set to one.

The Continue Sequence Condition bits are also meaningful on the ACK to the last Data frame (End_Sequence = 1) when Sequence Initiative is also transferred (Sequence Initiative bit = 1). In this case, the Continue Sequence Condition bits indicate how long it will be until the N_Port, which received Sequence Initiative, will transmit its first Sequence for this Exchange.

The Continue Sequence Condition bits may be used to manage link resources within a N_Port such as maintaining or removing an existing Class 1 Connection. The bits are informational and may be used to improve the performance and management of link resources within a N_Port. The bits are not binding. The time estimate is relative to the time to remove and reestablish a Class 1 Connection regardless of the Class of Service being used.

Sequence Condition bits:

- 0 0 = No information
- 0 1 = Sequence to follow - immediately
- 1 0 = Sequence to follow - soon
- 1 1 = Sequence to follow - delayed

When the Continue Sequence Condition bits are set to 0 0, no information is being offered regarding when the next Sequence is being transmitted.

When the Continue Sequence Condition bits are set to 0 1, this indicates that the next consecutive Sequence for this Exchange shall be transmitted immediately.

When the Continue Sequence Condition bits are set to 1 0, this indicates that the next consecutive Sequence for this Exchange shall be transmitted in a time period which is less than the time to remove and reestablish a Class 1 Connection (soon).

When the Continue Sequence Condition bits are set to 1 1, this indicates that the next consecutive Sequence transmission for this Exchange shall be delayed for a period of time which is longer than the time to remove and reestablish a Class 1 Connection. If the Sequence Initiator holds Sequence Initiative and indicates delayed transmission of the next Sequence, the Initiator shall wait until the final ACK (EOF_t) before transmitting the next Sequence for the Exchange.

- **Bits 5-4 - Abort Sequence Condition**

The Abort Sequence Condition bits shall be set to a value by the Sequence Initiator on the first Data frame of an Exchange to indicate that the Originator is requiring a specific error policy for this Exchange. For Classes 1, 2, 4 and 6, the Abort Sequence Condition bits shall not be meaningful on other Data frames within an Exchange. For Class 3 process policy support, the Abort Sequence Condition bits shall be set to 1 0 on each data frame with the sequence to indicate that the originator is requiring process policy to be used for the sequence. The error policy passed in the first frame of the sequence of the first Sequence of an Exchange shall be the error policy supported by both N_Ports participating in the Exchange. Additionally, for Class 3 operation the error policy passed in the first received frame of the sequence shall be the error policy supported by both N_Ports participating in the Exchange (see 23.6.1.1).

- 0 0 = Abort, Discard multiple Sequences
- 0 1 = Abort, Discard a single Sequence
- 1 0 = Process policy with infinite buffers
- 1 1 = Discard multiple Sequences with immediate retransmission

In the Abort, Discard multiple Sequences Error Policy, the Sequence Recipient shall deliver Sequences to the FC-4 or upper level in the order transmitted under the condition that the previous Sequence, if any, was also deliverable. If a Sequence is determined to be non-deliverable, all subsequent Sequences shall be discarded until the ABTS protocol has been completed. The Abort, Discard multiple Sequences Error Policy shall be supported in Class 1, 2, 3, 4 or 6.

In the Abort, Discard a single Sequence Error Policy, the Sequence Recipient may deliver Sequences to the FC-4 or upper level in the order that received Sequences are completed by the Sequence Recipient without regard to the deliverability of any previous Sequences. The Abort, Discard a single Sequence Error Policy shall be supported in Class 1, 2, 3, 4 or 6.

In the Process policy with infinite buffers, frames shall be delivered to the FC-4 or upper level in the order transmitted. Process policy with infinite buffers shall only be allowed in Class 1, 3 and 6. In Class 1 and 6, Process policy with infinite buffers shall use ACK_0 (see 14.3.3.2).

In the Discard multiple Sequences with immediate retransmission Error Policy, the Sequence Recipient shall deliver Sequences to the FC-4 or upper level in the order transmitted under the condition that the previous Sequence, if any, was also deliverable. If a Sequence is determined to be non-deliverable, all subsequent Sequences shall be discarded until a new Sequence is received with the Retransmission bit (Bit 9) in F_CTL set = 1 or until the ABTS protocol has been completed. The Discard multiple Sequences with immediate retransmission is a special case of the Discard multiple Sequences Error Policy. This policy is applicable to an Exchange where all transmission is in Class 1 or 6.

A N_Port during N_Port_Login shall indicate process policy support. Discard policy shall be supported.

NOTE - If the delivery order of Sequences, without gaps, is required by a FC-4 to match the transmission order of Sequences within an Exchange, then one of the two Discard multiple Sequence Error Policies is required. In the Discard a Single Sequence Error Policy, out of order Sequence delivery is to be expected and handled by the FC-4 or upper level.

The Abort Sequence Condition bits shall be set to a value other than zeros by the Sequence Recipient in an ACK frame to indicate to the Sequence Initiator that the Sequence Recipient has detected an abnormal condition, malfunction, or error.

0 0 = Continue Sequence
 0 1 = Abort Sequence requested
 1 0 = Stop Sequence
 1 1 = Immediate Sequence retransmission requested

A setting of 0 1 indicates a request by the Sequence Recipient to the Sequence Initiator to terminate this Sequence using the Abort Sequence protocol and then optionally perform Sequence recovery. See 15.2.1 and 23.7.1.1 for a description of the Abort Sequence protocol.

A setting of 1 0 indicates a request by the Sequence Recipient to the Sequence Initiator to stop this Sequence. This allows for a request for an early termination by the Sequence Recipient. Some of the data received may have been processed and some of the data discarded. Aborting the Sequence using the ABTS command is not necessary and shall not be used. Both the Sequence Initiator and Recipient end the Sequence in a normal manner. See 23.7.2 for a description of the Stop Sequence protocol.

A setting of 1 1 indicates that the Sequence Recipient has detected an error in a Class 1 or Class 6 Sequence and requests that the Sequence Initiator begin immediate retransmission of the Sequence, if able (see 23.7.1.2). Sequence retransmission also uses F_CTL bit 9. The Sequence status is saved by the Sequence Recipient in the Exchange Status Block associated with the aborted SEQ_ID.

- **Bit 3 - Relative Offset present**

When bit 3 is set to zero on a Data frame, the value of the Parameter Field is FC-4 protocol specific and may depend on the Information Category. When bit 3 is set to one in a Data frame, the Parameter Field contains the Relative Offset for the Payload of the frame as defined by the FC-4 protocol. Bit 3 is only meaningful on Data frames of a Sequence and shall be ignored on all other frames. Bit 3 is not meaningful on Link_Control or Basic Link Service Link Data frames.

NOTE - When bit 3 is set to 0 on a Data frame, the value in the Parameter Field shall be passed to the upper level.

- **Bit 2 - Exchange reassembly**

The Sequence Initiator shall set bit 2 to 0 to indicate that the Payload in this Data frame is associated with an Exchange between a single pair of N_Ports. Therefore, reassembly is confined to a single destination N_Port.

NOTE - Bit 2 being set to 1 is reserved for future use to indicate that the Payload in this Data frame is associated with an Exchange being managed by a single Node using multiple N_Ports at either the source, destination, or both.

- **Bits 1-0 - Fill Data Bytes**

When the bits associated with the Fill Data Bytes are non-zero, it notifies the Data frame receiver (Sequence Recipient) that one or more of the last three bytes of the Data Field shall be ignored, except for CRC calculation. The fill byte value is not specified by FC-FS but shall be a valid data byte.

Bits 1-0 shall only be meaningful on the last Data frame of a series of consecutive Data frames of a single Information Category within a single Sequence. For example, if a Sequence contains Data frames of a single Information Category, non-zero values for bits 1-0 shall only be meaningful on the last Data frame of the Sequence. The fill Data bytes shall not be included in the Payload.

12.6.1 F_CTL Summary

¹Table 23 and Table 24 summarize the relationship of F_CTL bits on Data frames and on Link_Response frames. The text describing each F_CTL bit more clearly elaborates on the manner in which F_CTL bits interact with each other. The tables provide an overall summary.

When a bit is designated as meaningful under a set of conditions, that bit shall be ignored if those conditions are not present. For example, bit 18 is only meaningful when bit 19 = 1; this means that bit 18 shall be ignored unless bit 19 = 1.

12.6.1.1 F_CTL bits on Data frames

Table 23 shows the key interactions between specific bits within the F_CTL field. The top part of Table 23 describes those bits that are unconditionally meaningful on the first, last, or any Data frame of a Sequence. The connect-request may reflect settings for either the first Data frame of a Sequence, the last Data frame of a Sequence, or both first and last.

NOTE - A key control function may become effective when a F_CTL bit is set to 1. The locations where the key function is meaningful are indicated in the top part of the Table 23.

The bottom part of Table 23 describes those bits, which are conditionally meaningful. For example, Bit 19 = 1 (vertical column) is only meaningful on the last Data frame of a Sequence. Bit 18 = 1 (vertical column) is only meaningful on the last Data frame when bit 19 = 1. Bit 17 (vertical column) is only meaningful on the last Data frame when bit 19 = 1, bit 18 = 0, and bit 16 = 1.

¹ The future enhancement to the Fibre Channel architecture will address the control bit needed for high speed hardware implementation of multiple categories in a Sequence.

Table 23 - F_CTL bit interactions on Data frames														
Bits associated with Data frame order:	23	22	21 =1	20 =1	19 =1	18 =1	17 =1	16 =1	9 =1	8 =1	7-6	5-4	3 =1	1-0
1 st frame of Seq	M	M	M				MF		M	M		MF	M	
last frame of Seq	M	M	M		M				M	M			M	M
any frame of Seq	M	M	M						M				M	
connect-request	M	M	M	ML	ML		MF			M	ML		M	
First_Sequence 21 = 0 21 = 1														
												ML		
Last_Sequence 20 = 0 20 = 1														
End_Sequence 19 = 0 19 = 1														
				ML		ML		ML			ML			
End_Connection 18 = 0 18 = 1														
Sequence Initiative 16 = 0 16 = 1														
Notes: M = Meaningful MF = Only meaningful on first Data frame of a Sequence ML = Only meaningful on last Data frame of a Sequence														

12.6.1.2 F_CTL bits on Link_Control

Table 24 shows the key interactions with F_CTL bits on ACK, BSY, and RJT frames and should be reviewed together with Table 23. F_CTL bits 19, 18, and 16 in an ACK frame are retransmitted to reflect confirmation (1) or denial (0) of those indications by the Sequence Recipient. For example, if bits 5-4 are set to (0 1) in response to a Data frame in which bit 19 = 1 and bit 16 = 1, setting bits 19 and 16 to zero in the ACK frame indicates that the Data frame was not processed as the last Data frame and that Sequence Initiative was not accepted by the Sequence Recipient of the Data frame since the Sequence Recipient is requesting that the Sequence Initiator transmit an ABTS frame to Abort the Sequence. See 18.3.8, 18.3.10, and 23.7.1.1 for additional information on setting the Abort Sequence Condition bits.

The ACK to a connect-request may reflect settings for either the first Data frame of a Sequence, the last Data frame of a Sequence, or both first and last.

NOTE - A key control function may become effective when a F_CTL bit is set to 1. The locations where the key function is meaningful are indicated in the top part of the Table 24.

Table 24 - F_CTL bit interactions on ACK, BSY or RJT															
Bits associated with ACK frame order:	23	22	21	20	19	18	16	15	14	9 =1	8 =1	7-6	5-4	3	1-0
ACK to 1 st frame	V	V	E					M+		M	M		Ma		
ACK to last frame	V	V	E					M+		M	M	ML	Ma		
ACK to any frame	V	V	E					M+		M	M		Ma		
ACK to Connect-request	V	V	E	E	ML		ML	M+	ML		M	ML	Ma		
Exchange Context															
23 = 0	V														
23 = 1	V														
Sequence Context															
22 = 0		V													
22 = 1		V													
First_Sequence															
21 = 0			E												
21 = 1			E												
Last_Sequence															
20 = 0				E											
20 = 1				E											
End_Sequence															
19 = 0					E										
19 = 1					ML	ML	ML		ML			ML			
End_Connection															
18 = 0						E									
18 = 1						ML									
Sequence Initiative															
16 = 0							E								
16 = 1							ML								
Notes:															
1. M = Meaningful															
2. Ma = Meaningful only on ACK frames															
3. ML = Meaningful only on last ACK, BSY and RJT frames of a Sequence															
4. E = Echo (meaningful)															
5. V = Inverse or invert (meaningful)															
6. M+ = Meaningful on first and following ACK frames until the Sequence Recipient X_ID appears in a Data frame (X_ID reassignment)															

12.7 Sequence_ID (SEQ_ID)

The SEQ_ID is a one byte field (Word 3, Bits 31-24) assigned by the Sequence Initiator which shall be unique for a specific D_ID and S_ID pair while the Sequence is Open. Both the Se-

quence Initiator and the Sequence Recipient track the status of frames within the Sequence using fields within the Sequence_Qualifier. If its X_ID is unassigned, it shall use any other field or fields such as S_ID, D_ID, or the other N_Port's X_ID for tracking (see 12.1.1 and 12.10).

If the Sequence Initiator initiates a new Sequence for an Exchange in any Class of service while it already has Sequences open for that Exchange, it is termed a streamed Sequence. If streamed Sequences occur, it is the responsibility of the Sequence Initiator to use X+1 different consecutive SEQ_IDs where X is the number of Open Sequences per Exchange (see 17.6.8.8). For example, if X=2 from Login, then consecutive SEQ_IDs of 11-93-22-11-93 is acceptable.

If consecutive non-streamed Sequences for the same Exchange occur during a single Sequence Initiative, it is the responsibility of the Sequence Initiator to use a different SEQ_ID for each consecutive Sequence. For example, consecutive SEQ_IDs of 21-74-21-74 is acceptable. The examples show when a SEQ_ID shall be allowed to be repeated. A series of SEQ_IDs for the same Exchange may also be random and never repeat (also see 18.3.4). See 18.6.2 for more discussion regarding reusing and timing out Recovery_Qualifiers following an aborted or abnormally terminated Sequence, or an aborted Exchange.

The combination of Initiator and Recipient Sequence Status Blocks identified by a single SEQ_ID describe the status of that Sequence for a given Exchange. See 18.8.2 for a description of the Sequence Status Block maintained by the Sequence Recipient.

12.8 DF_CTL

Data Field Control (DF_CTL) is a one-byte field (Word 3, Bits 23-16) that specifies the presence of optional headers at the beginning of the Data_Field for Device_Data or Video_Data frames. DF_CTL bits are not meaningful on Link_Control or Basic Link Service frames. Control bit usage is shown in Table 25.

Table 25 - DF_CTL bit definition	
Word 3, Bit(s)	Optional Header
23	Reserved for Extended Frame_Header
22	Reclaimed from Expiration_Security Header and reserved
21	0 = No Network_Header 1 = Network_Header
20	0 = No Association_Header 1 = Association_Header
19-18	Reserved
17-16	0 0 = No Device_Header 0 1 = 16 Byte Device_Header 1 0 = 32 Byte Device_Header 1 1 = 64 Byte Device_Header

The Optional Headers shall be positioned in the Data Field in the order specified with the bit 23 header as the first header in the Data Field, bit 22 header as the second header in the Data Field, and so forth, in a left to right manner corresponding to bits 23, 22, 21, and so forth as shown in Figure 24.

If either bit 17 or 16 are set to one, then a Device_Header is present. The size of the Device_Header is specified by the encoded value of bits 17 and 16 as shown.

If an Optional Header is not present as indicated by the appropriate bit in DF_CTL, no space shall be allocated for the Header in the Data Field of the frame. Therefore, for example, if bits 23 and 22 are zero and bit 21 is one, the first data byte of the Data Field contains the first byte of the Network_Header.

See clause 13 for a discussion on Optional Headers.

12.9 Sequence count (SEQ_CNT)

The sequence count (SEQ_CNT) is a two-byte field (Word 3, Bits 15-0) that shall indicate the sequential order of Data frame transmission within a single Sequence or multiple consecutive Sequences for the same Exchange. The sequence count of the first Data frame of the first Sequence of the Exchange transmitted by either the Originator or Responder shall be binary zero. The sequence count of each subsequent Data frame in the Sequence shall be incremented by one.

If a Sequence is streamed, the sequence count of the first Data frame of the Sequence shall be incremented by one from the sequence count of the last Data frame of the previous Sequence (this is termed continuously increasing SEQ_CNT). If a Sequence is non-streamed, the starting sequence count may be continuously increasing or binary zero.

The same SEQ_ID and SEQ_CNT shall identify ACK and Link_Response frames as the frame to which it is responding. Frames are tracked on a SEQ_ID, SEQ_CNT basis within the scope of the Sequence_Qualifier for that Sequence.

The sequence count shall wrap to zero after reaching a value of 65 535. The sequence count shall then only be incremented to (but not including) the sequence count of an unacknowledged frame of the same Sequence. Otherwise, data integrity is not ensured. Sequences of Data frames and sequence count values are discussed in clause 18. See 17.6.8.7 regarding data integrity, Credit, and sequence count.

12.10 Originator Exchange_ID (OX_ID)

The Originator Exchange_ID is a two-byte field (Word 4, Bits 31-16) that shall identify the Exchange_ID assigned by the Originator of the Exchange. Each Exchange shall be assigned an identifier unique to the Originator or Originator-Responder pair. If the Originator is enforcing uniqueness via the OX_ID mechanism, it shall assign a unique value for OX_ID other than hex 'FF FF' in the first Data frame of the first Sequence of an Exchange. An OX_ID of hex 'FF FF' indicates that the OX_ID is unassigned and that the Originator is not enforcing uniqueness via the OX_ID mechanism. If an Originator uses the unassigned value of hex 'FF FF' to identify the Exchange, it shall have only one Exchange (OX_ID= hex 'FF FF') with a given Responder.

An Originator Exchange Status Block associated with the OX_ID is used to track the progress of a series of Sequences which comprises an Exchange. See clause 18 for a discussion of Sequences and Exchanges. See 18.8.1 for a description of the Exchange Status Block.

NOTE - If hex 'FF FF' is used as the OX_ID throughout the Exchange, then the Originator uses an alternate Sequence tracking mechanism. If the OX_ID is unique, it may be used as an index into a control block structure which may be used in conjunction with other constructs to track frames.

12.11 Responder Exchange_ID (RX_ID)

The Responder Exchange_ID is a two byte field (Word 4, Bits 15-0) assigned by the Responder which shall provide a unique, locally meaningful identifier at the Responder for an Exchange established by an Originator and identified by an OX_ID. The Responder of the Exchange shall assign a unique value for RX_ID other than hex 'FF FF', if RX_ID is being used, in an ACK to a

Data frame in the first Sequence of an Exchange in Class 1 and 2, or in the first Sequence transmitted as a Sequence Initiator, if any, in Class 3. RX_ID is not used in Class 6 and shall be set to hex 'FF FF' for all Class 6 frames. An RX_ID of hex 'FF FF' shall indicate that the RX_ID is unassigned. If the Responder does not assign an RX_ID other than hex 'FF FF' by the end of the first Sequence, then the Responder is not enforcing uniqueness via the RX_ID mechanism.

When the Responder uses only hex 'FF FF' for RX_ID, it shall have the capability to identify the Exchange through the OX_ID and the S_ID of the Originator of the Exchange. Under all other circumstances, until a value other than hex 'FF FF' is assigned, hex 'FF FF' value for RX_ID shall be used indicating that RX_ID is unassigned. After a value other than hex 'FF FF' is assigned, the assigned value shall be used for the remainder of the Exchange (see 18.3.2, item e and 18.5.2).

A Responder Exchange Status Block associated with the RX_ID is used to track the progress of a series of Sequences which compose an Exchange. See 18.8.1 for a description of the Exchange Status Block.

See clause 18 for a discussion of Sequences and Exchanges.

NOTE - If hex 'FF FF' is used as the RX_ID throughout the Exchange, then the Responder uses an alternate Sequence tracking mechanism. If the RX_ID is unique, it may be used as an index into a control block structure which may be used in conjunction with other constructs to track frames.

12.12 Parameter

The Parameter field (Word 5, Bits 31-0) has meanings based on frame type. For Link_Control frames, the Parameter field is used to carry information specific to the individual Link_Control frame. For Data frames with the Relative Offset Present bit set to 1, the Parameter field specifies Relative Off-set, a four-byte field that contains the relative displacement of the first byte of the Payload of the frame from the base address as specified by the ULP. Relative Off-set is expressed in terms of bytes (see 11.7). The use of the Relative Offset field is optional and is indicated as a Login Service Parameter. If Relative Offset is being used, the number of bytes transmitted relative to the protocol-specific base address shall be less than the maximum value of the Relative Offset (Parameter) field (2^{32}). For Data frames with the Relative Offset Present bit set to 0, the Parameter field shall be set and interpreted in a protocol specific manner, which may depend on the type of Information Unit carried by the frame.

See clause 21 for a discussion concerning Relative Offset. See clause 14 for a discussion concerning use of the Parameter field in Link_Control frames. See clause 12 for a discussion concerning use of the Parameter field in Basic Link Data Frames.

13 Optional headers

13.1 Introduction

Optional headers defined within the Data Field of a frame are

- Reserved (reclaimed from Expiration_Security_Header)
- Network_Header
- Association_Header
- Device_Header.

Control bits in the DF_CTL field of the Frame_Header define the presence of optional headers. The sequential order of the optional headers, Payload, and their sizes are indicated in Figure 24.

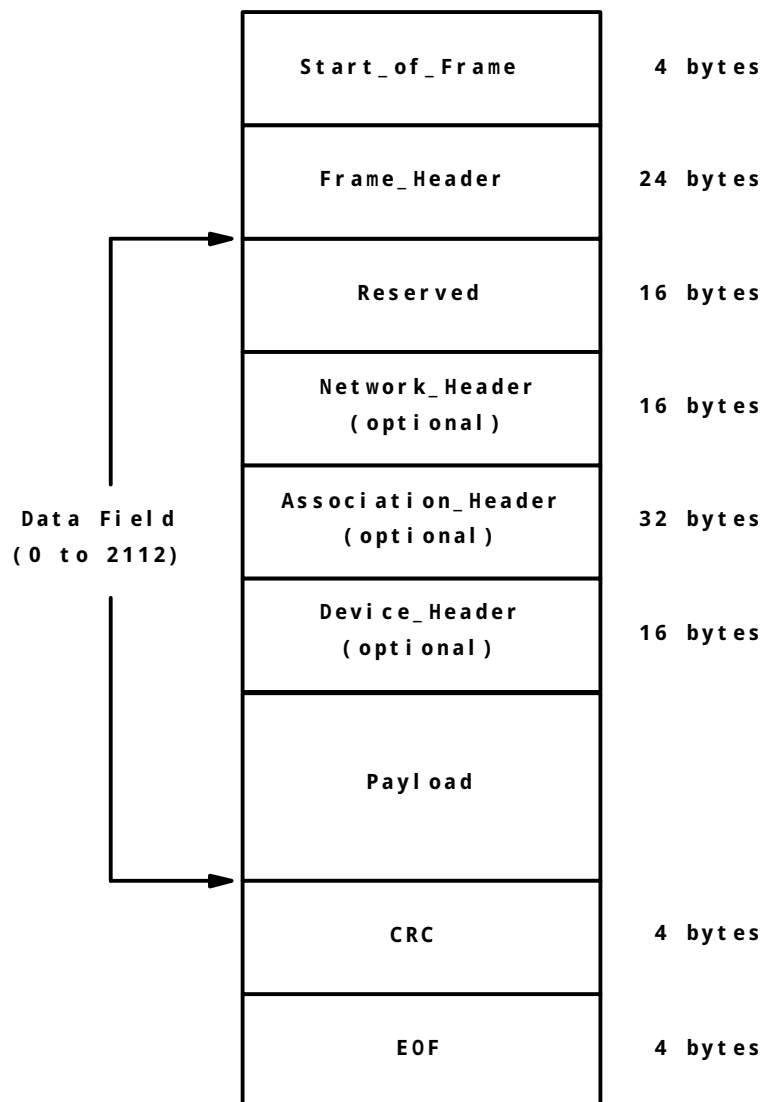


Figure 24 - Optional headers order

The first 16 bytes of the Data Field after the Frame_Header have been reclaimed from the Expiration_Security_Header and are reserved for future use. If present, a Network_Header shall be the next 16 bytes of the Data Field. If present, an Association_Header shall be the next 16 bytes of the Data Field. If present, a Device_Header shall be the next 16, 32, or 64 bytes of the Data Field. If none of the optional headers is present, no space in the Data Field shall be reserved.

13.2 Expiration_Security_Header

The Expiration_Security_Header is reclaimed and its space is reserved for future use.

13.3 Network_Header

A bridge or a gateway node that interfaces to an external Network may use the Network_Header. The Network_Header, if present, shall be 16 bytes in size.

The Network_Header, as shown in Figure 25, is an optional header within the Data_Field content. Its presence shall be indicated by bit 21 in the DF_CTL field, located in the Frame_Header, being set to one. The Network_Header may be used for routing between Fibre Channel networks of different Fabric address spaces, or Fibre Channel and non-Fibre Channel networks. The Network_Header contains Name_Identifier for Network_Destination_Address and Network_Source_Address. The Name_Identifier permitted in the Network_Header are shown in Table 27.

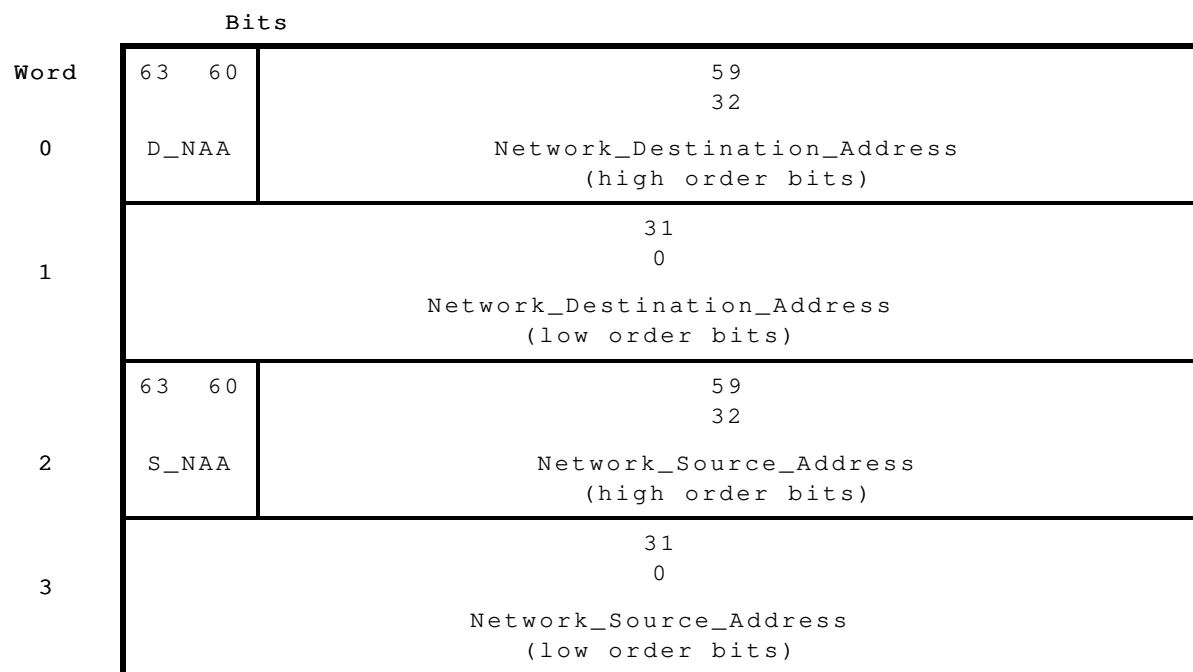


Figure 25 - Network_Header

13.3.1 D_NAA or S_NAA

Destination Network_Address_Authority (D_NAA) or Source Network_Address_Authority (S_NAA) field indicates the authority responsible for the administration of the network address (destination or source) used. The Network_Address_Authority indicators are shown in Table 26.

Table 26 - NAA identifiers	
Bits 63 62 61 60	NAA
0 0 0 0	Ignored
0 0 0 1	IEEE
0 0 1 0	IEEE extended
0 0 1 1	Locally assigned
0 1 0 0	IP
0 1 0 1	IEEE Registered
0 1 1 0	IEEE Registered Extended
0 1 1 1 - 1 0 1 1	Reserved
1 1 0 0	Reclaimed from CCITT
1 1 0 1	Reserved
1 1 1 0	Reclaimed from CCITT
1 1 1 1	Reserved

13.3.2 Network_Destination_ID or Network_Source_ID

The Network_Destination_ID or Network_Source_ID shall be a 60-bit field indicating the network address being used.

13.3.2.1 IEEE 48-bit address

When D_NAA (or S_NAA) is IEEE, Network_Destination_ID (or Network_Source_ID) field shall contain a 48-bit IEEE Standard 802.1A Universal LAN MAC Address (ULA). The ULA shall be represented as an ordered string of six bytes numbered from 0 to 5. The least significant bit of byte 0 shall be the Individual/Group Address (I/G) bit. The next least significant bit shall be the Universally or Locally Administered Address (U/L) bit. IEEE Standard 802.1A further specifies that the bytes be transferred in the order 0 to 5. Figure 26 shows how the bytes of an ULA shall be mapped to two words on the Network Header.

A unique 48-bit IEEE address may be assigned to a N_Port, Node, F_Port, or Fabric.

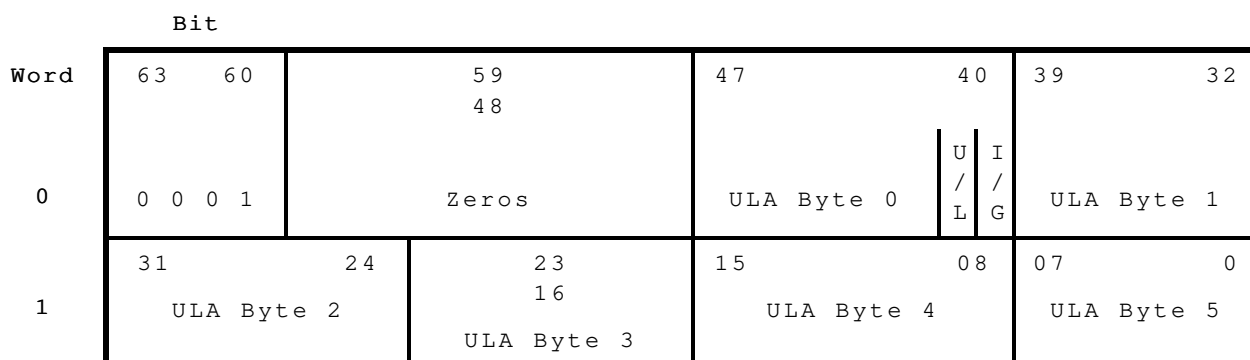


Figure 26 - IEEE 48-bit address format

13.3.2.2 IEEE extended

When D_NAA (or S_NAA) is IEEE extended, Network_Destination_ID (or Network_Source_ID) field shall contain the 48-bit IEEE address preceded by a 12 bit value which may be viewed as either of the following:

- An address uniquely indicating a F_Port or a N_Port contained in a unit
- An extension to the Locally Administered Address portion of the 48-bit address which forms a unique 60-bit value

A unique IEEE extended address may be assigned to a F_Port, N_Port, Fabric or Node.

13.3.2.3 Locally assigned

When D_NAA (or S_NAA) is locally assigned, Network_Destination_ID (or Network_Source_ID) shall be assigned by the local environment and shall be Fabric unique.

A Fabric unique address may be locally assigned to a N_Port, Node, F_Port, or Fabric.

13.3.2.4 32-bit IP address

When D_NAA (or S_NAA) is IP, Network_Destination_ID (or Network_Source_ID) field shall contain a 32-bit IP address. Figure 27 shows how the bytes of an IP address shall be mapped to two words on the Network Header.

A unique 32 bit IP address may be assigned to a Node. Assignment of IP addresses shall conform to accepted Internet Protocol conventions.

NOTE - It is reasonable to have a D_NAA indicate an IP address and S_NAA indicate that the Network_Source_ID is ignored.

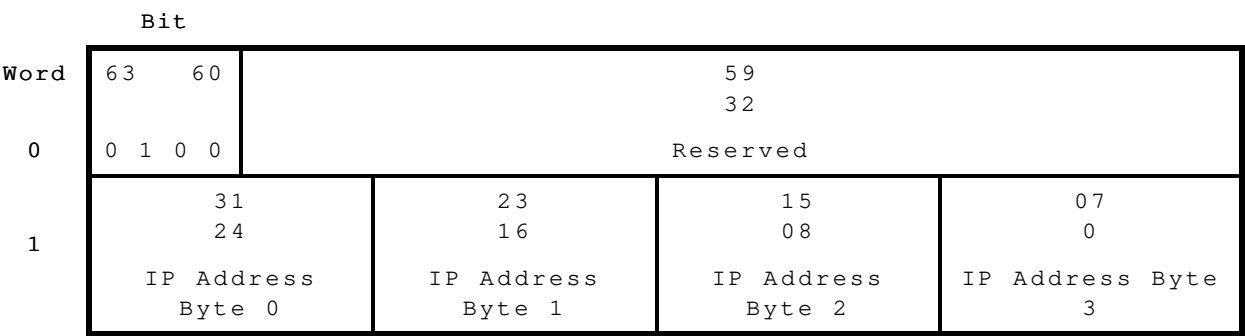


Figure 27 - 32-bit IP address format

13.3.2.5 IEEE registered

When D_NAA (or S_NAA) is IEEE registered, the Network_Destination_ID (or Network_Source_ID) field shall contain the 24-bit IEEE company_id in canonical form followed by a 36-bit vendor specified identifier (VSID) which uniquely indicates a Node, a N_Port, a F_Port, a Fabric, or other object. This format is described in Figure 28.

An IEEE registered name may be assigned to a N_Port, Node, F_Port, or Fabric.

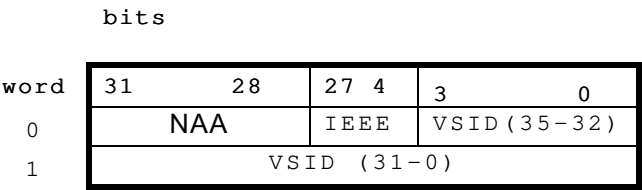


Figure 28 - IEEE Registered

13.3.2.6 IEEE registered extended

When D_NAA (or S_NAA) is IEEE registered extended, the Network_Destination_ID (or Network_Source_ID) field shall contain the 24-bit IEEE company_id in canonical form followed by a 36-bit vendor specified id. An additional 64-bit vendor specified identifier extension is defined. The separate vendor specified spaces allow a node to be identified by the 24-bit IEEE company_id and 36-bit vendor specified id. The node may then generate additional unique sub-ids using the vendor specified identifier extension. The location of the vendor specified identifier extension is not physically part of the Network header and is defined for each type of object. For example, this data may reside in a vendor-defined space and retrieved via a vendor-specific manner. The logical format of IEEE registered extended is described in Figure 29.

An IEEE registered extended name may be assigned to any Fibre Channel related object. The Fibre Channel architecture does not presently provide a large enough space for it to be used as a node, N_Port, F_Port or fabric identifier.

	bits		
Word	3128	274	30
0	NAA	IEEE	VSID (35-32)
1	VSID (31-0)		
2	Vendor Specified Identifier Extension (63-32)		
3	Vendor Specified Identifier Extension (31-0)		

Figure 29 - IEEE registered format

13.3.3 Application summary

The application of Name_Identifier in Network_Header for heterogeneous (FC to Non-FC) networks and homogeneous (FC to FC) networks is summarized in Table 27.

Table 27 - Network addresses		
NAA	Name_Identifier	Network
IEEE	WWN	Heterogeneous
CCITT – individual address	WWN	Heterogeneous
CCITT - group address	WWN	Heterogeneous
IP	WWN	Heterogeneous
IEEE registered	WWN	Heterogeneous
IEEE extended	WWN	FC Networks
Local	FCN	FC Networks
Note - WWN – Worldwide Name (worldwide unique address) FCN - Fibre Channel Name (Fibre Channel unique address)		

FC Name_Identifier

The Network_Addresses used in the Network_Header may refer to various FC entities. The Name_Identifier used for various FC entities are summarized in Table 28.

NOTE - FC-FS does not prevent a Fabric Element from being assigned a unique Worldwide Name in addition to those of its F_Ports. However such usage is outside the scope of FC-FS.

Name_Identifier formats

Formats for various Name_Identifier are summarized in Table 29.

Table 28 - Fibre Channel user identifiers		
NAA	Name_Identifier	Fibre Channel users

		N_Port	Node	F_Port	Fabric
IEEE	WWN	yes	yes	yes	yes
CCITT - individual address	WWN	yes	yes	yes	yes
CCITT - group address	WWN	yes	yes	yes	yes
IP	WWN	yes	yes	yes	yes
IEEE Registered	WWN	yes	yes	yes	yes
IEEE extended	WWN	yes	yes	yes	yes
Local	FCN	yes	yes	yes	yes
Note - yes - applicable to the user no - not applicable to the user WWN - Worldwide Name (worldwide unique identifier) FCN - Fibre Channel Name (Fibre Channel unique identifier)					

Table 29 - Name_Identifier formats			
	Name_Identifier (64 bits)		
NAA Description	NAA ID (4 bits)	60 bit field	
		(12 bits)	(48 bits)
IEEE	0001	zeros	IEEE address
IEEE extended	0010	N_Port number within the Node	IEEE address for Node
		F_Port number within the Fabric element	IEEE address for Fabric element
		Extension to the IEEE address which forms unique 60 bit address	IEEE Address
IEEE Registered	0101	IEEE Company ID (24 bits)	Vendor specified identifier (36 bits)
local	0011	Fabric unique	
IP	0100	zeroes (28 bits)	IP address (32 bits)
CCITT – individual address	1100	CCITT address	
CCITT - group address	1110	CCITT address	

13.4 Association_Header

The Association_Header is an optional header within the Data_Field content. Its presence shall be indicated by bit 20 in the DF_CTL field, located in the Frame_Header, being set to one. The Association_Header shall be 32 bytes in size.

The Association_Header may be used to identify a specific Process or group of Processes within a Node associated with an Exchange. When a N_Port has indicated during Login that an Initial Process_Associator is required to communicate with it, the Association_Header shall be used by that N_Port to identify a specific Process or group of Processes within a Node associated with an Exchange (see 19.1).

The Association_Header shall be subdivided into fields as illustrated in Figure 30. The Validity bits (V) shall indicate whether each of the four Associators contain meaningful (valid) information (bit = 1), or that the Associator shall be ignored (bit = 0) as defined in Table 30. The contents of each Associator of the Association_Header are meaningful to the Node, which generates that particular field. They may not be meaningful to the other Node receiving it.

Applicability

Association_Header is applicable to Class 1, 2, 3 or 6.

NOTE - The Association_Header is provided to support system architectures that require more than two levels of identifiers, i.e., X_ID and SEQ_ID.

Word	Bits	
	31	24
0	Validity	Originator Process_Associator (most significant three bytes)
1	Originator Process_Associator (least significant four bytes)	
2	Reserved	Responder Process_Associator (most significant three bytes)
3	Responder Process_Associator (least significant four bytes)	
4	Obsolete - shall be all zeros	
5	Obsolete - shall be all zeros	
6	Obsolete - shall be all zeros	
7	Obsolete - shall be all zeros	

Figure 30 - Association_Header

Table 30 specifies the meaning of the Validity bits (V) in Figure 30.

Table 30 - Association_Header Validity bits (Word 0, Bits 31 - 24)	
Bit	Description
31	Originator Process_Associator 0 = not meaningful 1 = meaningful
30	Responder Process_Associator 0 = not meaningful 1 = meaningful
29	Obsolete
28	Obsolete

27-25	Reserved
24	Multicast Process_Associator 0 = Unicast 1 = Multicast

13.4.1 Process_Associators

13.4.1.1 Originator and Responder Process_Associators

Process_Associators (Originator and Responder) have the following characteristics:

- A Process_Associator identifies a specific process or group of processes within a Node.
 - The Process_Associator is the mechanism by which a specific process or group of processes is addressed by another communicating process.
- NOTE - An example of a group of processes is a set of related processes controlled by a single instance of an operating system.
- A Process_Associator shall be specified by the Node owning the specific process or group of processes and shall be meaningful to that Node. The value specified may not be meaningful to the other communicating Nodes, although the value may have been made known to them. The other communicating Nodes shall return the Process_Associators given to them.
 - A Process_Associator, once assigned for an Exchange, shall not be changed within the life of the Exchange.
 - A Process_Associator shall not be required to be remembered after the logout of the communicating N_Port.
 - The contents of Process_Associator fields are implementation dependent.
 - A Process_Associator shall span, within the Node, all N_Ports having access to the related process.

13.4.1.2 Multicast Process_Associator

This bit set to one shall indicate that the passed Process_Associator shall be processed as a multicast Process_Associator by the recipient N_Port. That is, the N_Port shall perform an internal multicasting of the payload of the received frame to all images within the Multicast Group specified by the Process_Associator. This bit set to zero shall indicate that the passed Process_Associator shall be processed as a unicast Process_Associator by the Recipient N_Port. That is, the N_Port shall pass the payload of the received frame to the image specified by the Process_Associator. See Clause 25 for a description of Multicast support.

13.4.2 Operation_Associators

Operation_Associators are obsolete in FC-FS. See FC-PH, FC-PH-2, and FC-PH-3 for information about this obsolete field and its functions.

13.5 Device_Header

The Device_Header, if present, shall be 16, 32, or 64 bytes in size. The contents of the Device_Header are entirely under the control of a level above FC-2 based on the TYPE field.

13.6 Optional header usage

13.6.1 Network_Header

Network_Header, if used, shall be present only in the first Data frame of a Sequence. If the receiving N_Port does not support the header function, it shall ignore the header and skip the Data field by the header length (16 bytes).

13.6.2 Association_Header

Association_Header if present, shall be present only in the first Data frame of a Sequence (see 19.2).

13.6.3 Device_Header

Device_Header, if present, shall be present either in the first Data frame or in all Data frames of a Sequence. A ULP type may use a Device_Header. For that ULP type, the Device_Header is required to be supported. The Device_Header may be ignored and skipped, if not needed. If a Device_Header is present for a ULP that does not require it, the related FC-4 may reject the frame with the reason code of TYPE not supported.

13.6.4 Link Service

No Optional_Headers are permitted in Basic Link Service. Only the Expiration_Security_Header is permitted for use with Extended Link Service and FC-4 Link Service.

13.6.5 Summary

Table 31 summarizes the usage of Optional headers.

Table 31 - Optional header usage summary			
Optional Header	Where present	Sequence applicability	Receiving N_Port action
Network_Header	Only in the first Data frame of a Sequence	All Sequences except Basic and Extended Link Services	Skips if not supported or required
Association_Header	Only in the first Data frame of a Sequence	Only in Sequences defined in Association_Header management protocols (see 19.2)	See 19.2
Device_Header	Only in the first Data frame of a Sequence	All Sequences except Basic and Extended Link Services	Skips if not needed or FC-4 may reject if the related ULP Type does not support it.

14 Data frames and responses

14.1 Introduction

Table 32 identifies the types of frames that are possible to be transmitted and received within a N_Port.

14.2 Data frames

As shown in Table 32, two types of frames are defined:

- Data frames
- Link_Control frames

The following Data frames are defined:

- Link_Data
- Device_Data
- Video_Data

When the term "Data frame" is used in FC-FS, it refers to any of the types of Data frames that may be transmitted. The rules of transmission of Data frames and response frames (Link_Control) are equally applicable to each type of Data frame.

Data frames may be used to transfer information such as data, control information, header information, and status from a source N_Port to a destination N_Port. In Class 1, 2 and 6, each Data frame successfully transmitted shall be acknowledged to indicate successful delivery to the destination N_Port. An indication of unsuccessful delivery of a valid frame shall be returned to the transmitter by a Link_Response frame in Class 1, 2 and 6.

Data frames may be streamed, i.e., a single N_Port may transmit multiple frames before a response frame is received. The number of outstanding, unacknowledged Data frames allowed is specified by a Class Service Parameter during the Login procedure (end-to-end Credit). See clause 17 for the specification of Login and Service Parameters and clause 20 for the specification of flow control rules.

Table 32 - Frame formats			
Frame Types	Link_Control (FT-0)	Acknowledgement (ACK)	ACK ACK_0, ACK_1
		Link_Responses	Busy F_BSY, P_BSY
			Reject F_RJT, P_RJT
		Link_Command	LCR
			NTY (see 14.3.4.2), END (see 14.3.4.3)
	Data (FT-1)	FC-4 Device_Data	FC-4 Device Types IP, IPI-3, SCSI, SB, ...
		FC-4 Video_Data	FC-4 Video Types Reserved
		Link Data	Basic Link Service ABTS, BA_ACC, BA_RJT, NOP, RMC
			Extended Link Service ABTX, ACC, ADVC, ECHO, ESTC, ESTS, FLOGI, LOGO, LS_RJT, PLOGI, RCS, RES, RLS, RRQ, RSI, RSS, RTV, TEST
			Extended Link Service – Proc. (15.5.29) PRLI, PRLO, SCN,TPLS
			Extended Link Services - Alias (15.5.35) GAID, FACT, FDACT, NACT, NDACT
			Extended Link Services - Class 4 (15.5.36) QoSR, RVCS
			Extended Link Services - FC-AL (15.5.37) PDISC, FDISC, ADISC, TPRLO
			Extended Link Services - FC-SL (15.5.38) BALLOC, DALLOC

ACK and Link_Response frames are individual frames which indicate successful or unsuccessful frame delivery of a valid frame to the FC-2 level at the destination N_Port which also participate in end-to-end flow control. ACK frames shall indicate successful delivery to the N_Port, while Link_Response frames shall indicate unsuccessful delivery. The R_RDY Primitive Signal is used for buffer-to-buffer flow control, which is discussed in clause 20.

A set of one or more Data frames related by the same SEQ_ID transmitted unidirectionally from one N_Port to another N_Port is called a Sequence. See clause 18 for a discussion of Sequences and Exchanges.

Class 1 and 6 Data frames except a connect-request shall be retransmitted, only if the Discard multiple Sequences with immediate retransmission Exchange error policy is in effect and the

Sequence Recipient has requested Sequence retransmission on an ACK frame. Regardless of the error policy, a Class 1 and 6 connect-request or Class 2 Data frame shall be retransmitted, only in response to a corresponding Busy (F_BSY, P_BSY) frame. Except as above, Data frame recovery shall be by means of Sequence retransmission under the control of FC-4. See 23.6.2, 23.6.3, and 23.7 for a discussion of Sequence integrity, Sequence error detection, and Sequence recovery.

Each Data frame within a Sequence shall be transmitted within an E_D_TOV timeout period to avoid timeout errors at the destination N_Port.

Table 33 specifies the allowable delimiters for valid Data frames by class.

Table 33 - Data frame delimiters	
Data frame	Delimiters
Class 1	SOFc1, SOFi1, SOFn, EOFn
Class 2	SOFi2, SOFn2, EOFn
Class 3	SOFi3, SOFn3, EOFn, EOFt
Class 4	SOFc4, SOFi4, SOFn4, EOFn EOFt
Class 6	SOFc1, SOFi1, SOFn, EOFn
Dedicated Simplex	SOFc1, SOFi1, SOFn1, EOFn

Format: FT_1 (See 11.4)

Addressing: The S_ID field designates the source N_Port identifier (Sequence Initiator) transmitting the Data frame. The D_ID field designates the destination N_Port identifier (Sequence Recipient) of the Data frame.

Data Field: The Data_Field for Data frames is a multiple of four bytes and variable in length. The Data Field may contain optional headers whose presence is indicated by the DF_CTL field in the Frame_Header (see clause 13).

In order to accommodate message content within the Data field that is not a multiple of four bytes, fill bytes shall be appended to the end of the Data Field. The number of fill bytes is specified by F_CTL bits 1-0 (see 12.6) and shall only be meaningful on the last frame of an instance of an Information Category. The fill byte value is not specified by FC-FS.

Payload size

The payload size is determined by taking the overall frame length between the SOF and EOF

- minus the 24 byte Frame_Header,
- minus any Optional Headers,
- minus the fill bytes (0, 1, 2, or 3),
- minus the CRC

Responses

R_RDY Primitive (SOFc1, SOFx2, SOFx3)

ACK

- ACK_0
- ACK_1

Link_Response

- F_RJT, P_RJT
- F_BSY, P_BSY

14.2.1 R_RDY response

In Class 1 and 6, a connect-request (SOF_{c1}) frame shall be responded to by transmitting the R_RDY Primitive Signal. The R_RDY Primitive Signal shall only be used for flow control and shall not indicate that the requested Dedicated Connection has been made. In Class 2 and Class 3, Data and Link_Control frames received shall be responded to by transmitting the R_RDY Primitive Signal. R_RDY transmission shall indicate that the interface buffer, which received the frame, is available for further frame reception.

14.2.2 Data frame responses

Link_Control response frames are ACK frames (0, 1, and N) and Link_Response frames (P_BSY, P_RJT, F_BSY, and F_RJT). All Link_Control response frames (ACK_0, ACK_1, P_BSY, F_BSY, P_RJT, and F_RJT) shall be transmitted in the same Class of Service as the frame to which it is responding. However, the ACK (ACK_0, or ACK_1) to a connect-request frame (SOF_{c1}) is sent as a Class 1 frame and is not subject to buffer to buffer to flow control.

14.2.2.1 ACK frames

Successful Data frame delivery

Class 1

- ACK_0
- ACK_1

Class 2

- ACK_0
- ACK_1

Class 3

- No response

Class 6

- ACK_0
- ACK_1

14.2.2.2 Link_Response frames

Unsuccessful Data frame delivery

Class 1

- F_BSY (Fabric Busy)
- P_BSY (N_Port Busy)
- F_RJT (Fabric Reject)

- P_RJT (N_Port Reject)

Class 2

- F_BSY (Fabric Busy)
- P_BSY (N_Port Busy)
- F_RJT (Fabric Reject)
- P_RJT (N_Port Reject)

Class 3

- No response

Class 6

- F_BSY (Fabric Busy)
- P_BSY (N_Port Busy)
- F_RJT (Fabric Reject)
- P_RJT (N_Port Reject)

14.2.3 Transfer mechanisms

Several mechanisms are available for a Sequence which allow a FC-4 or Upper Level Protocol to convey information to a destination N_Port. Those mechanisms are:

- Information Category within R_CTL field
- Options within F_CTL field
- Device_Header
- Payload

Information Category

The Information Category is included in R_CTL to assist the receiver of a Data frame in directing the Data Field content to the appropriate buffer pool.

F_CTL field options

Within the F_CTL field, Exchange and Sequence CTL bits are used to manage the initiative to transmit Data frames, manage Sequences, and manage Exchanges.

Device_Header

Provisions have been made for a Device_Header to precede the actual Payload contained in the Data Field of a Data frame. The size of the Device_Header is encoded in the DF_CTL field of the Frame_Header. This provides an additional means to specify unique protocol-dependent information to an upper level.

Payload

The normal method of locating a ULP header as part of the Payload provides a straightforward approach to achieve interoperability.

14.3 Link_Control

Link_Control frames shall be used by the N_Port link facility functions to control frame transfer and provide N_Port control in Class 1, 2 and 6.

The Fabric in reply to a Class 1 or 6 preemption request frame shall send a Link_Response frame (F_RJT) if preemption is not enabled for that Fabric. The Fabric shall send an F_RJT Link_Response frame with a "Preemption Not Enabled" reason code to the requesting N_Port.

The Fabric in reply to a Class 1 or 6 preemption request frame shall send a Link_Response frame (F_BSY) if the Fabric was unable to complete the preemption. The Fabric shall send an F_BSY Link_Response frame to the requesting N_Port.

Link_Control frames are identified by the Routing bits 31-28 in the R_CTL field being set to 1 1 0 0. When a Link_Control frame is identified in R_CTL bits 31-28, the Information Category bits (27-24) contain the command code for each Link_Control frame as shown in Table 34. The TYPE field for Link_Control frames other than F_BSY shall be reserved. The DF_CTL field shall be set to zeros since a Link_Control frame contains a Data Field of zero length.

Table 34 - Link_Control codes			
Encoded Value Word 0, bits 27-24	Word 5 Parameter	Description	Abbr.
0000	= 1	Acknowledge_1	ACK_1
0001	≠ 0	reserved	
	= 0	Acknowledge_0	ACK_0
0010	NA	N_Port Reject	P_RJT
0011		Fabric Reject	F_RJT
0100		N_Port Busy	P_BSY
0101		Fabric Busy to Data frame	F_BSY
0110		Fabric Busy to Link_Control frame	F_BSY
0111		Link Credit Reset	LCR
1000		Notify	NTY
1001		End	END
Others		reserved	

Link_Control frames provide:

- indication of successful delivery
- indication of unsuccessful delivery
- flow control and buffer management feedback
- low-level control commands to a N_Port

A N_Port shall provide sufficient resources to receive Link_Control frames in response to Data frames transmitted. A N_Port shall not transmit P_BSY response frames in response to Link_Control frames.

NOTE - It is not necessary to save information in order to retransmit a Link_Control frame since F_BSY to a Link_Control frame contains all information required to retransmit and P_BSY is not allowed for Link_Control frames.

LCR may always be retransmitted in response to an F_BSY. For ACK (0, 1, N) and RJT frames, see individual commands for any restrictions on frame retransmission in response to

F_BSY. Link_Control frames shall be transmitted within an E_D_TOV timeout period of the event that causes transmission of the Link_Control frame.

Table 35 indicates allowable delimiters for valid Link_Control frames by class.

Table 35 - Link_Control frame delimiters	
ACK, BSY, RJT	Delimiters
Class 1	SOFn1, EOFn, EOFt, EOFdt,
Class 2	SOFn2, EOFn, EOFt,
Class 6	SOFn1, EOFn, EOFt, EOFdt
LCR	Delimiters
Class 2	SOFn2, EOFn,
Class 4	SOFn4, EOFn, EOFt, EOFdt, EOFdti, EOFrt, EOFrti
Dedicated Simplex	SOFn2, EOFn, EOFt, EOFdt

14.3.1 R_RDY response

In Class 2, all Link_Control frames shall be responded to by transmitting the R_RDY Primitive Signal when the interface buffer which received the frame is available for further frame reception.

14.3.2 Link_Continue function

The Link_Continue function in FC-FS provides a positive feedback mechanism to control the flow of Data frames on the link. A Data frame shall only be transmitted when the attached Port has indicated that a buffer is available for frame reception. The following list specifies flow control elements:

- R_RDY - buffer-to-buffer flow control for frames between F_Ports and N_Ports if a Fabric is present, or between N_Ports without a Fabric present. The R_RDY Primitive is transmitted on receipt of any Class 2 or 3 frame as well as a Class 1 or Class 6 connect-request frame.
- ACK_0 - successful or unsuccessful delivery of a Sequence (see 14.3.2.2) between N_Ports with or without a Fabric present. ACK_0 shall be applicable only to acknowledged class frames.
- ACK_1 - end-to-end flow control for a single Data frame transfer between N_Ports with or without a Fabric present. The ACK_1 frame is transmitted on receipt of acknowledged class frames.

A N_Port should transmit R_RDY and Link_Control frames before Data frames in order to avoid Credit problems (both buffer-to buffer and end-to-end). When both an R_RDY and a Link_Control frame (ACK, BSY, RJT) are being transmitted in response to a Data frame (Class 2 and Class 1 and 6 connect-request), the R_RDY shall be transmitted prior to the Link_Control frame.

ACK (0 or 1) may be used for acknowledgment of Data frames between N_Ports for a given Sequence, but usage shall follow the allowable forms based on support defined in Login. Prior to N_Port_Login, ACK_1 shall be used. Following N_Port_Login, the decision to use ACK_0 or ACK_1 shall be made based on the results of N_Port_Login.

ACK precedence

When multiple ACK forms are supported by both Sequence Initiator N_Port_Login parameters and the destination N_Port. Sequence Recipient N_Port_Login parameters, ACK_0 usage shall take precedence over ACK_1. ACK_1 shall be the default, if both ends support no other ACK form. Mixing ACK forms within a given Sequence is not allowed (i.e., only one ACK form shall be used within a single Sequence). ACK precedence is summarized in Figure 31.

ACK support by Sequence Recipient at Login			
ACK support by Sequence Initiator at Login	ACK_1		ACK_0
	ACK_1		ACK_1
	ACK_1	ACK_1	ACK_1
	ACK_1	ACK_1	ACK_1
	ACK_0	ACK_1	ACK_0
	ACK_1	ACK_1	ACK_0

Figure 31 - ACK precedence

14.3.2.1 Receiver Ready (R_RDY)

The R_RDY Primitive shall indicate that a single frame (valid or invalid) was received and that the interface buffer, which received the frame, is available for further frame reception. A N_Port or F_Port may choose to support multiple buffers for frames using the R_RDY Primitive for flow control. The number of buffers is specified during Login as buffer-to-buffer Credit.

R_RDY shall be transmitted between a N_Port and a F_Port with a Fabric present, or between two N_Ports without a Fabric present for all Class 2 frames, Class 3 frames, and connect-request frames in Class 1 and 6. The R_RDY Primitive is not forwarded to any higher levels within a N_Port or passed beyond the entry or exit point of the Fabric.

See 10.3.2 for specification of the number of Idles before and after the R_RDY Primitive during transmission.

Responses: There is no response to an R_RDY Primitive.

14.3.2.2 Acknowledge (ACK)

The ACK frame shall indicate that one or more valid Data frames were received by the destination N_Port for the corresponding Sequence_Qualifier and SEQ_CNT of a valid Exchange as specified in the Sequence_Qualifier, and that the interface buffers which received the frames or frame are available for further frame reception. ACK frames shall be used in Class 1, 2 and 6, except for Dedicated Simplex, shall be transmitted in the same Class as the Data frame or frames that are being acknowledged. In Dedicated Simplex, ACK shall be transmitted as Class 2, while the Data frames are in Class 1 (see clause 27).

NOTE - In Class 1 and 6, it is recommended that N_Ports transmit ACK_1 in the same order that the corresponding Data frames are received.

ACK_1

All N_Ports as the default, prior to Login shall support the ACK_1 form of ACK. The SEQ_CNT of the ACK_1 shall match the single Data frame being acknowledged. If a port can only support ACK_0 it shall Logout any port which attempts to Login which does not support ACK_0. The Parameter Field contains a value of binary one in ACK_CNT (bits 15-0) to indicate that a single Data frame is being acknowledged. The R_CTL field (Word 0, bits 27-24) shall be set to 0000.

ACK_0

ACK_0 is the designation used when the ACK_CNT (bits 15-0) of the Parameter Field of the ACK_0 frame contains a value of binary zero to indicate that all Data frames of a Sequence are being acknowledged. If the SEQ_CNT is allowed to wrap within a single Sequence, frame uniqueness is not being ensured. The SEQ_CNT of the ACK_0 shall match the SEQ_CNT of the last Data frame transmitted within the Sequence. The R_CTL field (Word 0, bits 27-24) shall be set to 0001.

The ACK_0 frame may be used for both Discard and Process Exchange Error Policies. For both policy types, ACK_0 support as indicated by Login also specifies that infinite buffering shall be used.

When multiple ACK forms are supported by both Sequence Initiator N_Port_Login parameters and the destination N_Port Sequence Recipient N_Port_Login Parameters, ACK_0 usage shall take precedence over ACK_1. ACK_1 shall be the default, if both ends support no other common ACK form.

If both Sequence Initiator and Sequence Recipient support ACK_0, a single ACK_0 per Sequence shall be used to indicate successful Sequence delivery or to set Abort Sequence Condition bits. An additional ACK_0 shall be used within a Sequence to:

- perform X_ID interlock or
- respond to a connect-request (SOF_{c1}).

ACK_0 shall not participate in end-to-end Credit management. Mixing ACK forms in a Sequence is not allowed.

NOTE - Although infinite buffers is indicated at the FC-FS level within a N_Port, individual FC-4s such as SCSI or IPI-3, for example, may agree on a maximum Sequence size that is specified at the upper level through Mode Select in SCSI and Attributes in IPI-3. In each protocol, a burst size is specified which indicates the largest single Sequence that shall be transferred. By further controlling the maximum number of concurrent Sequences, each N_Port may limit the amount of buffering that is actually required.

NOTE - The Sequence Recipient sends ACK_1 at least once with Abort Sequence Condition bits set to a value other than 0 0 (see 18.3.10.2, 18.3.10.3, and 18.3.10.4).

All ACK forms

For all forms of ACK, when the History bit (bit 16) of the Parameter Field is set = 0, it shall indicate that the Sequence Recipient N_Port has transmitted all previous ACKs (i.e., lower SEQ_CNT), if any, for this Sequence. When the History bit (bit 16) of the Parameter Field is set = 1, it shall indicate that at least one previous ACK has not been transmitted (withheld, Data frame not processed, or Data frame not received) by the Sequence Recipient N_Port. Using this historical information allows a N_Port to reclaim end-to-end Credit for a missing ACK frame.

Being able to reclaim end-to-end Credit does not relieve the N_Port of accounting for all ACK frames of a Sequence in Class 2. ACK frames shall not be retransmitted in response to an F_BSY (Class 2). The F_BSY frame to an ACK shall be discarded.

Support for ACK_0 may not be symmetrical for a single N_Port as a Sequence Initiator and Sequence Recipient (see 17.6.8.3 and 17.6.8.4).

NOTE - Throughout FC-FS, ACK refers to one of the two forms (ACK_1 or ACK_0) and although there are two command codes in R_CTL, the Parameter Field History bit (bit 16) and ACK_CNT (bits 15-0) are used in a consistent manner.

The ACK frame provides end-to-end flow control for one or more Data frames between two N_Ports as defined in ACK_0 or ACK_1. See 20.4.3.2 for usage rules. A specific Data frame shall be acknowledged once and only once. ACK reception does not imply Data delivery to a higher level.

ACK frames participate in a number of functions in addition to end-to-end flow control. The following list identifies many of those functions:

- X_ID assignment (see 18.5.3)
- X_ID interlock (see 18.5.4)
- Terminating a Sequence (see 18.3.8)
- Establishing a Dedicated Connection (see 22.4.1)
- Removing a Dedicated Connection (see 22.4.3)
- Abort Sequence condition (see 12.6 and 18.3.10)
- Stop Sequence condition (see 12.6 and 18.3.10)
- Abnormal Sequence termination (see 23.7.1)
- Sequence retransmission requested (see 23.7.1.2)

Format: FT_0

Addressing: The D_ID field designates the source of the Data frame (Sequence Initiator) being replied to by the ACK, while the S_ID field designates the source of the ACK frame (Sequence Recipient).

F_CTL: The F_CTL field is returned with both Sequence and Exchange Context bits inverted in the ACK frame. Other bits may also be set according to Table 24.

SEQ_ID: the SEQ_ID shall be equal to the SEQ_ID of the frame being replied to by ACK.

SEQ_CNT: The sequence count (SEQ_CNT) shall be equal to the sequence count of the highest Data frame being replied to by the ACK.

Parameter field: The Parameter Field is defined as follows:

- History Bit (bit 16)
 - 0 = all previous ACKs transmitted
 - 1 = at least one previous ACK not transmitted
- ACK_CNT (bits 15 - 0)
 - 0 = all Data frames (ACK_0)
 - N = 1 Data frames (ACK_1)

Responses

R_RDY Primitive (SOF_{x2})

Link_Response

- F_RJT, P_RJT
- F_BSY

14.3.3 Link_Response

Either the destination N_Port or a F_Port in reply to frames shall send Link_Response frames for Class 1, 2 and 6. Link_Response frames shall only be sent by a N_Port in reply to valid frames (see 11.8.1).

A Link_Response indicates that the frame identified by the Sequence_Qualifier and SEQ_CNT was not delivered to or processed by the destination N_Port. When a Link_Response frame

(either Reject or Busy) is generated by a F_Port or a facility internal to the Fabric, the frame is routed to the destination N_Port of the Link_Response frame. Link_Response frames may be:

- Busy - indicates a busy condition was encountered in the Fabric or destination N_Port.
- Reject - indicates that delivery of the frame is being denied.

14.3.3.1 Fabric Busy (F_BSY)

The F_BSY Link_Response shall indicate that the Fabric or the destination N_Port is temporarily occupied with other link activity and the Fabric is unable to deliver the frame. A reason code is identified in bits 31-28 of the TYPE field. In Class 1 and 6, F_BSY shall only be transmitted in response to the SOFc1 frame and shall be ended with EOFdt. In Class 2, any Data frame or ACK frame may receive an F_BSY response. A Busy response shall not be used in Class 3.

There are two different Link_Control codes defined for F_BSY as shown in Table 34. When word 0, bits 27-24 has a value of 0101, the F_BSY is in response to a Data frame. When word 0, bits 27-24 has a value of 0110, F_BSY is in response to a Link_Control frame.

A F_BSY frame shall not be transmitted in response to another busy frame (either F_BSY or P_BSY); the busy frame shall be discarded.

When a N_Port receives F_BSY in response to a Data frame transmission, the N_Port shall retransmit the busied frame if it has not reached its ability to retry. Therefore, a N_Port shall save sufficient information for Data frames with a SOFc1, or SOFx2 delimiter for retransmission until an ACK or RJT is received or retry is exhausted.

If a N_Port has exhausted its ability to retry Data frame transmission in response to an F_BSY, it shall notify the FC-4 or an upper level. The N_Port may perform the Abort Sequence Protocol based on the Exchange Error Policy.

It is not necessary to save information in order to retransmit a Link_Control frame since F_BSY to a Link_Control frame contains all information required to retransmit and P_BSY is not allowed for Link_Control frames. In Class 2, if a N_Port receives an F_BSY in response to an ACK frame, it shall discard the F_BSY frame.

The Fabric shall interchange the S_ID and D_ID fields, invert both the Exchange and Sequence Context bits, and select the correct Link_Control code value for the F_BSY depending on whether it is in response to a Data frame or Link_Control frame. The Parameter field is returned unchanged.

A reason code is indicated in the TYPE field (Word 2, bits 31-28). If the frame being busied is a Link_Control frame, the Link_Control command code of the busied frame in R_CTL (Word 0, bits 27-24) is moved to the TYPE field (Word 2, bits 27-24) of the F_BSY frame before the F_BSY command code is inserted in R_CTL bits 27-24 of the F_BSY frame. The Data Field of a Data frame is discarded. The Fabric shall use EOFdt for Class 1 and 6 F_BSY frames (connect-request) and EOFn for Class 2 F_BSY frames.

Format: FT_0

Addressing: The D_ID field designates the source of the frame encountering the busy condition while the S_ID field designates the destination of the frame encountering the busy condition.

F_CTL: The F_CTL field is returned as received with both Sequence and Exchange Context bits inverted in the F_BSY or P_BSY frame.

SEQ_ID: The SEQ_ID shall be equal to the SEQ_ID of the frame being busied.

SEQ_CNT: The SEQ_CNT shall be equal to the SEQ_CNT of the frame encountering the busy condition.

Parameter field: The Parameter field is returned unchanged for both Data frames and Link_Control frames.

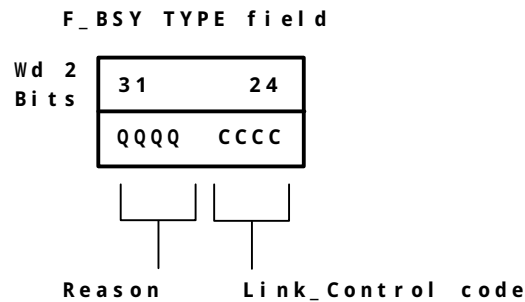


Figure 32 - F_BSY TYPE field

Table 36 - F_BSY reason codes	
Encoded Value Word 2, bits 31-28	Description
0001	Fabric busy
0011	N_Port busy
Others	Reserved

TYPE field: The F_BSY reason code format is defined in Figure 32. Table 36 specifies F_BSY reason codes.

Busy reason codes are defined as follows:

Fabric busy: The Fabric is unable to deliver the frame to the destination N_Port due to conditions internal to the Fabric.

N_Port busy: The destination N_Port is currently involved in a Class 1 Dedicated Connection and the Fabric is unable to deliver the frame.

Responses

R_RDY Primitive (SOF_{x2})

Link_Response

– none

14.3.3.2 N_Port Busy (P_BSY)

The P_BSY Link_Response shall indicate that the responding N_Port is temporarily occupied with other link activity and is not able to accept the frame. A reason code shall be identified in the Parameter field of a P_BSY frame. In Class 1 and 6, P_BSY shall only be transmitted in response to the SOF_{c1} frame and shall be ended with EOF_{dt}. In Class 2, any Data frame may receive a P_BSY response. A Busy response shall not be used in Class 3.

A P_BSY frame shall not be transmitted in response to another Busy frame (either F_BSY or P_BSY); the Busy frame shall be discarded.

When a N_Port receives P_BSY in response to a frame transmission, the N_Port shall re-transmit the busied frame if it has not reached its ability to retry. Therefore, a Port shall save sufficient information for Data frames with a SOF_{c1}, or SOF_{x2} delimiter for retransmission until an ACK or RJT is received or retry is exhausted.

If a N_Port has exhausted its ability to retry Data frame transmission in response to a P_BSY, it shall notify the FC-4 or an upper level. The N_Port may perform the Abort Sequence protocol based on the Exchange Error Policy.

N_Port Busy indicates that the Busy was issued by the destination N_Port. P_BSY shall not be issued in response to a Link_Control frame. A N_Port shall process a Link_Control frame for each unacknowledged Data frame transmitted.

Format: FT_0

Addressing: The D_ID field designates the source of the frame encountering the busy condition while the S_ID field designates the destination of the frame encountering the busy condition.

F_CTL: The F_CTL field is returned as received with both Sequence and Exchange Context bits inverted in the P_BSY frame. Other bits may also be set according to Table 24.

SEQ_ID: The SEQ_ID shall be equal to the SEQ_ID of the frame being busied.

SEQ_CNT: The SEQ_CNT shall be equal to the SEQ_CNT of the frame encountering the busy condition.

Parameter field: The four bytes of this field indicate the action and reason code for the P_BSY response as defined in Figure 33. Table 37 and Table 38 specify the P_BSY action and reason codes, respectively.

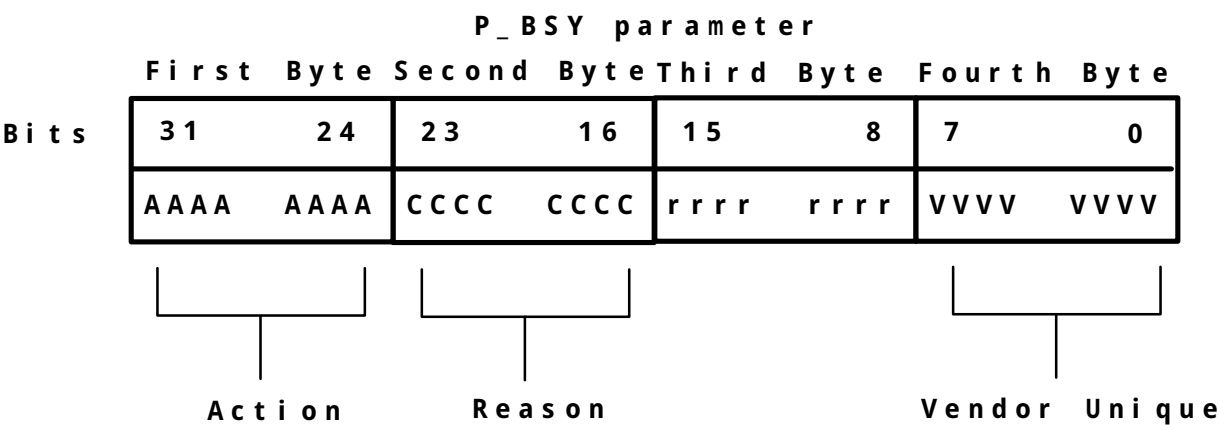


Figure 33 - P_BSY code format

Table 37 - P_BSY action codes

Encoded Value Wd 5, bits 31-24	Description
0000 0001 (1)	Sequence terminated - retry later
0000 0010 (2)	Sequence active - retry later
Others	Reserved

Table 38 - P_BSY reason codes	
Encoded Value Wd 5, bits 23-16	Description
0000 0001	Physical N_Port busy
0000 0011	N_Port Resource busy
0000 0111	Partial Multicast Busy (see Clause 25)
1111 1111	Vendor Unique Busy (See Bits 7-0)
Others	Reserved

Action 1

Action 1 shall indicate that the Sequence Recipient has busied the Sequence (EOF_t or EOF_{dt}). For a Class 1 or 6 connect-request the busy frame is ended with EOF_{dt}. The Sequence Recipient shall only terminate the Sequence on a Busy in response to a connect-request (SOF_{c1}) or in response to an interlocked Data frame associated with X_ID assignment (SOF_{i2}). The frame and Sequence are retryable at a later time.

Action 2

Action 2 shall indicate that the Sequence Recipient has busied a Class 2 frame and that the Sequence has not been terminated (EOF_n). The frame is retryable at a later time.

Busy reason codes are defined as follows:

Physical N_Port busy

P_BSY - the destination N_Port link facilities are currently busy and the N_Port is unable to accept the Payload of the frame.

N_Port Resource Busy

P_BSY - the destination N_Port is unable to process the Data frame at the present time.

Vendor Unique Busy

The Vendor Unique Busy bits (bits 7-0) shall be used by specific Vendors to specify additional reason codes.

Responses

R_RDY Primitive (SOF_{x2})

Link_Response

– none

14.3.3.3 Reject (P_RJT, F_RJT)

The Reject Link_Response shall indicate that delivery of a frame is being denied. A four-byte reject action and reason code shall be contained in the Parameter field. Rejects are transmitted for a variety of conditions. For certain conditions retry is possible, whereas other conditions may require specific intervention. FC-FS does not define the specific intervention required.

In Class 1, 2 and 6, if a N_Port detects an error or a F_Port in a Data frame, it shall transmit a Reject frame based on the reason codes specified in Table 36. If an error is detected on a Link_Control frame, a Reject frame shall only be transmitted under specific conditions. A Fabric shall only reject a Link_Control frame for the following reasons:

- Class not supported
- Invalid D_ID
- Invalid S_ID
- N_Port not available-temporary
- N_Port not available-permanent
- Login required (Fabric)
- Class 6 non-uniform Link_Control Frame

A N_Port shall only reject a Link_Control frame for the following reason (with Action code 2):

- Unexpected ACK

For other reasons, if a N_Port detects an error in a Link_Control frame for a valid Exchange, it shall initiate the Abort Sequence Protocol and not transmit a Reject frame. For an unidentified or invalid Exchange, if an error is detected in a Link_Control frame, the N_Port shall discard the frame and ignore the Link_Control frame error. If a Class 3 frame satisfies a rejectable condition, the frame shall be discarded. A Reject frame (F_RJT, P_RJT) shall not be transmitted in response to another Reject frame (either F_RJT or P_RJT); the received Reject frame in error shall be discarded.

Format: FT_0

Addressing: The D_ID field designates the source of the frame being rejected while the S_ID field designates the destination of the frame being rejected.

F_CTL: The F_CTL field is returned with both Sequence and Exchange Context bits inverted in the F_RJT or P_RJT frame. Other bits may also be set according to Table 24.

SEQ_ID: The SEQ_ID shall be equal to the SEQ_ID of the frame being rejected.

SEQ_CNT: The SEQ_CNT shall be equal to the SEQ_CNT of the frame being rejected.

Parameter field: The four bytes of this field shall indicate the action code and reason for rejecting the request (see Figure 34 and Table 39 and Table 40).

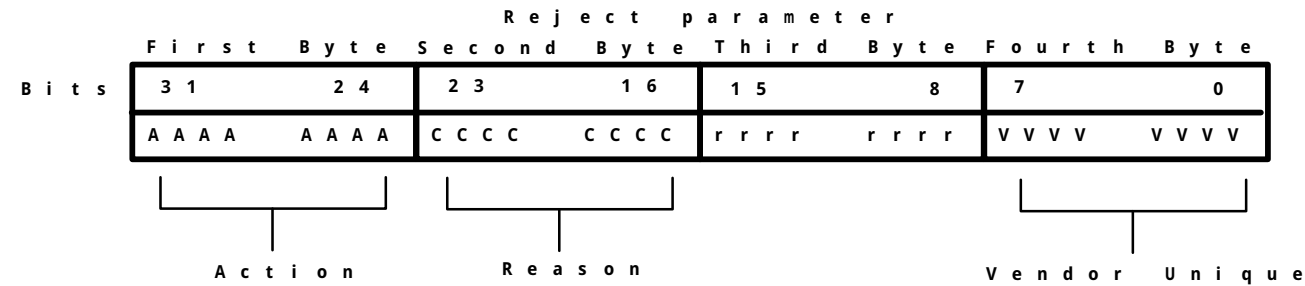


Figure 34 - Reject code format

Table 39 - Reject action codes	
Encoded Value Wd 5, bits 31-24	Description
0000 0001 (1)	Retryable error
0000 0010 (2)	Non-retryable error
Other codes	Reserved

Table 40 - Reject reason codes		
Encoded Value Wd 5, bits 23-16	Description	By
0000 0001	Invalid D_ID	B
0000 0010	Invalid S_ID	B
0000 0011	N_Port not available, temporary	F
0000 0100	N_Port not available, permanent	F
0000 0101	Class not supported	B
0000 0110	Delimiter usage error	B
0000 0111	TYPE not supported	B
0000 1000	Invalid Link_Control	P
0000 1001	Invalid R_CTL field	P
0000 1010	Invalid F_CTL field	P
0000 1011	Invalid OX_ID	P
0000 1100	Invalid RX_ID	P
0000 1101	Invalid SEC_ID	P
0000 1110	Invalid DF_CTL	F
0000 1111	Invalid SEQ_CNT	P
0001 0000	Invalid Parameter field	P
0001 0001	Exchange Error	P
0001 0010	Protocol Error	P
0001 0011	Incorrect length	B
0001 0100	Unexpected ACK	P
0001 0101	Class of service not supported by entity at 'FFFFFFF' (see 17.3.3)	F

Table 40 - Reject reason codes		
Encoded Value Wd 5, bits 23-16	Description	By
0001 0110	Login Required	B
0001 0111	Excessive Sequences attempted	P
0001 1000	Unable to Establish Exchange	P
0001 1001	Reserved	P
00001 1010	Fabric path not available	F
00001 1011	Invalid VC_ID (Class 4)	F
00001 1100	Invalid CS_CTL field	B
00001 1101	Insufficient resources for VC (Class 4)	B
00001 1110	Dedicated Simplex not supported	B
00001 1111	Invalid class of service	B
00010 0000	Preemption request rejected (see 22.9)	F
00010 0001	Preemption not enabled (see 17)	F
00010 0010	Multicast Error (see 25)	F
00010 0011	Multicast Error Terminate (see 25)	F
1111 1111	Vendor Unique Error (See bits 7-0)	P
others	Reserved	P
Notes: 1. F = F_RJT (F_Port) 2. P = P_RJT (N_Port) 3. B = Both F_RJT and P_RJT		

If a frame within a Sequence is rejected, the Sequence shall be abnormally terminated or aborted. If an EOF_t or EOF_{dt} ends the RJT frame, the Port transmitting the RJT frame has terminated the Sequence. A Port shall only terminate the Sequence using a Reject with an EOF_{dt} in response to a connect-request (SOF_{c1}). In Class 2 a N_Port shall only terminate the Sequence on a Reject in response to an interlocked Data frame associated with X_ID assignment (SOF_{i2}). If an EOF_n ends the RJT frame, the N_Port receiving the RJT frame shall perform the Abort Sequence protocol to abort the Sequence. Rejects shall only be transmitted in response to valid frames.

Action 1

Action 1 indicates that if the condition indicated in the Reject Reason code is changed or corrected, the Sequence may be retryable.

- Class not supported
- Invalid D_ID
- Invalid S_ID
- N_Port not available-temporary
- N_Port not available-permanent
- Login required
- Excessive Sequences attempted

- Unable to establish Exchange

Applicability:

- by Fabric when D_ID = Fabric
- by Fabric when D_ID = N_Port
- by N_Port when D_ID = N_Port

Action 2

Action 2 indicates that the Sequence is non-retryable and further recovery such as Abort Exchange may be required. The following reason codes are non-retryable:

- Delimiter usage error
- Type not supported
- Invalid Link_Control
- Invalid R_CTL
- Invalid F_CTL
- Invalid OX_ID
- Invalid RX_ID
- Invalid SEQ_ID
- Invalid DF_CTL
- Invalid SEQ_CNT
- Invalid Parameter field
- Exchange error
- Protocol error
- Incorrect length
- Unexpected ACK

Applicability

- by Fabric when D_ID = Fabric
- by N_Port when D_ID = N_Port

The first error detected shall be the error reported; the order of checking is not specified.

Invalid D_ID

F_RJT - the Fabric is unable to locate the destination N_Port address.

P_RJT - the N_Port that received this frame does not recognize the D_ID as its own Identifier.

Invalid S_ID

F_RJT - the S_ID does not match the N_Port Identifier assigned by the Fabric.

P_RJT - the destination N_Port does not recognize the S_ID as valid.

N_Port not available, temporary

F_RJT - The N_Port specified by the D_ID is a valid destination address but the N_Port is not functionally available. The N_Port is online and may be performing a Link Recovery Protocol, for example.

N_Port not available, permanent

F_RJT - The N_Port specified by the D_ID is a valid destination address but the N_Port is not functionally available. The N_Port is Offline or Powered Down.

Class not supported

F_RJT or P_RJT - The Class of Service specified by the SOF delimiter of the frame being rejected is not supported.

Delimiter usage error

F_RJT or P_RJT - The SOF or EOF is not appropriate for the current conditions. For example, a frame started by SOF_{c1} is received while a Class 1 Dedicated Connection already exists with the same N_Port. See Table 33 and Table 35 for allowable delimiters by Class.

TYPE not supported

F_RJT or P_RJT - The TYPE field of the frame being rejected is not supported by the Port replying with the Reject frame.

Invalid Link_Control

P_RJT - The command specified in the Information Category bits within R_CTL field in the frame being rejected is invalid or not supported as a Link_Control frame.

Invalid R_CTL field

P_RJT - The R_CTL field is invalid or inconsistent with the other Frame Header fields or conditions present.

Invalid F_CTL field

P_RJT - The F_CTL field is invalid or inconsistent with the other Frame_Header fields or conditions present.

Invalid OX_ID

P_RJT - The OX_ID specified is invalid or inconsistent with the other Frame_Header fields or conditions present.

Invalid RX_ID

P_RJT - The RX_ID specified is invalid or inconsistent with the other Frame_Header fields or conditions present.

Invalid SEQ_ID

P_RJT - The SEQ_ID specified is invalid or inconsistent with the other Frame_Header fields or conditions present.

Invalid DF_CTL

P_RJT - The DF_CTL field is invalid.

Invalid SEQ_CNT

P_RJT - The SEQ_CNT specified is inconsistent with the other Frame_Header fields or conditions present. A SEQ_CNT reject is not used to indicate out of order or missing Data frames (see 12.6 bits 5-4 (F_CTL Abort Sequence Condition)).

Invalid Parameter field

P_RJT - The Parameter field is incorrectly specified or invalid.

Exchange Error

P_RJT - An error has been detected in the identified Exchange (OX_ID). This could indicate Data frame transmission without Sequence Initiative or other logical errors in handling an Exchange.

Protocol Error

P_RJT - This indicates that an error has been detected which violates the rules of FC-2 signaling protocol which are not specified by other error codes.

Incorrect length

F_RJT or P_RJT - The frame being rejected is an incorrect length for the conditions present.

Unexpected ACK

P_RJT - An ACK was received from an unexpected S_ID. The ACK received was not for an Open Sequence or Exchange, but was received from a Logged-in N_Port. (The EOF delimiter shall be EOF_n.)

Login Required

F_RJT or P_RJT - An Exchange is being initiated before the interchange of Service Parameters (i.e., Login) has been performed. The Fabric may issue F_RJT in order to notify a N_Port that a Login with the Fabric is required due to changes within the Fabric. The Fabric shall not issue F_RJT in order to convey Login status of a destination N_Port.

Excessive Sequences attempted

P_RJT - A new Sequence was initiated by a N_Port which exceeded the capability of the Sequence Recipient as specified in the Service Parameters during Login.

Unable to Establish Exchange

P_RJT - A new Exchange was initiated by a N_Port which exceeded the capability of the Responder facilities.

Fabric path not available

F_RJT - The speed of the source and destination N_Ports does not match. Other Fabric characteristics related to multiple Fabric domains may also use this reason code.

Vendor Unique Reject

F_RJT or P_RJT - The Vendor Unique Reject bits (bits 7-0) shall be used by specific Vendors to specify additional reason codes.

ResponsesR_RDY Primitive (SOF_{x2})

Link_Response

– F_BSY

14.3.4 Link_Control commands

Link_Control commands are Link_Control frames that initiate a low-level action at the destination N_Port specified by the D_ID. These commands are limited in scope and are normally associated with functions such as reset. Link_Control commands do not require end-to-end Credit and do not participate in end-to-end flow control with regard to incrementing or decrementing Credit_CNT. Link_Control commands shall not be considered to be part of any existing Exchange or Sequence.

14.3.4.1 Link Credit Reset (LCR)

Class 4 behavior shall not be affected by LCR.

The Link Credit Reset frame shall indicate that the N_Port specified by the S_ID requests that the N_Port specified by the D_ID reset any buffers containing Data frames from the S_ID in order to allow the S_ID to reset its end-to-end Credit to its Login value, i.e., EE_Credit_CNT is set to zero. Both N_Ports abnormally terminate all Active Sequences with the S_ID as Sequence Initiator and the D_ID as Sequence Recipient for all classes.

The S_ID N_Port shall perform Exchange and Sequence recovery at the discretion of the appropriate Upper Level Protocol. After transmitting the LCR frame, the N_Port that requested the Credit Reset shall wait the R_A_TOV timeout period before initiating Sequences with the destination N_Port. The Link Credit Reset frame shall not be transmitted as part of an existing Sequence or Exchange. All fields other than R_CTL, D_ID, and S_ID are reserved and ignored by the Receiver except for CRC calculation.

Link Credit Reset shall only be transmitted in Class 2. See 23.2.4.3 for a discussion of end-to-end Credit loss in Class 2 resulting from Sequence timeout. Any Class 1, 3 and 6 Data frames in the destination N_Port buffers are also reset, although a Dedicated Connection, if present, is unaffected. Therefore, a N_Port should remove a Dedicated Connection prior to transmitting LCR in Class 2 to the same destination N_Port. LCR shall be transmitted with SOF_{n2} and EOF_n.

If a N_Port is out of Credit in Class 1 or 6 and times out all Sequences, it shall perform Connection Recovery (see 22.8). It shall not use LCR.

Format: FT_0

Addressing: The S_ID field designates the N_Port that is requesting a buffer reset by the destination N_Port or D_ID.

ResponsesR_RDY Primitive (SOF_{n2})

F_RJT, P_RJT

F_BSY

NOTE - F_RJT may be returned for any of the reasons allowed by the Fabric. P_RJT is only returned for "Invalid D_ID" or "Class not supported" in order to allow a N_Port to avoid special casing LCR in Class 2. However, the N_Port transmitting LCR should be aware of possibility of F_RJT or P_RJT in order to avoid end-to-end Credit_CNT problems. In

particular, the zero values of OX_ID, RX_ID, SEQ_ID, and SEQ_CNT should be noted for possible conflict with an existing Exchange.

14.3.4.2 Notify (NTY)

The Notify (NTY) Link_Control frame shall be used by the Fabric to inform a destination N_Port that a Camp-On exists on its F_Port. The NTY command is a request to the destination N_Port to initiate the End Connection protocol. The NTY command frame shall be transmitted in Class 2.

The NTY frame shall be sent by the Fabric in Class 2 to the destination N_Port of the connect-request frame with the Camp-On CS_CTL bit set (=1) when that destination's F_Port is busy with a previous connection, and the destination N_Port supports intermix. If the destination does not support intermix and Class 2, the Fabric shall not transmit the NTY command to it. The NTY, like other Link_Control frames does not require end-to-end credit and does not participate in end-to-end flow control (see 14.3.4).

Protocol:

- NTY
- No reply frame

Format: FT_0

Addressing:

The D_ID field of the NTY frame designates the destination N_Port of the connect-request frame. The S_ID field of the NTY frame designates the Fabric controller.

Payload:

- The NTY Link_Control frame has no payload.

Class of Service

- Class 2

Reply Sequence

- None

Code Point

- Word 0, bits 27-24 = 1000

14.3.4.3 End (END)

The END Link_Control command is used by the Dedicated Simplex to remove a Dedicated Simplex Connection, by the Camp-On to remove an outstanding Camp-On request and by the Class 4 to deactivate or remove the Class 4 circuit.

In the case of Dedicated Simplex, the Connection Initiator or Connection Recipient may issue an END Link_Control command with an EOFdt delimiter to remove the Dedicated Simplex connection (see 27.9.2, 27.10.4, and 27.11.2).

In the case of Camp-On, the Connection Initiator may issue an END Link_Control command with an EOFdt delimiter to remove the outstanding Camp-On request (see 29.4).

In the case of Class 4, the END frame shall indicate that the Class 4 circuit is deactivated or removed. The Class 4 circuit is removed if the End Of Frame delimiter is EOFrt or EOFrti. The Class 4 circuit is deactivated if the End Of Frame delimiter is EOFdt or EOFdti. The END Link_Control frame shall be discarded if the End Of Frame delimiter is EOFn, EOFt, EOFni, 132

EOFa or if the end of frame delimiter is missing. Upon the reception of the END Link_Control frame all active Sequences between the communicating N_Ports, using the Class 4 circuit are abnormally terminated.

The communicating N_Ports, at the discretion of the appropriate Upper Level protocol shall perform Exchange and Sequence recovery.

N_Ports shall use the END Link_Control frame to remove or deactivate a Class 4 circuit if the N_Port is unable to determine the state of the Class 4 circuit.

The Fabric shall send the END Link_Control command to remove Class 4 circuits when:

- Fabric re-Login is attempted,
- N_Port or the F_Port enters the Offline state,
- Link failure is detected at the N_Port,
- error recovery for the setup process (see 28.2.4.1), or
- internal Fabric failures of the Class 4 circuit path are detected (i.e., the Fabric is unable to provide the guaranteed quality of service).

The Fabric shall send the END Link_Control command to both N_Ports in the Class 4 circuit, unless prevented by Link failure, Offline, etc.

Protocol:

- END
- No reply frame

Format: FT_0

Addressing:

The D_ID field shall designate the N_Port receiving the END Link_Control frame. In the case of Dedicated Simplex, the S_ID shall indicate the Source N_Port. In Class 4, the S_ID and VC_ID field shall designate the N_Port on which behalf the QoS is transmitting the END command.

Payload:

The END Link_Control frame has no payload.

Class of Service:

Class 1, 2 or 4

Reply Sequence:

None

Code Point:

Word 0, bits 27-24 = 1001

14.4 Data frame summary

Table 41 summarizes Data frame responses by Class.

Table 41 - Data frame summary		
Data Frame	Frame Response (Link_Control)	
Data frame Sequence	Expected (successful)	Alternate Link_Response (unsuccessful)
Class 1 Data Frame	R_RDY Primitive to SOFc1, ACK_0 or ACK_1	F_BSY / PBSY or F_RJT / P_RJT
Class 2 Data frame	R_RDY primitive plus ACK_0, or ACK_1	R_RDY primitive plus F_BSY / P_BSY or F_RJT / P_RJT
Class 3 Data frame	R_RDY primitive	None

14.5 ACK generation assistance

If a Sequence Recipient supports multiple ACK forms, an indication about the required ACK form by the Sequence Initiator will be assistance to the Sequence Recipient in generating it. This shall be done in accordance with Table 42.

Table 42 - N_Port Class Service Parameters - ACK support		
Sequence Recipient word 1, bit 30	Sequence Initiator word 0, bit 10	ACK form required
ACK_0	ACK_0	
0	0	ACK_1
0	1	ACK_1
1	0	ACK_1
1	1	ACK_0

14.5.1 N_Port_Login

14.5.1.1 Capability Indicator

The ACK generation assistance capability is indicated during N_Port_Login in word 0, bit 9 of N_Port Class Service Parameters.

The Initiator Control Flags (D) are specified in 17.6.8.3 and Table 204.

14.5.1.2 Precedence requirements

The ACK precedence requirements specified in 14.3.2 are indicated for easy reference in Figure 31.

14.5.2 Terms

ACK_Form refers to F_CTL bits 13 and 12 of Frame_Header Word 2.

14.5.3 Applicability

The ACK precedence determined during Login is applicable to all Class 1 and 6 Data frames, Class 2 Data frames, and connect-request frames.

ACK_Form is meaningful on all Class 1, 2 or 6 Data frames of a Sequence and on all connect-request frames. ACK_Form is not meaningful on Class 1, 2 or 6 Link_Control frames or any Class 3 frames.

14.5.4 F_CTL bits

F_CTL Bits 13-12 (ACK_Form field) are set by Sequence Initiator to provide an optional assistance to the Sequence Recipient by indicating in this F_CTL field (see Table 21) its ACK capability determined during N_Port_Login.

14.5.5 Login rules

Only ACK_1 shall be used for Classes 1, 2 and 6, during or before the establishment of Login parameters. Additional rules are specified for ACK_Form field usage during these conditions:

- In Classes 1, 2 and 6, ACK_1 shall be used to acknowledge LOGI (PLOGI or FLOGI) and the ACC to LOGI. This rule shall be followed whether LOGI and the ACC to LOGI flow in the same Class 1 or 6 connection or two separate connections.
- If an additional frame with SOFc1 is used to establish Class 1 or 6 connection for LOGI or ACC to LOGI, ACK_1 shall be used to acknowledge all these frames including SOFc1 frame. This rule shall be followed whether LOGI and the ACC to LOGI flow in the same Class 1 or 6 connection or two separate connections.
- If ACK generation assistance is not provided, the ACK_Form field in Class 1, 2 or 6 LOGI, ACC frames to LOGI, or any SOFc1 frames used shall be set to 00.
- If ACK generation assistance is provided, the ACK_Form field in Class 1, 2 or 6 LOGI, ACC frames to LOGI, or any SOFc1 frames used shall be set to 01.
- Once established, the ability or inability to provide ACK generation assistance shall not change until Logout or Relogin occurs.

14.5.6 ACK_Form errors

If a Sequence Recipient receives an ACK_Form value that it does not support, it shall issue a P_RJT with the reason code "protocol error".

15 Link Services

15.1 Introduction

There are three types of Link Services:

- Basic
- Extended
- FC-4

Link Service frames and Sequences are composed of Link_Data frames and shall operate according to the R_RDY Primitive, ACK, and Link_Response rules specified in clause 14. Since DF_CTL bits are not meaningful on Basic Link_Data frames, Basic Link Service frames shall not contain optional headers.

15.1.1 Default Login values

Prior to Login or following Logout, the following default set of Service Parameters shall apply:

- Concurrent Sequences = 1
- Total Concurrent Sequences = 1
- End-to-end Credit = 1
- Buffer-to-buffer Credit = 1 (0 for a port operating in arbitrated loop mode)
- Receive_Data_Field Size = 128
- X_ID interlock required
- ACK_1
- Discard multiple Sequences Error Policy
- Relative Offset not used
- E_D_TOV Resolution = 0
- BB_SC_N = 0
- Other optionally supported features shall not be used or required.

Following Login with the destination, a N_Port shall use the Service Parameters obtained through Login.

15.1.2 Sequence and Exchange management

Basic Link Service commands consist of only a single Basic Link_Data frame and are interspersed or are part of a Sequence for an Exchange performing a specific protocol other than Basic Link Service. In such cases, the Basic Link Service command does not constitute a separate Information Category in specifying the number of Information Categories in a Sequence as a Login parameter. Basic Link Service commands support low-level functions such as passing control bit information in a NOP, or aborting a Sequence using ABTS. Login shall not be required prior to using Basic Link Service commands.

The protocols supported by the Extended Link Services are performed within a single Exchange, intended exclusively for the purpose. Most of Extended Link Service protocols are performed as a two Sequence Exchange. Each of these two Sequence Exchanges consists of a request Sequence by the Originator (N_Port), transfer of Sequence Initiative, and a reply Sequence from the Responder (N_Port or F_Port) which terminates the Exchange by setting the Last_Sequence bit (Bit 20) in F_CTL.

The Sequence Initiator and Sequence Recipient shall follow the rules for Sequence management and Recovery_Qualifier reuse as specified in clause 18. The following rules regarding Sequence and Exchange management apply to Extended Link Services in addition to the rules specified in clause 18:

- Basic and Extended reply frames and Sequences shall be transmitted in the same Class as the request
- If Login has not been completed successfully, each Port shall use the default Login values. Since Concurrent Sequences is limited to one, originating a second Exchange to perform an Extended Link Service, such as Read Exchange Status, is not possible. Therefore, Abort Sequence Protocol (ABTS) is the preferred recovery action.
- If Login has not been completed successfully, the ADVC, ESTC, ESTS, LIRR, PRLI, PRLO, RNC, RNFT, RNID, RTIN, TPRLO, TPLS protocols shall not be attempted or supported. If the recipient N_Port or NL_Port is not logged in with the requesting port it shall reply with an LS_RJT ELS Sequence with a reason code of 'unable to perform command request' with a reason code of 'N_Port_Login Required.'
- If Login has been completed successfully, the Originator of the Exchange shall use the Discard multiple Sequences Error Policy for all Extended Link Service Exchanges.
- The Originator of an Extended Link Service Exchange shall detect an Exchange error following Sequence Initiative transfer if the Reply Sequence is not initiated and received within a timeout interval equal to twice the value of R_A_TOV.
- If the Exchange Originator of an Extended Link Service Exchange detects an Exchange error, it shall abort the Exchange (ABTX) and retry the protocol of the aborted Exchange with a different Exchange.
- If the Sequence Initiator aborts a Sequence using ABTS (Abort Sequence Protocol) due to receiving an ACK with the Abort Sequence bits (5-4) set to 0 1, the Sequence Initiator shall retry the Sequence after the Basic Accept is received for the aborted Sequence one time only. If the retry fails, the Extended Link Service Exchange shall be aborted (ABTX).
- If the Sequence Initiator attempts to abort a Sequence using ABTS (Abort Sequence Protocol) and it detects a Sequence timeout (E_D_TOV) waiting for the ACK frame in response to the ABTS, it shall abort the Exchange (ABTX), if conditions permit, and retry the protocol of the aborted Exchange with a different Exchange (see 15.5.2).
- The timeout for a Basic Link Service shall be $2 \times R_A_TOV$.

15.2 Basic Link Service commands

All Basic Link Service commands (see Table 43) shall be supported by a N_Port. Remove Connection (RMC) Link Service command is applicable to only Class 1.

R_CTL bits 31-28 (Word 0) are set = to 1000 to indicate a Basic Link_Data frame. The TYPE field (Word 2, bits 31-24) shall = 0000 0000 to indicate Basic Link Service. The command code of Basic Link Service commands shall be specified in the R_CTL bits 27-24. Table 43 specifies the Basic Link Service command codes.

Table 43 - Basic Link Service Command Codes					
Word 0		Word 1	Description	Abbreviation	Reference
Routing (Bits 31-28)	Category (bits 27-24)	Type (Bits 31-24)			
1000	0000	0000 0000	No Operation	NOP	15.2.4
	0001		Abort Sequence	ABTS	15.2.1
	0010		Remove Connection	RMC	15.2.5
	0100		Basic_Accept	BA_ACC	15.2.2
	0101		Basic_Reject	BA_RJT	15.2.3
	0110		Preempted	PRMT	15.2.6
	Others		Reserved		

15.2.1 Abort Sequence (ABTS)

The Abort Sequence Basic Link Service frame shall be used

- by the Sequence Initiator to request that the Sequence Recipient abort one or more Sequences (see 15.2.1.1 and 23.7.1.1)
- by the Sequence Initiator or Sequence Recipient to request that the ABTS Recipient abort the entire Exchange (see 15.2.1.2).

The decision to attempt to abort one or more Sequences may be determined by the Sequence Initiator (Sequence timeout) or the Sequence Recipient (ACK frame Abort Sequence Condition bits 5-4 or P_RJT frame).

The Sequence Initiator may optionally require that the Sequence Recipient abort one or more sequences by setting bit 0 of the parameter field to 1. If the parameter field bit 0 is set to 0, the Sequence Recipient may elect to abort one or more Sequences or elect to abort the entire Exchange in a protocol specific manner.

The Sequence Recipient may request that one or more Sequences in progress be aborted by setting the Abort Sequence Condition bits to a value of 0 1 on an ACK frame (see 12.6). The ABTS frame may be transmitted without regard to which N_Port holds, or may hold, the Sequence Initiative.

Whether a sequence or exchange is aborted shall be based on the value of bit 0 in the parameter field. FC-FS does not specify a mechanism to determine which behavior an ABTS should create. Bit 0 of the parameter field shall be as specified in Table 44.

Table 44 - ABTS Parameter Field Definitions	
Parameter Field, bit 0 Value	Meaning
0	Abort Exchange
1	Abort Sequence

15.2.1.1 Aborting Sequences using ABTS

Using ABTS to abort one or more Sequences is specified in this section and 23.7.1.1. In this method,

- none, one or multiple Sequences are aborted;
- ABTS is transmitted by the Sequence Initiator of the last Sequence; and
- ABTS is transmitted as part of the Open Sequence.

The SEQ_ID of ABTS frame shall match the SEQ_ID of the last Sequence transmitted by the Sequence Initiator of the ABTS frame. Since ABTS is a continuation of the last transmitted Sequence, it shall be transmitted in the same Class of Service. Since Sequences shall not be streamed in more than one Class, the Class in which the ABTS is transmitted shall be the same Class in which an error, if any, occurred. The RX_ID and OX_ID specified in the ABTS Frame_Header shall be associated with the Exchange in which the Sequence Initiator has detected a potential error.

F_CTL bits, such as First_Sequence, shall be set to match previous Data frames within this Sequence since the ABTS frame is part of the Sequence. F_CTL bits for Sequence Initiative (bit 16) and End_Sequence (bit 19) shall be set to one in order to transfer Sequence Initiative.

ABTS Initiator

Since ABTS is used for error recovery, the following relaxed behaviors are allowed. An ABTS Initiator may transmit ABTS, even if

- There is no end-to-end Credit available,
- it does not hold the Sequence Initiative,
- there is no Sequence Open,
- maximum number of Concurrent Sequences supported are in use, and
- it is a Connection Recipient in Unidirectional Class 1 Connection.

After transmitting the ABTS frame, a N_Port shall consider the status of the Exchange in which it was transmitted to be in an indeterminate condition and shall not deliver any Sequences or notification of Sequence delivery to an upper level until the Basic Accept is received, processed, and recovery, if any, is performed. Due to out of order delivery and special ACK transmission rules, an ACK to a Data frame within a Recovery_Qualifier range may mislead the Sequence Initiator of the ABTS prior to reception of the Basic Accept.

NOTE - The ABTS frame is allowed to be transmitted after a Sequence timeout. The Sequence Initiator of the ABTS frame should reset the E_D_TOV and R_A_TOV timers when the ABTS frame is transmitted, just as any other Data frame transmitted for a Sequence.

ABTS Recipient

When the ABTS request frame is received, the Sequence Recipient may abort no Sequences, one Sequence, or multiple Sequences based on the status of each Sequence within an Exchange and the Exchange Error Policy (see 23.6.1.1). After receiving the ABTS frame, the Recipient shall determine a range of SEQ_CNT values found in error, if any, associated with the identified Exchange. Data frames for any deliverable Sequences may be processed after the ABTS frame is received based on the policy for the Exchange, but before the Basic Accept is transmitted.

Transmission of the Basic Accept to the ABTS frame is an atomic function in that any Data frames identified in the Recovery_Qualifier range (identified in Basic Accept Payload) shall be discarded after the Basic Accept is transmitted to the Sequence Initiator. The Basic Accept provides a synchronization point between the Sequence Initiator and Sequence Recipient. The ABTS Sequence Recipient is not required to timeout waiting for any missing frames before

transmitting the Basic Accept. The ABTS Sequence Recipient shall set F_CTL bit 16 = 0 in the Basic Accept to indicate that it holds the Sequence Initiative for the Exchange or set it = 1 to indicate that the ABTS Sequence Initiator holds the Sequence Initiative.

The format of the Basic Accept Payload is shown in Table 45. The SEQ_ID, if indicated as valid, shall be the last deliverable Sequence transmitted by the Sequence Initiator (of ABTS). If the SEQ_ID is indicated as invalid, then the Sequence Recipient has no information on the last deliverable Sequence. The low SEQ_CNT value shall be equal to the SEQ_CNT of the last Data frame of the last deliverable Sequence. The high SEQ_CNT value shall be equal to the SEQ_CNT of the ABTS frame.

In the Basic Accept Payload, if the low SEQ_CNT = high SEQ_CNT and the last valid SEQ_ID in the Basic Accept matches the last Sequence that was transmitted, then no Sequences have been aborted (all were deliverable), no Recovery_Qualifier is identified, and no recovery is required. If the low SEQ_CNT is not equal to the high SEQ_CNT or the last SEQ_ID is not the last Sequence transmitted, then at least one Sequence is in error.

Recovery Qualifier

If the ABTS frame was transmitted in Class 1 or 6, there shall be no Recovery_Qualifier. The Sequence Initiator of the ABTS frame may reuse SEQ_IDs at its discretion following the normal rules for SEQ_ID and SEQ_CNT and there is no timeout required.

If the ABTS frame was transmitted in Class 2 or 3 and a Sequence error has been indicated, a Recovery_Qualifier range shall be established for both N_Ports. If a Recovery_Qualifier exists, the Sequence Initiator of the ABTS frame shall discard ACK and Link_Response frames received which correspond to the Recovery_Qualifier between the low and high SEQ_CNT values. After transmission of the Basic Accept to the ABTS frame the Sequence Recipient of the ABTS frame shall discard Data frames received which correspond to the Recovery_Qualifier between the low and high SEQ_CNT values if a Recovery_Qualifier exists. While the Recovery_Qualifier exists, the Sequence Initiator shall not transmit Data frames for the Recovery_Qualifier within the specified low and high SEQ_CNT values.

If a Recovery_Qualifier has been established, based on the Basic Accept Payload, the Sequence Initiator of the ABTS shall issue a Reinstate Recovery Qualifier (RRQ) Extended Link Service request Sequence after waiting an R_A_TOV timeout period after reception of the Basic Accept (see 15.5.14).

After the Basic Accept has been transmitted and the Sequence status has been posted in the Exchange Status Block as Aborted, if the Sequence Recipient receives any Data frames for the Aborted Sequence or Aborted Sequences (based on Recovery_Qualifier range), the frames shall be discarded. See 23.7.1 and 23.2.4 for more discussion on abnormal termination of Sequences and Sequence timeout. See 23.7.1.1 for examples of the ABTS protocol which include several special cases such as the start of an Exchange and Class 3. Additional information regarding Sequence recovery and the effects of ABTS based on different Exchange Error Policies is also discussed.

Following reception of the Basic Accept to the Abort Sequence frame, the Sequence Initiator may perform Sequence recovery under guidance from the appropriate FC-4.

The addressing is fully specified by the D_ID and S_ID fields.

Protocol:

- Abort Sequence request frame
- Basic Accept reply frame

Format: FT_1

Addressing: The D_ID field designates the destination N_Port (Sequence Recipient) while the S_ID field designates the source N_Port (Sequence Initiator) which is requesting that a Sequence or Sequences be aborted.

X_ID: Both the RX_ID and OX_ID shall correspond to the current values as determined by the Sequence Initiator of the ABTS frame.

SEQ_ID, SEQ_CNT: The SEQ_ID shall be the same as the last Sequence transmitted for this Exchange by the N_Port transmitting ABTS, even if the last Data frame has been transmitted. The SEQ_CNT shall be set to a value one greater than the previous Data frame transmitted, indicating the highest SEQ_CNT transmitted for this SEQ_ID and the highest SEQ_CNT for this range of SEQ_CNTs over multiple Sequences.

Payload: The Abort Sequence Basic Link Service command has no Payload.

Reply Sequence

- Basic Reject (BA_RJT)
 - signifies rejection of the ABTS command, however, the Sequence may have been aborted without Sequence information (see 15.2.3).
- Basic Accept (BA_ACC)
 - signifies that the destination N_Port has aborted and discarded no Sequences, one Sequence, or multiple Sequences.
 - BA_ACC Frame Header (see 15.2.2)
 - Basic Accept Payload
 - The format of the Basic Accept Payload is shown in Table 45.
 - The SEQ_ID, if indicated as valid, shall be the last deliverable Sequence transmitted by the Sequence Initiator. If the SEQ_ID is indicated as invalid, then the Sequence Recipient has no information on the last deliverable Sequence.
 - The high SEQ_CNT shall be equal to the SEQ_CNT of the ABTS frame. The low SEQ_CNT value shall be one of the following:
 - same as SEQ_CNT of the ABTS frame
 - equal to the SEQ_CNT of the last Data frame of the last deliverable Sequence
 - equal to hex '00 00'.
 - The payload is specified for each of the permitted cases.
 - To indicate that the current Sequence in which ABTS has been received is the last deliverable Sequence, and no Sequences are aborted at its end, the Sequence Recipient shall set, in the BA_ACC payload,
 - ❑ SEQ_ID Validity = valid (hex '80')
 - ❑ SEQ_ID = the SEQ_ID of the Sequence in which the ABTS has been received from the Sequence Initiator, and
 - ❑ Low SEQ_CNT = High SEQ_CNT = SEQ_CNT of the ABTS frame
 - To indicate that it has the information on the last deliverable Sequence but one or more Sequences are aborted at its end, the Sequence Recipient shall set, in the BA_ACC payload,
 - ❑ SEQ_ID Validity = valid (hex '80')

- ❑ SEQ_ID = the SEQ_ID of the last deliverable Sequence received from the Sequence Initiator but is not equal to the SEQ_ID of the Sequence in which ABTS frame has been received,
- ❑ Low SEQ_CNT = the SEQ_CNT of the last Data frame of the last deliverable Sequence
- ❑ High SEQ_CNT = the SEQ_CNT of the ABTS frame
 - To indicate that it has no information on the last deliverable Sequence, and One or more Sequences are aborted at its end, the Sequence Recipient shall set, in the BA_ACC payload, independent of Continuously Increasing SEQ_CNT use,
- ❑ SEQ_ID Validity = invalid (hex '00')
- ❑ SEQ_ID = invalid in this case
- ❑ Low SEQ_CNT = hex '00 00'
- ❑ High SEQ_CNT = the SEQ_CNT of the ABTS frame.

15.2.1.2 Aborting an Exchange using ABTS

Using ABTS to abort an Exchange is specified in this section. In this method,

- an entire Exchange is aborted;
- ABTS is transmitted by the Sequence Initiator or the Sequence Recipient of the last Sequence; and
- ABTS is transmitted as part of the Open Sequence or in a new Sequence.

ABTS sent by the last Sequence Initiator in an Open Sequence

If the last Sequence is Open and the Sequence Initiator of the last Sequence transmits the ABTS frame, the SEQ_ID of this ABTS frame shall match the SEQ_ID of the last Sequence transmitted by the last Sequence Initiator. The SEQ_CNT of the ABTS frame shall be one greater than the SEQ_CNT of the last Data frame transmitted for this last Sequence.

ABTS sent by the last Sequence Initiator in a new Sequence

If the last Sequence has been completed and is therefore not Open, and the Sequence Initiator of the last Sequence transmits the ABTS frame, the ABTS shall be transmitted in a new Sequence with a valid SEQ_ID not in use at that time.

ABTS sent in an Open or new Sequence

Since ABTS is a continuation of the last transmitted Sequence, it shall be transmitted in the same Class of Service. Since Sequences shall not be streamed in more than one Class, the Class in which the ABTS is transmitted shall be the same Class in which an error, if any, occurred. The RX_ID and OX_ID specified in the ABTS Frame_Header shall be associated with the Exchange in which the Sequence Initiator has detected a potential error.

F_CTL bits for Sequence Initiative (bit 16) and End_Sequence (bit 19) shall be set to one in order to transfer Sequence Initiative. If the ABTS frame is part of the last Sequence, F_CTL bits, such as First_Sequence, shall be set to match previous Data frames within this Sequence. If the ABTS is transmitted in a new Sequence, F_CTL bits shall be set to match the new Sequence.

ABTS by the last Sequence Recipient

If the last Sequence Recipient chooses to transmit an ABTS frame, it shall transmit ABTS in a new Sequence with a SEQ_ID available for use from its N_Port as the Sequence Initiator. The new Sequence shall follow applicable rules for the Sequence. The Class in which the ABTS is transmitted shall be the same Class in which an error, if any, occurred. The RX_ID and OX_ID

specified for the new Sequence shall be associated with the Exchange in which the Sequence Recipient has detected a potential error.

If the Sequence Initiator has not transferred the Sequence Initiative or has transferred the Sequence Initiative but has not received the confirmation, but receives the ABTS frame then the Sequence Initiator shall abort the Exchange. Aborting the Exchange may be performed by one of the following two methods specified:

On receiving the ABTS, the ABTS Recipient may set

- the Last_Sequence bit to one in BA_ACC or
- the Last_Sequence bit to zero in BA_ACC followed by ABTX command.

NOTE - If the Sequence Initiator has transferred the Sequence Initiative, received the confirmation but receives ABTS, then it is treated as the ABTS sent by the new Sequence Initiator and the corresponding rules are followed.

Protocol:

- Abort Sequence request frame
- Basic Accept reply frame

Format: FT_1

Addressing: The D_ID field designates the destination N_Port (ABTS Recipient) while the S_ID field designates the source N_Port (ABTS Initiator) which is requesting that an Exchange be aborted.

X_ID: Both the RX_ID and OX_ID shall correspond to the current values as determined by the Sequence Initiator of the ABTS frame.

SEQ_ID, SEQ_CNT:

- If the Sequence Initiator is the ABTS initiator and a Sequence is Open, the SEQ_ID shall be the same as the last Sequence transmitted for this Exchange by the N_Port transmitting ABTS, even if the last Data frame has been transmitted. The SEQ_CNT shall be set to a value one greater than the previous Data frame transmitted, indicating the highest SEQ_CNT transmitted for this SEQ_ID and the highest SEQ_CNT for this range of SEQ_CNTs over multiple Sequences.
- If the Sequence Initiator is the ABTS Initiator and no Sequence is Open, the SEQ_ID shall be a new valid value unused at that time and the SEQ_CNT shall be either continuously increasing from the latest Data frame transmitted in the last Sequence or binary zero.
- If the Sequence Recipient is the ABTS Initiator, the SEQ_ID shall be a new valid value unused at that time by that N_Port as a Sequence Initiator and the SEQ_CNT shall be either continuously increasing from the latest Data frame transmitted in the last Sequence or binary zero.

Payload: The Abort Sequence Basic Link Service command has no Payload.

Reply Sequence:

- Basic Reject (BA_RJT)
 - signifies rejection of the ABTS command, however, the Sequence may have been aborted without Sequence information (see 15.2.3).
- Basic Accept (BA_ACC)
 - signifies that the destination N_Port has aborted and discarded no Sequences, one Sequence, multiple Sequences, or the entire Exchange.

- BA_ACC Frame Header (see 15.2.2)
 - The SEQ_ID, if indicated as valid, shall be the last deliverable Sequence received from the Sequence Initiator. If the SEQ_ID is indicated as invalid, then the Sequence Recipient has no information on the last deliverable Sequence.
 - To abort an Exchange, the Last_Sequence bit shall be set to 1 and Low SEQ_CNT shall be hex '00 00' and High SEQ_CNT hex 'FF FF'.
- BA_ACC Payload
 - The format of the Basic Accept Payload is shown in Table 45.
- The Payload is specified for each of the permitted cases:
 - To indicate that it has the information on the last deliverable Sequence, and nothing is aborted at its end, the ABTS Recipient shall set, in the BA_ACC payload,
 - ❑ SEQ_ID Validity = valid (hex '80')
 - ❑ SEQ_ID = the SEQ_ID of the last deliverable Sequence received from the ABTS Initiator, and
 - ❑ Low SEQ_CNT = High SEQ_CNT = SEQ_CNT of ABTS frame
 - To indicate that it has no information on the last deliverable Sequence, and it is aborting the entire Exchange, the ABTS Recipient shall set the Last_Sequence F_CTL bit to 1 and shall set, in the BA_ACC payload,
 - ❑ SEQ_ID Validity = invalid (hex '00'),
 - ❑ SEQ_ID = invalid in this case
 - ❑ Low SEQ_CNT = hex '00 00' and
 - ❑ High SEQ_CNT = hex 'FF FF'
 - To indicate that it has information on the last deliverable Sequence, but it is aborting the entire Exchange due to uncertainty such as Sequence Initiative ownership, or lack of its capability to resolve the conflict, the ABTS Recipient shall set the Last_Sequence F_CTL bit to 1 and shall set, in the BA_ACC payload,
 - ❑ SEQ_ID Validity = valid (hex '80')
 - ❑ SEQ_ID = the SEQ_ID of the last deliverable Sequence received from the ABTS Initiator,
 - ❑ Low SEQ_CNT = hex '00 00' and
 - ❑ High SEQ_CNT = hex 'FF FF'.

Table 45 - ABTS Basic Accept Payload	
Item	Size - Bytes
SEQ_ID Validity (hex '80' = valid, hex '00' = invalid)	1
SEQ_ID of last Sequence deliverable to ULP (if valid indicated)	1
Reserved	2
OX_ID	2
RX_ID	2
Low SEQ_CNT	2
High SEQ_CNT	2

15.2.2 Basic Accept (BA_ACC)

The Basic Accept (BA_ACC) is a single frame Link Service reply Sequence that notifies the transmitter of a Basic Link Service request frame that the request has been completed. The Basic Accept Link Service reply Sequence shall transfer the Sequence Initiative by setting the Sequence Initiative bit (Bit 16) to 1 in F_CTL on the last Data frame of the reply Sequence if the Sequence Initiative for the Exchange is held by the transmitter of the ABTS frame. The Sequence Initiative (Bit 16) shall be set to 0 to indicate that the transmitter of the Basic Accept holds the Sequence Initiative for the Exchange. The OX_ID and RX_ID shall be set to match the Exchange in which the ABTS frame was transmitted. The SEQ_ID shall be assigned following the normal rules for SEQ_ID assignment.

Protocol: Basic Accept (BA_ACC) is the reply Sequence to Abort Sequence Basic Link Service command.

Format: FT_1

Addressing: The D_ID field designates the source of the Link Service frame being accepted while the S_ID field designates the destination of the request Data frame Sequence being accepted.

Payload: The Payload content is defined within individual Basic Link Service command (ABTS).

Reply Sequence: none

15.2.3 Basic Reject (BA_RJT)

The Basic Reject (BA_RJT) is a single frame Link Service reply Sequence that notifies the transmitter of a Basic Link Service request frame that the request has been rejected. A four-byte reason code is contained in the Payload. Basic Reject may be transmitted for a variety of conditions which may be unique to a specific Basic Link Service request. The OX_ID and RX_ID shall be set to match the Exchange in which the Basic request frame was transmitted. The SEQ_ID shall be assigned following the normal rules for SEQ_ID assignment.

Protocol: BA_RJT may be a reply Sequence to ABTS.

Format: FT_1

Addressing: The D_ID field designates the source of the Basic Link Service request being rejected while the S_ID field designates the destination of the request Data frame Sequence being rejected.

Payload: The first word of the Payload shall contain four bytes to indicate the reason for rejecting the request (see Figure 35 and Table 46 and Table 47).

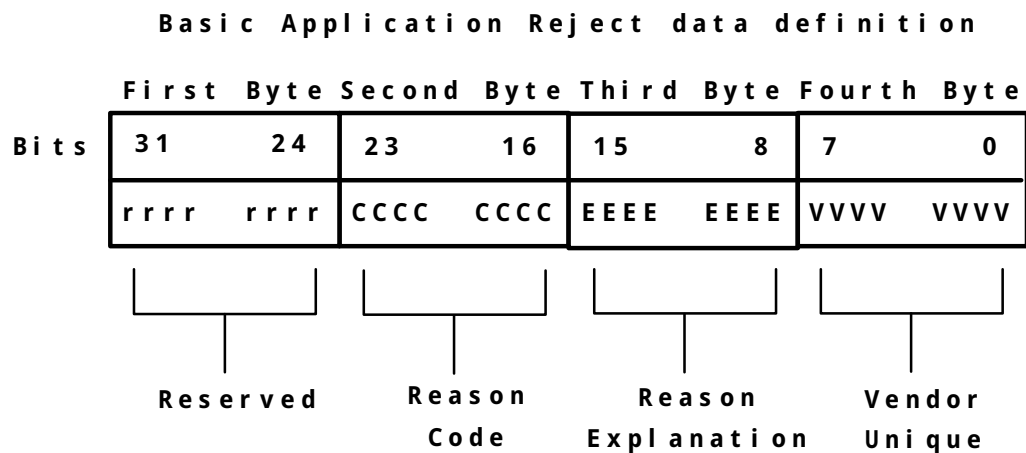


Figure 35 - BA_RJT format

Table 46 - BA_RJT reason codes	
Encoded Value (Bits 23-16)	Description
0000 0001	Invalid Command code
0000 0011	Logical error
0000 0101	Logical busy
0000 0111	Protocol error
0000 1001	Unable to perform command request
Others	Reserved
1111 1111	Vendor Unique Error (see Bits 7-0)

Table 47 - BA_RJT reason code explanation		
Encoded Value (Bits 15-8)	Description	Applicable commands
0000 0000	No additional explanation	ABTS
0000 0011	Invalid OX_ID-RX_ID combination	ABTS
0000 0101	Sequence Aborted, no Sequence information provided	ABTS
Others	Reserved	

The first error condition detected shall be the error reported.

- Bits 23-16 Description**

- Invalid Command code**

The Command code in the Sequence being rejected is invalid.

Logical error

The request identified by the Command code is invalid or logically inconsistent for the conditions present.

Logical busy

The Basic Link Service is logically busy and unable to process the request at this time.

Protocol Error

This indicates that an error has been detected which violates the rules of FC-2 protocol which are not specified by other error codes.

Unable to perform command request

The Recipient of a Link Service command is unable to perform the request at this time.

Vendor Unique Error

The Vendor Unique Error bits may be used by Vendors to specify additional reason codes.

- **Bits 15-8 Reason explanation**

Table 47 shows expanded explanations for Basic Link Service commands.

Reply Link Service Sequence: none

15.2.4 No Operation (NOP)

The No Operation Basic Link Service frame shall be used with delimiters appropriate to the Class in which it is being used. The Data_Field of a NOP frame shall be of zero size. However, the F_CTL field and the SOF and EOF delimiters shall be examined and the appropriate action shall be taken by both the N_Port and Fabric, if present. A NOP frame may be used to initiate a Class 1 or 6 Connection using SOFc1, initiate Sequences, terminate Sequences, or terminate Class 1 or 6 Connections in place of a normal Data frame when there is no Data to send. When used to remove Class 1 or 6 Connections, NOP shall use the normal remove connection procedure by using the E_C bit (see 22.4.3).

The OX_ID and RX_ID shall be set to match the Exchange in which the NOP is being transmitted. The SEQ_ID shall be assigned following the normal rules for SEQ_ID assignment.

Protocol:

- No Operation request
- No reply frame

Format: FT_1

Addressing: The D_ID field designates the destination of the frame while the S_ID field designates the source of the frame.

Payload: The NOP Basic Link Service command has no Payload.

Reply Sequence: none

15.2.5 Remove Connection (RMC)

The Remove Connection Basic Link Service frame shall be used to request immediate removal of a Class 1 or 6 Connection. An ACK frame ended by EOF_{dt} shall be transmitted in response.

This protocol overrides the normal negotiated remove connection procedure using the E_C bit in F_CTL. This method should not be used as the normal method to remove Class 1 Connections since all Open Sequences shall be abnormally terminated. See 22.4.3 for specific rules on removing connections.

The OX_ID and RX_ID shall be set to match the Exchange in which the RMC is being transmitted. The SEQ_ID shall be assigned following the normal rules for SEQ_ID assignment.

Protocol:

- Remove Connection request
- No reply frame

Format: FT_1

Addressing: The D_ID field designates the N_Port being requested to terminate the Class 1 or 6 Connection while the S_ID field designates the N_Port requesting removal.

Payload: The RMC Basic Link Service command has no Payload.

Reply Sequence: none

15.2.6 Dedicated Connection Preempted (PRMT)

The PRMT basic link service command shall indicate that the connection for which this N_Port is participating has been preempted and no longer exists. All sequences associated with this connection have ended abnormally.

Protocol:

- Connection Preempted Notification
- No reply frame

Format: FT_1

Addressing: The D_ID field shall designate the N_Port to which the Fabric directs this basic link service command. The S_ID field shall be equal to the S_ID field in the SOFc1 preemption request frame that triggered this preemption notification.

Priority: Bit 17 of the F_CTL field of the PRMT frame shall be set to one and the Priority field shall be set to the value of the Priority field of the SOFc1 preemption request frame which triggered this preemption.

Payload: None

Reply Sequence: None

15.3 Basic Link Service summary

Table 48 summarizes the Basic Link Service request and reply frames. The Payload content and size in bytes are specified. The Payload for the BA_RJT is 4 bytes in size.

Table 48 - Basic Link Service Payload	
REQUEST SEQUENCE	REPLY SEQUENCE

Request	Payload	Reply	Accept Payload
Abort Sequence (ABTS)	None	BA_ACC or BA_RJT	SEQ_ID Validity, SEQ_ID, OX_ID, RX_ID, Low SEQ_CNT, High SEQ_CNT(12 bytes)
No Operation (NOP)	None	None	
Remove Connection (RMC)	None	None	
Basic Accept	Defined in ABTS	None	
Basic Reject	Reject Reason Code	None	

15.4 Extended Link Services

An Extended Link Service request solicits a destination Port (F_Port or N_Port) to perform a function or service. The Information Category for a request is specified as Unsolicited Control. An Extended Link Service reply may be transmitted in answer to an Extended Link Service request. The Information Category for a reply is specified as Solicited Control. Each request or reply is composed of a single Sequence with the LS_Command code being specified in the first word of the Payload of the first frame of the Sequence.

Each Sequence may be composed of one or more frames. Normal rules for Exchange and Sequence management apply to Extended Link Service frames, Sequences, and Exchanges. The Accept to the following requests shall terminate the Exchange by setting the Last Sequence bit (Bit 20) to one on the last frame of the reply Sequence. The following Extended Link Service requests and the corresponding replies shall be performed within a single Exchange:

- N_Port_Login (PLOGI)
- F_Port Login (FLOGI)
- Logout (LOGO)
- Abort Exchange (ABTX)
- Echo (ECHO)
- Read Connection Status (RCS)
- Read Exchange Status Block (RES)
- Read Sequence Status Block (RSS)
- Establish Streaming (ESTS)
- Estimate Credit (ESTC)
- Advise Credit (ADVC)
- Read Timeout Value (RTV)
- Read Link Error Status Block (RLS)
- Reinstate Recovery Qualifier (RRQ)
- Process Login (PRLI)
- Process Logout (PRLO)
- State Change Notification (SCN)
- Test Process Login State (TPLS)
- Third Party Process Logout (TPRLO)

- Login Control List Management (LCLM)
- Get Alias_ID (GAID)
- Fabric Activate Alias_ID (FACT)
- Fabric Deactivate Alias_ID (FDACT)
- N_Port Activate Alias_ID (NACT)
- N_Port Deactivate Alias_ID (NDACT)
- Quality of Service Request (QoSR)
- Read Virtual Circuit Status (RVCS)
- Discover N_Port Service Parameters (PDISC)
- Discover F_Port Service Parameters (FDISC)
- Discover Address (ADISC)
- Fibre Channel Address Resolution Protocol Request (FARP-REQ)
- Fibre Channel Address Resolution Protocol Reply (FARP-REPLY)
- Read Port Status Block (RPS)
- Read Port List (RPL)
- Bandwidth Allocation (BALLOC)
- Bandwidth Deallocation (DALLOC)
- Fabric Address Notification (FAN)
- Registered State Change Notification (RSCN)
- State Change Registration (SCN)
- Report Node FC-4 Types (RNFT)
- Loop Initialize (LINIT)
- Loop Port Control (LPC)
- Loop Status (LSTS)
- Request Topology Information (RTIN)
- Request Node Identification Data (RNID)
- Registered Link Incident Record (RLIR)
- Link Incident Record Registration (LIRR)
- Request Sequence Initiative (RSI)
- Test (TEST)
- Clock Synchronization Request (CSR)
- Clock Synchronization Update (CSU)

15.4.1 Extended Link Service command codes

R_CTL bits 31-28 (Word 0) are set = to 0010 to indicate an Extended Link_Data frame. The TYPE field for Extended Link Service frames shall = 0000 0001. Unless otherwise indicated for individual ELSSs, the Information Category bits shall be set to unsolicited control (0010) for request Sequences and solicited control (0011) for reply Sequences.

When R_CTL bits 31-28 and TYPE indicate Extended Link Service, the first byte of the Payload (LS_Command code) of the request or reply Sequence shall be as shown in Table 49. The

remainder of the payload is ELS unique. Subsequent frames, if any, for a request or reply Sequence shall only contain additional Payload in the Payload field (i.e., the LS_Command code is not repeated in each frame).

Table 49 - LS_Command codes				
Value (Bits 31-24)		Description	Abbr.	Reference
Binary	Hex			
0000 0001	01	Link Service Reject	LS_RJT	15.6.2
0000 0010	02	Accept	ACC	15.6.1
0000 0011	03	N_Port_Login	PLOGI	15.5.7
0000 0100	04	F_Port Login	FLOGI	15.5.7
0000 0101	05	Logout	LOGO	15.5.8
0000 0110	06	Abort Exchange	ABTX	15.5.2
0000 0111	07	Read Connection Status	RCS	15.5.9
0000 1000	08	Read Exchange Status Block	RES	15.5.10
0000 1001	09	Read Sequence Status Block	RSS	15.5.12
0000 1010	0A	Request Sequence Initiative	RSI	15.5.15
0000 1011	0B	Establish Streaming	ESTS	15.5.6
0000 1100	0C	Estimate Credit	ESTC	15.5.5
0000 1101	0D	Advise Credit	ADVC	15.5.3
0000 1110	0E	Read Timeout Value	RTV	15.5.13
0000 1111	0F	Read Link Error Status Block	RLS	15.5.11
0001 0000	10	Echo	ECHO	15.5.4
0001 0001	11	Test	TEST	15.5.16
0001 0010	12	Reinstate Recovery Qualifier	RRQ	15.5.14
0010 0000	20	Process Login	PRLI	15.5.26
0100 0001	21	Process Logout	PRLO	15.5.27
0100 0010	22	State Change Notification	SCN	15.5.28
0010 0011	23	Test Process Login State	TPLS	15.5.29
0100 0100	24	Third Party Process Logout	TPRLO	15.5.37.3
0100 0101	25	Login Control List Management	LCLM	15.5.40
0011 0000	30	Get Alias_ID	GAID	15.5.34.1
0011 0001	31	Fabric Activate Alias_ID	FACT	15.5.34.2
0011 0010	32	Fabric Deactivate Alias_ID	FDACT	15.5.34.3
0011 0011	33	N_Port Activate Alias_ID	NACT	15.5.35.1
0011 0100	34	N_Port Deactivate Alias_ID	NDACT	15.5.35.2
0100 0000	40	Quality of Service Request	QoSR	15.5.36.1
0100 0001	41	Read Virtual Circuit Status	RVCS	15.5.36.2
0101 0000	50	Discover N_Port Service Parameters	PDISC	15.5.37.1

Table 49 - LS_Command codes				
Value (Bits 31-24)		Description	Abbr.	Reference
Binary	Hex			
0101 0001	51	Discover F_Port Service Parameters	FDISC	15.5.37.1
0101 0010	52	Discover Address	ADISC	15.5.37.2
0101 0011	53	Report Node Capability	RNC	15.5.17
0101 0100	54	Fibre Channel Address Resolution Protocol Request	FARP-REQ	15.5.30.1
0101 0101	55	Fibre Channel Address Resolution Protocol Reply	FARP-REPLY	15.5.30.2
0101 0110	56	Read Port Status Block	RPS	15.5.41
0101 0111	57	Read Port List	RPL	15.5.42
0101 1000	58	Bandwidth Allocation	BALLOC	15.5.38.1
0101 1001	59	Bandwidth Deallocation	DALLOC	15.5.38.2
0110 0000	60	Fabric Address Notification	FAN	15.5.18
0110 0001	61	Registered State Change Notification	RSCN	15.5.22
0110 0010	62	State Change Registration	SCR	15.5.23
0110 0011	63	Report Node FC-4 Types	RNFT	15.5.43
0110 1000	68	Clock Synchronization Request	CSR	15.5.39.1
0110 1001	69	Clock Synchronization Update	CSU	15.5.39.2
0111 0000	70	Loop Initialize	LINIT	15.5.19
0111 0001	71	Loop Port Control	LPC	15.5.20
0111 0010	72	Loop Status	LSTS	15.5.21
0111 0111	77	Request Topology Information	RTIN	15.5.31.1
0111 1000	78	Request Node Identification Data	RNID	15.5.32.1
0111 1001	79	Registered Link Incident Record	RLIR	15.5.33.2
0111 1010	7A	Link Incident Record Registration	LIRR	0
Others	Others	Reserved		

15.5 Extended Link Service requests

A Sequence Initiator shall transmit a Link Service Sequence in order to solicit the destination N_Port to perform a link-level function or service. Unless otherwise noted, Extended Link Service requests may be issued prior to completion of N_Port_Login.

If an Extended Link Service request Sequence that has a reply Sequence is transmitted without the transfer of Sequence Initiative, the Recipient shall abort the Exchange and not perform the request. An Extended Link Service Protocol is composed of an Extended Link Service request Sequence and, for some requests, an Extended Link Service reply Sequence. The last Data frame of an Extended Link Service request Sequence that has a reply Sequence shall transfer the Sequence Initiative to the Recipient in order to allow the reply to be transmitted (see clause 18).

The following Extended Link Service requests have no reply Sequence:

- Estimate Credit

- Test
- Clock Synchronization Update (CSU)

The following Extended Link Service requests and their replies shall be supported by a N_Port (others are optional):

- FLOGI
- PLOGI
- LOGO
- RRQ (Class 2 and 3 only)

A N_Port is not required to generate and send the PLOGI Extended Link Service request. However, if a N_Port receives PLOGI Extended Link Service request, the N_Port shall respond with the specified Link Service reply (ACC), or if PLOGI frame has invalid information, shall respond with LS_RJT. LS_RJT shall not be issued for the reason "PLOGI command not supported".

NOTE - If a N_Port which does not generate PLOGI, is in a point-to-point topology, and has a N_Port_Name greater than the other N_Port's, the other N_Port may timeout, waiting to receive PLOGI.

All Extended Link Service requests, excluding ESTS, ESTC, and ADVC, and the corresponding replies shall be performed within a single Exchange (see 17.7.4.9 for the procedure using ESTS, ESTC, and ADVC). The Advise Credit request may also be performed in a separate Exchange.

The Originator of the Link Service request shall assign an OX_ID when the request Sequence and Exchange are originated. If the only Exchange, which the Responder has, Open with the Originator's N_Port is the one for the Link Service being performed, a value of hex 'FF FF' may be used. The Responder shall assign an RX_ID when the request Sequence is received. A value of hex 'FF FF' may be used if only one Exchange is Open with the Originator.

Normal Sequence and Exchange management rules shall apply. See clause 18 for more information regarding use of F_CTL bits for Exchange and Sequence control as well as OX_ID, RX_ID, SEQ_ID, and SEQ_CNT.

Payload

Link Service requests and Link Service reply frames use an FT_1 frame type. The Payload shall contain a multiple of four bytes of data based on the individual Link Service request or reply. If required, more than one frame may be used to form a request or reply Sequence.

15.5.1 Link Service request collisions

There are two Extended Link Service request Sequences in which a collision is possible with the other N_Port involved in the same target Exchange. Those two requests are Abort Exchange (ABTX) and Request Sequence Initiative (RSI). For example, N_Port (A) may transmit an ABTX request to N_Port (B) at the same time that N_Port (B) transmits an ABTX request to N_Port (A) for the same target Exchange.

If such an instance occurs, the Originator N_Port of the target Exchange shall reject the ABTX or RSI request Sequence with an LS_RJT with a reason code of Command already in progress. The Responder N_Port of the target Exchange shall honor and process the ABTX or RSI request Sequence normally.

15.5.2 Abort Exchange (ABTX)

The Abort Exchange Link Service request shall be used to request abnormal termination of an Open Exchange, in progress. A request to abort an Exchange shall only be accepted if the request is made by the Originator N_Port or the Responder N_Port of the target Exchange.

A separate Exchange shall be used to abort the existing Exchange. The Payload shall contain the OX_ID and RX_ID for the Exchange being aborted, as well as the S_ID of the N_Port that originated the Exchange.

Transmission of an ABTX frame is allowed while the identified Exchange is Open (see 18.3.14). Both the Originator and Responder shall ensure that the OX_ID and RX_ID pair being terminated are currently associated with the OX_ID and RX_ID pair specified in the ABTX request.

If an RX_ID other than hex 'FF FF' has not been received by the Sequence Initiator or the First Sequence of the Exchange has not completed successfully, the Abort Sequence Protocol shall be performed prior to attempting to Abort the Exchange using ABTX (see 23.7.1.1). This ensures that the Exchange Status Block exists and that both the OX_ID and RX_ID are assigned and known to both the Originator and Responder.

In Class 1, both the OX_ID and RX_ID are available by the respective N_Ports for reuse after the Exchange has been successfully aborted (Accept transmitted and received). If all frames are not accounted for in Class 2 or 3, a Recovery_Qualifier shall be established with a low SEQ_CNT of binary zero and a high SEQ_CNT of hex 'FF FF'.

The N_Port which issues the ABTX request shall set the Recovery_Qualifier status in the Payload of the ABTX request to indicate whether it believes a Recovery_Qualifier is required or not required. If the ABTX indicates that a Recovery_Qualifier is not required, the Recipient of the ABTX may agree by transmitting an Accept reply Sequence or it may disagree by transmitting an LS_RJT with the reason code of Recovery_Qualifier required.

If the ABTX requires a Recovery_Qualifier in its Payload and it is Accepted by the destination N_Port, then the N_Port which requested the ABTX shall transmit a Reinstate Recovery Qualifier (RRQ) Extended Link Service request following an R_A_TOV timeout in order to free those Exchange resources.

Each N_Port abnormally terminates any Active or Open Sequences associated with the Exchange. The ACC reply confirms that all Sequences and the Exchange have been abnormally terminated. It also confirms the presence or absence of a Recovery_Qualifier. If ABTX is used, then the use of ABTS to precede or follow ABTX is optional.

The addressing is fully specified by the D_ID and S_ID fields.

Protocol:

- Abort Exchange request Sequence
- Accept reply Sequence

Format: FT_1

Addressing: The D_ID field designates the destination N_Port of the Exchange being aborted while the S_ID field designates the source N_Port which is requesting that the Exchange be aborted.

X_ID: A separate and distinct Exchange shall be required other than the Exchange being aborted in order to properly track status.

SEQ_ID, SEQ_CNT: The SEQ_ID and the SEQ_CNT shall be appropriate for an Active Sequence.

Payload: The format of the Payload is shown in Table 50.

Table 50 - ABTX Payload	
Item	Size -Bytes
hex '06 00 00 00'	4
Recovery_Qualifier status hex '00' no Recovery Qualifier hex '80' Recovery_Qualifier required	1
Originator S_ID	3
OX_ID	2
RX_ID	2
Association_Header (optionally required)	32

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the ABTX command (see 15.6.2)

Accept (ACC)

- signifies that the destination N_Port has terminated the Exchange.
- The format of the Accept Payload is shown in Table 51.

Table 51 - ABTX Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4

15.5.3 Advise Credit (ADVC)

The ADVC Link Service request shall be used to advise the destination N_Port of the estimated end-to-end Credit that the source N_Port requests to be allocated. The ADVC Link Service request shall be a separate Sequence. It may also be requested in a separate Exchange. See 17.8.2.3 for the usage of this frame. The ADVC request may also be used independently from the Estimate Credit procedure (see 17.8). If the recipient N_Port or NL_Port is not logged in with the requesting port it shall reply with an LS_RJT ELS Sequence with a reason code of 'unable to perform command request' with a reason code of 'N_Port_Login Required.'

Protocol:

- Advise Credit request Sequence
- Accept reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting Credit revision. The D_ID field designates the destination N_Port.

Payload: The format of the Payload is shown in Table 52. The Payload shall contain estimated Credit (M+1) in the end-to-end Credit field of the appropriate Class Service Parameters (see 17.6) as indicated by the Class Validity bit. The Class Validity bit = 1 for each Class Service Parameters of the ADVC Payload identifies the Class for which a revised end-to-end Credit is requested. The receiver shall ignore the other Service Parameter fields.

Table 52 - ADVC Payload	
Item	Size -Bytes
hex '0D 00 00 00'	4
Common Service Parameters	16
N_Port_Name	8
Node_Name	8
Class 1 Service Parameters	16
Class 2 Service Parameters	16
Class 3 Service Parameters	16
Reserved	16
Vendor Version Level	16

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the ADVC command (see 15.6.2)

Accept (ACC)

- signifies successful completion of the ADVC function and permanently replaces the end-to-end Credit in effect for the current N_Port_Login.
- Accept Payload

The format of the Accept Payload is shown in Table 53. The Payload shall contain the revised end-to-end Credit allocated in the Credit field for the appropriate Class Service Parameters as indicated by the Class Validity bit. The revised end-to-end Credit shall replace the end-to-end Credit for the current Login for the N_Port transmitting the Accept Sequence (see 17.6.8.7). The Class Validity bit = 1 for each Class Service Parameter group of the ADVC Payload identifies each Class for which the end-to-end Credit is updated or revised. The receiver shall ignore the other Service Parameter fields. This revised end to-end Credit value is determined by the destination N_Port based on its buffering scheme, buffer management, buffer availability, and N_Port processing time. See 17.8.2.3 for the determination of this value.

Table 53 - ADVC Accept Payload	
Item	Size - Bytes

hex '02 00 00 00'	4
Common Service Parameters	16
N_Port_Name	8
Node_Name	8
Class 1 Service Parameters	16
Class 2 Service Parameters	16
Class 3 Service Parameters	16
Reserved	16
Vendor Version Level	16

15.5.4 Echo (ECHO)

The Echo Extended Link Service request Sequence shall consist of a single frame requesting the Recipient to transmit the Payload contents, following the LS_Command, back to the Initiator of the Echo command in the same order as received using the ACC reply Sequence consisting of a single frame. The Echo frame shall indicate End_Sequence and Sequence Initiative transfer as well as other appropriate F_CTL bits. The Echo Extended Link Service request provides a means to transmit a Data frame and have the Payload content returned for a simple loop-back diagnostic function. The Echo command shall be transmitted as a one frame Sequence and the ACC reply Sequence is also a one frame Sequence. The Echo Protocol shall be transmitted as an Exchange that is separate from any other Exchange. The Echo Protocol is applicable to Class 1, 2, and 3.

Protocol:

- Echo request Sequence
- Accept reply Sequence

Format: FT_1

Addressing: The D_ID field designates the destination of the request while the S_ID field designates the source of the request.

Payload: The format of the Payload is shown in Table 54. The Payload size is limited by the smallest Data Field size supported by the destination N_Port, the Fabric, and the source destination N_Port for the Class of Service being used since the Accept frame shall be equal in size to the Echo Data Field size.

Table 54 - ECHO Payload	
Item	Size - Bytes
hex '10 00 00 00'	4
ECHO data	max frame

Reply Sequence:

Service Reject (LS_RJT)

- signifies rejection of the ECHO command (see 15.6.2)

Accept (ACC)

- signifies successful completion of the ECHO function.
- Accept Payload

The format of the Accept Payload is shown in Table 55. The Payload shall contain the ECHO data contained in the Payload of the ECHO request frame.

Table 55 - ECHO Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4
ECHO data	Echo payload - 4

15.5.5 Estimate Credit (ESTC)

The ESTC Link Service request shall be used to estimate the minimum Credit required to achieve the maximum bandwidth for a given distance between a N_Port pair. The ESTC Link Service request shall have the frame size as determined by Login with the destination N_Port. If the recipient N_Port or NL_Port is not logged in with the requesting port it shall reply with an LS_RJT ELS Sequence with a reason code of 'unable to perform command request' with a reason code of 'N_Port_Login Required.'

The Class of the SOF delimiter of the ESTC request identifies the Class for which Credit is being estimated. The destination N_Port shall acknowledge Data frames as specified by its Login parameters. See 17.8.2.2 for the usage of this frame.

Protocol:

- Estimate Credit request Sequence
- No reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting the Credit estimate. The D_ID field designates the destination N_Port specified in the Establish Streaming frame.

Payload: The format of the Payload is shown in Table 56. The first word of the Payload of the first frame of the Sequence shall contain the LS_Command code. The remainder of the Payload shall be the size as determined by Login. The content of the Payload after the LS_Command and for subsequent frames shall be valid data bytes.

Table 56 - ESTC Payload	
Item	Size - Bytes
hex '0C 00 00 00'	4
Any data	max

Reply Link Service Sequence: None.

15.5.6 Establish Streaming (ESTS)

The ESTS Link Service request Sequence requests a temporary allocation of Credit known as Streaming Credit large enough to perform continuous streaming of Data frames. The SOF delimiter of the ESTC request identifies the Class for which Credit is being estimated. See 17.8.2.1 for the usage of this frame.

If the recipient N_Port or NL_Port is not logged in with the requesting port it shall reply with an LS_RJT ELS Sequence with a reason code of 'unable to perform command request' with a reason code of 'N_Port_Login Required.'

Protocol:

- Establish Streaming request Sequence
- Accept reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting Streaming. The D_ID field designates the destination N_Port addressed.

Payload: The format of the Payload is shown in Table 57.

Table 57 - ESTS Payload	
Item	Size - Bytes
hex '0B 00 00 00'	4

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the ESTS command (see 15.6.2)

Accept (ACC)

- signifies successful completion of the ESTS function.
- Accept Payload

The format of the Accept Payload is shown in Table 58. The Payload shall contain Streaming Credit (L) allocated in the Credit field of the appropriate Class Service Parameters. The Class Validity bit = 1 identifies the Class which contains the Streaming Credit (L). The receiver shall ignore the other Service Parameter fields.

Table 58 - ESTS Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4
Common Service Parameters	16

N_Port_Name	8
Node_Name	8
Class 1 Service Parameters	16
Class 2 Service Parameters	16
Class 3 Service Parameters	16
Reserved	16
Vendor Version Level	16

15.5.7 Login (FLOGI/PLOGI)

The Login Link Service request shall transfer Service Parameters from the initiating N_Port to the N_Port or F_Port associated with the Destination Identifier. The PLOGI frame provides the means by which a N_Port may request Login with another N_Port prior to other Data frame transfers. The FLOGI sequence provides the means by which a N_Port may request Login with the Fabric (see 17.1). The term LOGI shall be considered to be equivalent to PLOGI or FLOGI.

The interchange of Login information shall establish the operating environment between the two N_Ports or between a N_Port and F_Port. Three Classes of Service are available depending on the support provided by the Fabric, if present. See 17.6 for a definition of Service Parameters.

In order to Login with the Fabric and determine the Fabric operating characteristics, a N_Port shall specify the Destination Identifier as the well-known F_Port Identifier (i.e., hex 'FF FF FE').

In order to direct the Login Link Service frame to a Fibre Channel Service, a N_Port shall specify the appropriate well-known Address Identifier (see Table 11 in 12.3).

When a N_Port receives a Login from a N_Port all Active or Open Sequences with the N_Port performing Rlogin shall be abnormally terminated.

Protocol:

- Login request Sequence
- Accept reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting Login. If unidentified, as in Fabric Login, binary zeros are used. The D_ID field designates the destination N_Port or F_Port of the Login.

Payload: The format of the Payload is shown in Table 59.

Table 59 - LOGI Payload	
Item	Size - Bytes
hex '03 00 00 00' for PLOGI hex '04 00 00 00' for FLOGI	4
Common Service Parameters	16

N_Port_Name	8
Node_Name	8
Class 1 Service Parameters	16
Class 2 Service Parameters	16
Class 3 Service Parameters	16
Class 4 Service Parameters	16
Vendor Version Level	16
Services Availability (see note)	8
Reserved (see note)	116
NOTE - These fields are only present when the Payload Length bit (see 17.6.2.3) is set to 1. When the Payload Length bit is set to 0, these fields are not present in the payload, i.e., the payload is 116 bytes long.	

Service Parameters are defined in 17.6.

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the LOGI command (see 15.6.2)

Accept (ACC)

- signifies successful completion of the LOGI command
- Accept Payload

The format of the Accept Payload is shown in Table 60 (see 17.6).

Table 60 - LOGI Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4
Common Service Parameters	16
N_Port_Name	8
Node_Name or Fabric_Name	8
Class 1 Service Parameters	16
Class 2 Service Parameters	16
Class 3 Service Parameters	16
Class 4 Service Parameters	16
Vendor Version Level	16
Services Availability (see note)	8
Reserved (see note)	116

NOTE - These fields are only present when the Payload Length bit (see 17.6.2.3) is set to 1. When the Payload Length bit is set to 0, these fields are not present in the payload, i.e., the payload is 116 bytes long.

15.5.8 Logout (LOGO)

The Logout Link Service request shall request invalidation of the Service Parameters and Port_Name that have been saved by a N_Port, freeing those resources. This provides a means by which a N_Port may request Logout or remove service between two N_Ports. (see 17.5)

The N_Port Identifier and Port_Name of the N_Port requesting Logout are identified in the Payload. This allows a N_Port to Logout its old Identifier using a new Identifier after its native N_Port Identifier has changed. Both the Source N_Port and the Destination N_Port of the Logout request Sequence shall abnormally terminate all Open Exchanges that used the N_Port identifier indicated in the Payload of the Logout request Sequence.

Protocol:

- Logout request Sequence
- Accept reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting Logout. The D_ID field designates the destination N_Port of the Logout request.

Payload: The format of the Payload is shown in Table 61.

Table 61 - LOGO Payload	
Item	Size - Bytes
hex '05 00 00 00'	4
Reserved	1
N_Port Identifier	3
Port_Name	8

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the LOGO command (see 15.6.2)

Accept (ACC)

- signifies that Service has been removed.
- Accept Payload

The format of the Accept Payload is shown in Table 62.

Table 62 - LOGO Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4

15.5.9 Read Connection Status (RCS)

The RCS Link Service request Sequence requests the Fabric Controller to return the current Dedicated Connection status for the N_Port specified in the Payload of the RCS frame. The RCS request provides the means by which a N_Port may interrogate the Fabric for the Connection status of other N_Ports within the Fabric.

In order to direct the RCS Link Service request frame to the Fabric, a N_Port specifies the Destination Identifier as a well-known Fabric Controller address (i.e., hex 'FF FF FD').

Protocol:

- Read Connection Status request Sequence
- Accept (ACC) reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting Connection status. The D_ID field is the Fabric Controller, hex 'FF FF FD'.

Payload: The format of the Payload is shown in Table 63. The first word of the Payload shall contain the LS_Command code. The second word shall contain the N_Port address identifier for which Connection status is being requested.

Table 63 - RCS Payload	
Item	Size - Bytes
hex '07 00 00 00'	4
reserved	1
N_Port_Identifier	3

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the RCS command (see 15.6.2)

Accept (ACC)

- signifies that the Fabric has completed the request.
- Accept Payload

The format of the Accept Payload is shown in Table 64.

Table 64 - RCS Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4
Connection Status (Word 1, bits 31-24)	1
N_Port Identifier (Word 1, bits 23-0)	3

Connection Status Codes:

- **Bit 31 - Connect-request delivered**

If bit 31 is zero, the specified N_Port is either not Connected, or is involved in an Established Connection based on the setting of bit 29. If bit 31 is one, a connect-request has been delivered to the specified N_Port, but the N_Port has not yet responded with a proper response frame and a Dedicated Connection does not yet exist.

- **Bit 30 - Connect-request stacked**

If bit 30 is zero, no connect-request is stacked for the specified N_Port on behalf of the requesting N_Port. If bit 30 is one, one or more connect-requests are stacked, but have not been delivered to the specified N_Port on behalf of the requesting N_Port.

- **Bit 29 - Connection established**

If bit 29 is zero, the specified N_Port in the RCS request is not in a Dedicated Connection. If bit 29 is one, the specified N_Port is involved in a Dedicated Connection. When bit 29 is one, the address identifier in bits 23-0 identifies the other N_Port involved in the Dedicated Connection.

- **Bit 28 - Intermix mode**

If bit 28 is zero, the N_Port specified in the RCS frame is not functioning in Intermix mode. If bit 28 is one, the N_Port specified in the request is functioning in intermix mode. A N_Port is functioning in intermix mode if both the N_Port and the F_Port have both previously indicated that each supports intermix during Login. See 17.6 and 16.4 for more discussion of intermix.

- **Bits 23 - 0**

Bits 23 through 0 specify the address identifier of the N_Port involved in a Dedicated Connection with the other N_Port specified by the RCS Link Service request Sequence frame (i.e., bit 29 = 1).

15.5.10 Read Exchange Status Block (RES)

The RES Link Service request Sequence requests a N_Port to return the Exchange Status Block for the RX_ID or OX_ID originated by the S_ID specified in the Payload of this request Sequence. The specification of OX_ID and RX_ID may be useful or required information for the destination N_Port to locate the status information requested. A Responder destination N_Port would use RX_ID and ignore the OX_ID. Similarly, the Originator would use the OX_ID and ignore the RX_ID. This provides the N_Port transmitting the request with information regarding the current status of the Exchange specified.

If the destination N_Port of the RES request determines that the SEQ_ID, Originator S_ID, OX_ID, or RX_ID are inconsistent, then it shall reply with an LS_RJT Sequence with a reason code that it is unable to perform the command request.

Protocol:

- Read Exchange Status request Sequence
- Accept (ACC) reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting the Exchange Status Block. The D_ID field designates the destination N_Port to which the request is being made.

Payload: The format of the Payload is shown in Table 65.

Table 65 - RES Payload	
Item	Size - Bytes
hex '08 00 00 00'	4
reserved	1
Originator S_ID	3
OX_ID	2
RX_ID	2
Association_Header (optional)	32

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the RES command (see 15.6.2)

Accept (ACC)

- signifies that the N_Port has transmitted the requested data.
- Accept Payload

The format of the Accept Payload is shown in Table 66. The format of the Exchange Status Block is specified in Table 209.

Table 66 - RES Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4
Exchange Status Block (see Table 209)	N
Association_Header (optional)	32

15.5.11 Read Link Error Status Block (RLS)

The RLS Link Service request Sequence requests a N_Port or F_Port to return the Link Error Status Block associated with the Port Identifier specified in the Payload of this frame. This provides the N_Port transmitting the request with information regarding Link Errors detected

within the destination N_Port or F_Port. If a F_Port does not support the Link Error Status Block, it shall reply with an LS_RJT with a reason code of unable to perform command request.

Protocol:

- Read Link Error Status request Sequence
- Accept (ACC) reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting the Link Error Status Block. The D_ID field designates the destination N_Port or F_Port (FFFFFE) to which the request is being made.

Payload: The format of the Payload is shown in Table 67.

Table 67 - RLS Payload	
Item	Size - Bytes
hex '0F 00 00 00'	4
reserved	1
Port Identifier	3

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the RLS command (see 15.6.2)

Accept (ACC)

- signifies that the N_Port has transmitted the requested data.
- Accept Payload:

The format of the Accept Payload is shown in Table 68. The format of the Link Error Status Block is specified in Figure 65.

Table 68 - RLS Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4
Link Error Status Block (see Figure 65)	24

15.5.12 Read Sequence Status Block (RSS)

The RSS Link Service request Sequence requests a N_Port to return the Sequence Status Block for the SEQ_ID specified in the Payload of this frame. The Payload also specifies the S_ID of the Exchange Originator as well as the associated OX_ID and RX_ID. The specification of OX_ID and RX_ID may be useful or required information for the destination N_Port to locate the status information requested. A Responder destination N_Port may use RX_ID and

ignore the OX_ID. Similarly, the Originator would use the OX_ID and ignore the RX_ID. This provides the N_Port transmitting the request with information regarding the current status of the Sequence it identified.

If the destination N_Port of the RSS request determines that the SEQ_ID, Originator S_ID, OX_ID, or RX_ID are inconsistent then it shall reply with an LS_RJT Sequence with a reason code that it is unable to perform the command request.

Protocol:

- Read Sequence Status request Sequence
- Accept (ACC) reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting the Sequence Status Block. The D_ID field designates the destination N_Port to which the request is being made.

Payload: The format of the Payload is shown in Table 69.

Table 69 - RSS Payload	
Item	Size - Bytes
hex '09 00 00 00'	4
SEQ_ID	1
Originator S_ID	3
OX_ID	2
RX_ID	2

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the RSS command (see 15.6.2)

Accept (ACC)

- signifies that the N_Port has transmitted the requested data.
- Accept Payload:

The format of the Accept Payload is shown in Table 70. The format of the Sequence Status Block is specified in 18.8.2.

Table 70 - RSS Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4
Sequence Status Block (see 18.8.2)	N

15.5.13 Read Timeout Value (RTV)

The RTV Link Service request Sequence requests a N_Port or F_Port (hex 'FF FF FE') to return the Resource_Allocation_Timeout Value (R_A_TOV) and the Error_Detect_Timeout Value (E_D_TOV) in the Accept reply Sequence. This provides the N_Port transmitting the request with information regarding these values from another N_Port or from the F_Port. Usage of E_D_TOV and R_A_TOV is discussed in 23.2.1.

Protocol:

- Read Timeout Value (RTV) request Sequence
- Accept (ACC) reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting the timeout interval values. The D_ID field designates the destination N_Port or F_Port to which the request is being made.

Payload: The format of the Payload is shown in Table 71.

Table 71 - RTV Payload	
Item	Size - Bytes
hex '0E 00 00 00'	4

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the RTV command (see 15.6.2)

Accept (ACC)

- signifies that the N_Port or F_Port has transmitted the requested data.
- Accept Payload

The format of the Accept Payload is shown in Table 72. Timeout values are specified as a count of either 1 ms or 1 ns increments, depending on the setting of Bit 26 in the Timeout Qualifier.

Table 72 - RTV Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4
Resource_Allocation_Timeout Value (R_A_TOV) (see 23.2.1)	4
Error_Detect_Timeout Value (E_D_TOV) (see 23.2.1)	4
Timeout Qualifier (see 17.6.3.7)	4

R_A_TOV: In a point-to-point topology, this value shall have the same resolution as E_D_TOV, as indicated by the Timeout Qualifier.

NOTE - In a point-to-point topology, R_A_TOV must be twice E_D_TOV, which means it must have the same resolution. In other topologies, R_A_TOV will always be in 1 ms increments.

Timeout Qualifier:

- **Bits 31-27:** Reserved
- **Bit 26:** E_D_TOV Resolution

If Bit 26 is zero, the value specified in the E_D_TOV field shall indicate a count of 1 ms increments. If Bit 26 is one, the value specified in the E_D_TOV field shall indicate a count of 1 ns increments.

- **Bits 25-0:** Reserved

15.5.14 Reinstate Recovery Qualifier (RRQ)

The Reinstate Recovery Qualifier Link Service request shall be used to notify the destination N_Port that the Recovery_Qualifier shall be available for reuse. The Recovery_Qualifier (S_ID, D_ID, OX_ID, RX_ID, and low SEQ_CNT - high SEQ_CNT) shall be associated with an Exchange in which the Abort Sequence or Abort Exchange was previously performed.

In the case of Abort Exchange, the ESB and Recovery_Qualifier are immediately available for reuse. In the case of Abort Sequence Protocol, the Recovery_Qualifier is purged. A request to Reinstate the Recovery_Qualifier shall only be accepted if the Originator N_Port or the Responder N_Port of the target Exchange makes the request.

A separate Exchange shall be used to reinstate the Recovery_Qualifier. The Payload shall contain the OX_ID and RX_ID for the Exchange Recovery_Qualifier, as well as the S_ID of the N_Port that originated the Exchange. Resources associated with the OX_ID in the Originator, and with the RX_ID in the Responder, shall be released following transmission and reception of the Accept reply Sequence if the Exchange had been aborted with ABTX.

Both the Originator and Responder shall ensure that the OX_ID and RX_ID pair being terminated are currently associated with the OX_ID and RX_ID pair specified in the RRQ request.

The Recovery_Qualifier range shall be timed out for an R_A_TOV timeout period (i.e., RRQ shall not be transmitted until an R_A_TOV timeout period after BA_ACC for ABTS or ACC for ABTX has been received) by the N_Port which transmitted and successfully completed either the ABTX or the ABTS frame.

The addressing is fully specified by the D_ID and S_ID fields.

Protocol:

- Reinstate Recovery Qualifier request Sequence
- Accept reply Sequence

Format: FT_1

Addressing: The D_ID field designates the destination N_Port of the RRQ request Sequence while the S_ID field designates the source N_Port which is requesting that the Recovery_Qualifier be reinstated.

X_ID: A separate and distinct Exchange is required.

SEQ_ID, SEQ_CNT: The SEQ_ID and the SEQ_CNT shall be appropriate for an Active Sequence.

Payload: The format of the Payload is shown in Table 73.

Table 73 - RRQ Payload	
Item	Size - Bytes
hex '12 00 00 00'	4
reserved	1
Originator S_ID	3
OX_ID	2
RX_ID	2
Association_Header (optional)	32

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the RRQ command (see 15.6.2)

Accept (ACC)

- signifies that the destination N_Port reinstated the Recovery_Qualifier.
- Accept Payload

The format of the Accept Payload is shown in Table 74.

Table 74 - RRQ Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4

15.5.15 Request Sequence Initiative (RSI)

The Request Sequence Initiative Link Service request shall be used to request that Sequence Initiative be passed to the Sequence Recipient of an Exchange in progress. A request to pass Sequence Initiative shall only be accepted if the request is made by the Originator N_Port or the Responder N_Port of the target Exchange. A separate Exchange shall be used to perform the Request Sequence Initiative. The Payload shall contain the OX_ID and RX_ID for the target Exchange, as well as the S_ID of the N_Port that originated the Exchange. The Accept reply is sent subsequent to the transfer of Sequence Initiative on the target Exchange.

Transmission of RSI is allowed while the identified Exchange is Open. Both the Originator and Responder shall ensure that the OX_ID and RX_ID pair for which Sequence Initiative is being passed are currently associated with the OX_ID and RX_ID pair specified in the RSI request.

If there are any Sequences Active for the target Exchange, the Sequence Initiator of the Active Sequence of the target Exchange shall terminate them and transfer Sequence Initiative as follows:

- If there is an Active Sequence for which the last Data frame has not been transmitted, the Sequence Initiator of the target Exchange shall terminate the Sequence by transmitting a Data frame with the End_Sequence and Sequence Initiative bits set to 1.
- If there are no Data frames to be sent for the Active Sequence, the Sequence Initiator of the target Exchange shall transmit a NOP Basic Link Service frame with the End_Sequence and Sequence Initiative bits set to 1 in F_CTL.

If there are no Sequences Active, the Sequence Initiator of the target Exchange shall transfer Sequence Initiative by initiating a new Sequence consisting of a single NOP Basic Link Service frame (a one frame Sequence) with the End_Sequence and Sequence Initiative bits set to 1 in F_CTL.

The Accept to the Exchange requesting Sequence Initiative shall be transmitted after Sequence Initiative has been passed (see 18.6.4) on the target Exchange.

The addressing is fully specified by the D_ID and S_ID fields.

Protocol:

- Request Sequence Initiative request Sequence
- Accept reply Sequence

Format: FT_1

Addressing: The D_ID field designates the destination N_Port of the Exchange for which Sequence Initiative is being requested and the S_ID field designates the source N_Port that is requesting Sequence Initiative.

X_ID: A separate and distinct Exchange is required other than the Exchange for which Sequence Initiative is being requested in order to properly track status.

SEQ_ID, SEQ_CNT: The SEQ_ID and the SEQ_CNT shall be appropriate for an Active Sequence.

Payload: The format of the Payload is shown in Table 75.

Table 75 - RSI Payload	
Item	Size - Bytes
hex '0A 00 00 00'	4
reserved	1
Originator S_ID	3
OX_ID	2
RX_ID	2
Association_Header (optionally required)	32

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the RSI command (see 15.6.2)

Accept (ACC)

- signifies that the destination N_Port has transferred the Sequence Initiative for the target Exchange.
- Accept Payload

The format of the Accept Payload is shown in Table 76.

Table 76 - RSI Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4

15.5.16 Test (TEST)

The Test Extended Link Service request Sequence shall consist of a single Sequence being transmitted from the Sequence Initiator to the Sequence Recipient. The Test request may be used in diagnostic or testing procedures to provide system loading. There is no reply Sequence. The Payload may consist of any frame size up to the maximum allowable for the Class and other normal Sequence and frame limitations. The Test Link Service request is applicable to Class 1, 2, and 3.

Protocol: Test request Sequence

Format: FT_1

Addressing: The D_ID field designates the destination of the request while the S_ID field designates the source of the request.

Payload: The format of the Payload is shown in Table 77.

Table 77 - TEST Payload	
Item	Size - Bytes
hex '11 00 00 00'	4
TEST data	variable

The Payload size is limited by the smallest Data Field size supported by the destination N_Port and the Fabric for the Class of Service being used.

Reply Sequence: none

15.5.17 Report Node Capability Information (RNC)

New implementations shall not rely on support of RNC. For a description of RNC see FC-PH-3. For an ELS that reports FC-4 types, see 15.5.43.

15.5.18 Fabric Address Notification (FAN)

The Fabric Address Notification Extended Link Service is sent by the Fabric F_Port (well-known address hex 'FF FF FE') to all known previously logged in (FLOGI) and attached ports following an initialization event. (This initialization event is typically Loop Initialization on an Arbitrated Loop, though other events that may cause a port to change its ID may also be considered.) The Fabric F_Port shall report the Fabric_Port_Name and Fabric_Name as they were reported in the prior login (FLOGI), and shall report the current Loop Fabric Address if the fabric attach is a FL_Port. The Fabric F_Port shall send this service using the default login parameters; i.e., the parameters that are in effect prior to a FLOGI request.

The attached ports may use this information to authenticate active Exchanges and operating parameters (e.g., login BB_Credit).

The Fabric F_Port shall report identical information to all attached ports. If the information changes in any way before the Fabric F_Port is able to send the service to all attached ports, the Fabric F_Port shall initiate a new initialization event.

The attached ports shall not initiate an Accept Reply Sequence to this Extended Link Service. A Service Reject may be initiated if the port does not recognize or support this service.

Protocol:

- Fabric Address Notification request Sequence
- No Reply Sequence

Format: FT-1

Addressing: The S_ID is the Fabric F_Port (hex 'FF FF FE'). The D_ID is the N_Port or NL_Port receiving the notification.

Payload: The format of the FAN request Payload is shown in Table 78.

Table 78 - FAN Payload	
Item	Size - Bytes
hex '60 00 00 00'	4
Reserved	1
Loop Fabric Address	3
Fabric_Port_Name	8
Fabric_Name	8

15.5.19 Loop Initialize (LINIT)

The Loop Initialize Extended Link Service requests the initialization of Loop Initialization on a designated loop. The FL_Port connected to the loop addressed by the Loop Fabric Address shall originate a LIP of the type specified in the Payload and enter the INITIALIZING Arbitrated Loop state. The Accept Reply Sequence shall not be initiated before the FL_Port has transitioned from the OPEN-INIT to MONITORING state, and has completed sending any FAN ELS.

Protocol:

- Loop Initialize request Sequence

- Accept (ACC) Reply Sequence

Format: FT-1

Addressing: The S_ID designates the N_Port or NL_Port requesting Loop Initialization of the loop. The D_ID field shall be the Loop Fabric Address (LFA) of the loop to be initialized.

Payload: The format of the LINIT request Payload is shown in Table 79.

Table 79 - LINIT Payload	
Item	Size Bytes
hex '70 00 00 00'	4
Reserved	1
Initialization Function	1
LIP Byte 3	1
LIP Byte 4	1

Initialization Function: This field defines modifications of the initialization to be performed. The format of the Initialization Function is shown in Table 80.

Table 80 - Initialization Function	
Function	Value
Normal Initialization - The fabric determines the best method by which to complete the initialization.	0
Force Login - The L_bit shall be set in the LISA Sequence to force all affected NL_Ports to perform FLOGI.	1
Reserved	2 - 255

LIP Byte 3: The 3rd character of the LIP Primitive Sequence to be originated.

LIP Byte 4: The 4th character of the LIP Primitive Sequence to be originated.

NOTE – LIP Byte 3 and LIP Byte 4 should only be set to values permitted by FC-AL or FC-AL-2.

Reply Extended Link Service Sequence:

Service Reject (LS_RJT)

- Signifies the rejection of the LINIT command

Accept (ACC)

- Signifies acceptance of the LINIT request and initiation of Loop Initialization.
- Accept Payload

The format of the Accept Payload is shown in Table 81.

Table 81 - LINIT Accept Payload	
Item	Size Bytes
hex '02 00 00 00'	4
Reserved	3
Status	1

The format of the Status field is shown in Table 82.

Table 82 - LINIT Status	
State	Value
Reserved	0
Success - The requested function was completed.	1
Failure - The requested function could not be completed.	2
Reserved	3-255

15.5.20 Loop Port Control (LPC)

The Loop Port Control Extended Link Service requests the Loop Bypass and/or Loop Enable of specific L_Ports on a designated loop. The FL_Port connected to the loop addressed by the Loop Fabric Address shall gain access to the loop in the manner selected by the requester, and originate LPB and/or LPE Primitive Sequences to the AL_PAs specified in the Payload.

The Accept Reply Sequence shall not be initiated before the FL_Port has either originated all requested LPBs and LPEs and left the loop in the state selected by the requester, or has detected a failure to deliver any LPB or LPE.

Protocol:

- Loop Port Control request Sequence
- Accept (ACC) Reply Sequence

Format: FT-1

Addressing: The S_ID designates the N_Port or NL_Port requesting the bypass and/or enable of Loop Devices on a loop. The D_ID field shall be the Loop Fabric Address (LFA) of the loop with attached L_Ports to be bypassed and/or enabled.

Payload: The format of the LPC request Payload is shown in Table 83.

Table 83 - LPC Payload	
Item	Size Bytes
hex '71 00 00 00'	4
Reserved	3
Port Control Function	1
Loop Devices to Bypass (LPB) (AL_PA bit map)	16
Loop Devices to Enable (LPE) (AL_PA bit map)	16

– **Port Control Function:**

- The format of the Port Control Function field is shown in Table 84. It is a code that indicates how to perform the bypass function.
- The “Normal Bypass” function requests the FL_Port to Arbitrate for the loop, win, send OPN(0,0), receive OPN(0,0), and then issue the requested LPB and LPE Primitive Sequences as defined in FC-AL. After all LPB and LPE Primitive Sequences have been delivered the FL_Port shall issue a CLS and return to the MONITORING state. If the FL_Port is unable to successfully deliver any requested LPB or LPE Primitive Sequence within two AL_TIME delays (transmit the Primitive Sequence and receive it back), the FL_Port shall attempt to return to the MONITORING state and indicate Failure status.
- The “Initialize and Bypass” function requests the FL_Port to enter the INITIALIZING state (send at least 12 LIP(F7,F7) Primitive Sequences), and then issue the requested LPB and LPE Primitive Sequences as defined in FC-AL. After all LPB and LPE Primitive Sequences have been delivered the FL_Port shall re-enter the INITIALIZING state (send at least 12 LIP(F7,F7) Primitive Sequences), and then attempt to complete Loop Initialization. If the FL_Port is unable to successfully deliver any requested LPB or LPE Primitive Sequence within two AL_TIME delays (transmit the Primitive Sequence and receive it back), the FL_Port shall continue to attempt to complete Loop Initialization and indicate failure status.

– **Loop Devices to Bypass:**

- The format of the Loop Devices to Bypass field follows the AL_PA bit mapped format defined in FC-AL (see FC-AL). If a bit is set to one in this field, then the FL_Port shall originate a LPByx Primitive Sequence to the corresponding AL_PA.

– **Loop Devices to Enable:**

- The format of the Loop Devices to Enable field follows the AL_PA bit mapped format defined in FC-AL (see FC-AL). If a bit is set to one in this field, then the FL_Port shall originate a LPEyx Primitive Sequence to the corresponding AL_PA. If Word 0 bit 31 is set to one then the FL_Port shall originate the LPEfx Primitive Sequence.

Table 84 - Port Control Function	
Function	Value
Reserved	0
Normal Bypass	1
Initialize and Bypass	2

Reserved	3 - 255
----------	---------

Reply Extended Link Service Sequence:

Service Reject (LS_RJT)

- Signifies the rejection of the LPC command

Accept (ACC)

- Signifies acceptance of the LPC request.
- Accept Payload

The format of the Accept Payload is shown in Table 85.

Table 85 - LPC Accept Payload	
Item	Size Bytes
hex '02 00 00 00'	4
Reserved	3
Status	1

The format of the Status field is shown in Table 86.

Table 86 - LPC Status	
State	Value
Reserved	0
Success - The requested function was completed.	1
Failure - The requested function could not be completed.	2
Reserved	3 - 255

15.5.21 Loop Status (LSTS)

A Loop Status Extended Link Service requests the Fabric Controller to report on the state of the specified loop.

Protocol:

- Loop Status request Sequence
- Reply Sequence

Format: FT-1

Addressing: The S_ID is the N_Port or NL_Port requesting status for the specified loop. The D_ID field shall be the Loop Fabric Address (LFA) of the loop for which status is being requested.

Payload: The format of the LSTS Payload is shown in Table 87.

Table 87 - LSTS Payload	
Item	Size Bytes
hex '72 00 00 00'	4

Reply Extended Link Service Sequence:

Service Reject (LS_RJT)

- Signifies the rejection of the LSTS command

Accept (ACC)

- Supplies the requested status for the identified Loop.
- Accept Payload

The format of the Accept Payload is shown in Table 88.

Table 88 - LSTS Accept Payload	
Item	Size Bytes
hex '02 00 00 00'	4
Reserved	1
Failed Receiver	1
FC-FLA Compliance Level	1
Loop State	1
Current Public Loop Devices bit map	16
Current Private Loop Devices bit map	16
AL_PA position map	128

- Failed Receiver:
 - The AL_PA of the L_Port that detected the loop failure. This field is valid only if the Loop State indicates a Loop Failure. This field shall be set to hex '00' if the FL_Port detected the failure. This field shall be set to hex 'F7' if the detector of the failure cannot be determined.

NOTE – If the failure is not detected by the FL_Port, this field will typically be set to the “x” value of a received LIP (hex 'F8', x).
- FC-FLA Compliance Level:

- The Compliance Level of the loop, as detected by the FL_Port. The format of the FC-FLA Compliance Level field is shown in Table 89. See FC-FLA for the meaning of the various compliance levels.

Table 89 - FC-FLA Compliance Level	
Level	Value
Reserved	0
FC-FLA Level A	1
FC-FLA Level B	2
Reserved	3 - 255

- Loop State:
 - The format of the Loop State field is shown in Table 90.

Table 90 - Loop State	
State	Value
Reserved	0
Online – The loop is not performing Loop Initialization, and no Failure has been detected.	1
Loop Failure - A Loop Failure has been detected. The AL_PA that detected the failure is reported in the Failed Receiver field.	2
Initialization Failure - The FL_Port has not been able to complete Loop Initialization.	3
Initializing - The loop is currently performing Loop Initialization.	4
Reserved	5 - 255

- **Current Public Loop Devices:**
 - The format of this field follows the AL_PA bit mapped format defined in FC-AL (see FC-AL). If a bit is set to one in this field, then a Loop Device that has performed FLOGI is present at the identified location. This field is valid only when the value in the Loop State field is Online.
- **Current Private Loop Devices:**
 - The format of this field follows the AL_PA bit mapped format defined in FC-AL (see FC-AL). If a bit is set to one in this field, then a Loop Device is present at the identified location. This field is valid only when the value in the Loop State field is Online.
- **AL_PA position map:**
 - The format of this field follows the AL_PA position map format defined in FC-AL for the LILP Sequence. If byte 0 of word 0 has a value of zero, then no AL_PA position map is available. This field is valid only when the value in the Loop State field is Online.

15.5.22 Registered State Change Notification (RSCN)

A Registered State Change Notification (RSCN) Extended Link Service request shall be sent to registered N_Ports and NL_Ports when an event occurs which may have affected the state of a N_Port or NL_Port, or the ULP state within the N_Port or NL_Port. The term "state" is used here to refer to any condition of a N_Port or NL_Port that is considered important enough to notify other N_Ports and NL_Ports of a change in that state. The definition of "important" is specific to a N_Port or NL_Port implementation, but can include the Login state or Link state.

NOTE - The State Change Notification (SCN) ELS may be used to alert N_Ports and NL_Ports to state changes in Remote Ports.

RSCN is intended to provide a timely indication of changes in nodes to avoid the considerable traffic that polling can generate. RSCN may be used to detect a failed node and release resources tied up by the failed node. RSCN may also be used to notify interested nodes of new devices coming online, and of changes within an online node that affect the operation of the system (e.g., more storage has become available). The sender of the RSCN request may coalesce several events into a single report.

The normal use of an RSCN request is for the fabric to notify registered N_Ports and NL_Ports of changes detected by the fabric. The RSCN request is issued by the Fabric Controller (hex 'FF FF FD') to the registered N_Ports and NL_Ports. A N_Port or NL_Port may also issue a RSCN request to the Fabric Controller to indicate changes of state within the N_Port or NL_Port that are not otherwise detectable by the fabric. In either case, the payload of a RSCN request includes a list containing the addresses of the Affected N_Ports and NL_Ports. Note that the sending of RSCN between N_Ports or NL_Ports, neither of which are the Fabric Controller, is permitted.

15.5.22.1 RSCN issued by the Fabric Controller

The Fabric Controller shall issue an RSCN request to all registered N_Ports and NL_Ports for an affected N_Port or NL_Port when the fabric detects an event. The Fabric Controller shall ensure that any Fabric-provided resources (such as the Name Service) have received updates to reflect changes caused by the event, prior to issuing the RSCN for the event. An "event" may include any of the following:

- an implicit fabric Logout of the Affected N_Port or NL_Port, including loss of signal, NOS, and OLS, or when the fabric receives a FLOGI from a port that had already completed FLOGI;
- a loop initialization of the Affected NL_Port, and the L_bit was set in the LISA Sequence;
- a fabric Login from an Affected N_Port or NL_Port not previously logged in;
- the fabric path between the Affected N_Port or NL_Port and any other N_Port or NL_Port has changed (such as: a change to the fabric routing tables that affects the ability of the fabric to deliver frames in order, or an E_Port initialization or failure);
- any other fabric-detected state change of the Affected N_Port or NL_Port;
- the Affected N_Port or NL_Port issues an RSCN request to the Fabric Controller.

A registered N_Port or NL_Port that receives an RSCN request may perform any operation it deems necessary to determine the nature of the state change. These operations include the PDISC ELS, the ADISC ELS, a query to the Name Service, or a ULP query. The fabric may accumulate Affected N_Port or NL_Port addresses for subsequent delivery to reduce the volume of RSCN traffic.

15.5.22.2 RSCN issued by the Affected N_Port or NL_Port

A N_Port or NL_Port shall issue an RSCN request to the Fabric Controller or the N_Port or NL_Port that has registered for receipt of RSCN requests or both when an event is detected by

a N_Port or NL_Port. N_Ports and NL_Ports, neither of which are the Fabric Controller, that request registration for state-change notification (RSCN) are hereafter called "registered N_Ports or NL_Ports". A "event" may include any of the following:

- a failure within the Affected N_Port or NL_Port;
- any other important state change of the Affected N_Port or NL_Port (e.g. an event defined by a FC-4).

15.5.22.3 RSCN initiative

An Affected N_Port or NL_Port shall issue one RSCN request for all state changes that occur prior to the initiation of the RSCN request; multiple RSCN requests shall not be queued for initiation. An RSCN request shall be considered initiated when the SOF of the first frame of the RSCN request Sequence has been transmitted.

15.5.22.4 RSCN registration

The Fabric Controller shall only issue RSCN requests to N_Ports and NL_Ports that have registered to be notified of state changes in other N_Ports and NL_Ports. A N_Port or NL_Port shall issue RSCN requests to Registered N_Ports and NL_Ports that have registered to be notified of state changes in other N_Ports and NL_Ports. These registrations shall be performed via the State Change Registration (SCR) Extended Link Service. A N_Port or NL_Port may issue an RSCN to the Fabric Controller without having completed SCR with the Fabric Controller.

Protocol:

- Registered State Change Notification (RSCN) request Sequence
- Accept (ACC) Reply Sequence

Format: FT-1

Addressing:

- In the first case (the fabric notifies a registered N_Port or NL_Port - see 15.5.22.1), the S_ID is the Fabric Controller, (hex 'FF FF FD') and the D_ID is the address of the Registered N_Port or NL_Port destination.
- In the second case, (an affected N_Port or NL_Port notifies the fabric or a Registered N_Port or NL_Port destination of a change - see 15.5.22.2), the S_ID designates either the N_Port or NL_Port indicating a state change to the Fabric Controller or the Registered N_Port or NL_Port and the D_ID is either the Fabric Controller, (hex 'FF FF FD') or the address of the Registered N_Port or NL_Port destination.

Payload: The format of the RSCN request Payload is shown in Table 91.

Table 91 - RSCN Payload	
Item	Size Bytes
hex '61' = RSCN Command code	1
hex '04' = Page Length	1
Payload Length	2
Affected N_Port ID pages	4-max

- **Page Length:**
 - The length in bytes of an Affected N_Port ID page. This value is fixed at hex '04'.
- **Payload Length:**
 - The length in bytes of the entire payload, inclusive of the word 0. This value shall be a multiple of 4. The minimum value of this field is 4. The maximum value of this field is 256.
- **Affected N_Port ID page:**
 - Each Affected N_Port ID page contains the ID of an affected N_Port or NL_Port. The RSCN payload may contain zero or more of these pages. The generic format of the Affected N_Port ID page is shown in Table 92.

Table 92 - Generic Affected N_Port ID page	
Item	Size Bytes
Address Format	1
Affected N_Port ID	3

- **Address Format:**
 - The format of the Address Format field is shown in Table 93.

Table 93 - Address Format	
Format	Value
Port Address - Bytes 1, 2, and 3 of the Affected N_Port ID are valid, and indicate a single N_Port or NL_Port address. This format is illustrated in Table 94.	0
Area Address Group - Bytes 1 and 2 of the Affected N_Port ID are valid, and indicates a group of addresses that encompass an Area of N_Port or NL_Port addresses. Byte 3 shall be zero. This format is illustrated in Table 95.	1
Domain Address Group - Byte 1 of the Affected N_Port ID is valid, and indicates a group of addresses that encompass a Domain of N_Port and/or NL_Port addresses. Bytes 2 and 3 shall be zero. This format is illustrated in Table 96.	2
Fabric Address Group - This format indicates a group of addresses that encompass the entire Fabric of N_Port and/or NL_Port addresses. Bytes 1, 2 and 3 shall be zero. This format is illustrated in table 0.6.	3
Reserved	3 - 255

- **Affected N_Port ID:**
 - As defined by the Address Format. If the Address Format indicates an Area, Domain or Fabric Address Group, all N_Port or NL_Port IDs within the indicated Group are Affected. Table 94, Table 95, Table 96 and Table 97 detail the various address formats.

Table 94 - Port Address Format

- Affected N_Port ID page	
Item	Size Bytes
Address Format = hex '00'	1
Affected N_Port ID	3

Table 95 - Area Address Format - Affected N_Port ID page	
Item	Size Bytes
Address Format = hex '01'	1
Affected N_Port Domain	1
Affected N_Port Area	1
Reserved = hex '00'	1

Table 96 - Domain Address Format - Affected N_Port ID page	
Item	Size Bytes
Address Format = hex '02'	1
Affected N_Port Domain	1
Reserved = hex '00'	2

Table 97 - Fabric Address Format Affected N_Port ID page	
Item	Size Bytes
Address Format = hex '03'	1
Reserved = hex '00'	3

Reply Extended Link Service Sequence:

Service Reject (LS_RJT)

- Signifies the rejection of the RSCN command

Accept (ACC)

- Signifies acceptance of the RSCN request.
- Accept Payload

The format of the Accept Payload is shown in Table 98.

Table 98 - RSCN Accept Payload

Item	Size Bytes
hex '02 00 00 00'	4

15.5.23 State Change Registration (SCR)

The State Change Registration (SCR) Extended Link Service requests the Fabric Controller or a N_Port or NL_Port to add the N_Port or NL_Port that is sending the SCR request to the list of N_Ports and NL_Ports registered to receive the Registered State Change Notification (RSCN) Extended Link Service.

Protocol:

- State Change Registration request Sequence
- Accept (ACC) Reply Sequence

Format: FT-1

Addressing: The S_ID designates the N_Port or NL_Port requesting registration for State Change Notification. The D_ID designates either the Fabric Controller, hex 'FF FF FD', or the address of the N_Port or NL_Port from which RSCNs are desired.

Payload: The format of the SCR request Payload is shown in Table 99.

Table 99 - SCR Payload	
Item	Size Bytes
hex '62 00 00 00'	4
Reserved	3
Registration Function	1

- **Registration Function:** The format of the Registration Function field is shown in Table 100.

Table 100 - Registration Function	
Function	Value
Reserved	0
Fabric Detected registration - Register to receive all RSCN requests issued by the Fabric Controller for events detected by the fabric.	1
N_Port Detected registration - Register to receive all RSCN requests issued for events detected by the Affected N_Port or NL_Port.	2
Full registration - Register to receive all RSCN requests issued. The RSCN request shall return all Affected N_Port ID pages.	3
Reserved	4 - 254
Clear registration - Remove any current RSCN registrations.	255

Reply Extended Link Service Sequence:

Service Reject (LS_RJT)

- Signifies the rejection of the SCR command

Accept (ACC)

- Signifies acceptance of the SCR request and registration for RSCN.
- Accept Payload

The format of the Accept Payload is shown in Table 101.

Table 101 -SCR Accept Payload	
Item	Size Bytes
hex '02 00 00 00'	4

15.5.24 Process Extended link services

These Extended Link Services are optional. These services are considered to be generic to systems designed to support multiple FC-4s. The usage of Extended Link Services by a particular FC-4 is specified in the corresponding FC-4 document. Presently, these functions are required for FC-SB and SCSI-FCP support.

15.5.25 PRLI/PRLO command codes

When TYPE indicates Extended Link Service, byte 0 of the first word of the first frame of the Payload (LS_Command code) of the request or reply Sequence shall contain the encoded value shown in Table 102. Subsequent frames, if any, for a request or reply Sequence shall only contain additional Payload in the Payload field (i.e., the LS_Command code is not repeated in each frame).

Information Units associated with the Extended Link Services specified in Table 102 are described in Table 103.

Table 102 - LS_Command codes		
Value (Bits 31-24)	Description	Abbr.
0010 0000	Process Login	PRLI
0010 0001	Process Logout	PRLO
0010 0010	State Change Notification	SCN
0010 0011	Test Process Login State	TPLS

Table 103 - Extended link service information units				
IU Name	Category	Content	F/M/L	SI

PRLI	Unsol Control	Process Login	F	T
PRLO	Unsol Control	Process Logout	F	T
SCN	Unsol Control	State Change Notification	F/L	N/A
TPLS	Unsol Control	Test Process Login State	F	T
Legend: IU Name is the Information Unit name Category is the Information Category Content is the Payload content F/M/L is the First/Middle/Last Sequence of an Exchange SI is Sequence Initiative - Held or Transferred				

15.5.26 Process login

The Process Login (PRLI) Extended Link Service request shall be used to establish the operating environment between a group of related processes at the originating N_Port and a group of related processes at the responding N_Port.

Establishing the operating environment may include the establishment of image pairs and the exchange of Service Parameters. The establishment of image pairs is FC-4 independent and is system structure dependent. The exchange of Service Parameters is FC-4 dependent, and if required by a particular FC-4, shall be specified in the corresponding FC-4 standard.

Process Login is long lived. The number of concurrent Process Logins in effect at a N_Port is a function of the N_Port facilities available. Process Login is separate from N_Port_Login. Process Login may be either implicit or explicit.

If the recipient N_Port or NL_Port is not logged in with the requesting port it shall reply with an LS_RJT ELS Sequence with a reason code of 'unable to perform command request' with a reason code of 'N_Port_Login Required.'

Implicit Process Login

Implicit Process Login is a method of establishing an operating environment by means other than the explicit use of the PRLI Exchange. Specific methods of Implicit Process Login are not defined.

Explicit Process Login

Explicit Process Login is accomplished by using the PRLI Extended Link Service Sequence within a separate Exchange to establish an operating environment.

A group of related processes is known as an 'image' and is identified by the Process_Associator (see Figure 36). The combination of the D_ID, Responder Process_Associator, S_ID and Originator Process_Associator identify the image pair (see Figure 37). Either a single group or multiple groups of related processes may exist behind a N_Port. A single group of related processes behind a N_Port may be denoted by either a N_Port ID only or N_Port ID and Process_Associator.

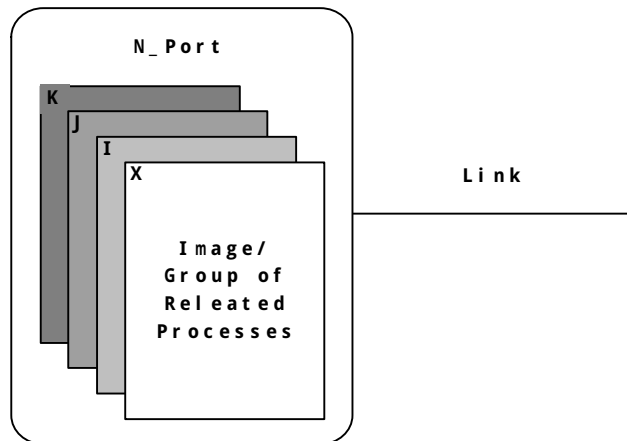


Figure 36 - Images/groups of related processes

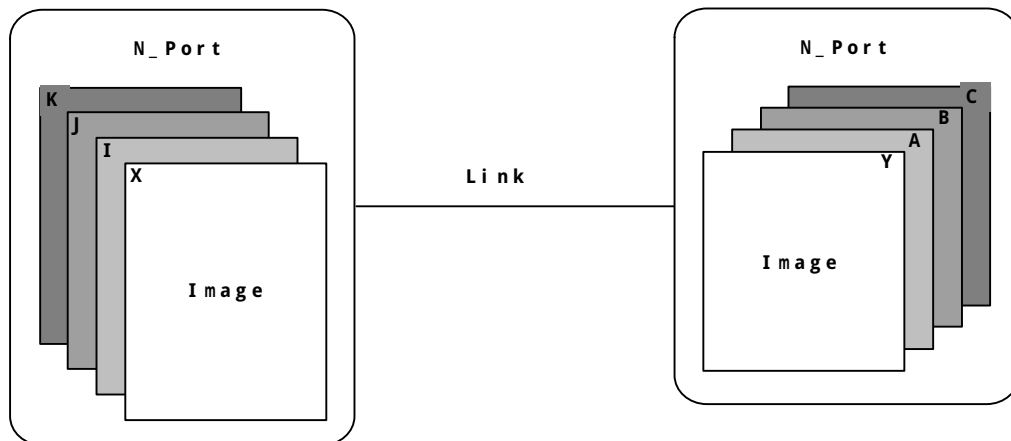


Figure 37 - Image pairs

PRLI, if required, is performed after N_Port_Login is successful and prior to other IU transfers. Examples of use of the Process Login function may include image initialization, image re-configuration, or when the N_Port receives an indication that the image pair no longer exists. PRLI allows each image behind a N_Port to separately manage its resources.

PRLI may be used to establish an operating environment between any of the following combinations of N_Port facilities:

- two N_Ports
- one N_Port and one N_Port image
- two N_Port images

Multiple image pairs may be established with a single PRLI request and Accept reply Sequence set. Failure to establish a particular image pair does not affect existing image pairs or the ability to establish other image pair.

PRLI may also be used to exchange Service Parameters without establishing image pairs. However, if an image pair is currently established, a subsequent PRLI request targeted to the

same N_Port pair shall identify an image pair in order to modify Service Parameter settings for that image pair.

A ULP may choose to establish or modify the operating environment for multiple image pairs with a single PRLI request. This can be accomplished by specifying multiple Service Parameter pages within a single PRLI request. However, a destination N_Port may be unable to execute a PRLI request containing multiple Service Parameter pages. No protocol to determine the capability of a PRLI Responder to execute PRLI requests containing multiple Service Parameter pages is specified. An Accept response containing a non-zero Response code is returned in the case that a PRLI Responder is unable to execute a PRLI request containing multiple Service Parameter pages (see Table 104).

Table 104 - PRLI accept response code	
Encoded Value Wd 0, Bits 11-8	Description
0000	Reserved
0001	Request executed
0010	The target image has no resources available for establishing image pairs between the specified source and destination N_Ports. The PRLI request may be retried.
0011	Initialization is not complete for the target image. The PRLI request may be retried.
0100	The target image corresponding to the Responder Process_Associator specified in the PRLI request and PRLI Accept response does not exist. The PRLI request shall not be retried.
0101	The target image has a predefined configuration, which precludes establishing this image pair. The PRLI request shall not be retried.
0110	Request executed conditionally. Some Service Parameters were not able to be set to their requested state (see Table 106)
0111	The destination N_Port is unable to process multiple page PRLI requests. The PRLI request may be retried as a single page request.
1000	Service parameters are invalid
1001 - 1111	Reserved

If a PRLI request is received for an established image pair, the existing image pair is unaffected and the PRLI request is processed normally. This allows the exchange of Service Parameters for a FC-4 not specified when the original image pair was established. PRLO shall be used to remove an established image pair.

It shall be the responsibility of the ULPs to ensure that all active operations over an image pair have been properly terminated prior to issuing a PRLI request that replaces Service Parameters. If the replacement of Service Parameters affects any active operations, all active Sequences shall be terminated by invoking Abort Sequence (ABTS) protocol. Following the completion of ABTS protocol, all active Exchanges shall be terminated by invoking either Abort Exchange (ABTX) protocol or Abort Sequence Protocol-Last Sequence (ABTS-LS) protocol. Whether or not the replacement of Service Parameters affects an active operation shall be specified for each Service Parameter by the associated FC-4.

The N_Port originating the PRLI request shall not consider the image pair to be established until it has taken the necessary action to establish the image pair, and has received an Accept reply Sequence indicating that the image pair has been established. The N_Port responding to the PRLI request does not consider the image pair to be established until the necessary action is taken at the N_Port to establish the image pair, and an Accept reply Sequence is sent.

If a link error is detected when a PRLI request is received, the appropriate response, if any, is made, and the image pair is not established. If the PRLI request cannot be accepted for reasons other than a link-error or link-busy condition, an LS_RJT reply Sequence containing the appropriate LS_RJT reason code is sent in response. If an LS_RJT is sent in response to a PRLI request for an image pair that is already established, the existing image pair is unaffected. If an LS_RJT, link-busy, or link-reject response is received to a PRLI request, the PRLI request may be retried until the image pair is established. The number of retries is system dependent. In the case of LS_RJT, whether or not the PRLI is retried depends on the LS_RJT reason code.

In the event that there is an error in the response to establish an image pair, the originating N_Port cannot assume that the requested action has or has not taken place. If no valid response is received by the N_Port to the PRLI request, the N_Port can retry the request. The number of retries is system dependent.

Protocol:

- Process Login request Sequence
- Accept reply Sequence

Format: FT-1

Addressing: The S_ID field designates the N_Port requesting Process Login. The D_ID field designates the destination N_Port of the Process Login.

Payload: The format of the Payload is shown in Table 105.

Table 105 - PRLI payload	
Item	Size (Bytes)
x'20' = PRLI Command code	1
x'10' = Page Length	1
Payload Length	2
Service Parameter pages	16-max

- **Page length:**
 - Byte 1 of word 0 contains an 8-bit unsigned binary integer that specifies the length of each Service Parameter page. The rightmost two bits shall be zeros.
- **Payload length:**
 - Bytes 2-3 of word 0 contain a 16-bit unsigned binary integer that specifies the length of the PRLI Payload. The rightmost two bits shall be zeros. The value specified shall be greater than or equal to 20, and less than or equal to 65 532.
- **Service parameter page(s):**
 - Words 1- max of the PRLI Payload contain one or more Service Parameter pages. Each Service Parameter page contains Service Parameters for a single image pair and is associated with either a single FC-4 TYPE or is common to all FC-4 TYPE codes for the specified image pair. The format of PRLI Service Parameter pages is specified in Table 106.

Table 106 - PRLI service parameter page format		
Item	Word	Bit
Service Parameter page	0-3	31-0
TYPE Code or Common Service Parameters	0	31-24
TYPE Code Extension	0	23-16
Originator Process_Associator Validity	0	15
Responder Process_Associator Validity	0	14
Establish Image Pair	0	13
Reserved	0	12-0
Originator Process_Associator	1	31-0
Responder Process_Associator	2	31-0
Service Parameters	3	31-0
NOTE - If byte 0 of the first word of a Service Parameter page is set to the value x'00', the Service Parameter page is common to all FC-4 Types at that image pair.		

- **Word 0, Bits 31-24 - TYPE code or common service parameters**

Identifies the protocol associated with this Service Parameter page. If byte 0 of the first word of a Service Parameter page contains the value x'00', the Service Parameter page contains Service Parameters common to all FC-4 Types at that image pair or N_Port pair. If byte 0 of the first word of a Service Parameter page contains the value other than x'00', the Service Parameter page contains Service Parameters for the FC-4 TYPE indicated.

- **Word 0, Bits 23-16 - TYPE code extension**

Reserved for future use.

- **Word 0, Bit 15 - Originator Process_Associator validity**

0 = not meaningful
1 = meaningful

- **Word 0, Bit 14 - Responder Process_Associator validity**

0 = not meaningful
1 = meaningful

- **Word 0, Bit 13 - Establish Image Pair**

0 = Exchange Service Parameters only
1 = Establish image pair and exchange Service Parameters

- **Word 3, Bits 31-0 - Service parameters**

Common Service Parameters are specified below. FC-4 TYPE code dependent Service Parameters shall be specified in the corresponding FC-4 standard.

NOTE - No Common Service Parameters are currently specified.

Reply link service sequence:

Link Service Reject (LS_RJT)

– signifies rejection of the PRLI request

Accept (ACC)

- signifies successful completion of the PRLI request
- Accept Payload

The format of the Accept Payload is shown in Table 107.

Table 107 - PRLI accept payload	
Item	Size (Bytes)
x'02' = ACC Command code	1
x'10' = Page Length	1
Payload Length	2
Service Parameter Response pages	16-max

- **Page length:**
 - Byte 1 of word 0 contains an 8-bit unsigned binary integer that specifies the length of each Service Parameter Response page. The rightmost two bits shall be zeros.
- **Payload length:**
 - Bytes 2-3 of word 0 contain a 16-bit unsigned binary integer that specifies the length of the PRLI Accept Payload. The rightmost two bits shall be zeros. The value specified shall be greater than or equal to 20, and less than or equal to 65 532.
- **Service parameter response page(s):**
 - Words 1:max of the PRLI Accept Payload contain one or more Service Parameter Response pages. Each Service Parameter Response page contains Service Parameter responses for a single image pair or N_Port pair and is associated with a single FC-4 TYPE or common to all FC-4 Types at that image pair or N_Port pair.
 - The number of Service Parameter pages and the Payload Length field of the PRLI Accept reply Sequence shall match the number of Service Parameter pages and the Payload Length field of the PRLI request Sequence.
 - The format of PRLI Accept Service Parameter Response pages is described in Table 108.

Table 108 - PRLI accept service parameter response page format		
Item	Word	Bit
TYPE Code or Common Service Parameters ¹	0	31-24
TYPE Code Extension	0	23-16
Originator Process_Associator Validity	0	15
Responder Process_Associator Validity	0	14
Image Pair Established	0	13
Reserved	0	12
Response Code (see Table 104)	0	11-8
Reserved	0	7-0
Originator Process_Associator	1	31-0
Responder Process_Associator	2	31-0
Service Parameter Response	3	31-0
NOTE - If byte 0 of the first word of a Service Parameter page is set to the value hex '00', the Service Parameter page is common to all FC-4 Types at that image pair.		

- **Word 0, Bits 31-24 - TYPE code or common service parameters**

Identifies the protocol associated with this Service Parameter Response page. If byte 0 of the first word of a Service Parameter Response page contains the value x'00', the Service Parameter page contains Service Parameters common to all FC-4 Types at that image pair or N_Port pair. If byte 0 of the first word of a Service Parameter Response page contains a value other than x'00', the Service Parameter page contains Service Parameters for the FC-4 TYPE indicated.

- **Word 0, Bits 23-16 - TYPE code extension**

Reserved for future use.

- **Word 0, Bit 15 - Originator Process_Associator validity**

0 = not meaningful
1 = meaningful

- **Word 0, Bit 14 - Responder Process_Associator validity**

0 = not meaningful
1 = meaningful

- **Word 0, Bit 13 - Image Pair Established**

Image Pair Established is valid only if bit 13 was set to one on the corresponding Service Parameter page of the PRLI request.

0 = Image pair not established, see response code for additional information
1 = Image pair established

- **Word 0, Bits 11-8 - Response code**

The Response code field contains an encoded binary value indicating the result of the PRLI request. The meanings of the encoded Response code values are shown in Table 104.

- **Word 3, Bits 31-0 - Service parameter response**

Provides feedback to the Originator as to the resultant state of the Service Parameters as returned by the Responder.

15.5.26.1 PRLI/PRLO Relationships

Process_Associator (PA) images can exist in the following relationships. Any of these relationships can be established as a default condition using mechanisms not specified by the standards.

PA not supported

If one or both of the Originator and Responder do not support PAs, no valid PA is specified in any frame of any exchange.

PRLI and PRLO commands may have no more than a single page for the Common Service Parameters and an additional single page for each additional TYPE supported by the Originator or Responder. Each page indicates that both the Originator PA and Responder PA are invalid.

The PRLI/PRLO can be thought of as enabling and disabling the specified services for the entire N_Port.

PA required by originator, supported by responder

If the Originator requires PAs, the Originator is expected to communicate only when a valid PA is included in the initial Association_Header. The Responder provides a final Association_Header and provides other Association_Headers as required. The PA establishes a routing to a particular Originator process (or group of related processes). A common set of processes is identified by setting the PA of one N_Port of a node to the same value as that of the PA of another N_Port of the same node. FC-EP, may define Hunt and Stripe Groups using N_Port aliases. If so, then a process defined by a PA for the alias of the group is considered as the same process for all N_Ports of the group. The PA may additionally be used by the FC-4 in a FC-4 dependent manner.

The PRLI request pages have valid Originator PA values and invalid Responder PA values. This PRLI informs the Responder of the Originator requirements and capabilities for each TYPE of FC-4.

The PRLI and PRLO can be thought of as enabling and disabling the access of each Originator Process to the Responder N_Port. Communication with image pairs that have not been established is not allowed, even if both PAs exist and have the proper requirements and capabilities.

PA required by responder, supported by originator

If the Responder requires PAs, the Responder is expected to communicate only when a valid Responder PA is included in the initial Association_Header. The Responder provides a final Association_Header and provides other Association_Headers as required. The PA establishes a routing to a particular Responder process. A common set of processes is identified by setting the PA of one N_Port of a node to the same value as that of the PA of another N_Port of the same node. FC-EP, may define Hunt and Stripe Groups using N_Port aliases. If so, then a process defined by a PA for the alias of the group is considered as the same process for all N_Ports of the group. The PA may additionally be used by the FC-4 in a FC-4 dependent manner.

The PRLI request pages have valid Responder PA values and invalid Originator PA values. The Responder PA values may be obtained through an informative PRLI operation or by other methods not specified.

The PRLI and PRLO can be thought of as enabling and disabling the access of each Responder Process to the Originating N_Port. Communication with image pairs that have not been established is not allowed, even if both PAs exist and have the proper requirements and capabilities.

PA required by originator and responder

Communication can only take place between an Originator and a Responder when valid PAs are specified. ANSI X3.xxx, FC-EP, may define hunt groups and striping groups using N_Port aliases. If so, then a process (or group of processes) defined by a PA for the alias of the group is considered as the same process for all N_Ports of the group. The PA may additionally be used by the FC-4 in a FC-4 dependent manner.

The PRLI request pages must have valid Originator and Responder PA values defined and must have the Establish Image Pair bit set to one in order to create image pairs in an binding manner. The Responder PA values may be obtained through an informative PRLI operation or by other methods not specified.

PRLI and PRLO pages with complete Originator PA and Responder PA information enable or disable communication between the specified image pair. PRLI pages with incomplete Originator PA and Responder PA information are invalid. Communication with image pairs that have not been established is not allowed, even if both PAs exist and have the proper requirements and capabilities.

15.5.26.2 Process login, mode of operation

PRLI has two modes of operation.

Informative mode

Service Parameter information is exchanged enabling subsequent negotiation for image pair establishment.

Binding mode

Information is exchanged that explicitly establishes a relationship between processes in the communicating N_Ports. The relationship does not allow any communication types or paths other than those established by the PRLI.

The use of a Binding PRLI page requires that the Originator have precise and detailed knowledge of the PAs and capabilities available in the Responder. That information may be obtained from Directory Services, implicitly from configuration information obtained outside the scope of FC, or by performing an Informative PRLI.

Binding or Informative mode is determined by the setting of the Establish Image Pair bit in the PRLI request page.

The Service Parameters included in a page may be either requirements or capabilities. Capabilities indicate those FC-4 properties that describe the role and state of the node in the FC-4. Such capabilities include channel or device for FC-SB, initiator or target for FCP, and similar values. Requirements indicate those FC-4 properties that must be agreed upon by both nodes for operation with a FC-4. Requirements include values such as the parameters controlling the FCP, IUs that are used in communication.

15.5.26.3 Process login, protocol

PA required by originator and responder

For each PRLI request page the Originator and Responder PA validity bits are set to one and valid Originator and Responder PAs are specified. Service Parameter requirements are set to those that correspond to the combined understanding expected to be agreed to by the Re-

sponder, given the knowledge that the Originator PA has of the Responder PA. Service Parameter capabilities are those of the Originator only. The specification of Service Parameters as either requirements or capabilities is specified in the corresponding FC-4 standard.

For each ACC response page the Originator and Responder PA validity bits are set to one and valid Originator and Responder PAs are specified. Service Parameter requirements are set to those that are agreed to by the Responder. Service Parameter capabilities are those of the Responder only. Error responses are possible.

Each page identifies a Binding mode PRLI operation between one image at the Originator N_Port and one image at the Responder N_Port.

PA required by originator, supported by responder

For each PRLI request page the Originator PA validity bit is set to one and the Responder PA validity bit is set to zero. A valid Originator PA is specified. Service Parameter requirements are set to those that correspond to the combined understanding expected to be agreed to by the Responder, given the knowledge that the Originator PA has of the Responder PA. Service Parameter capabilities are those of the Originator only. The specification of Service Parameters as either requirements or capabilities is specified in the corresponding FC-4 standard.

For each ACC response page the Originator PA validity bit is set to one and the Responder PA validity bit is set to zero. The Originator PA is returned. Service Parameter requirements are set to those that are agreed to by the Responder. Service Parameter capabilities are those of the Responder only. Error responses are possible.

Each page identifies a Binding mode PRLI operation between one image at the Originator N_Port and the Responder N_Port.

PA supported by originator, required by responder

For each PRLI request page, the Originator PA validity bit is set to zero and the Responder PA validity bit is set to one. A valid Responder PA is specified. Service Parameter requirements are set to those that correspond to the combined understanding expected to be agreed to by the Responder, given the knowledge that the Originator PA has of the Responder PA. Service Parameter capabilities are those of the Originator only. The specification of Service Parameters as either requirements or capabilities is specified in the corresponding FC-4 standard.

For each ACC response page, the Originator PA validity bit is set to zero and the Responder PA validity bit is set to one. The Responder PA is returned. Service Parameter requirements are set to those that are agreed to by the Responder. Service Parameter capabilities are those of the Responder only. Error responses are possible.

Each page identifies a Binding mode PRLI operation between the Originator N_Port and one image at the Responder N_Port.

15.5.27 Process logout

The Process Logout (PRLO) Extended Link Service request shall be used to request invalidation of the operating environment between an image at the initiating N_Port and an image at the recipient N_Port. PRLO frees resources committed by a previous PRLI function. ULP behavior following successful execution of the PRLO function is specified in the corresponding FC-4 standard.

Examples of PRLO usage include image re-configuration and TYPE-specific reset of Process Login Service Parameters.

One or more image pairs may be removed with a single PRLO request and Accept reply Sequence set. Other image pairs associated with the same or different N_Ports shall be unaffected. After an image pair is removed, IUs may not be sent or received using that image pair.

One or more TYPE-specific Service Parameter settings may be reset with a single PRLO request and Accept reply Sequence set. Other TYPE-specific Service Parameter settings associated with the same or different image pairs or N_Ports shall be unaffected. After TYPE-specific Service Parameter settings are reset, IUs may not be sent or received for the specified FC-4 TYPE using that image pair and N_Port combination specified in the PRLO request.

A ULP may choose to remove multiple image pairs or reset multiple TYPE-specific Service Parameter settings with a single PRLO request. This can be accomplished by specifying multiple Logout Parameter pages within a single PRLO request. However, a destination N_Port may be unable to execute a PRLO request containing multiple Logout Parameter pages. No protocol to determine the capability of a PRLO Responder to execute PRLO requests containing multiple Logout Parameter pages is specified. An Accept response containing a non-zero Response code is returned in the case that a PRLO Responder is unable to execute a PRLO request containing multiple Logout Parameter pages.

If a PRLO request is received for an image pair or FC-4 TYPE that does not exist, the request is accepted, provided that no link errors are detected, and the Accept response is sent.

If a link error is detected when a PRLO request is received, the Sequence is discarded, and the appropriate response, if any, for the link error recognized is sent. The N_Port originating the PRLO request shall not consider an image pair to be removed until it receives an error-free Accept reply Sequence. The N_Port responding to the PRLO request shall not consider an image pair to be removed or TYPE-specific Service Parameter settings to be reset, as appropriate, until the Accept reply Sequence is completely sent without error. A N_Port that receives a link-busy or link-reject reply in response to a PRLO request may retry the PRLO request. The number of retries is system dependent.

Unless the requesting N_Port receives a valid response to a PRLO request, that N_Port cannot assume that the requested action has or has not taken place. If a N_Port to the PRLO request receives an invalid response, the N_Port can retry the request. The number of retries is system dependent.

If the recipient N_Port or NL_Port is not logged in with the requesting port it shall reply with an LS_RJT ELS Sequence with a reason code of 'unable to perform command request' with a reason code of 'N_Port_Login Required.'

Protocol:

- Process Logout request Sequence
- Accept reply Sequence

Format: FT-1

Addressing: The S_ID field designates the N_Port requesting Process Logout. The D_ID field designates the destination N_Port of the Process Logout.

Payload: The format of the Payload is shown in Table 109.

Table 109 - PRLO payload	
Item	Size (Bytes)

x'21' = PRLO Command code	1
x'10' = Page Length	1
Payload Length	2
Logout Parameter pages	16-max

– **Page length:**

- Byte 1 of word 0 contains an 8-bit unsigned binary integer that specifies the length of each Logout Parameter page. The rightmost two bits shall be zeros.

– **Payload length:**

- Bytes 2-3 of word 0 contain a 16-bit unsigned binary integer that specifies the length of the PRLO Payload. The rightmost two bits shall be zeros. The value specified shall be greater than or equal to 20, and less than or equal to 65 532.

– **Logout parameter page(s):**

- Words 1:max of the PRLO Payload contain one or more Logout Parameter pages. Each Logout Parameter page contains logout parameters for a single image pair and is associated with either a single FC-4 TYPE or is common to all FC-4 TYPE codes for the specified image pair.
- The format of PRLO Logout Parameter pages is described in Table 110.

Table 110 - PRLO logout parameter page format		
Item	Word	Bit
Logout Parameter page	0-3	31-0
TYPE Code or Common Logout Parameters ¹	0	31-24
TYPE Code Extension	0	23-16
Originator Process_Associator Validity	0	15
Responder Process_Associator Validity	0	14
Reserved	0	13-0
Originator Process_Associator	1	31-0
Responder Process_Associator	2	31-0
Reserved	3	31-0
NOTE - If byte 0 of the first word of a Logout Parameter page is set to the value x'00', the Logout Parameter page is common to all FC-4 Types at that image pair.		

• **Word 0, Bits 31-24 - TYPE code or common logout parameters**

Identifies the protocol associated with this Logout Parameter page. If byte 0 of the first word of a Logout Parameter page contains the value x'00', the Logout Parameter page contains Logout Parameters common to all FC-4 Types at that image pair or N_Port pair. If byte 0 of the first word of a Logout Parameter page contains the value other than x'00', the Logout Parameter page contains Logout Parameters for the FC-4 TYPE indicated.

• **Word 0, Bits 23-16 - TYPE code extension**

Reserved for future use.

- **Word 0, Bit 15 - Originator Process_Associator validity**
0 = not meaningful
1 = meaningful
- **Word 0, Bit 14 - Responder Process_Associator validity**
0 = not meaningful
1 = meaningful

Reply link service sequence:

Link Service Reject (LS_RJT)

- signifies rejection of the PRLO request

Accept (ACC)

- signifies successful completion of the PRLO request
- Accept Payload

The format of the Accept Payload is shown in Table 111.

Table 111 - PRLO accept payload	
Item	Size (Bytes)
x'02' = ACC Command code	1
x'10' = Page Length	1
Payload Length	2
Logout Parameter Response pages	16-max

- **Page length:**
 - Byte 1 of word 0 contains an 8-bit unsigned binary integer that specifies the length of each Logout Parameter Response page. The rightmost two bits shall be zeros.
- **Payload length:**
 - Bytes 2-3 of word 0 contain a 16-bit unsigned binary integer that specifies the length of the PRLO Accept Payload. The rightmost two bits shall be zeros. The value specified shall be greater than or equal to 20, and less than or equal to 65 532.
- **Logout parameter response page(s):**
 - Words 1:max of the PRLO Accept Payload contain one or more Logout Parameter Response pages. Each Logout Parameter Response page contains a logout parameter response for a single image pair and is associated with a single FC-4 TYPE or common to all FC-4 Types at that image pair or N_Port pair.
 - The number of Logout Parameter pages and the Payload Length field of the PRLO Accept reply Sequence shall match the number of Logout Parameter pages and the Payload Length field of the PRLO request Sequence.
 - The format of PRLO Accept Logout Parameter Response pages is described in Table 112.

Table 112 - PRLO accept logout parameter response page format		
Item	Word	Bit
Logout Parameter page	0-3	31-0
TYPE Code or Common Service Parameters ¹	0	31-24
TYPE Code Extension	0	23-16
Originator Process_Associator Validity	0	15
Responder Process_Associator Validity	0	14
Reserved	0	13-12
Response Code (see Table 113)	0	11-8
Reserved	0	7-0
Originator Process_Associator	1	31-0
Responder Process_Associator	2	31-0
Reserved	3	31-0
NOTE - If byte 0 of the first word of a Logout Parameter page is set to the value x'00', the Logout Parameter page is common to all FC-4 Types at that image pair.		

- **Word 0, Bits 31-24 - TYPE code or common logout parameters**

Identifies the protocol associated with this Logout Parameter Response page. If byte 0 of the first word of a Logout Parameter Response page contains the value x'00', the Logout Parameter page contains Logout Parameters common to all FC-4 Types at that image pair or N_Port pair. If byte 0 of the first word of a Logout Parameter Response page contains a value other than x'00', the Logout Parameter page contains Logout Parameters for the FC-4 TYPE indicated.

- **Word 0, Bits 23-16 - TYPE code extension**

Reserved for future use.

- **Word 0, Bit 15 - Originator Process_Associator validity**

0 = not meaningful
1 = meaningful

- **Word 0, Bit 14 - Responder Process_Associator validity**

0 = not meaningful
1 = meaningful

- **Word 0, Bits 11-8 - Response code**

The Response code field contains an encoded binary value indicating the result of the PRLO request and the status of the image pair. The meanings of the encoded Response code values are shown in Table 113.

Table 113 - PRLO accept response code	
Encoded Value Word 0, Bits 11-8	Description
0000	Reserved

0001	Request executed
0010 - 0011	Reserved
0100	The target image corresponding to the Responder Process_Associator specified in the PRLO request and PRLO Accept response does not exist. The PRLO request shall not be retried.
0101 - 0110	Reserved
0111	The destination N_Port is unable to process multiple page PRLO requests. The PRLO request may be retried as a single page request.
1000	Service Parameters are invalid
1001 - 1111	Reserved

15.5.27.1 PRLO Operation

PRLO operates by referencing particular PA image pairs. Pages can be mixed in any combination in a PRLO request. The PRLO Accept is required to present a response page for every request page. Pages may be individually marked as being in error.

A PRLO page identifies a particular image pair to logout by marking either or both the Originator Process Associator and the Responder Process Associator. Only that image pair is logged out. No further communication for the affected FC-4(s) is possible between these two images. It is the responsibility of the ULPs to ensure that all active operations over an image pair have been orderly and properly terminated prior to issuing a PRLO request. Following PRLO execution, all active Sequences are terminated by invoking Abort Sequence (ABTS) protocol. Following the completion of ABTS protocol, all active Exchanges shall be terminated by invoking either Abort Exchange (ABTX) protocol or Abort Sequence Protocol-Last Sequence (ABTS-LS) protocol. On-going operations and states for other image pairs are not affected.

ULP attempts to communicate over an image pair that has not been established or has been abnormally terminated shall be acknowledged in the normal manner. The Originator may then perform a PRLO operation for the affected image pair in order to properly terminate the operating environment at both the Originator and Responder.

15.5.28 State change notification

A State Change Notification (SCN) Extended Link Service request shall be sent to selected N_Ports when an event occurs which may have affected the current N_Port_Login or Process Login (i.e., image pair) states. The term Login state is used here to refer to either or both the N_Port_Login state or the N_Port Process Login state. There is no reply to the SCN request Sequence.

The main purpose of SCN is communicating the release of shared resources. If a node fails, timely notification of dependent nodes is important. SCN provides an early warning, which, in turn, allows the release of resources, dedicated to the failed node. Release of serially reusable node resources of the failed N_Port or N_Port process may avoid timeouts and deadlocks in the sharing nodes. Early failure data capture, problem notification and determination can begin soon after the point of failure. Release of shared resources to other nodes also has benefits in non-error situations. Accounting and load balancing are two system dependent management functions that benefit from real-time reporting of Login state changes.

The direction of SCN requests is from the Fabric to its attached nodes. The ultimate notification of an SCN request is from a F_Port to a N_Port. Since the initiative to send an SCN request may have occurred at a source distant from the F_Port directly attached to the N_Port requiring notification (e.g., another N_Port, an intermediate Fabric element, a management server, etc.) SCN protocol shall be described in terms of SCN issued by a F_Port or N_Port.

15.5.28.1 SCN issued by a F_Port

The Login state changes that cause SCN to be sent from a F_Port are as follows:

- all implicit F_Port Logouts, including loss of signal, NOS, and OLS
- SCNs sent to the Fabric by a N_Port (a self-generated SCN from a N_Port)
- the Fabric's capability of establishing a communication path between a pair of N_Ports is altered

When a N_Port changes Login state and the state change is recognized by the Fabric, the Fabric then issues SCN to selected N_Ports, specifying the N_Port ID of the N_Port which changed state. Any N_Port, which was previously logged in to the state-changed N_Port, may then test the state of the Login via TPLS, to verify Process Login state, or other Link Services such as NOP, to verify Login state, prior to performing Login state recovery. The Affected N_Port ID is specified in the Payload of the SCN request.

If the D_ID field of the SCN request specifies the well-known address identifier for the Fabric Controller, hex 'FFFFFD', the Fabric is to report SCN to other N_Ports. The Fabric maintains the ID's of the N_Ports with changed state. For each of these the Fabric also maintains a list of N_Ports requiring notification. For each N_Port requiring notification, the Fabric ignores subsequent N_Port_Login state changes until the SCN is sent and a valid response is received. This is to limit the volume of SCN traffic.

The Fabric reports the state change by issuing SCNs to selected N_Ports, specifying the N_Port ID's of the affected N_Ports.

15.5.28.2 SCN issued by a N_Port

The Login state changes which, causes a SCN to be sent by a N_Port are as follows:

- all implicit N_Port Logouts. In the case of a node that has two or more N_Ports, if the failure of one of the N_Ports is undetected by the Fabric, then SCN is sent via a surviving N_Port.

A N_Port may optionally report a state change that is caused by an event which affects existing image pairs or affects the ability to accept new image pairs. Individual images behind a N_Port are capable of generating the SCN request. The N_Port reports the state change by sending an SCN request to the Fabric Controller or, optionally, to the destination N_Port directly. However, a N_Port shall not issue an SCN request if it has already signaled a condition which would cause the Fabric, if present, to issue an SCN request on behalf of that N_Port.

If the first SCN request has not been sent by the N_Port, subsequent state changes in that N_Port do not create initiative to send another SCN request. Once the first SCN request has been sent, a subsequent state change creates a new initiative. If the N_Port is sending SCN requests to destinations other than the Fabric Controller, this rule applies separately to each destination.

15.5.28.3 SCN initiative

Initiative to send an SCN request is discharged when one of the following takes place:

- An ACK frame is received in response to the SCN request (Class 1 or 2);
- A condition is recognized for which retry of an SCN request, if previously sent, is not performed;
- For a condition that does not require retry to be performed indefinitely, the system-dependent number of retries is exhausted without success while attempting to send an SCN request.

15.5.28.4 Interest registration

The Fabric shall only issue SCN to N_Ports, which have registered to be notified of Login state changes for all other N_Ports. One method is to register at the time a N_Port queries the Name Server to obtain the N_Port ID for the N_Port that changed Login state.

NOTE - Fabric structure may limit SCN to a Fabric Region.

When a state change occurs, initiative for SCN shall be established in the Fabric only for N_Ports currently logged into the Fabric, which are capable of communicating with the N_Port which changed state, and which have registered to be notified.

Protocol:

- State Change Notification request Sequence
- No reply Sequence

Format: FT-1

Addressing: The S_ID field designates the N_Port detecting or reporting the Login state change. The D_ID field designates the destination N_Port of the Login state change notification.

Payload: The format of the Payload is shown in Table 114.

Table 114 - SCN payload	
Item	Size (Bytes)
x'22' = SCN Command code	1
x'04' = Page Length	1
Payload Length	2
Affected N_Port ID pages	4-max

- **Page length:**
 - Byte 1 of word 0 contains an 8-bit unsigned binary integer that specifies the length of each Affected N_Port ID page. The rightmost two bits shall be zeros.
- **Payload length:**
 - Bytes 2-3 of word 0 contain a 16-bit unsigned binary integer that specifies the length of the SCN Payload. The rightmost two bits shall be zeros. The value specified shall be greater than or equal to 8, and less than or equal to 65 532.
- **Affected N_Port ID page(s):**
 - Words 1 - max of the SCN Payload contain one or more Affected N_Port ID pages. Each Affected N_Port ID page contains a single N_Port ID identified as sustaining a state change. The format of SCN Affected N_Port ID pages is described in Table 115.

Table 115 - SCN affected N_Port ID page format		
Item	Word	Bit

Affected N_Port ID page	0	31-0
Reserved	0	31-24
Affected N_Port ID	0	23-0

The Affected N_Port ID shall not be all ones or all zeros nor shall the address match the D_ID of the SCN frame. If word 1, byte 0 of the Payload is not zero, the request is discarded, and a link-protocol error is detected. If an SCN request is received and the Affected N_Port ID is all ones, all zeros, or matches the D_ID of the SCN request, the request is discarded, and a link-protocol error is detected.

15.5.29 Test process login state

The Test Process Login State (TPLS) Extended Link Service request shall be used to determine whether a image pairs are established for the image pairs specified by the combination of the S_ID || Originator Process_Associator || D_ID || Responder Process_Associator. Upon receiving a TPLS request, the receiving N_Port checks whether it has an image pair established for each specified image.

TPLS verifies the Login state for the source N_Port at the destination N_Port.

NOTE - For example, an SB CH can use TPLS to determine CU Login state for that CH and an SB CU can use TPLS to determine CH Login state for that CU. Editor's note - look in SB for CH and CU, otherwise delete

The Accept reply Sequence confirms the successful completion of the TPLS function and indicates whether or not an image pair is established for the source specified by the S_ID and image pair(s) contained in the Payload. The Payload of the Accept reply indicates the state of the image pair.

If the recipient N_Port or NL_Port is not logged in with the requesting port it shall reply with an LS_RJT ELS Sequence with a reason code of 'unable to perform command request' with a reason code of 'N_Port_Login Required.'

Protocol:

- Test Process Login State request Sequence
- Accept reply Sequence

Format: FT-1

Addressing: The S_ID field designates the source N_Port associated with the image pair. The D_ID field designates the destination N_Port associated with the image pair.

Payload: The format of the Payload is shown in Table 116.

Table 116 - TPLS payload	
Item	Size (Bytes)
x'23' = TPLS Command code	1
x'10' = Page Length	1
Payload Length	2
Image Pair ID pages	16-max

- **Page length:**
 - Byte 1 of word 0 contains an 8-bit unsigned binary integer that specifies the length of each Image Pair ID page. The rightmost two bits shall be zeros.
- **Payload length:**
 - Bytes 2-3 of word 0 contain a 16-bit unsigned binary integer that specifies the length of the TPLS Payload. The rightmost two bits shall be zeros. The value specified shall be greater than or equal to 20, and less than or equal to 65 532.
- **Image pair ID page(s):**
 - Words 1:max of the TPLS Payload contain one or more Image Pair ID pages. Each Image Pair ID page contains parameters required to identify a single image pair.

The format of Image Pair ID pages is described in Table 117.

Table 117 - TPLS image pair ID page format		
Item	Word	Bit
Image Pair ID page	0-3	31-0
Reserved	0	31-16
Originator Process_Associator Validity	0	15
Responder Process_Associator Validity	0	14
Reserved	0	13-0
Originator Process_Associator	1	31-0
Responder Process_Associator	2	31-0
Reserved	3	31-0

- **Word 0, Bit 15 - Originator Process_Associator validity**
0 = not meaningful
1 = meaningful
- **Word 0, Bit 14 - Responder Process_Associator validity**
0 = not meaningful
1 = meaningful

Reply link service sequence:

Link Service Reject (LS_RJT)

- signifies rejection of the TPLS request

Accept (ACC)

- signifies successful completion of the TPLS request
- Accept Payload

The format of the Accept Payload is shown in Table 118.

Table 118 - TPLS accept payload	
Item	Size (Bytes)
x'02' = ACC Command code	1
x'10' = Page Length	1
Payload Length	2
TPLS Response pages	16-max

- **Page length:**
 - Byte 1 of word 0 contains an 8-bit unsigned binary integer that specifies the length of each Image Pair ID page. The rightmost two bits shall be zeros.
- **Payload length:**
 - Bytes 2-3 of word 0 contain a 16-bit unsigned binary integer that specifies the length of the TPLS Accept Payload. The rightmost two bits shall be zeros. The value specified shall be greater than or equal to 20, and less than or equal to 65 532.
- **TPLS response page(s):**
 - Words 1:max of the TPLS Payload contain one or more Image Pair ID pages. Each TPLS Response page contains TPLS response information associated with a single image pair.
 - The number of Service Parameter pages and the Payload Length field of the TPLS Accept reply Sequence shall match the number of Service Parameter pages and the Payload Length field of the TPLS request Sequence.
 - The format of TPLS Response pages is described in Table 119.

Table 119 - TPLS response page format		
Item	Word	Bit
TPLS Response page	0-3	31-0
Reserved	0	31-16
Originator Process_Associator Validity	0	15
Responder Process_Associator Validity	0	14
Reserved	0	13-12
Response Code (see Table 120)	0	11-8
Reserved	0	7-0
Originator Process_Associator	1	31-0
Responder Process_Associator	2	31-0
Reserved	3	31-1
Image Pair State	3	0

- **Word 0, Bit 15 - Originator Process_Associator validity**
0 = not meaningful
1 = meaningful

- **Word 0, Bit 14 - Responder Process_Associator validity**

0 = not meaningful

1 = meaningful

- **Word 0, Bits 11-8 - Response code**

The Response code field contains an encoded binary value indicating the result of the PRLI request and the status of the image pair. The meanings of the encoded Response code values are shown in Table 120.

Table 120 - TPLS accept response code	
Encoded Value Word 0, Bits 11-8	Description
0000	Reserved
0001	Request executed
0010-0110	Reserved
0111	The destination N_Port is unable to process multiple page PRLI requests. The PRLI request may be retried as a single page request.
1000-1111	Reserved

- **Word 3, Bit 0 - Image pair state**

1 = image pair established

0 = image pair not established

15.5.30 Fibre Channel Address Resolution Protocol (FARP)

These Extended Link Services support the Internet Protocol over Fibre Channel.

15.5.30.1 Fibre Channel Address Resolution Protocol Request (FARP-REQ)

The Fibre Channel Address Resolution Protocol (FARP) is a method using ELS commands to resolve WWN or IP address to N_Port ID mapping in environments without a Directory Server. That is, FARP is used when the WWN or an IP address is known, but not the N_Port ID, and a suitable Directory Service does not exist. See GS-2 for the definition of Directory Services.

The FARP-REQ Extended Link Service request shall be used to resolve Port_IDs of communicating Fibre Channel devices. A FARP-REQ may be used to retrieve a specific N_Port's current Port_ID given a Port_Name, Node_Name or IP address. This is accomplished by requesting either a FARP-REPLY ELS (see 15.5.30.2) and/or by indicating that the responding N_Port shall perform Login with the requesting N_Port.

A FARP-REQ Match shall be considered successful at the Recipient when the FARP-REQ request is successfully received and recognized by a N_Port, and the requested Match Address Code Point is successful.

Protocol:

- FARP-REQ request Sequence
- No Reply Sequence

Format: FT-1

Addressing: The S_ID designates the Requester N_Port requesting addressing information, The D_ID is the Broadcast Alias_ID, hex 'FFFFFF' (see 12.3).

Payload: The format of the FARP-REQ request Payload is shown in Table 121.

Table 121 - FARP-REQ Request Payload Item	
Item	Size- Bytes
hex '54 00 00 00'	4
Match Address Code Points	1
Requesting N_Port Identifier	3
Responder Action	1
Responding N_Port Identifier	3
Requesting N_Port Port_Name	8
Requesting N_Port Node_Name	8
Responding N_Port Port_Name	8
Responding N_Port Node_Name	8
Requesting N_Port IP Address	16
Responding N_Port IP Address	16

- **Match Address Code Points:** The Match Address Code Points define what addresses to match based on these code points. The Match Address Code Points also indicate which Address fields shall be valid (see below). The values of these code points are defined in Table 197.

Table 122 – FARP-REQ Match Address Code Points	
Value (binary)	Definition
0000 0001	The Match is successful if all of the following are true: <ul style="list-style-type: none"> - the Port_Name of the N_Port which receives the FARP-REQ equals the Responding Port_Name; - ignore the Responding Node_Name; - ignore the Responding IP address.
0000 0010	The Match is successful if all of the following are true: <ul style="list-style-type: none"> - ignore the Responding Port_Name; - the Node_Name of the N_Port which receives the FARP-REQ equals the Responding Node_Name; - ignore the Responding IP address.
0000 0011	The Match is successful if all of the following are true: <ul style="list-style-type: none"> - the Port_Name of the N_Port which receives the FARP-REQ equals the Responding Port_Name;

Table 122 – FARP-REQ Match Address Code Points	
Value (binary)	Definition
	<ul style="list-style-type: none"> - the Node_Name of the N_Port which receives the FARP-REQ equals the Responding Node_Name; - ignore the Responding IP address.
0000 0100	<p>The Match is successful if all of the following are true:</p> <ul style="list-style-type: none"> - ignore the Responding Port_Name; - ignore the Responding Node_Name; - the IP address of the N_Port, which receives the FARP-REQ equals the Responding IP, address.
0000 0101	<p>The Match is successful if all of the following are true:</p> <ul style="list-style-type: none"> - the Port_Name of the N_Port which receives the FARP-REQ equals the Responding Port_Name; - ignore the Responding Node_Name; - the IP address of the N_Port, which receives the FARP-REQ equals the Responding IP, address.
0000 0110	<p>The Match is successful if all of the following are true:</p> <ul style="list-style-type: none"> - ignore the Responding Port_Name; - the Node_Name of the N_Port which receives the FARP-REQ equals the Responding Node_Name; - the IP address of the N_Port, which receives the FARP-REQ equals the Responding IP, address.
0000 0111	<p>The Match is successful if all of the following are true:</p> <ul style="list-style-type: none"> - the Port_Name of the N_Port which receives the FARP-REQ equals the Responding Port_Name; - the Node_Name of the N_Port which receives the FARP-REQ equals the Responding Node_Name; - the IP address of the N_Port, which receives the FARP-REQ equals the Responding IP, address.
all others	Reserved

- **Requesting N_Port Identifier:** The Requesting N_Port Identifier contains the Address Identifier of the N_Port issuing the FARP-REQ.
- **Responder Action:** The Responder Action defines the Responder action if the match requested by the Match Address Code Points is successful. These actions are defined in Table 123.
- **Requesting N_Port Port_Name:** The Requesting N_Port Port_Name contains the Port_Name of the N_Port issuing the FARP-REQ.

Table 123 - Responder Action	
Value (hex)	Definition
00	Perform no action
01	If the Match is successful, the receiving N_Port shall perform Login with the Requesting N_Port Identifier.
02	If the Match is successful, the receiving N_Port shall originate FARP-REPLY with the Requesting N_Port Identifier. The receiving N_Port shall not perform Login with the Requesting N_Port Identifier.
03	If the Match is successful, the receiving N_Port shall perform Login with the Requesting N_Port Identifier; and, if the Login is successful, the receiving N_Port shall originate FARP-REPLY with the Requesting N_Port Identifier.
all others	Reserved

- **Requesting N_Port Node_Name:** The Requesting N_Port Node_Name contains the Node_Name of the N_Port issuing the FARP-REQ.
- **Responding N_Port Port_Name:** The Responding N_Port Port_Name contains the Port_Name, which the receiving N_Port compares with its own Port_Name to determine a Match.
- **Responding N_Port Node_Name:** The Responding N_Port Node_Name contains the Node_Name, which the receiving N_Port compares with its own Node_Name to determine a Match.
- **Requesting N_Port IP Address:** The Requesting N_Port IP address contains the IP address of the N_Port issuing the FARP-REQ. The format of the IP address is IPv6 (see reference IP Version 6 Addressing Architecture).
- **Responding N_Port IP Address:** The Responding N_Port IP address contains the IP address, which the receiving N_Port compares with, its own IP address to determine a Match. The format of the IP address is IPv6 (see reference IP Version 6 Addressing Architecture).

The Match Address Code Points indicate which of the above fields shall be specified, as defined in Table 124. In this table, “correct” means the value of the field shall be correct for the N_Port; “ignored” means that the value shall be either zero, or any valid value, since the field is not examined for comparison.

Reply Link Service Sequence: none

Table 124 - FARP-REQ Address Fields						
Code Point Value (binary)	Requesting N_Port Port_Name	Requesting N_Port Node_Name	Requesting N_Port IP Address	Responding N_Port Port_Name	Responding N_Port Node_Name	Responding N_Port IP Address
0000 0001	correct	ignored	ignored	correct	ignored	ignored
0000 0010	ignored	correct	ignored	ignored	correct	ignored
0000 0011	correct	correct	ignored	correct	correct	ignored
0000 0100	ignored	ignored	correct	ignored	ignored	correct
0000 0101	correct	ignored	correct	correct	ignored	correct

0000 0110	ignored	correct	correct	ignored	correct	correct
0000 0111	correct	correct	correct	correct	correct	correct
all others	Reserved					

15.5.30.2 Fibre Channel Address Resolution Protocol Reply (FARP-REPLY)

The FARP-REPLY Extended Link Service request shall be used to respond to a FARP-REQ, which resulted in a successful Match, with a FARP-REQ Responder Action requesting a FARP-REPLY. Note that the Responder action indicates whether the Responder shall or shall not perform Login prior to the FARP-REPLY. Under no circumstances shall the Responder issue a LS_RJT to a FARP-REQ.

Protocol:

- FARP-REPLY request Sequence
- Accept (ACC) reply Sequence

Format: FT-1

Addressing: The S_ID designates the Responding N_Port providing addressing information. The D_ID designates the Requesting N_Port requesting addressing information.

Payload: The format of the FARP-REPLY request Payload is shown in Table 125.

Table 125 - FARP-REPLY Request Payload	
Item	Size Bytes
hex '55 00 00 00'	4
Match Address Code Points	1
Requesting N_Port Identifier	3
Responder Action	1
Responding N_Port Identifier	3
Requesting N_Port Port_Name	8
Requesting N_Port Node_Name	8
Responding N_Port Port_Name	8
Responding N_Port Node_Name	8
Requesting N_Port IP Address	16
Responding N_Port IP Address	16

- **Match Address Code Points:** The Match Address Code Points define the address(es) that were matched. The values of these code points are defined in table x2. The value of this field shall equal the value sent in the corresponding FARP-REQ request.
- **Requesting N_Port Identifier:** The Requesting N_Port Identifier contains the Address Identifier of the N_Port that issued the FARP-REQ.
- **Responding N_Port Identifier:** The Responding N_Port Identifier contains the Address Identifier of the N_Port, which is responding to the FARP-REQ.

- **Responder Action:** The Responder Action field shall be set to the same value as in the corresponding FARP-REQ.
- **Requesting N_Port Port_Name:** The Requesting N_Port Port_Name contains the Port_Name of the N_Port, which issued the FARP-REQ.
- **Requesting N_Port Node_Name:** The Requesting N_Port Node_Name contains the Node_Name of the N_Port, which issued the FARP-REQ.
- **Responding N_Port Port_Name:** The Responding N_Port Port_Name contains the Port_Name of the N_Port, which is responding to the FARP-REQ.
- **Responding N_Port Node_Name:** The Responding N_Port Node_Name contains the Node_Name of the N_Port, which is responding to the FARP-REQ.
- **Requesting N_Port IP Address:** The Requesting N_Port IP address contains the IP address of the N_Port, which issued the FARP-REQ. The format of the IP address is IPv6 (see reference IP Version 6 Addressing Architecture).
- **Responding N_Port IP Address:** The Responding N_Port IP address contains the IP address of the N_Port, which is responding to the FARP-REQ. The format of the IP address is IPv6 (see reference IP Version 6 Addressing Architecture).

The Match Address Code Points indicate which of the above fields shall be specified, as defined in Table 126. In this table, “correct” means the value of the field shall be correct for the N_Port; “ignored” means that the value shall be either the correct value, zero, or any valid value, since the field was not examined for comparison; “REQ” means that the value shall be equal to the value sent in the corresponding FARP-REQ request.

Table 126 - FARP-REPLY Address Fields						
Code Point Value (binary)	Requesting N_Port Port_Name	Requesting N_Port Node_Name	Requesting N_Port IP Address	Responding N_Port Port_Name	Responding N_Port Node_Name	Responding N_Port IP Address
0000 0001	REQ	REQ	REQ	correct	ignored	ignored
0000 0010	REQ	REQ	REQ	ignored	correct	ignored
0000 0011	REQ	REQ	REQ	correct	correct	ignored
0000 0100	REQ	REQ	REQ	ignored	ignored	correct
0000 0101	REQ	REQ	REQ	correct	ignored	correct
0000 0110	REQ	REQ	REQ	ignored	correct	correct
0000 0111	REQ	REQ	REQ	correct	correct	correct
all others	Reserved					

Reply Link Service Sequence

Service Reject (LS_RJT)

- signifies rejection of the FARP-REPLY command (see 15.6.2)

Accept (ACC)

- signifies that the N_Port has transmitted the requested data.
- The format of the Accept Payload is shown in Table 127.

Table 127 - FARP-REPLY Accept Payload

Item	Size Bytes
hex '02 00 00 00'	4

15.5.31 Topology-Discovery Extended Link Service

This clause defines the Extended Link Service, which supports the acquisition of topology information. *Topology Discovery* is the ability of a product to provide identifying information on request to management applications. Through the use of topology discovery, management applications are able to identify the elements of a storage area network. For this application, node specific information including Node Identification Data, that contains topology information, shall be acquired.

The ELS frame shall be transmitted using the FT-1 frame format via Class 2, Class 3, or Class F service. The following defines the header fields of the ELS frame:

- R_CTL: This field shall be set to hex '22' for request frames, and to hex '23' for Accept reply frames.
- CS_CTL: This field shall be set to hex '00'.
- D_ID and S_ID: Set as indicated for the specific ELS.
- TYPE: This field shall be set to hex '01', indicating Extended Link Service.

All other fields shall be set as appropriate according to the rules defined in FC-FS. The first word in the payload specifies the Command Code.

Requirements on discovery:

- topology independent (i.e. applicable for operation in a Fabric, a Loop, point-to-point, or bridged environment);
- Inband;
- utilizes existing protocols and Extended Link Services;
- FC-4 independent (e.g. doesn't require a SCSI stack in a VI node); discovery information is not bound to a specific node (i.e. the information can be disseminated to any node that requests it);
- edge nodes are not encumbered with fabric topology information; that is, edge nodes need only be cognizant of their own information and the attached (nearest neighbor) node's information.

N_Ports, NL_Ports, and Fabrics that comply with the Topology-Discovery function shall implement the following ELS:

- Request Topology Information (RTIN): Enables a requestor to acquire node specific information.

15.5.31.1 Request Topology Information (RTIN)

The Request Topology Information (RTIN) ELS relies on implementation of the Request Node Identification Data (RNID) functionality as a mechanism to collect Node Identification Data of the attached node. (See 15.5.32.1). Once all Node Identification Data has been collected, a host or any attached Nx_Port may query the Interconnection Element with the RTIN ELS to retrieve topology information. The normal response shall be an Accept (ACC) ELS Sequence

with records of topology information in its payload. If the recipient Nx_Port does not support the RTIN ELS, it shall reply with an LS_RJT ELS Sequence with a reason code of "command not supported".

If the recipient N_Port or NL_Port is not logged in with the requesting port it shall reply with an LS_RJT ELS Sequence with a reason code of 'unable to perform command request' with a reason code of 'N_Port_Login Required.'

It is permissible for a node to request topology information from any other node in the fabric. The RTIN ELS shall be sent to any node identified by its address identifier. The RTIN ELS is typically sent by a node to a switch via the Management Server at well-known address hex 'FF FF FA' or to an embedded NL_Port within a Hub identified by its address identifier.

When the RTIN ELS recipient (i.e. Interconnection Element) supports Topology-Discovery, it shall return the requested topology information in the payload of the Accept reply Sequence.

The management application may acquire this topology information for its own unique purposes. The usage of Topology-Discovery is beyond the scope of this standard, and will be described in the appropriate standard or technical report. Examples of Topology-Discovery usage are:

Protocol:

- Request Topology Information (RTIN) request Sequence
- Accept (ACC) reply Sequence

Format: FT-1

Addressing: The S_ID field designates the requesting source port. The D_ID field designates the port receiving the RTIN request.

Payload: The format of the RTIN Payload is shown in Table 128.

Table 128 - RTIN Payload	
Item	Size Bytes
hex '77 00 00 00'	4
Flag	1
Address Identifier of the Node	3
Maximum Size	2
Reserved	2

- **Flag:** Byte 0 of word 1 describes the manner in which selected fields of the RTIN payload are to be interpreted. The meaning of bits 0-7 is as follows:
 - **Bit 0 - Proxy:** The setting of this bit provides the interpretation of RTIN payload word 1 bytes1-3. When zero, bit 0 specifies that word 1 bytes1-3 contain the Domain_ID of the switch that topology information is being requested from; when one, the bit specifies that the recipient of the RTIN ELS should collect and provide topology information for the node identified by the FC address indicated in word 1 bytes1-3.
 - **1-7 - Reserved.**

- **Address Identifier of the node:** When the Flag field bit 7 is set to zero, this field specifies the Domain_ID of the switch that topology information is being requested from. When the Flag field bit 7 is set to one (i.e. Proxy), this field specifies the FC address to be proxied by the recipient.
- **Maximum Size:** This field manages the size of the information returned in an RTIN Accept ELS. The sender of an RTIN ELS may specify the maximum size of the RTIN ACC it is able to receive. If the recipient of the RTIN has more available information to send than allowed by the maximum size, it shall indicate the excess residual size in the RTIN ACC. The values for Maximum Size are interpreted as follows:

Value (hex)	Meaning
'0000'	No Maximum Size indicated; the RTIN may be any size.
'FFFF'	Reserved.
'xxxx'	Any other value: The encoded value indicates the maximum number of words that shall be sent in the RTIN ACC ELS, not inclusive of the command word 0.

Reply Extended Link Service Sequence:

Service Reject (LS_RJT)

- Signifies the rejection of the RTIN request

Accept (ACC)

- Signifies acceptance of the RTIN request and presents topology information.
- Accept Payload

The format of the Accept Payload is shown in Table 129.

Table 129 - RTIN Accept Payload	
Item	Size Bytes
hex '02 00 00 00'	4
Residual Size	2
Reserved	2
Connectivity Unit Header Record	24
Domain Record	4-240
Topology Information Records	140-max

- **Residual Size:** This field indicates the availability of more information that was not returned in the RTIN Accept ELS. When the recipient of the RTIN has more available information to send than allowed by the maximum size, it shall indicate the excess residual size in the RTIN ACC. The values for Maximum Size are interpreted as follows:

Value (hex)	Meaning
'00 00'	No Maximum Size indicated; the RTIN may be any size.

- 'FF FF' Reserved. Receipt of an RTIN request with a Maximum Field size set to 'FFFF' will result in and LS_RJT with a reason code of Protocol Error.
- 'xxxx' Any other value: The encoded value indicates the maximum number of words that shall be sent in the RTIN ACC ELS, not inclusive of the command word 0.

- **Connectivity-Unit-Header Record:** This record provides information about the Connectivity Units. A Connectivity Unit is an element that provides Fibre Channel connectivity between one or more nodes (e.g. fabric switch element, hub, etc.). The format of the Connectivity Unit Header Record is shown in Table 130.

Table 130 - Connectivity Unit Header Record	
Format	Size Bytes
Global ID	16
Domain ID or Address ID of the node	3
Reserved	1
Reserved	1
Total Connectivity Units in Interconnection Element	1
Total Entity Nodes in Connectivity Unit	2

- **Global ID:** The Global ID provides a World Wide unique Name using a format similar to the Name_Identifier format. This field, which is left justified, shall include the leading NAA bits, the remaining identifier bits, and any necessary trailing zeros. If a vendor unique value is used, the NAA field shall be '0011'b (i.e. locally assigned), and the remaining 124-bit field is set to a value specified by the vendor.
- **Domain ID or Address ID of the node:** This field specifies the Address Identifier of the node to which the RTIN was directed.
- **Total Connectivity Units in Interconnection Element:** This field indicates the total number of Connectivity Units in this Interconnection Element, and provides the number of Domain_IDs in the Domain Record of this RTIN ACC ELS.
- **Total Entity Nodes in Connectivity Unit:** This field indicates the total number of Entity nodes in this Connectivity Unit. The value specifies the number of Topology Information Records that are included in this RTIN ACC ELS. An "Entity node" is a node to which another node is attached.
- **Domain Record:** This record specifies the Domain_IDs of all Connectivity Units within this Interconnection Element. For a managed Hub, this record is a list of Hub vendor-unique identifiers, which is under the Hub controller's management. In a switched topology, the RTIN requestor uses the Domain_IDs to further collect topology information for other switches in the Fabric. The minimum size of this field is 4 bytes. The maximum size of this field is 240 bytes, since the highest Domain_ID is 239. The Domain Record shall be a multiple of 4 bytes with pad bytes as needed, and is shown in Table 131.

Table 131 - Domain Record

Format	Size Bytes
Domain_ID 1	1
Domain_ID 2	1
...	
Domain_ID n	1
Pad bytes (if necessary)	0-3

– **Topology Information Records:**

- One Topology-Information Record shall be provided for each Entity node in the Connectivity Unit. Each record contains a set of topology information, which includes an Entity-node Node-Identification-Data field, one or more Attached-Node-Port-Information fields, and one or more Attached-node Node-Identification-Data fields.
- A typical Topology Information Record for an Entity node in a non-loop configuration would contain an Entity-node Node-Identification-Data field (68 bytes), one Attached-Node-Port-Information field (4 bytes), and one Attached-node Node-Identification-Data field (68 bytes), for a total of 140 bytes. A typical Topology Information Record for an Entity node in a loop configuration would contain an Entity-node Node-Identification-Data field (68 bytes), a multiple of Attached-Node-Port-Information fields ($n * 4$ bytes) - one for each attached NL_Port, and the same multiple of Attached-node Node Identification Data fields ($n * 68$ bytes), for a total of $68 + (n * 4) + (n * 68)$ bytes. In the case of multiple attached nodes, each Attached-Node-Port-Information field shall be adjacent to its corresponding Attached-node Node-Identification-Data field. If no Attached-node Node Identification Data is available to an Entity node (i.e. Fabric login has not been completed), then no Topology Information Record shall be provided for this Entity node.
- The contents of an individual Topology Record is shown in Table 132.

Table 132 - Topology Information Record	
Format	Size Bytes
Entity-node Node Identification Data	68
Attached Node Port Information 1	4
Attached-Node Node Identification Data 1	68
• •	
Attached Node Port Information n	4
Attached-Node Node Identification Data n	68

The contents of an Entity-node Node Identification Data record are shown in Table 133.

Table 133 - Entity-node Node Identification Data Record	
Format	Size Bytes
N_Port_Name	8

Node_Name	8
Global ID	16
Unit Type	4
Physical Port Number	4
Number of Attached Nodes	4
IP Version	2
UDP Port Number	2
IP Address	16
Reserved	2
Topology Discovery Flags	2

The contents of an Attached Node Port Information record are shown in Table 134.

Table 134 - Attached Node Port Information Record	
Format	Size Bytes
Acquisition Flags	1
Address Identifier	3

The contents of an Attached-node Node Identification Data record are shown in Table 135.

Table 135 - Attached-node Node Identification Data Record	
Format	Size Bytes
N_Port_Name	8
Node_Name	8
Global ID	16
Unit Type	4
Physical Port Number	4
Number of Attached Nodes	4
IP Version	2
UDP Port Number	2
IP Address	16
Reserved	2
Topology Discovery Flags	2

The format of the Entity-node Node-Identification-Data and the Attached-node Node-Identification-Data fields are specified in the “RNID ELS Extension for Topology Discovery”. The first two parameters (16 bytes) comprise the Common Node Identification Data and the next nine parameters (52 bytes) comprise the Specific Node Identification Data for topology discovery using the Topology Discovery RNID ELS acquisition procedure.

The Number-of-Attached-Nodes parameter in the Entity-node Node-Identification-Data field shall be used to indicate the number of Attached-Node-Port-Information fields and Attached-node Node-Identification-Data fields that are contained in this Topology Information Record. For this application, the Number-of-Attached-Nodes parameter shall have a minimum value of one and a maximum value of 126.

The Attached-Node-Port-Information field shall contain a one-byte Acquisition Flags field and a three-byte Address Identifier. The Flags field shall contain two parameters; a two-bit validity/progress mask and a four-bit method value as shown in Figure 38.

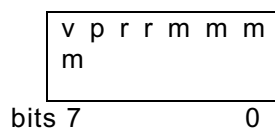


Figure 38 - Acquisition Flags

- **Validity bit 7 (v):** When one, the Attached-node Node-Identification-Data field is valid. When zero, the Attached-node Node-Identification-Data field is not valid.
- **Progress bit 6 (p):** When bit 7 is zero, indicating that the Attached-node Node-Identification-Data field is invalid, and bit 6 is one, the Attached-node Node-Identification-Data acquisition procedure is in progress, but has not completed. This bit is not applicable when bit 7 is one.
- **Method - bits 3-0 (m):** These four bits contain a coded value which specifies the procedure (method) used or being used to acquire the Attached-node Node-Identification-Data.

Value	Meaning
0	Topology Discovery RNID ELS
1 - F	Reserved

Inter-switch Handling of Request Topology Information

The Management Server is defined as a distributed service just as the Name Server in FC-SW-2. For RTIN ELS requests received at a local switch, that are bound for a remote switch, distributed Management Server requests are built. The local switch encapsulates the RTIN request using the FC-CT header, and forwards the request to the domain controller of the destination Domain_ID specified in the RTIN ELS. The remote switch services the RTIN request, assembles the RTIN response, and forwards the response to the local Management Server using the FC-CT header. The local switch re-moves the FC-CT header and forwards the RTIN Accept ELS to the requester. (Note for complete description of FC-CT see FC-GS-2).

FC-CT header (CT_HDR) values:

- FC_Type: hex 'FA' (Management Service Application);
- FC_Subtype: hex '01' (Management Service);
- Command Code: hex '01 00' (Request Topology Information);
- Maximum/Residual Size (same value as RTIN and RTIN ACC payloads' associated fields);
- Application Information Unit: RTIN payload.

15.5.32 Self-Description Extended Link Service

This clause defines the Extended Link Service, which supports the acquisition of Self-Description information. Self-Description is the ability of a product to provide identifying information (Node Identification Data) on request to compatible FC-4 facilities. Node Identification Data may be comprised of Common Node Identification Data, or Specific Node Identification Data, or both. Specific Node Identification Data contains product dependent information.

The ELS frame shall be transmitted using the FT-1 frame format via Class 2, Class 3, or Class F service.

The following defines the header fields of the ELS frame:

- R_CTL: This field shall be set to hex '22' for request frames, and to hex '23' for Accept reply frames.
- CS_CTL: This field shall be set to hex '00'.
- D_ID and S_ID: Set as indicated for the specific ELS.
- TYPE: This field shall be set to hex '01', indicating Extended Link Service.

All other fields shall be set as appropriate according to the rules defined in FC-FS. The first word in the payload specifies the Command Code. This optional Extended Link Service may be applicable for operation on a Fabric or point-to-point. N_Ports and Fabrics that comply with the Self-Description function shall implement the following ELS:

- Request Node Identification Data (RNID): Enables a requestor to acquire Self-Description Node Identification Data.

15.5.32.1 Request Node Identification Data (RNID)

The Request Node Identification Data (RNID) function shall provide a method for acquiring the Node Identification Data of the recipient node. The normal response shall be an Accept (ACC) ELS Sequence with Node Identification Data in its payload. If the recipient N_Port does not support the RNID ELS, it shall reply with an LS_RJT ELS Sequence with a reason code of "command not supported". If the recipient N_Port does not support the requested Node Identification Data format, it shall either reply with an LS_RJT ELS Sequence with a reason code of "unable to perform command request" and reason-code explanation of "unable to supply requested data" or reply with an Accept ELS containing only Common Node Identification Data in its payload.

If the recipient N_Port or NL_Port is not logged in with the requesting port it shall reply with an LS_RJT ELS Sequence with a reason code of 'unable to perform command request' with a reason code of 'N_Port_Login Required.'

The RNID ELS is typically sent by a N_Port to its nearest neighbor node. The node attached to the other end of the link is hereafter referred to as the nearest neighbor node. It is permissible for a node to request Node Identification Data from any other node in the Fabric. If the nearest neighbor node has been determined from the FLOGI to be a Fabric, then the RNID ELS shall be sent to the Fabric Controller at the well-known address hex 'FF FF FD'. If the nearest neighbor node has been determined from the FLOGI to be a N_Port, then the RNID ELS shall be sent to that N_Port identified by its address identifier.

The Fabric may acquire the Node Identification Data for all nodes attached to the switch's F_Ports or E_Ports with the address identifier of the attached node. Multiple switches in the same Fabric may acquire the Node Identification Data of the node attached to the other end of their Inter-Switch Link (ISL). For Node Identification Data acquisition between E_Port nodes, each switch may issue and respond to RNID ELSs across the ISL using Class F (see FC-SW-2).

When the RNID ELS recipient supports Self-Description, it shall return the requested Node Identification Data in the payload of the Accept reply Sequence.

The ULP may acquire this Node Identification Data for its own unique purposes. The usage of Self-Description is beyond the scope of this standard, and will be described in the appropriate ULP standard or technical report. Examples of Self-Description usage are:

- authentication of a system configuration;
- verification of a restored configuration with the known system configuration before a link disruption;
- coordination of Link-Incident reports.

Protocol:

- Request Node-Identification (RNID) request Sequence
- Accept (ACC) reply Sequence

Format: FT-1

Addressing: The S_ID field designates the requesting source port. The D_ID field designates the port receiving the RNID request.

Payload: The format of the RNID Payload is shown in Table 136.

Table 136 - RNID Payload	
Item	Size Bytes
hex '78 00 00 00'	4
Node Identification Data Format	1
Reserved	3

Node Identification Data Format: The format of the requested Node-Identification-Data is shown in Table 137.

Table 137 - Node Identification Data Format	
Description	hex
No format specified	'00'
Specific-coded value	'01' –'FF'

The Node Identification Data Format field shall indicate either no specified format requested or the specific-coded format requested by the client (e.g. ULP) that processes the data. The Node Identification Data Format shall be set to hex '00' when the requesting N_Port does not want Specific Node Identification Data (i.e. it is requesting only Common Node Identification Data). The Node Identification Data Format shall be set to the Specific-coded value when requesting data for a specific ULP (e.g. FCP, SBCCS, etc.). The Specific-coded value uses the TYPE codes defined in FC-4 data structures of FC-FS. These codes were selected because most cli-

ents will define Node Identification Data Formats for their own particular usage according to the unique FC-4 protocol.

Reply Extended Link Service Sequence:

Service Reject (LS_RJT)

- Signifies the rejection of the RNID request

Accept (ACC)

- Signifies acceptance of the RNID request and presents Node Identification Data.
- Accept Payload

The format of the Accept Payload is shown in Table 138.

Table 138 - RNID Accept Payload	
Item	Size Bytes
hex '02 00 00 00'	4
Node Identification Data Format	1
Common Node-Identification-Data Length	1
Reserved	1
Specific Node-Identification-Data Length	1
Common Node Identification Data	0 or 16
Specific Node Identification Data	0-max

– Node Identification Data Format:

- The format of the Node Identification Data Format field is shown in Table 139.
- The Node Identification Data Format field must be specified in the ACC ELS to an RNID ELS for indication of the Specific Node Identification Data Format in the response payload. The Node Identification Data Format shall be set to a Specific-coded value (FC-4 Device Type) for the client (e.g. FCP, SBCCS, etc.) requesting the data. If only the Common Node Identification Data is being provided (i.e. the Specific Node-Identification-Data length is zero), then the Node Identification Data Format field is meaningless.

Table 139 - Node Identification Data Format	
Description	hex
Reserved	'00'
Specific-coded value	'01' –'FF'

- **Common Node-Identification-Data Length:** When Common Node Identification Data is included in the RNID Accept payload, the Common Node-Identification-Data length shall specify 16 bytes, indicating the size of the Common Node-Identification-Data payload. (This

value does not include word0, word1, and the Specific Node Identification Data of the RNID Accept payload). When no Common Node Identification Data is included in the RNID Accept payload, the Common Node-Identification-Data length shall specify 0 bytes.

- **Specific Node-Identification-Data Length:** The Specific Node-Identification-Data length shall be specified according to the Node Identification Data Format requested for the respective ULP. The length shall specify the number of bytes in the Specific Node-Identification-Data payload. (This value does not include word0, word1, and the Common Node Identification Data of the RNID Accept payload). This value shall be a multiple of 4. The minimum value of this field is 0 bytes. The maximum value of this field is 252 bytes.
- **Common Node Identification Data:** When the Common Node-Identification-Data length indicates that Common Node Identification Data exists, the field shall contain the unit's Worldwide_Names (i.e. the N_Port_Name and the Node_Name), as shown in Table 140.

Table 140 - Common Node Identification Data	
Format	Size Bytes
N_Port_Name	8
Node_Name	8

- **Specific Node Identification Data:** The format of the Specific Node-Identification-Data field shall be dependent on the specified ULP (FC-4 Device Type). Specific Node Identification Data is subject to the ULP and its intended usage. For the specified FC-4 Device-Type Specific Node Identification Data, the ULP selects a set of product-identifying-information parameters. The ULP defines the format, fields used, and size of the parameters from the following list:
 - **Flags:** Describe the manner in which Specific Node Identification Data fields are to be interpreted. An example is a Validity Flag that indicates the validity of the information contained in the Specific Node Identification Data. Typical validity codes and their meanings are:
 - Specific Node Identification Data is valid.
 - Specific Node Identification Data is valid, however, it may not be current. This value may be used when the ULP has obtained the requested Specific Node Identification Data of its nearest neighbor but subsequently has observed some incident event (such as the loss-of-signal on a link) which may have resulted in a configuration change. The ULP has been unable to obtain the Specific Node Identification Data again.
 - Specific Node Identification Data is not valid. The ULP is unable to obtain the requested Specific Node Identification Data. Except for the flag-validity field, the contents of the Specific Node Identification Data will have no meaning.
 - **Node-Type:** Specifies the type of node described by this Specific Node Identification Data. Typical node-type specification includes SCSI target, SCSI initiator, SBCCS channel-subsystem, SBCCS device, etc.
 - **Node-parameters:** Contains additional classification information and ULP interface identification about the node-type. Examples are channel-subsystem, disk, tape, printer, etc.
 - **Type-number:** Provides a type number of the product.
 - **Model-number:** Provides a model number of the product.

- **Manufacturer:** Contains a field that identifies the manufacturer of the product.
- **Plant-of-manufacture:** Contains a plant code that identifies the plant of manufacture for the product.
- **Sequence-number:** Contains a multi-character sequence number of the product. The sequence number may represent part of the product's serial number.
- **Tag field:** Contains additional physical interface information about the product. An example for an SBCCS node is the physical address identifier for the product's interface that is identified by the other fields of the Specific Node Identification Data.

NOTE – The information in the Specific Node Identification Data should only be used to determine the product identity of the node. The information in the Specific Node Identification Data should not be used to determine the functional characteristics or the service capabilities of the node.

Special Handling of Node Identification Data

When a condition occurs that causes regression into the Login (FLOGI or PLOGI) portion of the initialization process for the nearest neighbor node, the N_Port may discard the neighboring Node Identification Data or may consider the neighboring Node Identification Data to be no longer current. When conditions permit, the N_Port shall attempt to acquire the current Node Identification Data of the attached node. If a N_Port sends a RNID ELS and receives an LS_RJT Sequence with a reason code of “command not supported”, the ULP shall be notified to establish the Node Identification Data for that node with an indication of “not valid”.

If a N_Port sends an RNID ELS and receives an LS_RJT Sequence with a reason code of “unable to perform command request” and reason-code explanation of “unable to supply requested data”, the ULP may cause the N_Port to retry the RNID ELS with a different Node Identification Data Format request, or establish the Node Identification Data with an indication of “not valid”. If a N_Port sends an RNID ELS and receives a reply with an Accept ELS containing only Common Node Identification Data in its payload, the ULP usage of this Node-Identification-Data and any further Self-Description processing is beyond the scope of this standard.

Self-Description processing for the purpose of discovering the topology of a fabric is not oriented or restricted to a specific ULP. Since the usage of the RNID ELS functionality is general and universal for this purpose, its description is provided in the standard.

Special Handling for Topology Discovery

As part of the Topology Discovery process being defined in FC-GS-3, a node shall issue RNID ELSs to all locally attached nodes (i.e. nearest neighbor nodes). The Specific-coded value in the Node Identification Data Format field shall be set to the value of hex 'DF' when requesting data for topology discovery. The Specific-coded value of hex 'DF' is a reserved value in the TYPE codes defined in FC-4 data structures of FC-FS. This coded value was selected because the Node Identification Data Format for topology discovery is universal to all clients that process this data, and is not unique to a particular FC-4 protocol. The expected response is an Accept ELS containing Node Identification Data in its payload. The RNID Accept ELS payload (see Table 138) shall contain the following:

- Node Identification Data Format field set to hex 'DF';
- Common Node-Identification-Data length set to decimal 16;
- Specific Node-Identification-Data length set to decimal 52;
- Common Node Identification Data (see Table 140);
- Specific Node Identification Data (see Table 141).

Table 141 - Topology Discovery Specific Node Identification Data	
Format	Size Bytes
Global ID	16
Unit Type	4
Physical Port Number	4
Number of Attached Nodes	4
IP Version	2
UDP Port Number	2
IP Address	16
Reserved	2
Topology Discovery Flags	2

Global ID: The Global ID provides a World Wide unique Name using a format similar to the Name_Identifier format. This field, which is left justified, shall include the leading NAA bits, the remaining identifier bits, and any necessary trailing zeros. If a vendor unique value is used, the NAA field shall be '0011'b (i.e. locally assigned), and the remaining 124-bit field is set to a value specified by the vendor.

Unit Type: Type of unit as shown in Table 142.

Table 142 - Topology Discovery Unit Type	
Value - hex	Type
'00 00 00 00'	Reserved
'00 00 00 01'	Unknown
'00 00 00 02'	Other (none of the following)
'00 00 00 03'	Hub
'00 00 00 04'	Switch
'00 00 00 05'	Gateway
'00 00 00 06'	Converter
'00 00 00 07'	HBA
'00 00 00 08'	Proxy-agent
'00 00 00 09'	Storage device (disk, CD, tape, etc)
'00 00 00 0A'	Host
'00 00 00 0B'	Storage subsystem (raid, library, etc)
'00 00 00 0C'	Module (subcomponent of a system)
'00 00 00 0D'	Software driver
'00 00 00 0E' - 'FF FF FF FF'	Reserved

NOTE – The Topology Discovery Unit types are exemplary of the OBJECT-TYPE values specified in the INTERNET-DRAFT “FC Mgmt Integration MIB”, dated Sept 14, 1999 (see www.ietf.org/internet-drafts/draft-ietf-ipfc-fcmgmt-int-mib-01.txt).

Physical Port Number: The numerical value that identifies the physical port, which has a Fibre-Channel link attached.

Number of Attached Nodes: The numerical count of attached node descriptors (i.e. an FL_Port may have multiple nodes attached to this port). The minimum value is zero. The maximum value is 126.

IP Version: The IP versions are shown in Table 143.

Table 143 - IP Version	
Version Description	Value
None	0
IP (IP version 4)	1
IP6 (IP version 6)	2
Reserved	3-65535

A value of zero (None) indicates that this node is not providing IP support, and the UDP Port and IP Address fields are meaningless.

UDP Port: The numerical value that identifies a port that is using the User Datagram Protocol (UDP).

IP Address: The IP address of the management entity.

Topology Discovery Flags: This flags field is a bit vector and is shown in Figure 39.

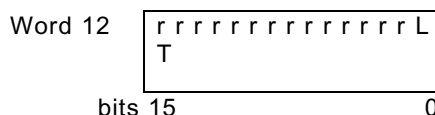


Figure 39 - Topology Discovery Flags

- **Word 12, bit 0 - Topology Discovery Support (T):** When asserted, the value signals that the node supports further topology-discovery inquiries.
- **Word 12, bit 1 - Loop Position Valid (L):** When asserted and multiple nodes are reported, the value signals that the Node-Identification-Data records reported are in the order detected from the successful execution of a Loop Initialization Report Primitive.

A node that supports topology-discovery processing (i.e. capable of handling queries for topology information) shall have stored internally its Node Identification Data. In the case of a Fabric Element or an embedded NL_Port within a Hub, the node collects the topology information for each port that has an attachment (including FL_Ports). Management requests or Inband topology-discovery requests may use this stored Node Identification Data. The usage of this Node-Identification-Data and any further discovery processing is beyond the scope of this standard.

15.5.33 Link-Incident-Reporting Extended Link Services

15.5.33.1 Description of link incident reporting Extended Link Services

This clause defines the Extended Link Services that are provided for the support of Link-Incident reporting. Link-Incident reporting is the ability of a product to provide link-incident information to minimize and reduce problem determination in a Fibre-Channel environment. Specifically, it assists in correlating and isolating a link-incident condition to the failing node or link interface. The Link-Incident-Reporting function asynchronously provides Link Incident Records to compatible FC-4 facilities.

The following terms are used to reference the ports, nodes, and records related to link incident reporting.

- incident port: The port on which the link incident occurred and which detected the link incident.
- connected port or connected node: The port or node connected directly to the incident port by a Fibre Channel link.
- reporting port: The port which sends the Link Incident Record.
- registered port: A port which is registered with the incident port to receive link incident reports.
- reporting node: The node which creates the Link Incident Record describing the incident on the incident port.

These ELS frames shall be transmitted using the FT-1 frame format via Class 2, Class 3, or Class F service. The following defines the header fields of these ELS frames:

- R_CTL: This field shall be set to hex '22' for request frames, and to hex '23' for Accept reply frames.
- CS_CTL: This field shall be set to hex '00'.
- D_ID and S_ID: Set as indicated for the specific ELS.
- TYPE: This field shall be set to hex '01', indicating Extended Link Service.

All other fields shall be set as appropriate according to the rules defined in FC-FS. The first word in the payload specifies the Command Code.

The following optional Extended Link Services may be applicable for operation on a Fabric, a Loop, or point-to-point. N_Ports, NL_Ports, and Fabrics that comply with the Link-Incident-Reporting functions shall implement these ELSs:

- Registered Link Incident Record (RLIR): Sends a Link Incident Record after a link-incident condition is detected at a particular node.
- Link Incident Record Registration (LIRR): Allows a N_Port or NL_Port to indicate that it is a registered port capable of receiving the Registered Link Incident Record (RLIR) ELS.

15.5.33.2 Registered Link Incident Record (RLIR)

The Registered Link Incident Record (RLIR) function shall provide a method for a reporting N_Port or NL_Port to send a Link Incident Record to a registered N_Port or NL_Port. The normal response to an RLIR ELS Sequence shall be an Accept (ACC) ELS Sequence with no payload. If the recipient N_Port or NL_Port does not support the RLIR ELS, it shall reply with an LS_RJT ELS Sequence with a reason code of 'command not supported'. If the recipient N_Port or NL_Port cannot accept the specified Link Incident Record type, it shall reply with an LS_RJT ELS Sequence with a reason code of 'unable to perform command request'. If the recipient N_Port or NL_Port is not logged in with the requesting port it shall reply with an LS_RJT ELS

Sequence with a reason code of 'unable to perform command request' with a reason code of 'N_Port_Login Required.'

A port shall recognize a link incident when a condition is detected for which an incident code is defined. When a link incident is recognized and a recipient of Link Incident Records is registered for the incident port, a Link Incident Record that contains information related to the link incident shall be created. When a Link Incident Record is generated, the reporting port shall use the link incident reporting procedure to pass the Link Incident Record to a registered port. The link incident reporting mechanism, if any, for a port that does not use this link incident reporting procedure is beyond the scope of this standard.

In the event that the incident port or its attached link is not operational, a reporting node is allowed to transmit the Link Incident Record to the selected registered port(s) from any other single port which is aware of the Link Incident Record.

A reporting N_Port or NL_Port shall attempt to report a Link Incident Record on behalf of the incident port by sending an RLIR ELS after it has determined that at least one port is registered for link incident reporting. The registered N_Port or NL_Port shall respond with an Accept (ACC) ELS with no payload.

The usage of Link Incident Records is beyond the scope of this standard. Link Incident Record usage may include reporting link anomalies, problem determination, and isolation of link errors to the failing node's Field Replaceable Unit (FRU).

Protocol:

- Registered-Link Incident Record (RLIR) request Sequence
- Accept (ACC) reply Sequence

Format: FT-1

Addressing: The S_ID designates the reporting port that is providing the Link Incident Record. The D_ID field designates the registered port.

Payload: The format of the RLIR Payload is shown in Table 144.

Table 144 - RLIR Payload	
Item	Size Bytes
hex '79 00 00 00'	4
Link Incident Record Format	1
Common Link Incident Record Length(= 16 or 64)	1
Common Link-Incident-Descriptor Length (=4)	1
Specific Link Incident Record Length	1
Common Link Incident Record	16 or 64
Common Link-Incident Descriptor	4
Specific Link Incident Record	0-max

Link Incident Record Format: The format of the Link Incident Record-Format field is shown in Table 145.

Table 145 - Link Incident Record Format	
Description	hex
Reserved	'00'
Specific-coded value	'01'-'FF'

The Specific-coded value field of the Link Incident Record Format shall be set to the FC-4 Device Type registered for by the client that is being presented with the incident record (e.g. FCP, SBCCS, etc.). The Specific-coded value uses the TYPE codes defined in FC-4 data structures of FC-FS. These codes were selected because most clients will define Link Incident Record formats for their own particular usage according to the unique FC-4 protocol. A value of zero in the Specific-coded value field indicates that only the Common Link Incident Record and the Common Link Incident Descriptor are being reported and that the Specific Link Incident Record Length shall be zero.

Common Link Incident Record Length: When only the incident port's Name_Identifier (i.e. the N_Port_Name and the Node_Name) are included in the RLIR ELS payload, the Common Link Incident Record length shall specify 16 bytes, indicating the size of the Common Link Incident Record field. When the additional Common Link Incident Record fields are included in the payload, the Common Link Incident Record Length shall be 64.

Link Incident Descriptor Length: The Link Incident Descriptor length shall be set to 4 bytes specifying the number of bytes in the Link Incident Descriptor field.

Specific Link Incident Record Length: The Specific Link Incident Record Length shall be specified according to the Link Incident Record Format indicated for the respective ULP. The length shall specify the number of bytes in the Specific Link Incident Record field. This value shall be a multiple of 4. The minimum value of this field is 0 bytes. The maximum value of this field is 252 bytes.

Common Link Incident Record Data: The Common Link Incident Record shall contain the unit's Name_Identifier (i.e. the N_Port_Name and the Node_Name), as shown in Table 146. The Common Link Incident Record may optionally contain the additional fields shown in Table 146. Optional fields that have unknown values shall be set to zero.

Table 146 - Common Link Incident Record Data		
Format	Size - Bytes	Mandatory/ Optional
Incident port N_Port_Name	8	Mandatory
Incident port Node_Name	8	
Incident Port Type	1	Optional
Incident Port Identifier	3	
Connected port Port_Name	8	
Connected port Node_Name	8	
Fabric_Name	8	
Incident physical port number	4	

transaction id	4	
reserved	3	
time stamp format	1	
time stamp	8	

Incident port N_Port Name: The Name_Identifier that is the incident port's N_Port_Name.

Incident port Node_Name: The Name_Identifier that is the incident port's Node_Name.

Incident Port Type: The value in the Incident Port Type field specifies the type of the incident port. Port Type values are defined in FC-GS-3. A value of zero in the field indicates the Incident Port Type is unknown or unspecified.

Incident Port Identifier: The 24 bit N_Port Identifier of the incident port. All ports shall supply a valid Domain address (bits 23 - 16). Ports that do not have complete N_Port Identifiers may use zeros for the lower two bytes. A value of all zero in the Port Identifier field indicates the Port Identifier is unknown or unspecified.

Connected Port Port_Name: The Name_Identifier that is the connected port's N_Port_Name.

- If the incident port is an NL_Port in a private loop, the connected port shall be the port associated with the link incident, if known.
- If the incident port is an NL_Port in a public loop, the connected port shall be the FL_Port.
- If the incident port is an N_Port connected to a fabric, the connected port shall be the F_Port.
- If the incident port is an FL_Port, the connected port shall be the port associated with the link incident, if known.

If the connected port's Port_Name is unknown or unspecified, the Connected Port Port_Name may be zero.

Connected Port Node_Name: The Name_Identifier that is the Node_Name of the port described by the Connected Port Port_Name. If the connected port is an F_Port, the Node_Name may be the Fabric_Name, the Node_Name of the local fabric element, or unspecified. If the Node_Name is unknown or unspecified, the corresponding field shall be zero.

Fabric_Name: The Fabric_Name of the incident port's fabric. For Nx_Port devices this is the value obtained in the FLOGI ACC payload. Private loop devices do not have a Fabric_Name and shall report zeros.

Incident Physical Port Number: The vendor specific identification of the incident port within the unit having Fibre Channel ports. When this field is presented in the Common Link Incident Record, the Incident Physical Port Number shall be valid.

Transaction ID: A 32 bit value that starts at one and is incremented by one for each link incident record generated by the reporting node. Applications receiving link incident records may use this value to eliminate duplicates or detect missing records. The value of zero shall be used only to indicate that the Transaction ID is unknown or unspecified.

Time Stamp Format: This field specifies the format of the Time Stamp field.

Value	Meaning
'00'hex	The Time Stamp field is unknown or unspecified.

- '01'hex Time Server: The 64 bit time stamp is reported in units of seconds and fractions of a second. The time stamp uses the value obtained from the Time Server (see FC-GS-3).
- '02'hex Clock synchronization format: The 64-bit time stamp is reported as defined for the Clock Synchronization Update (CSU) ELS.
- 03h-FFh reserved

Time Stamp: The time stamp value in the format specified by the Time Stamp Format field. If the Time Stamp Format value is 0, the Time Stamp field is unknown or unspecified and may have any value.

Link-Incident Descriptor: The format of the Link-Incident Descriptor is shown in Figure 40.

Byte/bit Word	0 31	1 23	2 15	3 7
0	IQ	IC	EPAI (domain/area of ILS)	

Figure 40 – Common Link-incident Descriptor

Incident Qualifier (IQ)

Word 0 bits 31-24 (byte 0 of the Common Link-Incident Descriptor) qualify the manner in which the contents of the Link Incident Record shall be interpreted. The meaning of bits 31-24 is as follows:

Bits Meaning

31 Reserved.

30 Reserved.

29 **Switch:** When one, bit 29 indicates that the incident port is a port on a switch node. When zero, bit 29 indicates that the incident port is not a port on a switch node.

28 bit **Expansion Port:** When bit 29 is one, indicating that the incident port is a switch port, indicates whether or not the switch port is an Inter-Switch-Link Expansion port (E_Port). When one, bit 28 indicates that the switch port is an Inter-Switch-Link Expansion port. When zero, bit 28 indicates that the switch port is not an Inter-Switch-Link Expansion port.

27-26 **Reporting-status order:** Bits 27-26 constitute a two-bit code that identifies the reporting status order for the link incident. The codes and their meanings are as follows:

Value Meaning

0 **Informational report:** Link incidents use reporting-status order 0 to provide information only.

1 **Link degraded but operational:** Link incidents use reporting-status order 1 if the link associated with the incident port is not in a Link-Failure or Offline

state as a result of the event which generated the Link Incident Record.

- 2 **Link not operational:** Link incidents use reporting-status order 2 if the link associated with the incident port is in a Link-Failure or Offline state as a result of the event which generated the Link Incident Record.
- 3 Reserved.
- 25 **Subassembly type:** When one, bit 25 specifies that the type of subassembly used for the port that is the subject of this Link Incident Record is laser. When zero, bit 25 specifies that the type of subassembly used for the port that is the subject of this Link Incident Record is not laser.
- 24 **FRU identification:** When one, bit 24 specifies that the Specific-Link Incident Record Data is in a format that provides field-replaceable-unit (FRU) identification. When zero, bit 24 specifies that the Specific-Link Incident Record Data is not in a format that provides field-replaceable-unit (FRU) identification.

Incident Code (IC)

- **Word 0 bits 23-16 (byte 1 of the Link-Incident Descriptor):** contains the incident code that describes the incident that was observed by the incident node.
- **Bit 23 is reserved.**
- **Bits 22-16:** value that specifies the type of incident that was observed. The values that can be specified and their meanings are as follows:

Value	Meaning
0	Reserved.
1	Implicit incident: A condition, caused by an event known to have occurred within the incident port, has been recognized by the incident port. The condition affects the attached link in such a way that it may cause a link incident to be recognized by the connected port.
2	Bit-error-rate threshold exceeded: The number of code-violation errors recognized by the incident port has exceeded a threshold (see 23.2.7).
3	Link Failure - loss of signal or synchronization: The incident port has recognized a Loss-of-Synchronization condition, and it persisted for more than the R_T_TOV timeout period. The incident port has recognized a Loss-of-Signal condition. (see 6.1.3.1).
4	Link Failure - Not-Operational Primitive Sequence (NOS) recognized: The Not-Operational Primitive Sequence has been recognized by the incident port (see 10.5.3.2).
5	Link Failure - Primitive Sequence timeout: The incident port has recognized either a Link-Reset-Protocol timeout (see 10.5.2.1, 10.5.2.2 and 10.5.2.3) or a timeout when timing for the appropriate response while in the NOS Receive state and after NOS is no longer recognized (see 10.5.3.2).
6	Link Failure - Invalid Primitive Sequence for port state: The incident port recognized either a LR or LRR Primitive Sequence while in the Wait-for-OLS state (see 10.5.4.3).
7	Link Failure - Loop Initialization time out: The incident port failed to

- complete loop initialization within the normal loop time out period (see FC-AL-2).
- 8 **Link Failure – receiving LIP(F8):** The incident port is receiving LIP(F8) indicating some other port on the loop is experiencing a loss of signal condition.
- 9 -127 Reserved.

After a Link Failure is detected, the reporting node may delay recognizing and considering this link failure as a reportable link incident condition, for a specific period of time. Some FC-4 protocols may be capable of transparently recovering from a Link Failure condition, and may not generate a Link Incident Record. If the Link Failure condition persists for longer than the FC-4 specified time out period, then a Link Incident Record is generated and reported.

Expansion Port Address Identifier (EPAI): When word 0 bits 29 and 28 (IQ field) are both ones, indicating that the incident port is an Inter-Switch-Link Expansion port, word 0 bits 15-0 (bytes 2 and 3 of the Link-Incident Descriptor) contain the Domain and Area address identifiers (hex 'dd aa') of the connected Expansion port on this ISL.

Specific Link Incident Record Data: The format of the Specific Link Incident Record-Data field shall be dependent on the Link Incident Record Format. Specific Link Incident Record Data is subject to the client's requirements and intended usage. The Specific Link Record Data may include summaries of RNID node information and other parameters such as:

- Incident port control descriptors: Contains flag, qualifier, and control parameters pertaining to the incident record. The contents of Incident-Control Descriptors are described in the appropriate ULP standard or technical report.
- Incident port's node descriptor: Contains the Node Identification Data of the incident node. The contents of Node Identification Data are described in the appropriate ULP standard or technical report.
- Connected port's node descriptor: Contains the node identification data of the connected port's node. The contents of node identification data are described in the appropriate ULP standard or technical report.
- Incident specific information: When word 0 bit 24 (IQ field) indicates that the Specific Link Incident Record Data is in a format that provides field replaceable unit (FRU) identification, then the incident specific information contains the FRU identification. When word 0 bit 24 indicates that the Specific Link Incident Record Data is not in a format that provides field replaceable unit identification, then the incident specific information contains model dependent incident information.

Reply Extended Link Service Sequence:

Service Reject (LS_RJT)

- Signifies the rejection of the RLIR request

Accept (ACC)

- Signifies acceptance of the RLIR request and its Link Incident Record.
- Accept Payload

The format of the Accept Payload is shown in Table 147.

Table 147 - RLIR Accept Payload
--

Item	Size Bytes
hex '02 00 00 00'	4

Link-Incident reporting procedure

The detecting node shall generate one Link Incident Record per link incident. The number of Link Incident Record Format for each link incident. The number of Link Incident Records a reporting node may hold is model dependent. If the incident port's reporting node attempts to generate a Link Incident Record but does not have resources to hold the record, the oldest Link Incident Record shall be discarded, and the new Link Incident Record shall be retained.

The port selected as the reporting port by the reporting node shall present the Link Incident Records by sending RLIR ELSs to ports registered as valid recipients of Link Incident Records. After sending all required RLIR ELSs containing the Link Incident Record, the port may discard the Link Incident Record.

The reporting port shall select ports from the registrant list for each Link Incident Record Format. The RLIR will first be sent to all registered ports that have registered to always receive RLIRs. If no registered ports are registered to always receive RLIR or if RLIR was not successfully transmitted to at least one of those ports registered to always receive RLIRs, then the reporting port shall send the RLIR to a registered port selected from among those that have registered to conditionally receive RLIR. If RLIR is not successfully transmitted to the selected registered port registered to conditionally receive RLIR, the sending port shall select another port, if any, from the established registration list. The RLIR ELS is considered successfully transmitted if an ACC payload is received for the RLIR within 2xR_A_TOV.

If a reporting port cannot successfully deliver an RLIR ELS to a registered port, the registered port's registration for the incident port may be discarded. If a reporting node cannot successfully deliver an RLIR ELS through a particular reporting port, it may select a different reporting port and try again. If no registration is established, it is model dependent as to whether the incident port's reporting node generates and retains or discards the Link Incident Record.

15.5.33.3 Link Incident Record Registration (LIRR)

The Link Incident Record Registration (LIRR) Extended Link Service requests the recipient to add or remove this source N_Port or NL_Port to or from the list of N_Ports and NL_Ports registered to receive the Registered Link Incident Record (RLIR) ELS. The normal response to a LIRR ELS Sequence shall be an Accept (ACC) ELS Sequence with no payload. If the recipient N_Port or NL_Port does not support the LIRR ELS, it shall reply with a LS_RJT ELS Sequence with a reason code of "command not supported". If the recipient N_Port or NL_Port cannot perform the requested LIRR registration function or the specified format is not supported, it shall reply with a LS_RJT ELS Sequence with a reason code of "unable to perform command request".

When a N_Port or NL_Port is registering or de-registering for receipt of Link Incident Records with a Fabric, the LIRR ELS shall be sent to the Management Server at well-known address hex 'FFFFFFA'.

Protocol:

- Link Incident Record-Registration (LIRR) request Sequence
- Accept (ACC) reply Sequence

Format: FT-1

Addressing: The S_ID designates the source port requesting registration for Link Incident Records. The D_ID field designates the port receiving the registration request.

Payload: The format of the LIRR Payload is shown in Table 148.

Table 148 - LIRR Payload	
Item	Size Bytes
hex '7A 00 00 00'	4
Registration Function	1
Link Incident Record-Registration Format	1
Reserved	2

Registration Function: The Registration function shall specify the mode of registration; that is, whether the recipient node adds or removes the sending N_Port or NL_Port to or from its list of registered nodes, hereafter referred to as the established registration list. The format of the Registration Function field is shown in Table 149.

Table 149 - Registration Function	
Function	value
Reserved	0
Set registration – conditionally receive:	1
Set registration – always receive:	2
Reserved	3 - 254
Clear registration:	255

Set registration – conditionally receive: The source port is registered as a valid recipient of subsequent RLIR ELSs for the format specified. The port is added to the appropriate format specific established registration list. This source port is chosen as the recipient of a link incident record only if no other recipients from this established registration list have been chosen.

Set registration – always receive: The source port is registered as a valid recipient of subsequent RLIR ELSs for the format specified. The port is added to the appropriate format specific established registration list. This source port is always chosen as a recipient of a link incident record.

Clear Registration: The source port is de-registered as a valid recipient of subsequent RLIR ELSs for the format specified (i.e. remove from the established registration list).

Link Incident Record-Registration Type: The type of the requested Link Incident Record Registration is shown in Table 150.

Table 150 - Link Incident Record-Registration Format	
Description	hex
Common Format	'00'
Specific-coded value	'01'-'FF'

A Link Incident Record Registration Format of 00 specifies that only the Common Link Incident Record and the Link Incident Descriptor is reported. If the Link Incident Record Registration format is non-zero, it shall be set to the Specific-coded value (FC-4 Device Type) of the registration format being requested for a specific client ULP (e.g. SCSI-FCP, SBCCS, etc.). The Specific-coded value uses the TYPE codes defined in FC-4 data structures of FC-FS. These codes allow clients to define and register for Link Incident Record formats for their own particular usage according to the unique FC-4 protocol

Reply Extended Link Service Sequence:

Service Reject (LS_RJT)

- Signifies the rejection of the LIRR request

Accept (ACC)

- Signifies acceptance of the LIRR request and the registration for RLIR ELSs.
- Accept Payload

The format of the Accept Payload is shown in Table 151.

Table 151 - LIRR Accept Payload	
Item	Size Bytes
hex '02 00 00 00'	4

Registration for Link Incident Records

A ULP that desires to obtain and process Link Incident Records must cause its port to register with other ports of interest. The term 'ports of interest' refers to all ports which the registered port can address and for which it desires their Link Incident Records.

A N_Port or NL_Port shall attempt to register as a valid recipient of subsequent Link Incident Records by sending a LIRR ELS as soon as the requesting N_Port or NL_Port has determined its address identifier and has determined the address identifiers for the ports of interest. The ULP must indicate the format of Link Incident Records of interest that are being registered. The recipient N_Port or NL_Port shall respond with an Accept (ACC) ELS with no payload.

When a registered ULP is no longer interested in receiving Link Incident Records from any port or ports of interest, it shall send a LIRR ELS to de-register their port as a valid recipient of Link Incident Records. The ULP must indicate the format of Link Incident Records that are being de-registered.

If a LIRR ELS requesting a de-registration function for a port that is not registered is received, the recipient port accepts the request by sending an Accept (ACC) ELS and performs no further action. If a LIRR ELS requesting a registration for a port that is already registered for the

specified format, the recipient port shall accept by sending an Accept (ACC) ELS request and perform no further action.

The recipient of a LIRR ELS shall maintain a separate established registration list of valid Link Incident Record recipients for each registration format specified. The size of the established registration list maintained by each port for each specified format is model dependent. If the established registration list is full when an add registration function occurs, the port does not accept this registration, and rejects this request indicating “unable to perform command request”.

A Logout occurring between a node and a valid-registered-recipient node shall cause de-registration of the valid-recipient port in the established registration list maintained by the port.

Responsibilities of Valid-Registered Recipients

A valid-registered recipient shall be capable of receiving and processing Link Incident Records that are generated according to the Link-Incident reporting procedure (see 15.5.33). The usage and coordination of Link Incident Records by ULPs is beyond the scope of this standard.

If a valid registered recipient receives a Link Incident Record for a format that does not match its registration format, the recipient port shall reject the RLIR ELS with a reason code of “unable to perform command request”.

15.5.34 Fabric Link Services for Alias

The following additional Fabric Controller Link Services are defined:

- Get Alias_ID (GAID)
- Fabric Activate Alias_ID (FACT)
- Fabric Deactivate Alias_ID (FDACT)

15.5.34.1 Get Alias_ID (GAID)

This request is sent to the Fabric Controller by Alias Server (see Figure 69) to request a unique alias address identifier to be associated with the Alias Group indicated in the passed Alias_Token. The payload of the request contains, among other parameters, the Alias_Token of the Alias Group to be created. The format of the payload is shown in Table 152.

Table 152 - Get Alias_ID Payload	
Item	Size - Bytes
hex '30 00 00 00'	4
Alias_Token	12
Alias_SP	80
NP_List_Length	4
NP_List (1)	4
NP_List (2 to n-1)	(n-2) * 4
NP_List (n)	4

Alias Group Token (Alias_Token): This is the token identifying the Alias Group. See FC-GS for a description of the Alias_Token.

Alias Group Service Parameters (Alias_SP): The Alias_SP defines the Service Parameters to be used for all operations with this Alias Group. The Service Parameters are passed as defined by the Common Service Parameters and the appropriate Class Service Parameters.

NOTE - This field is used by the Fabric Controller to ensure that the Service Parameters to be used by the Alias Group are not in conflict with the Fabric Service Parameters.

N_Port List Length (NP_List_Length): The NP_List_Length specifies the number of entries in the following NP_List.

N_Port List (NP_List): The NP_List contains one entry for each N_Port address identifier to be included in the Alias Group. The N_Port address identifier shall be right aligned within the NP_List entry. That is, the high-order byte of the entry shall be ignored and the low-order 3 bytes shall contain the N_Port address identifier. The N_Port address identifier shall not be an Alias_ID.

NOTE - The Fabric Controller uses the NP_List to determine whether it can support an Alias Group composed of these N_Ports. For example, if the Fabric cannot support an Alias Group that spans Domains, and the NP_List contains N_Ports in multiple Domains, the request may be rejected. Alternatively, the Fabric may form an Alias Group from the N_Ports within a single Domain.

The format of the payload for the LS_ACC indicating that this request has been successfully completed is shown in Table 153.

Table 153 - Get Alias_ID Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4
Alias_ID	4

Alias Group Identifier (Alias_ID): This is the alias address identifier that the N_Port will now recognize, in addition to its native address identifier.

a LS_RJT shall be returned if an Alias_ID cannot be generated. A Reason Code of “Unable to Perform Command Request” shall be generated, along with one of the Reason Explanations in Table 196.

15.5.34.2 Fabric Activate Alias_ID (FACT)

This request is sent to a Fabric by Alias Server (see Figure 69) to cause it to assign the passed Alias_ID as an alias for the passed N_Ports. The payload of the request contains, among other parameters a list of the N_Ports for which the alias identifier is to be recognized and the Alias_ID to be recognized as an alias identifier. The format of the payload is shown in Table 154.

Table 154 - Fabric Activate Alias_ID Payload	
Item	Size - Bytes
hex '31 00 00 00'	4
Alias_ID	4
NP_List_Length	4
NP_List(1)	4
NP_List (2 to n-1)	$(n - 2) * 4$
NP_List(n)	4

Alias Group Identifier (Alias_ID): This is the alias address identifier that the Fabric shall recognize for the N_Ports in the list, in addition to their native address identifiers.

N_Port List Length (NP_List_Length): The NP_List_Length specifies the number of entries in the following NP_List.

N_Port List (NP_List): The NP_List contains one entry for each N_Port for which the Alias_ID is to be assigned as an alias identifier. The N_Port address identifier shall be right aligned within the NP_List entry. That is, the high-order byte of the entry shall be ignored and the low-order 3 bytes shall contain the N_Port address identifier.

The format of the payload for the LS_ACC indicating that this request has been successfully completed is shown in Table 155.

Table 155 - Activate Alias_ID Accept Payload	
Item	Size (Bytes)
hex '02 00 00 00'	4

An LS_RJT shall be returned if an Alias_ID cannot be activated as an alias identifier. A Reason Code of “Unable to Perform Command Request” shall be generated, along with one of the Reason Explanations in Table 196.

15.5.34.3 Fabric Deactivate Alias_ID (FDACT)

This request is sent to the Fabric Controller by the Alias Server (see Figure 69) to request that it deassign the indicated Alias_ID as an Alias Group alias identifier for the passed N_Ports. The payload of the request contains, among other parameters, the Alias_ID to be deactivated, and the list of N_Ports for which this Alias_ID is to be deactivated. When there are no longer any N_Ports for which the Alias ID is active, the Fabric Controller shall free up the Alias ID for subsequent reassignment. The format of the payload is shown in Table 156.

Table 156 - Fabric Deactivate Alias_ID Payload	
Item	Size (Bytes)
hex '32 00 00 00'	4
Alias_ID	4
NP_List_Length	4
NP_List(1)	4
NP_List (2 to n-1)	$(n - 2) * 4$
NP_List(n)	4

Alias Group ID (Alias_ID): This is the alias identifier to be deactivated.

N_Port List Length (NP_List_Length): The NP_List_Length specifies the number of entries in the following NP_List.

N_Port List (NP_List): The NP_List contains one entry for each N_Port for which the MG_ID is to be deactivated as an alias identifier. The N_Port address identifier shall be right aligned within the NP_List entry. That is, the high-order byte of the entry shall be ignored and the low-order 3 bytes shall contain the N_Port address identifier.

The format of the payload for the LS_ACC indicating that this request has been successfully completed is shown in Table 157.

Table 157 - Fabric Deactivate Alias_ID Accept Payload	
Item	Size (Bytes)
hex '02 00 00 00'	4

An LS_RJT shall be returned if the Alias_ID cannot be deactivated as an alias identifier. A Reason Code of “Unable to Perform Command Request” shall be generated, along with one of the Reason Explanations in Table 196.

15.5.35 N_Port Link Services for Alias

The following additional N_Port Link Services are defined:

- N_Port Activate Alias_ID (NACT)
- N_Port Deactivate Alias_ID (NDACT)

15.5.35.1 N_Port Activate Alias_ID (NACT)

This request is sent to a N_Port by Alias Server (see Figure 69) to cause it to assign the passed alias address identifier as an alias. The payload of the request contains, among other parameters, the Alias_Token of the Alias Group which the N_Port will be joining, the Service Parameters associated with that Alias Group, and the alias address identifier to be assigned to the N_Port. The format of the payload is shown in Table 158.

Table 158 - N_Port Activate Alias_ID Payload	
Item	Size (Bytes)
hex '33 00 00 00'	4
Alias_Token	12
Alias_ID	4
Alias_SP	80

Alias Group Token (Alias_Token): This is the token identifying the Alias Group. It contains the Alias_Class and Alias_Qualifier of the Alias Group the N_Port is being requested to join. The N_Port validates that it supports that particular Alias Class. See FC-GS for a description of the Alias_Token.

Alias Group Identifier (Alias_ID): This is the alias address identifier that the N_Port will now recognize, in addition to its native address identifier.

Alias Group Service Parameters (Alias_SP): The Alias_SP defines the Service Parameters to be used for all operations with this Alias Group.

NOTE - These Service Parameters may differ from those passed during Login.

For Multicast Groups, only the Common Service Parameters and Class 3 Service Parameters are actually used. For Hunt Groups, all Service Parameters may be used.

These Service Parameters are used to perform an implicit Login among the members of the Alias Group.

If the N_Port can successfully perform the following functions, it shall transmit an LS_ACC to notify the Alias Server that it:

- Can support the Alias_Class in the Alias_Token.
- Can support the Alias Group Service Parameters specified in Alias_SP.
- Has assigned the passed Alias_ID as an alias for this N_Port.

If the N_Port cannot perform all of the above functions, it shall send an LS_RJT as a reply.

The format of the payload for the LS_ACC indicating that this request has been successfully completed is shown in Table 159.

Table 159 - N_Port Activate Alias_ID Accept Payload	
Item	Size (Bytes)
hex '02 00 00 00'	4

An LS_RJT shall be returned if the Alias_ID cannot be activated as an alias identifier. A Reason Code of “Unable to Perform Command Request” shall be generated, along with one of the Reason Explanations in Table 196.

15.5.35.2 N_Port Deactivate Alias_ID (NDACT)

This request is sent to a N_Port by Alias Server (see Figure 69) to cause it to deactivate the passed Alias_ID as an alias identifier. The payload of the request contains, among other parameters, the alias address identifier to be deactivated by the N_Port. The format of the payload is shown in Table 160.

Table 160 - N_Port Deactivate Alias_ID Payload	
Item	Size (Bytes)
hex '34 00 00 00'	4
Alias_ID	4

Alias Group Identifier (Alias_ID): This is the alias address identifier that the N_Port will no longer recognize.

If the N_Port can successfully deactivate the Alias_ID, it shall transmit an LS_ACC to notify the Alias Server.

The format of the payload for the LS_ACC indicating that this request has been successfully completed is shown in Table 161.

Table 161 - N_Port Deactivate Alias_ID Accept Payload	
Item	Size (Bytes)
hex '02 00 00 00'	4

An LS_RJT shall be returned if the Alias_ID cannot be deactivated as an alias identifier. A Reason Code of “Unable to Perform Command Request” shall be generated, along with one of the Reason Explanations in Table 196.

15.5.36 Class 4 Extended Link Services

The following defines these Extended Link Service commands for Class 4.

The following Extended Link Service requests shall be supported for Class 4:

- Quality of Service Request (QoSR)
- Read Virtual Circuit status (RVCS).

15.5.36.1 Quality of Service Request (QoSR)

The QoSR Extended Link Service Sequence requests setup of a Class 4 circuit and that the requested level of service be granted by the Quality of Service Facilitator (QoS_F) for this Class 4 circuit. The N_Port requesting a Class 4 circuit is the circuit initiator (CTI), the target N_Port is the circuit recipient (CTR). The CTR shall grant the setup of the Class 4 circuit before the Class 4 circuit setup is granted to the CTI (see Figure 41). A QoSR transaction from the QoS_F to the CTR serves to obtain acceptance of the CTR (see 28.2.4.1).

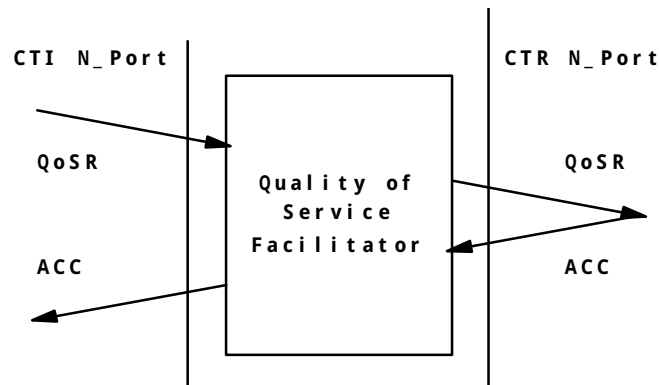


Figure 41 - QoS Sequence flow

Protocol:

- Quality of Service Request (QoSR) request Sequence
- Accept (ACC) reply Sequence

Format: FT_1

Addressing: When transmitted by a CTI, the S_ID field designates the source N_Port requesting the Class 4 circuit. The D_ID field designates the QoSF (hex 'FF FF F9').

When transmitted by the QoSF, the S_ID field designates the QoSF. The D_ID field designates the CTR Address Identifier.

Payload: The format of the Payload is shown in Table 162.

Table 162 - QoS SR Payload	
Item	Size Bytes
hex '40 00 00 00'	4
Reserved	4
CTI VC_ID	1
CTI Address Identifier	3
CTI Maximum Bandwidth (B/s)	4
CTI Minimum Bandwidth (B/s)	4
CTI Maximum Delay (μ s)	4
CTI VC Data Field Size	4
Reserved	12
CTR VC_ID	1
CTR Address Identifier	3
CTR Maximum Bandwidth (B/s)	4
CTR Minimum Bandwidth (B/s)	4
CTR Maximum Delay (μ s)	4
CTR VC Data Field Size	4
Reserved	12
Reserved	1
Live VC credit limit	1
Class 4 end-to-end credit	2

CTI VC_ID: The CIVC_ID for the CTI. The CTI shall choose a currently unassigned VC_ID value in the range of hex '01' through hex 'FE'. This same value shall be in the request from the QoSF to the CTR.

CTI Address Identifier: The Address Identifier in the originating request S_ID field. This same value shall be in the request from the QoSF to the CTR.

The CTI address identifier may be different from the S_ID in the QoSR from the CTI. However, the CTI address identifier shall be a valid, native or alias address identifier for the CTI.

The CTI address identifier shall be the address identifier used as the S_ID or D_ID on all Class 4 frames carried on the Class 4-circuit setup by this QoSR.

CTI Maximum Bandwidth (B/s): The maximum bandwidth in bytes per second (B/s), from the CTI to the CTR, the CTI requires for this VC. This same value shall be in the request from the QoSF to the CTR.

The Fabric may enforce the maximum bandwidth limit. A N_Port may use end-to-end flow control to further limit the available bandwidth.

CTI Minimum Bandwidth (B/s): The minimum bandwidth in bytes per second (B/s) acceptable from the CTI to the CTR. This same value shall be in the request from the QoSF to the CTR.

The Fabric shall guarantee this bandwidth if in any 1 ms interval, the average frame size satisfies the following condition:

Equation 1 - Average frame size related to Responder

$$\overline{FrameSize} \geq \frac{ResponderVCDataFieldSize + 180}{3}$$

CTI Maximum Delay (μs): The maximum acceptable delay in microseconds from the CTI to the CTR. This same value shall be in the request from the QoSF to the CTR. The CTI maximum delay value shall satisfy the following relationship:

Equation 2 - CTI maximum delay

$$E_D_TOV > 2(CIMD + CRMD)$$

where

CIMD CTI maximum delay

CRMD CTR maximum delay

CTI VC Data Field Size: The maximum acceptable frame size in bytes from the CTI to the CTR. This value shall not exceed the minimum value resulting from Fabric Login and N_Port_Login with the CTR. This value shall not be less than 256 and shall be a multiple 4. This same value shall be in the request from the QoSF to the CTR.

CTR VC_ID: When sent from the CTI, this field shall be zero. This same value shall be in the request from the QoSF to the CTR.

CTR Address Identifier: The Address Identifier of the CTR for this Class 4 circuit. This same value shall be in the request from the QoSF to the CTR.

CTR Maximum Bandwidth (B/s): The maximum bandwidth in bytes per second (B/s), from the CTR to the CTI. This same value shall be in the request from the QoSF to the CTR.

The Fabric may enforce the maximum bandwidth limit. A N_Port may use end-to-end flow control to further limit the available bandwidth.

CTR Minimum Bandwidth (B/s): The minimum bandwidth in bytes per second (B/s) acceptable from the CTR to the CTI. This same value shall be in the request from the QoSF to the CTR.

The Fabric shall guarantee this bandwidth if in any 1 ms interval, the average frame size satisfies the following condition:

Equation 3 - Average frame size related to Originator

$$\overline{FrameSize} \geq \frac{OriginatorVCDataFieldSize + 180}{3}$$

CTR Maximum Delay (μs): The maximum acceptable delay in microseconds from the CTR to the CTI. This same value shall be in the request from the QoSF to the CTR. The CTR maximum delay value shall satisfy the following relationship:

Equation 4 - CTR maximum delay

$$E_D_TOV > 2(CIMD + CRMD)$$

where

CIMD CTI maximum delay

CRMD CTR maximum delay

CTR VC Data Field Size: The maximum acceptable frame size in bytes from the CTR to the CTI. This value shall not exceed the minimum value resulting from Fabric Login and N_ Port Login with the CTR. This value shall not be less than 256 and shall be a multiple 4. This same value shall be in the request from the QoSF to the CTR.

Live VC credit limit: The VC credit limit for the Fabric in the QoSR shall set the VC in the live state from the QoSF to the CTR.

Class 4 end-to-end Credit: For the CTR, this shall represent the maximum Class 4 end-to-end Credit limit as well as the Class 4 end-to-end Credit for a dormant Class 4 circuit. This same value shall be in the request from the QoSF to the CTR.

Reply Link Service Sequence (Reply to QoSR):

Service Reject (LS_RJT)

- signifies rejection of the QoSR command

Accept (ACC)

- From the CTR to the QoSF - signifies that the CTR has accepted the QoS parameters.
- From the QoSF - signifies that the Fabric has accepted the QoS parameters and has setup the Class 4 circuit. The QoSF shall await the ACC from the CTR before it transmits the ACC to the CTI.
- Accept Payload

The format of the Accept Payload is shown in Table 163.

Table 163 - QoS SR Accept Payload	
Item	Size Bytes
hex '02 00 00 00'	4
Reserved	4
CTI VC_ID	1
CTI Address Identifier	3
CTI Maximum Bandwidth (B/s)	4
CTI Minimum Bandwidth (B/s)	4
CTI Maximum Delay (μ s)	4
CTI VC Data Field Size	4
Reserved	12
CTR VC_ID	1
CTR Address Identifier	3
CTR Maximum Bandwidth (B/s)	4
CTR Minimum Bandwidth (B/s)	4
CTR Maximum Delay (μ s)	4
CTR VC Data Field Size	4
Reserved	12
Reserved	1
Live VC credit limit	1
Class 4 end-to-end credit	2

CTI VC_ID: The value from the request shall be in the ACC from the CTR to the QoSF and from the QoSF to the CTI.

CTI Address Identifier: The value from the request shall be in the ACC from the CTR to the QoSF and from the QoSF to the CTI.

CTI Maximum Bandwidth (B/s): The value from the request shall be in the ACC from the CTR to the QoSF and from the QoSF to the CTI.

CTI Minimum Bandwidth (B/s): The value from the request shall be in the ACC from the CTR to the QoSF and from the QoSF to the CTI.

CTI Maximum Delay (μ s): The value from the request shall be in the ACC from the CTR to the QoSF and from the QoSF to the CTI.

CTI VC Data Field Size: The value from the request shall be in the ACC from the CTR to the QoSF and from the QoSF to the CTI.

CTR VC_ID: The field is only meaningful in the ACC and shall contain the CRVC_ID for the CTR. The CTR N_Port shall choose an unused value in the range of hex '01' through hex 'FE'. This same value shall be in the ACC from the QoSF to the CTI.

CTR Address Identifier: The value from the ACC from the CTR shall be in the ACC from the QoSF to the CTI. The value in the ACC from the CTR may be different from the value in the QoS SR from the QoSF.

The CTR address identifier may also be different from the S_ID in the ACC from the CTR. However, the CTR address identifier shall be a valid, native or alias address identifier for the CTR.

The CTR address identifier shall be the address identifier used as the S_ID or D_ID on all Class 4 frames carried on the Class 4-circuit setup by this ACC.

CTR Maximum Bandwidth (B/s): The value from the request shall be in the ACC from the CTR to the QoSF and from the QoSF to the CTI.

CTR Minimum Bandwidth (B/s): The value from the request shall be in the ACC from the CTR to the QoSF and from the QoSF to the CTI.

CTR Maximum Delay (μs): The value from the request shall be in the ACC from the CTR to the QoSF and from the QoSF to the CTI.

CTR VC Data Field Size: The value from the request shall be in the ACC from the CTR to the QoSF and from the QoSF to the CTI.

Live VC credit limit: The VC credit limit for the Fabric in the ACC shall set the VC in the live state from the QoSF to the CTI.

Class 4 end-to-end Credit: For the CTI, this shall represent the maximum Class 4 end-to-end Credit limit as well as the Class 4 end-to-end Credit for a dormant Class 4 circuit. The value in the ACC from the CTR shall be in the ACC from the QoSF to the CTI.

15.5.36.2 Read Virtual Circuit Status (RVCS)

Read VC status (RVCS).

Protocol:

- Read VC status Sequence
- Accept (ACC) reply Sequence

Format: FT_1

Addressing: The S_ID shall designate the source N_Port requesting the VC status. The D_ID field shall designate the QoSF (hex 'FF FF F9').

Payload: The format of the payload is shown in Table 164.

Table 164 - RVCS Payload	
Item	Size (Bytes)
hex '41 00 00 00'	4
Reserved	1
N_Port Identifier	3

Reply Link Service Sequence (Reply to RVCS):

Service Reject (LS_RJT)

- Signifies rejection of the RVCS command

Accept (ACC)

- Signifies that the QoSF has completed the request

Accept payload

- The format of the Accept payload is shown in Table 165.

Table 165 - RVCS Accept Payload	
Item	Size (Bytes)
hex '02 00 00 00'	4
Reserved	4
Class 4 VC status block (see Table 166), where N = number of Class 4 circuits for the designated N_Port.	N*8

The accept Sequence contains one entry for each Class 4 circuit setup with the designated N_Port. The format of the Class 4 VC status block entry is shown in Table 166.

Table 166 - Class 4 VC status block entry	
Item	Size (Bytes)
CTI VC_ID	1
CTI Address Identifier	3
CTR VC_ID	1
CTR Address Identifier	3

Link Service Reject (LS_RJT)

Table 196 shows additional LS_RJT reason code explanations applicable to Class4.

15.5.37 Enhanced Login/Logout Link Service**15.5.37.1 Discover N_Port/F_Port Service Parameters (PDISC/FDISC)**

The PDISC/FDISC Extended Link Service requests shall transfer Service Parameters from the initiating N_Port to the N_Port or F_Port associated with the Destination Identifier without affecting the operating environment between the two ports. In the case of PDISC, this means the exchange of Service Parameters without terminating open Sequences or open Exchanges. In the case of FDISC, this means the exchange of Service Parameters without establishing service with the fabric or affecting the internal configuration of the fabric.

The payload in PDISC and the ACC are, except for the LS_Command, identical to the payloads in PLOGI and the ACC. The payload in FDISC and the ACC are, except for the LS_Command, identical to the payloads in FLOGI and the ACC. The interchange of FDISC or PDISC information shall not modify the operating environment or Service Parameters between the two ports. PDISC and FDISC may be transmitted in any class of service. Class 1, 2, 3, or 4 PDISC and FDISC transmission may be attempted in any order.

A response shall not be sent to a PDISC if a LR or FC-AL initialization occurs before the ACC is sent. An ACC for a PDISC shall be ignored if a LR or FC-AL initialization occurred between the PDISC and the ACC.

Table 167 summarizes the responses to FDISC/PDISC when the receiving Port is logged in or not logged in.

Table 167 - Response summary to FDISC/PDISC				
ELS command	Port Status	N_Port	F_Port	
		Class 1, 2, 3, 4	Class 1, 2, or 4	Class 3
FDISC	Logged in	LS_RJT ¹	ACC	ACC
	Not Logged in	LS_RJT ¹	F_RJT ²	Discard
PDISC	Logged in	ACC	F_RJT ²	Discard
	Not Logged in	LS_RJT ¹	F_RJT ²	Discard
NOTE -				
1) LS_RJT may be issued with a reason code = "unable to supply requested data"				
2) P_RJT may be issued with a reason code = "Login required". If an unintelligent drive receives a PDISC from a N_Port with which it has not logged in or receives an FDISC, it may choose to issue a Logout or OLS.				

15.5.37.2 Discover Address (ADISC)

The ADISC Extended Link Service request shall exchange addresses and identifiers of communicating N_Ports. The ADISC and ADISC ACC Payload formats are shown in Table 168 and Table 169, respectively.

Table 168 - ADISC Payload	
Item	Size (Bytes)
hex'52000000'	4
Reserved	1
Hard Address of Originator	3
Port_Name of Originator	8
Node_Name of Originator	8
Reserved	1
N_Port ID of Originator	3

Table 169 - ADISC ACC Payload	
Item	Size (Bytes)
hex'02000000'	4
Reserved	1
Hard Address of Responder	3
Port_Name of Responder	8
Node_Name of Responder	8
Reserved	1
N_Port ID of Responder	3

A Hard Address is the 24-bit NL_Port Identifier, which consists of

- the 8-bit Domain address in the most significant byte,
- the 8-bit Area address in the next most significant byte, and
- the 8-bit AL_PA which an NL_Port attempts to acquire in the LIHA sequence during FC-AL initialization in the least significant byte (see FC-AL).

If a NL_Port does not have a hard address, or if a port does not have FC-AL capability, it shall report zeroes in this field.

Port_Name in ADISC is the 8-byte Port_Name of the Originator and Port_Name in ACC to ADISC is the 8-byte Port_Name of the Responder.

Node_Name in ADISC is the 8-byte Node_Name of the Originator and Node_Name in ACC to ADISC is the 8-byte Node_Name of the Responder.

N_Port ID is the 24-bit S_ID used in the header of the ADISC frame (for the Originator) or ACC frame (for the Responder).

A response shall not be sent to an ADISC if an LR or FC-AL initialization occurs before the ACC is sent. An ACC for a ADISC shall be ignored if an LR or FC-AL initialization occurred between the ADISC and the ACC.

15.5.37.3 Third Party Process Logout (TPRLO)

The Third Party Process Logout (TPRLO) Extended Link Service request shall be used to invalidate the operating environments between the specified image(s) at the recipient N_Port and the specified image(s) in the specified N_Port(s) which have performed Process Login with the recipient N_Port for the specified TYPE.

If the recipient N_Port or NL_Port is not logged in with the requesting port it shall reply with an LS_RJT ELS Sequence with a reason code of 'unable to perform command request' with a reason code of 'N_Port_Login Required.'

TPRLO has the same effect on the recipient N_Port as if all specified Third Party Originators performed PRLO with the recipient N_Port using the specified Process Associators. The TPRLO format is shown in Table 170.

Table 170 - TPRLO Payload	
Item	Size (Bytes)
x'24 ' = TPRLO Command code	1
x'10' = Page Length	1
Payload length	2
Logout Parameter pages	20-max

The TPRLO logout parameter page format is shown in Table 171.

Table 171 - TPRLO logout parameter page format		
Item	Word	Bit
TYPE Code or Common Service Parameters	0	31-24
TYPE Code Extension	0	23-16
Third Party Originator Process _Associator Validity	0	15
Responder Process_Associator Validity	0	14
Third Party Originator N_Port ID Validity	0	13
Global Process Logout	0	12
Reserved	0	11-0
Third Party Originator Process_Associator	1	31-0
Responder Process_Associator	2	31-0
Reserved	3	31-24
Third Party Originator N_Port ID	3	23-0

The TPRLO ACC Payload format is shown Table 172.

Table 172 - TPRLO ACC Payload	
Item	Size - Bytes
hex'02 '	1
x'10' = Page Length	1
Payload length	2
Logout Parameter pages	20-max

Third Party Originator Process Associator validity, if set to 1, indicates that word 2, bits 31-0 are meaningful.

The Global bit, if set to 1, specifies that all image pairs for all N_Ports with which Process Login has been performed be removed from the recipient N_Port for the specified TYPE. If the

Global bit is set to 1, only one logout parameter page shall be transmitted, and only the TYPE code and TYPE code extension fields shall have meaning (aside from the Global bit itself). If the Global bit is set to 1, it frees all resources associated with the establishment of all image pairs of the specified TYPE at the recipient N_Port.

Third Party Originator N_Port ID validity, if set to 1, indicates that word 1, bits 23-0 are meaningful.

The Third Party Originator Process_Associator specifies the Process Associator associated with the image to be removed.

The Third Party Originator N_Port ID specifies the N_Port associated with the image to be removed.

All other fields have the same meaning as described in PRLO.

The minimum TPRLO Payload length is 20 bytes.

15.5.38 Extended Link Services for Bandwidth Allocation

For the purposes of the dynamic allocation of bandwidth, the BALLOC and DALLOC extended link services are defined. These two Extended link services are of particular importance to FC-SL devices. Each service shall be transmitted within the context of a single exchange and a single sequence.

15.5.38.1 Bandwidth Allocation (BALLOC)

The Bandwidth Allocation Link service request shall be used to request access to isochronous bandwidth from a bandwidth management function such as a Slotted Loop Master.

Protocol:

- Bandwidth Allocation request Sequence
- Accept reply Sequence

Format: FT_1

Addressing: the S_ID field designates which loop device is making the request. The D_ID field designates the bandwidth management device.

Payload: The format of the Payload is shown in Table 173. Each BALLOC request shall indicate the isochronous frame rate, the number of frame positions, the access type (read or write), the starting frame position number (for read requests only), and the error policy. For the Read/Write field, 0x00 shall indicate Read and 0x01 shall indicate Write. For the Error Policy field, 0x00 shall indicate Discard, and 0x01 shall indicate Process. Note that the error policy of a given frame rate is established by the first request for bandwidth at a given rate.

Table 173 - BALLOC Payload Format	
Item	Size – Bytes
hex '58 00 00 00'	4
# Frame Positions	4
Frame Rate	2
Read/Write	1

Error Policy	1
--------------	---

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the BALLOC command.
- The bandwidth manager must have the entire amount of requested bandwidth available to respond to the BALLOC with an ACC. The bandwidth manager shall not allocate a part of the requested bandwidth. If all of the requested bandwidth can not be allocated, it will respond with a LS_RJT with a Bandwidth Not Available reason code. It is the responsibility of the requestor to reissue the request for bandwidth at a later time if desired.

Accept (ACC)

- signifies that the bandwidth manager accepted the command and has allocated the requested bandwidth
- Accept Payload

The format of the Accept Payload for the BALLOC Link Service is shown in Table 174. In addition to acknowledging the number of frame positions, the frame rate, the read/write policy, and the error policy, the bandwidth manager shall respond with the frame position at which the requestor's bandwidth access begins within the specified frame (Starting Frame Position). All frame positions are contiguous from the starting Frame Position. Use of the Read/Write and Error Policy fields shall be identical to that defined in the BALLOC payload. The bandwidth manager must have the entire amount of requested bandwidth available to respond to the BALLOC with an ACC.

Table 174 - BALLOC Accept Payload format	
Item	Size – Bytes
hex '02 00 00 00'	4
# Frame Positions	4
Frame Rate	2
Read/Write	1
Error Policy	1
Starting Frame Position	4

15.5.38.2 Bandwidth Deallocation (DALLOC)

The Bandwidth Deallocation Link service request shall be used to relinquish access to isochronous bandwidth to which a device has previously been granted by a bandwidth manager. The DALLOC can be used by any device to relinquish access to bandwidth voluntarily, or by the bandwidth manager to force a device to relinquish access to bandwidth. If the request applies to the ability of a device to write to specific frame positions, the bandwidth manager shall return the bandwidth specified in the request to its bandwidth pool upon acceptance.

Protocol:

- Bandwidth Deallocation request Sequence

- Accept reply Sequence

Format: FT_1

Addressing: the S_ID field designates the device making the request. The D_ID field designates the destination device. The D_ID field shall designate the bandwidth management device if a non-manager device is initiating the bandwidth reallocation. The D_ID field shall designate a non-bandwidth management device if the bandwidth manager is forcing the reallocation of bandwidth.

Payload: The format of the Payload is shown in Table 175. Each DALLOC request shall indicate the isochronous frame rate, the number of frame positions, the access type (read or write), the starting frame position number (for read requests only), and the error policy. For the Read/Write field, 0x00 shall indicate Read and 0x01 shall indicate Write. For the Error Policy field, 0x00 shall indicate Discard, and 0x01 shall indicate Process. Note that the error policy of a given frame rate is established by the first request for bandwidth at a given rate. All frame positions are contiguous from the Starting Frame Position.

The bandwidth manager is permitted to initiate the DALLOC frame in an unsolicited manner to free previously allocated bandwidth at its discretion. As long as the extended link service is valid, the destination of the DALLOC is required to respond with an Accept extended link service and immediately stop the origination of data into the boxcar frame positions identified in the DALLOC. The Master's use of the DALLOC extended link service beyond this is application specific.

Table 175 - DALLOC Payload	
Item	Size – Bytes
hex '59 00 00 00'	4
# Frame Positions	4
Frame Rate	2
Read/Write	1
Error Policy	1
Starting Frame Position	4

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the DALLOC command (see 15.6.2).

Accept (ACC)

- signifies that the bandwidth manager accepted the command and has allocated the requested bandwidth
- Accept Payload

The format of the Accept Payload for the DALLOC Link Service is shown in table 129. Use of the Read/Write and Error Policy fields shall be identical to that defined in the DALLOC payload.

Table 176 - DALLOC Accept Payload	
Item	Size – Bytes
hex '02 00 00 00'	4
# Frame Positions	4
Frame Rate	2
Read/Write	1
Error Policy	1
Starting Frame Position	4

15.5.39 Extended Link Services for Clock Synchronization

15.5.39.1 Clock Synchronization Request (CSR)

The CSR Link Service request shall be used to request the Clock Synchronization Server to either send or to quit sending periodic Clock Synchronization Update (CSU) ELS frames. See 32.2 for the usage of this frame.

Protocol:

- Clock Synchronization request Sequence
- Accept reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting clock synchronization updates. The D_ID field designates the Clock Synchronization Server (well-known address hex 'FF FF F6').

Payload: The format of the Payload is shown in Table 177. See 32.2.4.3 for permissible values and meanings of the Clock Sync Mode item.

Table 177 - CSR Payload	
Item	Size -Bytes
hex '68 00 00 00'	4
Clock Sync Mode	1
Reserved	3

Reply Link Service Sequence:

Service Reject (LS_RJT)

- signifies rejection of the CSR command (see 15.6.2)

Accept (ACC)

- signifies that the Clock Synchronization Server agrees to perform the action requested in the CSR payload.

- Accept Payload

The format of the Accept Payload is shown in Table 178. The Clock Sync Mode item in the CSR Accept payload shall contain the value that was received in the Clock Sync Mode item of the CSR payload.

Table 178 - CSR Accept Payload	
Item	Size - Bytes
hex '02 00 00 00'	4
Clock Sync Mode	1
Reserved	3

15.5.39.2 Clock Synchronization Update (CSU)

The CSU Link Service request shall be used by the Clock Synchronization Server to send its current clock value to its clients. See 32.2.4.4 for the usage of this frame.

Protocol:

- Clock Synchronization Update request Sequence
- No Reply Sequence

Format: FT_1

Information Category: The Information Category value (Header word 0, bits 24-27) shall indicate Solicited Data (0001).

Addressing: The S_ID field designates the Clock Synchronization Server (well-known address hex 'FF FF F6'). The D_ID field designates the N_Port(s) that is/are to receive the clock information.

Payload: The format of the Payload is shown in Table 179. See 32.2.4.4 for a description of the Clock Count item.

Table 179 - CSU Payload	
Item	Size - Bytes
hex '69 00 00 00'	4
Clock Count	8

Reply Link Service Sequence: None.

15.5.40 Login Control List Management

This clause defines a new Extended Link Service, which supports the management of a Login Control List (LCL) for Login Control. Login Control is the authentication of PLOGIs by the device based on information stored in the device about the originator of the PLOGI. The device can use a Login Control List (LCL) to determine which originators can perform a PLOGI.

The ELS frame shall be transmitted using the FT-1 frame format via Class 2, or Class 3 service. The following defines the header fields of the ELS frame:

- R_CTL: This field shall be set to hex '22' for request frames, and to hex '23' for Accept reply frames.
- CS_CTL: This field shall be set to hex '00'.
- D_ID and S_ID: Set as indicated for the specific ELS.
- TYPE: This field shall be set to hex '01', indicating Extended Link Service.

All other fields shall be set as appropriate according to the rules defined in FC-FS. The first word in the payload specifies the Command Code. This optional Extended Link Service is applicable for operation between an Originator NL_Port and a Responder NL_Port. NL_Ports that comply with the Login Control function shall implement the Login Control List Management ELS. It enables an originator to manage an LCL in a device that is utilizing Login Control. It may be sent prior to PLOGI in order to perform the clear LCL command.

15.5.40.1 Login Control List Management (LCLM)

The Login Control List Management (LCLM) function shall provide a method for managing the LCL on a device. The normal response shall be an Accept (ACC) ELS Sequence. This accept shall indicate that the LCL modification has been received by the destination device's local storage, but not necessarily that they have been copied to permanent memory. If the recipient NL_Port is unable to queue commands or is temporarily unable to process the ELS, it shall respond with a LS_RJT ELS with a reason code of "Logical Busy". If the recipient NL_Port does not support the LCLM ELS, it shall reply with an LS_RJT ELS Sequence with a reason code of "command not supported". If the recipient NL_Port does not allow this NL_Port to perform Login Control List Management, it shall reply with an LS_RJT ELS Sequence with a reason code of "unable to perform command request" and reason-code explanation of "invalid originator S_ID".

The LCLM ELS is sent by a NL_Port to a destination on which it has the capability of managing the LCL. More than one NL_Port may be allowed to manage the LCL on a given device. This is determined by the capabilities that the first NL_Port to send a LCL to the device sets for other originators.

When the LCLM ELS recipient supports LCL management by the originator of the LCLM ELS, it shall perform the requested LCLM command and return an Accept reply Sequence.

The mechanisms whereby the device implements login control based on the LCLM ELS is beyond the scope of this standard, and will be described in the appropriate device standard or technical report.

Protocol:

- Login Control List Management (LCLM) request Sequence
- Accept (ACC) reply Sequence

Format: FT-1

Addressing: The S_ID field designates the requesting source port. The D_ID field designates the port receiving the LCLM request.

Payload: The format of the LCLM Payload is shown in Table 180.

Table 180 – LCLM payload			
Word	Byte	Item	Size Bytes
0	0	hex '25'	1
0	1	Page Size hex '24'	1
0	2-3	Payload Length hex '1C'	2
1	0-1	Reserved	2
1	2	LCLM command	1
1	3	Global Capabilities	1
2-3		N_Port_Name	8
4-5		Node_Name	8
6	0-1	Reserved	2
6	2-3	Index	2

Payload length: Bytes 2-3 of word 0 contain the payload length of this ELS. This shall be set to hex '1C'.

LCLM Command: Byte 2 of word 1 contains an 8-bit value that indicates one of the following LCLM commands:

- hex '00' = Read LCL entry
- hex '01' = Add LCL entry
- hex '02' = Delete LCL entry
- hex '03' = Read LCL list
- hex '04' = Clear LCL

Global Capabilities: Byte 3 of Word 1 contains global capabilities. The format of the Global capabilities byte is described in Table 181.

Table 181 - Global capabilities		
Item	Byte	Bits
Reserved	3	7-1
LCL Management	3	0

- **Byte 0, Bit 0 – LCL Management**

Identifies that the N_Port_Name/Node_Name in this LCLM is allowed to manage the LCL. If any entry in the LCL has this bit set, then only those entries that have this bit set are allowed to perform LCLM commands, except the Clear LCL command which the device at the Well Known Address of the Management Server (hex 'FFFFFFA') may also perform to this device. If no entry in the LCL has this bit set then any device can perform LCLM commands.

- **Index:** Byte 2-3 of Word 6 is a 16 bit signed integer index into the LCL. The index is based on the number of the Port/Node_Name pair, not a byte index. This is used to indi-

cate the starting index of the group of Port/Node_Names that a Read LCL list request is requesting. The first entry is at index hex '0'. For all other LCLM commands this field is reserved and shall be set to hex'0'.

Reply link service sequence:

Link Service Reject (LS_RJT)

- signifies rejection of the LCLM request

Accept (ACC)

- signifies successful completion of the LCLM request
- Accept Payload

The format of the Accept Payload is shown in Table 182.

Table 182 – LCLM accept payload			
Word	Byte	Item	Size Bytes
0	0	hex '02'	1
0	1	Page Size	1
0	2-3	Payload Length	2
1	0-1	Reserved	2
1	2	LCLM command	1
1	3	Global Capabilities	1
2-3		N_Port_Name	8
4-5		Node_Name	8
6	0-1	List length	2
6	2-3	Index	2
7-max		Port/Node_Name List	n-max

- **Payload length:** Bytes 2 - 3 of word 0 contain a 16-bit unsigned binary integer that specifies the length of the LCLM accept payload. The rightmost four bits shall be bin '1100'. The value specified shall be greater than or equal to 28, and less than or equal to 252.
- **LCLM Command:** Byte 2 of word 1 contains an 8-bit value that indicates one of the following LCLM commands, which this response corresponds to: (reference command)
 - hex '00' = Read LCL entry
 - hex '01' = Add LCL entry
 - hex '02' = Delete LCL entry
 - hex '03' = Read LCL list
 - hex '04' = Clear LCL
- **Global Capabilities:** Byte 3 of Word 1 contains the Global capabilities. The format of the Global capabilities byte is described in Table 183. This is reserved and set to zero for the “read LCL list” and “clear LCL list” commands.

Table 183 – Global capabilities		
Item	Byte	Bits
Reserved	0	7-1
LCL Management	0	0

- **Byte 0, Bit 0 – LCL Management**

Identifies that the N_Port_Name/Node_Name in this LCLM is allowed to manage the LCL. If any entry in the LCL has this bit set, then only those entries that have this bit set are allowed to perform LCLM commands to this device. If no entry in the LCL has this bit set then any device can perform LCLM commands.

- **List Length:** Byte 0-1 of Word 6 is a 16 bit signed integer reporting the number of Port/Node_Names in the LCL list after performing the requested LCL command. This field is required for all responses. It is reserved and set to hex '0' for all commands.
- **Index:** Byte 2-3 of Word 6 is a 16 bit signed integer index into the LCL list. The index is based on the number of the Port/Node_Name pair, not a byte index. This is used to indicate the starting index of the group of Port/Node_Names that a Read LCL list accept is reporting. The first entry is at index hex '0'. The ACC for any LCL command other than a Read LCL shall return hex '0' in this field. All LCLM commands other than a Read LCL list shall set this value to hex '0'.
- **Port/Node_Name List:** Words 7:max of the LCLM Payload contain a Port/Node_Name list. For code hex '03' the Port/Node_Name List contains a list of the N_Port_Name/Node_Name pairs that are currently registered in the device's LCL. For all other codes, word 5 of the payload shall be the last word.

The format of the Port/Node_Name list is described in Table 184.

Table 184 – Port/Node_Name List		
Item	Word	Bits
N_Port_Name	0 – 1	31-0
Node_Name	2 – 3	31-0

- **Word 0 - 1 , Bits 31- 0 – N_Port_Name**

Contains the N_Port_Name of an entry of the LCL. This is reserved and set to zero for the “read LCL list” and “clear LCL list” commands.

- **Word 2 - 3 , Bits 31- 0 – Node_Name**

Contains the Node_Name of an entry of the LCL. This is reserved and set to zero for the “read LCL list” and “clear LCL list” commands.

15.5.40.2 LCLM Operation

LCLM operates by sending a command to a device. The command can be an addition of a Port/Node_Name to the LCL stored on the device, the deletion of a Port/Node_Name from the LCL stored on the device, a Read of the LCL stored on the device for a given Port/Node_Name, a Read of the list of Port/Node_Names in the LCL on the device, or a clear of the entire LCL.

The originator of any LCL request shall be able to accept LS_RJT, reason Code = Logical Busy, with Reason Code Explanation hex '19' "Command (request) already in progress" or hex '2A' "Unable to supply requested data".

A LCLM addition shall add a Port/Node_Name and the associated capabilities to the LCL on the device. If the originator of the LCLM Addition is not in the LCL and there are Port/Node_Names in the LCL with the LCL Management bit set a LS_RJT shall be returned with a reason code of "unable to perform command request" and reason code explanation of "invalid originator S_ID". If the originator of the LCLM Addition is in the LCL and there are Port/Node_Names in the LCL with the LCL Management bit set and the LCL Management bit for this Port/Node_Name is not set, a LS_RJT shall be returned with a reason code of "unable to perform command request" and reason code explanation of "invalid originator S_ID".

A LCLM deletion shall delete any LCL record for the Port/Node_Name in the ELS. If the originator of the LCLM Deletion is not in the LCL and there are Port/Node_Names in the LCL with the LCL Management bit set a LS_RJT shall be returned with a reason code of "unable to perform command request" and reason code explanation of "invalid originator S_ID". If the originator of the LCLM Deletion is in the LCL and there are Port/Node_Names in the LCL with the LCL Management bit set and the LCL Management bit for this Port/Node_Name is not set, a LS_RJT shall be returned with a reason code of "unable to perform command request" and reason code explanation of "invalid originator S_ID". If the Port/Node_Name in the LCLM deletion is not currently in the LCL of the device, the device shall return the ACC as if the Port/Node_Name had been in the LCL, and had been deleted. If the Port/Node_Name was in the LCL, and the Port was currently logged in, a LOGO shall be sent to the Port.

A LCLM Read shall return the global capabilities of the given Port/Node_Name for the device that was queried. If the originator of the LCLM Read is not in the LCL and there are Port/Node_Names in the LCL with the LCL Management bit set, a LS_RJT shall be returned with a reason code of "unable to perform command request" and reason code explanation of "invalid originator S_ID". If the originator of the LCLM Read is in the LCL and there are Port/Node_Names in the LCL with the LCL Management bit set and the LCL Management bit for this Port/Node_Name is not set, a LS_RJT shall be returned with a reason code of "unable to perform command request" and reason code explanation of "invalid originator S_ID". The format of the ACC shall be the same as the format for the LCLM ELS that added the Port/Node_Name to the LCL on the device. If the Port/Node_Name is not in the LCL of the device, the device shall return an LS_RJT ELS Sequence with a reason code of "unable to perform command request" and reason-code explanation of "Invalid Port/Node_Name". If the device had not finished copying previous LCL modifications/additions to permanent storage – or (in the case of rotating media storage) if the media had not spun up, a LS_RJT ELS Sequence with reason code of "Logical Busy" and Reason Code Explanation hex '2A' "Unable to supply requested data" shall be returned.

A LCLM Read List shall return a list of the N_Port_Name/Node_Name pairs currently in the LCL of the device starting at Index. The number of entries that are returned in the accept to the LCL read list is determined by the maximum frame length of the LCL accept (256 bytes). The end of the list can be determined using the value returned in Max Index. If the originator of the LCLM Read List is not in the LCL and there are Port/Node_Names in the LCL with the LCL Management bit set and the LCL Management bit for this Port/Node_Name is not set, a LS_RJT shall be returned with a reason code of "unable to perform command request" and reason code explanation of "invalid originator S_ID". If there are no Port/Node_Names in the LCL, the device shall return an ACC with a payload length of hex '1C' and no Port/Node_Names listed.

A LCLM clear shall clear all entries in the LCL. If the originator of the LCLM Read List is not in the LCL and there are Port/Node_Names in the LCL with the LCL Management bit set and the LCL Management bit for this Port/Node_Name is not set, a LS_RJT shall be returned with a reason code of "unable to perform command request" and reason code explanation of "invalid

originator S_ID". If there are no Port/Node_Names in the LCL, the device shall return an ACC with a payload length of hex '1C' and no Port/Node_Names listed.

This allows recovery from a missing LCL manager, or a move of a device with LCL from one configuration to another. This should only be used by a device, which has knowledge that it should be allowed to rebuild the LCL.

15.5.41 Read Port Status Block (RPS)

The RPS Extended Link Service request Sequence requests a N_Port or F_Port to return port status associated with the port specified in the Payload of this frame. The ability to request status information for ports other than the Port addressed in the frame header is useful in the absence of a management server.

Protocol:

- Read Port Status request Sequence
- Accept (ACC) reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting the Link Error Status Block. The D_ID field designates the destination N_Port or the Domain Controller (FFFF<Domain ID>) to which the request is being made.

Payload: The format of the Payload is shown in Table 185.

Table 185 – RPS Payload	
Item	Size - Bytes
hex '56'	1
Reserved	2
Flag	1
Port Selection	8

- **Flag:** Only one bit may be set in the Flag field at a time. If no bits are set, the port selection field is ignored and the RPS request is for the port identified in the frame header D_ID.
 Bits 7-2 Reserved.
 Bit 1 When a one, the port selection field contains a Port_Name of a WWN format.
 Bit 0 When a one, the port selection field contains a physical port number in the last 4 bytes of the port selection field.
- **Port selection:** contains either a Port_Name or physical port number of the port for which the port status information is being requested. The type of port selection is indicated in the Flag field. A physical port number uniquely identifies a N_Port or F_Port contained in a unit. Physical port numbers may be obtained using the RPL ELS, RTIN ELS, RNID ELS or by implicit means.

RPS Accept Payload:

Table 186 – RPS Accept Payload	
Item	Size - Bytes
hex '02'	1
Reserved	2
Flag	1
Reserved	2
Port Status	2
LESB	24
L_Port Extension	0-24

– **Flag**

Bits 7-1 Reserved

Bit 0 L_Port Extension Valid - When a one, the L_Port Extension block is included in the RPS ACC payload. If a zero, the RPS ACC payload ends with the LESB data.

– **Port Status**

Bit 15-6 Reserved

Bit 5 Point to Point- When a one, the port is connected point to point with another port.

Bit 4 Loop mode- When a one, the port is in FC-AL mode.

Bit 3 Fabric present- When a one, the port has detected a F_Port in a point to point connection or a FL_Port in loop mode.

Bit 2 Loss of Signal- When a one, the port is not receiving an input signal. A value of zero indicates the port is receiving an input signal or the status is not supported.

Bit 1 Loss of Synchronization - When a one, the port is in the Loss-of-Synchronization state.

Bit 0 Link Reset Protocol- The port is currently in the Link Reset Protocol.

The format of the L_Port Extension field is defined in Table 187.

Table 187 - L_Port Extension	
Item	Size- Bytes
Reserved	2
L_Port Status	2
Reserved	3
LIP AL_PS	1
LIP F7 Initiated Count	4
LIP F7 Received Count	4

LIP F8 Initiated Count	4
LIP F8 Received Count	4
LIP Reset Initiated Count	4
LIP Reset Received Count	4

– **L_Port Status**

If all bits in L_Port Status are zero, the port is not currently functional or does not support this status.

- Bit 15 OLD-PORT - When a one, the Loop Port State Machine is in the OLD-PORT state.
- Bit 14 MONITORING - When a one, the Loop Port State Machine is in the MONITORING state.
- Bit 13 ABITRATING - When a one, the Loop Port State Machine is in the ABITRATING state.
- Bit 12 OPEN - When a one, the Loop Port State Machine is in the OPEN state.
- Bit 11 OPENED - When a one, the Loop Port State Machine is in the OPENED state.
- Bit 10 XMITTED_CLOSE - When a one, the Loop Port State Machine is in the XMITTED_CLOSE state.
- Bit 9 RECEIVED_CLOSE - When a one, the Loop Port State Machine is in the RECEIVED_CLOSE state.
- Bit 8 TRANSFER - When a one, the Loop Port State Machine is in the TRANSFER state.
- Bits 7-3 Reserved
- Bit 2 LIP(F7,xx) Loop Initialization in process - When a one, the port is currently performing the Loop Initialization Process defined by FC-AL initiated by a LIP(F7,xx).
- Bit 1 LIP(F8,xx) Loop Initialization in process - When a one, the port is currently performing the Loop Initialization Process defined by FC-AL initiated by a LIP(F8,xx).
- Bit 0 LIP(AL_PD,xx) Loop Initialization in process - When a one, the port is currently performing the Loop Initialization Process defined by FC-AL initiated by a LIP(AL_PD,xx).

- **LIP AL_PS** - field contains the fourth character of the received LIP. A value of 'FF' indicates the port does not support this field.
- **LIP F7 Initiated Count** - The number of loop initialization processes originated by the port with LIP (F7,xx) (Initialize LIP).
- **LIP F7 Received Count** - The number of loop initialization processes initiated on the port by receiving (F7,xx) (Initialize LIP).
- **LIP F8 Initiated Count** - The number of loop initialization processes originated by the port with (F8,xx) (Failure LIP).
- **LIP F8 Received Count** - The number of loop initialization processes initiated on the port by receiving (F8,xx) (Failure LIP).

- **LIP Reset Initiated Count** - The number of loop initialization processes originated by the port with (AL_PD,xx) (Reset LIP).
- **LIP Reset Received Count** - The number of loop initialization processes initiated on the port by receiving (AL_PD,xx) (Reset LIP).

15.5.42 Read Port List (RPL)

The RPL Extended Link Service request provides internal node port identification information. This information allows the requesting port to direct a request for status of other ports using selection protocol outside the normal Fibre Channel address structure.

The internal port identification information is in the form of a list of available port LESB entries addressed by an entry number index.

Protocol:

- Report Port List (RPL) request Sequence
- Accept (ACC) reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting the port numbers information. The D_ID field designates the destination N_Port or the Domain Controller (FFFF<Domain ID>) to which the request is being made.

Payload: The format of the RPL Payload is shown in Table 188.

Table 188 - RPL Payload	
Item	Size- Bytes
hex '57'	1
Reserved	5
Maximum Size	2
Reserved	1
Index	3

Maximum Size: This field manages the size of the information returned in an RPL Accept. The sender of an RPL may specify the maximum size of the RPL ACC it is able to receive. The recipient of the RPL shall return information equal to or less than the maximum size value in the RPL. The values for Maximum Size are interpreted as follows:

Value (hex)	Meaning
'00 00'	No Maximum Size indicated; the RPL ACC may be any size.
'FFFF'	Reserved.
'xxxx' that	Any other value: The encoded value indicates the maximum number of words that shall be sent in the RPL ACC, not inclusive of the command word 0.

Index: The entry number of the first port number block being requested in this RPL. The index for the first entry in the list is 0.

Read Port List Accept (ACC)

- The RPL ACC returns a list of the port number information.
- Payload: The format of the RPL ACC Payload is shown Table 189.

Table 189 - RPL ACC Payload	
Item	Size- Bytes
hex '02'	1
Reserved	1
Payload length	2
Reserved	1
List Length	3
Reserved	1
Index	3
Port Number Blocks	16*n

- **Payload Length** - an unsigned binary integer that specifies the length of the RPL ACC payload in bytes.
- **List Length** - The total number of Port Number Blocks accessible through the recipient port.
- **Index** - The entry number for the first Port Number Block in the ACC payload.
- **Port Number Blocks** – See Table 190 for the contents of the Port Number Block.
- n - the number of Port Number Blocks in this ACC.

Table 190 - Port Number Block	
Item	Size- Bytes
Physical port number	4
Reserved	1
Port_Identifier	3
Port_Name	8

A Port_Identifier of all zeros indicates the port currently does not have a valid address.

The physical port number is unique identification of the port within the node.

15.5.43 Report Node FC-4 Types (RNFT)

The RNFT Link Service request Sequence provides for the exchange of supported FC-4 protocol lists. It may be used any time after N_Port_Login to verify that the remote N_Port supports a given FC-4 protocol.

The RNFT requests a list of the FC-4 protocols supported by the responder, and the RNFT ACC provides the requested list to the originator.

A specific FC-4 may require that its N_Ports support RNFT, and thus can conclude that a remote N_Port that returns LS_RJT with reason code explanation hex '2C' (Request not supported) does not support that FC-4.

If the recipient N_Port or NL_Port is not logged in with the requesting port it shall reply with an LS_RJT ELS Sequence with a reason code of 'unable to perform command request' with a reason code of 'N_Port_Login Required.'

Protocol:

- Report Node FC-4 Types (RNFT) request Sequence
- Accept (ACC) reply Sequence

Format: FT_1

Addressing: The S_ID field designates the source N_Port requesting the FC-4 Types information. The D_ID field designates the destination N_Port to which the request is being made.

Payload: The format of the request Payload is shown in Table 191.

Table 191 - RNFT Payload	
Item	Size-Bytes
hex '63'	1
Reserved	1
Maximum Size	2
Reserved	3
Index	1

Maximum Size: Bytes 2-3 of Word 0 contain a 16-bit unsigned binary integer that specifies the maximum length of the RNFT ACC that the originator can accept. The value zero implies the RNFT ACC may be any size.

Index: Byte 3 of Word 1 contains an 8-bit unsigned integer that specifies the index of the first FC-4 Entry to be returned in the RNFT reply.

Each FC-4 protocol supported by the responder has an index in the range from zero to (List Length – 1) that can be used to specify a subset of the entries when the entire list will not fit into one reply. The index of the entry for a particular FC-4 Type may change at any time.

Reply Link Service Sequence:

Service Reject (LS_RJT)

- Signifies rejection of the RNFT command (see 15.6.2)

Accept

- Signifies that the destination N_Port has transmitted the requested data.
- Accept Payload

The format of the Accept Payload is shown in Table 192.

Table 192 - RNFT ACC Payload	
Item	Size-Bytes
hex '02'	1
Reserved	1
Payload Length (M)	2
Reserved	1
List Length	1
Reserved	1
Index	1
FC-4 Entries	M-8

Payload Length: Bytes 2-3 of Word 0 contain a 16-bit unsigned binary integer that specifies the length M of the RNFT ACC payload in bytes.

$M = 8 + N \times 4$ where N is the number of FC-4 Entries contained in the payload.

List Length: Byte 1 of Word 1 contains an 8-bit unsigned binary integer that specifies the total number of FC-4 protocols supported by the responder.

If List Length exceeds Index+N then the originator may request additional records with another RNFT in which Index is increased by N.

Index: Byte 3 of Word 1 contains an 8-bit unsigned integer that specifies the index of the first FC-4 Entry returned in the RNFT reply.

FC-4 Entries: The remainder of the payload contains a list of N FC-4 Entry records.

FC-4 Entry: The FC-4 Entry record contains a FC-4 Entry and is shown in Table 193.

Table 193 - RNFT FC-4 Entry	
Item	Size-Bytes
FC-4 Type	1
FC-4 Qualifier	3

FC-4 Type: The FC-4 TYPE code of a FC-4 protocol that is supported by the sending N_Port. The values are defined in Table 20.

FC-4 Qualifier: The FC-4 Qualifier may be used to distinguish between two protocols that use the same FC-4 TYPE code.

For Vendor Unique FC-4 TYPE codes (hex '69' through hex 'FF'), the FC-4 Qualifier shall be selected from one of the 24-bit company_id values assigned by the IEEE Registration Authority to the organization that defines the Vendor Unique FC-4 protocol, and that company_id shall be used to qualify that FC-4 TYPE in all implementations. It is up to the organization that defines the Vendor Unique FC-4 protocol to assure that the protocol has a unique qualified FC-4 Type. For information about the proper use of the company_id, see the tutorial in Annex N.

15.5.44 Extended Link Service summary

Table 194 summarizes the Extended Link Service request and reply Sequences. The Payload content and size in bytes are specified. The Payload for the LS_RJT is 8 bytes in size.

Table 194 - Extended Link Services Summary			
Request Sequence		Reply Sequence	
Request	Payload	Reply	Accept Payload
Abort Exchange (ABTX)	LS_Command code, Recovery_Qualifier Status, Originator S_ID, OX_ID, RX_ID, Association_Header (optional)	Accept (ACC) or LS_RJT	LS_Command code
Advise Credit (ADVC)	LS_Command code, Common Service Parameters, Port_Name, Node_Name, Service Parameters (Classes 1, 2, 3), Vendor Version Level	Accept (ACC) or LS_RJT	LS_Command code, Common Service Parameters, Port_Name, Node_Name, Service Parameters (Class 1, 2, 3), Vendor Version Level
Echo (ECHO)	LS_Command code, up to Max frame	Accept (ACC) or LS_RJT	LS_Command code, Echo Data
Estimate Credit (ESTC)	LS_Command code, Data	None	
Establish Streaming (ESTS)	LS_Command code	Accept (ACC) or LS_RJT	LS_Command code, Common Service Parameters, Port_Name, Node_Name, Service Parameters (Class 1, 2, 3), Vendor Version
Login (FLOGI/PLOGI)	LS_Command code, Common Service Parameters, Port_Name, Node_Name, Service Parameters (Class 1, 2, 3), Vendor Version	Accept (ACC) or LS_RJT	LS_Command code, Common Service Parameters, Port_Name, Node_Name, Service Parameters, (Class 1, 2, 3), Vendor Version
Logout (LOGO)	LS_Command code, Reserved, N_Port Identifier, Port_Name	Accept (ACC) or LS_RJT	LS_Command code
Read Connection Status (RCS)	LS_Command code, Reserved, N_Port Identifier	Accept (ACC) or LS_RJT	LS_Command code, Connection Status, N_Port Identifier
Read Exchange Status Block (RES)	LS_Command code, Reserved, Originator S_ID, OX_ID, RX_ID, Association_Header (optional)	Accept (ACC) or LS_RJT	LS_Command code, Exchange Status Block, Association_Header (optional)
Read Link Error Status Block (RLS)	LS_Command code, Reserved, Port Identifier	Accept (ACC) or LS_RJT	LS_Command code, Link Error Status Block
Read Sequence Status Block (RSS)	LS_Command code, SEQ_ID, Originator S_ID, OX_ID, RX_ID	Accept (ACC) or LS_RJT	LS_Command code, Sequence Status Block
Read Timeout Value (RTV)	LS_Command code	Accept (ACC) or LS_RJT	LS_Command code, R_A_TOV, E_D_TOV, Timeout Qualifier

Table 194 - Extended Link Services Summary

Request Sequence		Reply Sequence	
Request	Payload	Reply	Accept Payload
Reinstate Recovery Qualifier (RRQ)	LS_Command code, reserved, S_ID, OX_ID, RX_ID, Association_Header (optional)	Accept (ACC) or LS_RJT	LS_Command code
Request Sequence Initiative (RSI)	LS_Command code, reserved, Originator S_ID, OX_ID, RX_ID, Association_Header (optional,)	Accept (ACC) or LS_RJT	LS_Command code
Test (TEST)	LS_Command code, up to Max frame	None	
Process Login (PRLI)	LS_Command code, Page Length, Payload Length, Service Parameter pages	Accept (ACC) or LS_RJT	LS_Command code, Page Length, Payload Length, Service Parameter pages
Process Logout (PRLO)	LS_Command code, Page Length, Payload Length, Logout Parameter pages	Accept (ACC) or LS_RJT	LS_Command code, Page Length, Payload Length, Logout Parameter Response pages
State Change Notification (SCN)	LS_Command code, Page Length, Payload Length, Affected N_Port ID pages	None	
Test Process Login State (TPLS)	LS_Command code, Page Length, Payload Length, Image Pair ID Pages	Accept (ACC) or LS_RJT	LS_Command code, Page Length, Payload Length, TPLS Response pages
Third Party Process Logout (TPRLO)	LS_Command code, Page Length, Payload Length, Logout Parameter Pages	Accept (ACC) or LS_RJT	LS_Command code, Page Length, Payload Length, Logout Parameter Pages
Get Alias_ID (GAID)	LS_Command code, Alias_Token, Alias_SP, NP_List_Length, NP_List (1), NP_List (2 to n-1), NP_List (n)	Accept (ACC) or LS_RJT	LS_Command code, Alias_ID
Fabric Activate Alias_ID (FACT)	LS_Command code, Alias_ID, NP_List_Length, NP_List (1), NP_List (2 to n-1), NP_List (n)	Accept (ACC) or LS_RJT	LS_Command code
Fabric Deactivate Alias_ID (FDACT)	LS_Command code, Alias_ID, NP_List_Length, NP_List (1), NP_List (2 to n-1), NP_List (n)	Accept (ACC) or LS_RJT	LS_Command code
N_Port Activate Alias_ID (NACT)	LS_Command code, Alias_Token, Alias_ID, Alias_SP	Accept (ACC) or LS_RJT	LS_Command code
N_Port Deactivate Alias_ID (NDACT)	LS_Command code, Alias_ID	Accept (ACC) or LS_RJT	LS_Command code
Quality of Service Request (QoSR)	LS_Command code, Reserved, CTI VC_ID, CTI Address Identifier, CTI Maximum Bandwidth (B/s), CTI Minimum Bandwidth (B/s), CTI Maximum Delay (μs), CTI VC Data Field Size, Reserved, CTR VC_ID, CTR Address Identifier, CTR Maximum Bandwidth (B/s), CTR Minimum Bandwidth (B/s), CTR Maximum Delay (μs), CTR VC Data Field Size, Reserved, Reserved, Live VC credit limit, Class 4 end-to-end credit	Accept (ACC) or LS_RJT	LS_Command code, Reserved, CTI VC_ID, CTI Address Identifier, CTI Maximum Bandwidth (B/s), CTI Minimum Bandwidth (B/s), CTI Maximum Delay (μs), CTI VC Data Field Size, Reserved, CTR VC_ID, CTR Address Identifier, CTR Maximum Bandwidth (B/s), CTR Minimum Bandwidth (B/s), CTR Maximum Delay (μs), CTR VC Data Field Size, Reserved, Reserved, Live VC credit limit, Class 4 end-to-end credit
Read Virtual Circuit Status (RVCS)	LS_Command code, Reserved, N_Port Identifier	Accept (ACC) or LS_RJT	LS_Command code, Reserved, Class 4 VC Status Block
Discover N_Port	LS_Command code, Common Serv-	Accept (ACC) or	LS_Command code, Common Serv-

Table 194 - Extended Link Services Summary

Request Sequence		Reply Sequence	
Request	Payload	Reply	Accept Payload
Service parameters (PDISC)	ice Parameters, Port_Name, Node_Name, Service Parameters (Class 1, 2, 3), Vendor Version	LS_RJT	ice Parameters, Port_Name, Node_Name, Service Parameters, (Class 1, 2, 3), Vendor Version
Discover F_Port Service parameters (FDISC)	LS_Command code, Common Service Parameters, Port_Name, Node_Name, Service Parameters (Class 1, 2, 3), Vendor Version	Accept (ACC) or LS_RJT	LS_Command code, Common Service Parameters, Port_Name, Node_Name, Service Parameters, (Class 1, 2, 3), Vendor Version
Discover Address (ADISC)	LS_Command code, Reserved, Hard Address of Originator, Port_Name of Originator, Node_Name of Originator, Reserved, N_Port ID of Originator	Accept (ACC) or LS_RJT	LS_Command code, Reserved, Hard Address of Responder, Port_Name of Responder, Node_Name of Responder, Reserved, N_Port ID of Responder
Fibre Channel Address Resolution Protocol Request (FARP-REQ)	LS_Command code, Match Address Code Points, Requesting N_Port Identifier, Responder Action, Responding N_Port Identifier, Requesting N_Port Port_Name, Requesting N_Port Node_Name, Responding N_Port Port_Name, Responding N_Port Node_Name, Requesting N_Port IP Address, Responding N_Port IP Address	None	
Fibre Channel Address Resolution Protocol Reply (FARP-REPLY)	LS_Command code, Match Address Code Points, Requesting N_Port Identifier, Responder Action, Responding N_Port Identifier, Requesting N_Port Port_Name, Requesting N_Port Node_Name, Responding N_Port Port_Name, Responding N_Port Node_Name, Requesting N_Port IP Address, Responding N_Port IP Address	Accept (ACC) or LS_RJT	LS_Command code
Bandwidth Allocation (BALLOC)	LS_Command code, # Frame Positions, Frame Rate, Read/Write, Error Policy	Accept (ACC) or LS_RJT	LS_Command code, # Frame Positions, Frame Rate, Read/Write, Error Policy, Starting Frame Position
Bandwidth Deallocation (DALLOC)	LS_Command code, # Frame Positions, Frame Rate, Read/Write, Error Policy, Starting Frame Position	Accept (ACC) or LS_RJT	LS_Command code, # Frame Positions, Frame Rate, Read/Write, Error Policy, Starting Frame Position
Fabric Address Notification (FAN)	LS_Command code, Reserved, Loop Fabric Address, Fabric_Port_Name, Fabric_Name	None	
Registered State Change Notification (RSCN)	LS_Command code, Page Length, Payload Length, Affected N_Port ID Pages	Accept (ACC) or LS_RJT	LS_Command code
State Change Registration (SCR)	LS_Command code, Reserved, Registration Function	Accept (ACC) or LS_RJT	LS_Command code
Clock Synchronization Request (CSR)	LS_Command code, Clock Sync Mode, Reserved	Accept (ACC) or LS_RJT	LS_Command code, Clock Sync Mode, Reserved
Clock Synchronization Update (CSU)	LS_Command code, Clock Count	None	
Loop Initialize (LINIT)	LS_Command code, Reserved, Initialization Function, LIP Byte 3,	Accept (ACC) or LS_RJT	LS_Command code, Reserved, Status

Table 194 - Extended Link Services Summary			
Request Sequence		Reply Sequence	
Request	Payload	Reply	Accept Payload
	LIP Byte 4		
Loop Port Control (LPC)	LS_Command code, Reserved, Port Control Function, Loop Devices to Bypass, Loop Devices to Enable	Accept (ACC) or LS_RJT	LS_Command code, Reserved, Status
Loop Status (LSTS)	LS_Command code	Accept (ACC) or LS_RJT	LS_Command code, Reserved, Failed Receiver, FC-FLA Compliance Level, Loop State, Current Public Loop Devices bit map, Current Private Loop Devices bit map, AL_PA position map
Request Topology Information (RTIN)	LS_Command code, Flag, Domain_ID or FC Address, Maximum Size, Reserved	Accept (ACC) or LS_RJT	LS_Command code, Residual size, reserved, Connectivity Unit Header Record, Domain Record, Topology Information Records
Request Node Identification Data (RNID)	LS_Command code, Node Identification Data Format, Reserved	Accept (ACC) or LS_RJT	LS_Command code, Node Identification Data Format, Common Node Identification Data Length, Reserved, Specific Node Identification Data Length, Common Node Identification Data, Specific Node Identification Data
Registered Link Incident Record (LRIR)	LS_Command code, Link Incident Record Format, Common Link Incident Record Length, Link Incident Descriptor Length, Specific Common Link Incident Record Length, Common Link Incident Record, Link Incident Descriptor, Specific Link Incident Record	Accept (ACC) or LS_RJT	LS_Command code
Link Incident Record Registration (LIRR)	LS_Command code, Registration Function, Link Incident Record Registration Type, Reserved	Accept (ACC) or LS_RJT	LS_Command code
Login Control List Management (LCLM)	LS_Command code, Page Size, Payload Length, Reserved, LCLM Command, Global Capabilities, N_Port_Name, Node_Name, Reserved, Index	Accept (ACC) or LS_RJT	LS_Command code, Page Size, Payload Length, Reserved, LCLM Command, Global Capabilities, N_Port_Name, Node_Name, Reserved, Index, Port/Node_Name List
Read Port Status Block (RPS)	LS_Command code, Reserved, Flag, Port Selection	Accept (ACC) or LS_RJT	LS_Command code, Reserved, Flag, Reserved, Port Status, LESB, L_Port Extension
Read Port List (RPL)	LS_Command code, Reserved, Maximum Size, Reserved, Index	Accept (ACC) or LS_RJT	LS_Command code, Reserved, Payload Length, Reserved, List Length, Reserved, Index, Port Number Blocks
Report Node FC-4 Types (RNFT)	LS_Command code, Reserved, Maximum Size, Reserved, Index	Accept (ACC) or LS_RJT	LS_Command code, Reserved, Payload Length, Reserved, List Length, Reserved, Index, FC-4 Entries

15.6 Extended Link Service Reply Sequences

An Extended Link Service Reply Sequence shall signify that the Extended Link Service request Sequence is completed. The reply Sequence may contain data in the Payload following the

LS_Command code word. The format and meaning of the Payload is specified in the request Extended Link Service definition..

15.6.1 Accept (ACC)

The Accept Link Service reply Sequence shall notify the transmitter of an Link Service request that the Extended Link Service request Sequence has been completed. The Responder shall terminate the Exchange by setting the Last Sequence bit (Bit 20) in F_CTL on the last Data frame of the reply Sequence. The first byte of the Payload shall contain hex '02'. The remainder of the Payload is unique to the Link Service request.

Protocol:

Accept is the reply Sequence to Login, Logout, Abort Exchange, Read Connection Status, Read Exchange Status Block, Read Link Error Status, Read Timeout Value, Read Sequence Status Block, Reinstate Recovery Qualifier, Request Sequence Initiative, Establish Streaming, Advise Credit, and Echo request Sequences. (TBD - need to add the others to this list or remove the list - rwk)

Format: FT_1

Addressing: The D_ID field designates the source of the Link Service Sequence being accepted while the S_ID field designates the destination of the request Sequence being accepted.

Payload: The Payload content following the LS_Command code (hex '02 xx xx xx') is defined within individual Link Service requests.

Reply Link Service Sequence: none

15.6.2 Link Service Reject (LS_RJT)

The Link Service Reject (LS_RJT) shall notify the transmitter of a Link Service request that the Link Service request Sequence has been rejected. A four-byte reason code shall be contained in the Data_Field. Link Service Reject may be transmitted for a variety of conditions, which may be unique to a specific Link Service request.

For example, if the Service Parameters specified in a Login frame were logically inconsistent or in error, a P_RJT frame would not be transmitted in response, but rather a Link Service Reject.

Protocol:

LS_RJT may be a reply Sequence to any Extended Link Service request excluding ESTC and TEST.

Format: FT_1

Addressing: The D_ID field designates the source of the Link Service request being rejected while the S_ID field designates the destination of the request Data frame Sequence being rejected.

Payload: The first word of the Payload shall contain the LS_Command code (hex '01 00 00 00'). The next four bytes of this field shall indicate the reason for rejecting the request (see Figure 42 and Table 195 and Table 196).

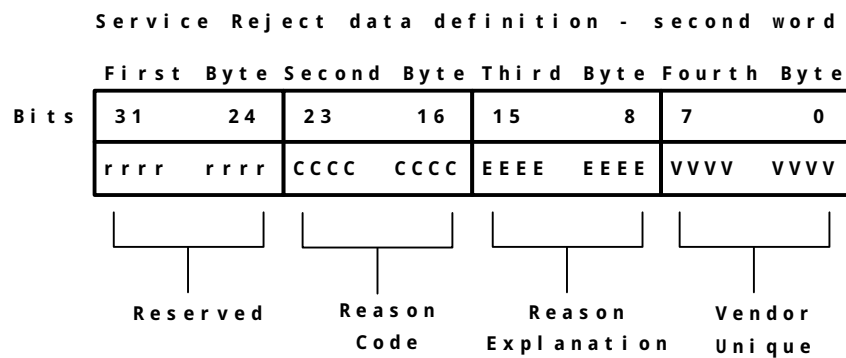


Figure 42 - LS_RJT format

Table 195 - LS_RJT reason codes	
Encoded Value (Bits 23-16)	Description
hex '01'	Invalid LS_Command code
hex '03'	Logical error
hex '05'	Logical busy
hex '07'	Protocol error
hex '09'	Unable to perform command request
hex '0B'	Command not supported
hex '0D'	Unable to Perform Command Request
Others	Reserved
hex 'FF'	Vendor Unique Error (See Bits 7-0)

The first error condition encountered shall be the error reported.

- Bits 23-16 Description**

Invalid LS_Command code

The LS_Command code in the Sequence being rejected is invalid or not supported.

Logical error

The request identified by the LS_Command code and Payload content is invalid or logically inconsistent for the conditions present.

Logical busy

The Link Service is logically busy and unable to process the request at this time.

Protocol Error

This indicates that an error has been detected which violates the rules of the Extended Link Service Protocol which are not specified by other error codes.

Unable to perform command request

The Recipient of a Link Service command is unable to perform the request at this time.

Command not supported

The Recipient of a Link Service command does not support the command requested.

Vendor Unique Error

The Vendor Unique Error bits may be used by Vendors to specify additional reason codes.

- Bits 15-8 Reason explanation**

Table 196 shows expanded explanations for Link Service commands with the applicable Extended Link Service commands.

Reply Link Service Sequence: none

Table 196 - LS_RJT reason code explanation		
Encoded Value (Bits 15-8)	Description	Applicable commands
hex '00'	No additional explanation	ABTX, ADVC, ESTS, FLOGI, PLOGI, LOGO, RCS, RES, RLS, RSS, RTV, RSI, PRLI, PRLO, TPLS, TPRLO, GAID, FACT, FFACT, NACT, NDACT, QoS, RVCS, PDISC, FDISC, ADISC, RNC, CSR, RNFT
hex '01'	Service Parm error - Options	FLOGI, PLOGI
hex '03'	Service Parm error - Initiator Ctl	FLOGI, PLOGI
hex '05'	Service Parm error - Recipient Ctl	FLOGI, PLOGI
hex '07'	Service Parm error - Rec Data Field Size	FLOGI, PLOGI
hex '09'	Service Parm error - Concurrent Seq	FLOGI, PLOGI
hex '0B'	Service Parm error - Credit	ADVC, FLOGI, PLOGI
hex '0D'	Invalid N_Port/F_Port_Name	FLOGI, PLOGI
hex '0E'	Invalid Node/Fabric Name	FLOGI, PLOGI
hex '0F'	Invalid Common Service Parameters	FLOGI, PLOGI
hex '11'	Invalid Association_Header	ABTX, RES, RRQ, RSI
hex '13'	Association_Header Required	ABTX, RES, RRQ, RSI
hex '15'	Invalid Originator S_ID	ABTX, RES, RRQ, RSI, RSS
hex '17'	Invalid OX_ID-RX_ID combination	ABTX, RES, RRQ, RSI, RSS
hex '19'	Command (request) already in progress	ABTX, PLOGI, RSI
hex '1E'	N_Port_Login Required	ADVC, ESTC, ESTS, LIRR, PRLI, PRLO, RNC, RNFT, RNID, RTIN, TPRLO, TPLS
hex '1F'	Invalid N_Port Identifier	RCS, RLS
hex '21'	Invalid SEQ_ID	RSS
hex '23'	Attempt to abort invalid Exchange	ABTX
hex '25'	Attempt to abort inactive Exchange	ABTX
hex '27'	Recovery_Qualifier required	ABTX
hex '29'	Insufficient resources to support Login	FLOGI, PLOGI
hex '2A'	Unable to supply requested Data	ADVC, ESTS, RCS, RES, RLS, RSS, RTV

Table 196 - LS_RJT reason code explanation

Encoded Value (Bits 15-8)	Description	Applicable commands
hex '2C'	Request not supported	ADVC, ESTS, ESTC, PRLI, PRLO, TPLS, TPRLO, GAID, FACT, FDACT, NACT, NDACT, QoSR, RVCS, PDISC, FDISC, ADISC, RNC, RNFT
hex '2D'	Invalid Payload Length	FLOGI, PLOGI
hex '30'	No Alias IDs available for this Alias Type	Get Alias_ID
hex '31'	Alias ID cannot be activated (no resources available)	Fabric Activate Alias ID, N_Port Activate Alias ID
hex '32'	Alias ID cannot be activated (invalid Alias ID)	Fabric Activate Alias ID, N_Port Activate Alias ID
hex '33'	Alias ID cannot be deactivated (doesn't exist)	Fabric Deactivate Alias ID, N_Port Deactivate Alias ID
hex '34'	Alias ID cannot be deactivated (resource problem)	Fabric Deactivate Alias ID, N_Port Deactivate Alias ID
hex '35'	Service Parameter conflict	N_Port Activate Alias ID
hex '36'	Invalid Alias_Token	Get Alias_ID
hex '37'	Unsupported Alias_Token	N_Port Activate Alias ID
hex '38'	Alias Group can not be formed	Get Alias ID
hex '40'	QoS Parm Error	QoSR
hex '41'	VC_ID Not Found	QoSR
hex '42'	Insufficient Resources to support a Class4 connection	QoSR
hex '44'	Invalid Port/Node_Name	LCLM
Others	Reserved	

15.7 FC-4 Link Service

A FC-4 Link Service request solicits a destination Port (F_Port or N_Port) to perform a function or service in order to support an individual FC-4 Device_Data protocol. The Information Category for a request shall be specified as Unsolicited Control. A FC-4 Link Service reply may be transmitted in answer to a FC-4 Link Service request. The Information Category for a reply shall be specified as Solicited Control. Each request or reply shall be composed of a single Sequence. The format of the request or reply shall be specified by the individual FC-4 being supported and is outside the scope of FC-FS. Each Sequence may be composed of one or more frames.

The protocols supported by the FC-4 Link Services shall be performed within a single Exchange, intended exclusively for the purpose. FC-4 Link Service protocols are performed using a two Sequence Exchange. The protocols consist of a request Sequence by the Originator (N_Port), transfer of Sequence Initiative, and a reply Sequence from the Responder (N_Port or F_Port). The execution of a FC-4 Link Service may perform sequence abort functions and modify sequence initiative of other exchanges in a protocol specific manner. The Sequence Initiator and Sequence Recipient shall follow the rules for Sequence management and Recov-

ery_Qualifier reuse as specified in clause 18. The following rules regarding Sequence and Exchange management apply to FC-4 Link Services in addition to the rules specified in clause 18:

- FC-4 Link Services shall only be Exchanges originated following N_Port_Login.
- the Originator of the Exchange shall use the Discard multiple Sequences Exchange Error Policy for all FC-4 Link Service Exchanges.
- the Originator of an 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.
- if the Exchange Originator of an FC-4 Link Service Exchange detects an Exchange error, it shall abort the Exchange (ABTX) and retry the protocol of the aborted Exchange with a different Exchange.
- if the Sequence Initiator aborts a Sequence using ABTS (Abort Sequence Protocol) due to receiving an ACK with the Abort Sequence bits (5-4) set to 0 1, the Sequence Initiator shall retry the Sequence after the Basic Accept is received for the aborted Sequence one time only.

R_CTL bits 31-28 (Word 0) are set = to 0011 to indicate a FC-4 Link_Data frame. The TYPE field for each FC-4 Link Service frame shall match the FC-4 Device_Data TYPE field as specified in Table 20.

16 Classes of service

Five Classes of service applicable to a Fabric and a N_Port are specified. These Classes of service are distinguished primarily by the methodology with which the communication circuit is allocated and retained between the communicating N_Ports and by the level of delivery integrity required for an application.

A given Fabric or N_Port may support one or more Classes of service. These Classes of service are:

- Class 1 - Dedicated Connection
- Class 2 - Multiplex
- Class 3 – Datagram
- Class 6 - Fractional
- Class 6 - Connected Multicast

Classes 1, 2, and 3 may be supported with any of the communication models. Classes 4 and 6 shall only be supported with the Fabric model. Intermix is specified as an option of Class 1 service.

In all Classes of service, the FC-2 Segmentation and Reassembly function makes available to the receiving ULP, the same image of application data as transmitted by the sending ULP (see clause 21

In all Classes of service, for each frame received, the Fabric shall do one of the following:

- deliver only one instance of the frame to any single N_Port
- issue a F_BSY (except Class 4)
- issue a F_RJT
- discard the frame without issuing any response.

16.1 Class 1 - Dedicated Connection

Class 1 is a service that provides Dedicated Connections. A Class 1 Connection is requested by a N_Port with another N_Port. An acknowledgement (ACK) is returned by the other N_Port to establish the Connection (see clause 22 In general, once a Connection is established, it shall be retained and guaranteed by the Fabric, until one of these N_Ports requests removal of the Connection. See 10.4 for special conditions in which a Connection may be removed.

NOTE - If a Class 1 Connection can be established between N_Ports of unlike speeds, the method and configuration by which this Connection is established will be transparent to FC-FS.

16.1.1 Class 1 function

A Class 1 Connection is requested by a N_Port with another N_Port via transmission of a frame containing a SOF_{c1} (Class 1/SOF_{c1}) . The Fabric, if present, allocates a circuit between the requesting N_Port and the destination N_Port. The destination N_Port transmits an ACK indicating its acceptance to the requesting N_Port. Upon successful receipt of the ACK at the requesting N_Port, the Connection is established (see 22.4.1). The Fabric retains the allocated circuit between these two N_Ports, until one of these N_Ports requests the Dedicated Connection be removed.

Even if a Fabric is not present, the requesting N_Port and the destination N_Port follow the same protocol.

Class 1 Delimiters as specified in 16.1.3 are used to establish and remove the Dedicated Connection and to initiate and terminate one or more Sequences within the Dedicated Connection.

16.1.2 Class 1 rules

The following rules apply to exclusive Class 1 Connections. See 16.4 for additional rules applicable to Intermix. To provide a Class 1 Connection, the transmitting and receiving N_Ports, and the Fabric shall obey the following rules:

- a) Except for some Link Service Protocols (see clause 15 a N_Port requesting a Class 1 Connection is required to have previously logged in with the Fabric and the N_Ports with which it intends to communicate, either implicitly or explicitly. To login explicitly, the requesting N_Port shall use Fabric and N_Port_Login protocols (see clause 17). (login)
- b) The Fabric is responsible for establishing a Connection at the request of a N_Port and retaining it until one of the communicating N_Ports explicitly requests removal of the Connection or one of the links attached to the N_Ports participates in a primitive sequence protocol (see 10.4). To establish or remove the Dedicated Connection, the requesting N_Port shall use the Class 1 Delimiters as specified in 16.1.3. (connection through Connection Sub-Fabric)
- c) After transmitting a Class 1/SOFc1 frame, the N_Port requesting the Connection shall not transmit additional frames to the destination N_Port, until it receives from that destination N_Port, an ACK that shall establish the Connection. (establish Connection)
- d) Once a Connection is established between two N_Ports, each N_Port shall send frames only to the other N_Port in the Connection until the Connection is removed. All these frames shall contain respective S_ID and D_ID for these N_Ports. (established Connection)
- e) A destination N_Port shall acknowledge delivery of every valid Data frame with an ACK_1 or the entire Sequence with a single ACK_0 (see 16.1.5). (acknowledgement)
- f) The Sequence Initiator shall increment the SEQ_CNT field of each successive frame transmitted within a Sequence. The Fabric shall guarantee delivery of the frames at the receiver in the same order of transmission within the Sequence (see 18.3.6). (sequential delivery)
- g) Each N_Port of an established Connection may originate multiple Exchanges and initiate multiple Sequences within that Connection. The N_Port originating an Exchange shall assign an X_ID unique to the Originator called OX_ID and the Responder of the Exchange shall assign an X_ID unique to the Responder called RX_ID. Thus within a given Connection, the value of OX_ID or RX_ID is unique to the respective N_Port. The Sequence Initiator shall assign a SEQ_Qualifier, for each Sequence it initiates, which is unique to the Sequence Initiator and the respective Sequence Recipient pair. (concurrent Exchanges and Sequences)
- h) Communicating N_Ports shall be responsible for end-to-end flow control, without any Fabric involvement. ACK frames are used to perform end-to-end flow control (see 16.1.5). All Class 1 frames except Class 1/SOFc1, participate only in end to-end flow control. A Class 1/SOFc1 frame participates in both end-to-end and buffer-to-buffer flow controls. (end-to-end flow control)
- i) The Fabric may reject a request for a Class 1 Connection or issue a busy with a valid reason code. After the Dedicated Connection is established, the Fabric shall not interfere with the Connection. The Fabric shall issue a F_BSY to any Class 2 frame or discard any Class 3 frame, transmitted from a third N_Port and destined to one of the N_Ports engaged in the Connection (see 14.3.3.1 and 14.3.3.3). (fabric reject or busy)
- j) The destination N_Port specified in Class 1/SOFc1 frame may respond with a busy or a reject with a valid reason code to this frame. Once the Dedicated Connection is estab-

lished, the destination N_Port shall not issue a busy but may issue a reject (see 14.3.3.2 and 14.3.3.3). (N_Port busy or reject)

- k) The End-to-end Credit established during the Login protocol by interchanging service Parameters shall be honored (see 20.3). At the beginning of a Connection, the End-to-end Credit_Count is reinitialized to the value determined during Login. Class 1/SOFc1 frame shall share the End-to-end Credit with Class 2 frames (see 20.3). (credit)
- l) The Fabric shall guarantee full bandwidth availability to the connected N_Ports (see clause 3) (bandwidth)
- m) Frames within a Sequence are tracked on a Sequence_Qualifier and SEQ_CNT basis (see 12.1.1). (tracking)
- n) A N_Port or F_Port shall be able to recognize SOF delimiters for all Classes of service, whether or not the Port supports all classes, and provide appropriate responses for all Classes with appropriate delimiters. (invalid Class)
 - If a N_Port which supports only Class 1 receives a Class 2 frame, and
 - is not engaged in a Dedicated Connection, the N_Port shall issue a P_RJT with appropriate Class 2 delimiters and obey buffer-to-buffer flow control rules.
 - is engaged in a Dedicated Connection, the N_Port response is unpredictable.
 - If a N_Port which supports only Class 1 receives a Class 3 frame, and
 - is not engaged in a Dedicated Connection, the N_Port shall discard the frame and obey the buffer-to-buffer flow control rules.
 - is engaged in a Dedicated Connection, the N_Port shall discard the frame and not follow the buffer-to-buffer flow control rules.
 - If a F_Port which supports only Class 1 receives a Class 2 frame, and
 - is not engaged in a Dedicated Connection, the F_Port shall issue a F_RJT with appropriate Class 2 delimiters and obey buffer-to-buffer flow control rules.
 - is engaged in a Dedicated Connection, the F_Port response is unpredictable.
 - If a F_Port which supports only Class 1 receives a Class 3 frame, and
 - is not engaged in a Dedicated Connection, the F_Port shall discard the frame and obey buffer-to-buffer flow control rules.
 - is engaged in a Dedicated Connection, the F_Port response is unpredictable.
- o) If a N_Port does not support Class 1 and receives a Class 1/SOFc1 frame, the N_Port shall issue a P_RJT with a SOFn1 and EOFdt with a reason code of Class not supported. If a F_Port does not support Class 1 and receives a Class 1/SOFc1 frame, the F_Port shall issue a F_RJT with a SOFn1 and EOFdt with a reason code of Class not supported. **(Class 1 not supported)**
- p) If a F_Port, not engaged in a Dedicated Connection, receives a frame with a SOFi1 or SOFn1, the F_Port response is unpredictable. However, the buffer-to-buffer control shall remain unaffected. **(invalid protocol)**

16.1.3 Class 1 delimiters

A Dedicated Connection is requested by transmitting a Data frame using a SOFc1 delimiter. SOFc1 initiates the first Sequence; subsequent Sequences are initiated with a SOFi1. SOFn1 starts all frames other than the first within a sequence.

EOFn terminates all frames other than the last frame within a Sequence. Each Sequence is terminated using an EOFt (see 16.1.5). An EOFdt contained in a frame terminates the Sequence in which the frame is sent and it also serves to remove the Dedicated Connection.

Other open Sequences in progress are also terminated. Exchanges and Sequences may be left in indeterminate state from the perspective of ULPs.

16.1.4 Class 1 frame size

The size of Data_Field of a frame using the SOF_{c1} delimiter is limited by the smaller value of the maximum Data_Field size supported for frames with SOF_{c1} by the Fabric and the destination N_Port. Subsequent frames, after a Dedicated Connection is established, are limited only by the maximum Data Field size supported by the destination N_Port.

16.1.5 Class 1 flow control

ACK frames are used to perform Class 1 end-to-end flow control. ACK frames are started with SOF_{n1}. The ACK used to terminate a Sequence shall end with EOF_t. The ACK used to terminate the Connection shall end with EOF_{dt}. All ACK frames, which do not terminate a Sequence, shall end with EOF_n.

All Class 1 frames shall follow end-to-end flow control rules (see 20.4.1). The Class 1/SOF_{c1} frame shall follow both end-to-end and buffer-to-buffer flow control rules (see 20.5.1).

16.1.6 Stacked connect-requests

Stacked connect-requests is a feature which may be provided by the Fabric (see 22.5.2 and 30).

16.2 Class 2 - Multiplex

This operating environment provides Connectionless service with notification of non-delivery between two N_Ports. This service allows one N_Port to transmit consecutive frames to multiple destinations without establishing a Dedicated Connection with any specific N_Port. Conversely, this service also allows one N_Port to receive consecutive frames from one or more N_Ports without having established Dedicated Connections with any of them.

16.2.1 Class 2 function

A Class 2 service is requested by a N_Port on a frame by frame basis. The Fabric, if present, routes the frame to the destination N_Port. If the N_Port transmits consecutive frames to multiple destinations, the Fabric demultiplexes them to the requested destinations.

Class 2 Delimiters are used to indicate the requested service and to initiate and terminate one or more Sequences as described in 16.2.3. Since Class 2 is Connectionless, the question of service removal does not arise.

16.2.2 Class 2 rules

To provide Class 2 service, the transmitting and receiving N_Ports, and the Fabric shall obey the following rules:

- a) Except for some Link Service Protocols (see clause 15), a N_Port supporting Class 2 service is required to have logged in with the Fabric and the N_Ports with which it intends to communicate, either explicitly or implicitly. To login explicitly, the requesting N_Port shall use Fabric and N_Port_Login protocols (see clause 17). (login)
- b) The Fabric routes the frames through Connectionless Sub Fabric, without establishing a Dedicated Connection between communicating N_Ports. To obtain Class 2 service from the Fabric, the N_Port shall use the Class 2 Delimiters as specified in 16.2.3. (connectionless service)

- c) A N_Port is allowed to send consecutive frames to one or more destinations. This enables a N_Port to demultiplex multiple Sequences to a single or multiple destinations concurrently (see 16.2.3). (demultiplexing)
- d) A given N_Port may receive consecutive frames from different sources. Each source is allowed to send consecutive frames for one or more Sequences. (multiplexing)
- e) A destination N_Port shall provide an acknowledgement to the source for each valid Data frame received. As in Class 1, the destination N_Port shall use ACK for the acknowledgement (see 16.2.5). If unable to deliver ACK, the Fabric shall return a F_BSY or F_RJT. (acknowledgement)
- f) The Sequence Initiator shall increment the SEQ_CNT field of each successive frame transmitted within a Sequence. However, the Fabric may not guarantee delivery to the destination in the same order of transmission (see 18.3.6). (non-sequential delivery)
- g) A N_Port may originate multiple Exchanges and initiate multiple Sequences with one or more destination N_Ports. The N_Port originating an Exchange shall assign an X_ID unique to the Originator called OX_ID and the Responder of the Exchange shall assign an X_ID unique to the Responder called RX_ID. The value of OX_ID or RX_ID is unique to a given N_Port. The Sequence Initiator shall assign a SEQ_ID, for each Sequence it initiates, which is unique to the Sequence Initiator and the respective Sequence Recipient pair while the Sequence is Open (see 18.5). (concurrent Exchanges and Sequences)
- h) Each F_Port (the local and the remote) exercises buffer-to buffer flow control with the N_Port to which it is directly attached. Communicating N_Ports performs end-to-end flow control. ACK frames are used to perform end-to-end flow control and R_RDY is used for buffer-to-buffer flow control. (dual flow control)
- i) If the Fabric is unable to deliver the frame to the destination N_Port, then the source is notified of each frame not delivered by an F_BSY or F_RJT frame from the Fabric with corresponding D_ID, S_ID, OX_ID, RX_ID, SEQ_ID, and SEQ_CNT. The source is also notified of valid frames busied or rejected by the destination N_Port by P_BSY or P_RJT. (non-delivery)
- j) A busy or reject may be issued by a F_Port or the destination N_Port with a valid reason code (see 17.7, 17.6, 14.3.3.3, and 14.3.3.2). (busy/reject)
- k) If a Class 2 Data frame is busied, the sender shall retransmit the busied frame up to the ability of the sender to retry, including zero. (retransmit)
- l) The Credit established during the Login protocol by interchanging Service Parameters shall be honored (see 20.3 for more on Credit.). Class 2 may share the Credit for Connectionless service with Class 3 and Class 1/SOFC1 frames (see 20.3). (credit)
- m) Effective transfer rate between any given N_Port pair is dependent upon the number of N_Ports a given N_Port is demultiplexing to and multiplexing from. (bandwidth)
- n) Frames within a Sequence are tracked on a Sequence_Qualifier and SEQ_CNT basis (see 12.1.1). (tracking)
- o) A N_Port or F_Port shall be able to recognize SOF delimiters for all Classes of service, whether or not the Port supports all Classes, and provide appropriate responses for all Classes with appropriate delimiters. A N_Port, which supports only Class 2 shall issue a P_RJT for Class 1 frames with appropriate Class 1 delimiters and discard Class 3 frames, while obeying the buffer-to-buffer flow control, rules in both cases. A F_Port, which supports only Class 2 shall issue a F_RJT for Class 1 frames with appropriate Class 1 delimiters and discard Class 3 frames, while obeying the buffer-to-buffer flow control, rules in both cases. (invalid class)
- p) The Class 2 PREF field is a Class of Service specific use of the CS_CTL field. Then PREF is zero, the Fabric shall deliver the frame normally. When PREF is one, the Fabric may deliver the frame to the destination N_Port prior to frames that have PREF set to zero. If the Fabric indicated through login that it guarantees order-of-delivery, the Fabric shall de-

liver frames with the same PREF value to a destination in the same order received from the source.

16.2.3 Class 2 delimiters

Sequences are initiated by transmitting a frame started by a SOF_{i2}. A SOF_{n2} starts subsequent frames within a Sequence. A Sequence is normally terminated with a frame ended by EOF_t. All frames other than the last frame within the Sequence are ended with an EOF_n.

16.2.4 Class 2 frame size

The size of each frame transmitted is limited by the smaller value of the maximum Data Field size supported by the Fabric or by the receiving N_Port. Each frame is routed through the Fabric, if present, as a separate entity.

16.2.5 Class 2 flow control

Class 2 service uses both buffer-to-buffer and end-to-end flow controls. R_RDY (Receiver Ready) is used for buffer to buffer flow control. R_RDY is transmitted by the F_Port to the N_Port originating the Class 2 frame to indicate that a buffer is available for further frame reception at the F_Port. R_RDY is transmitted by the destination N_Port to the attached F_Port to indicate that a buffer is available for further frame reception at the destination N_Port.

ACK frames are used to perform end-to-end flow control. ACK frames shall begin with SOF_{n2}. The ACK used to terminate a Sequence shall end with EOF_t. All ACK frames that do not terminate a Sequence shall end with EOF_n.

All Class 2 frames shall follow both buffer-to-buffer and end-to-end flow control rules (see 20.5.1 and 20.5.1).

16.3 Class 3 - Datagram

This operating environment provides Connectionless service without any notification of non-delivery (BSY or RJT), delivery (ACK), or end-to-end flow control between two N_Ports. The Fabric, if present, and the destination N_Port are allowed to discard Class 3 frames without any notification to the transmitting N_Port. This service allows one N_Port to transmit consecutive frames to multiple destinations without establishing a Dedicated Connection with any specific N_Port. Conversely, this service also allows one N_Port to receive consecutive frames from one or more N_Ports without having established Dedicated Connections with any of them.

16.3.1 Class 3 function

A Class 3 service is requested by a N_Port on a frame by frame basis. The Fabric, if present, routes the frame to the destination N_Port. If the N_Port transmits consecutive frames to multiple destinations, the Fabric demultiplexes them to the requested destinations.

Class 3 Delimiters are used to indicate the requested service and to initiate and terminate one or more Sequences as described in 16.3.3. Since Class 3 is Connectionless, the question of service removal does not arise.

16.3.2 Class 3 rules

To provide Class 3 service, the transmitting and receiving N_Ports, and the Fabric shall obey the following rules:

- a) Except for some Link Service Protocols (see clause 15), a N_Port supporting Class 3 service is required to have logged in with the Fabric or the N_Ports, either explicitly or implicitly. To login explicitly, the requesting N_Port shall use Fabric and N_Port_Login protocols (see clause 17). (login)

- b) The Fabric routes the frames through Connectionless Sub_Fabric, without establishing a Dedicated Connection between communicating N_Ports. To obtain Class 3 service from the Fabric, the N_Port shall use the Class 3 Delimiters as specified in 16.3.3. (connectionless service)
- c) A given N_Port is allowed to send consecutive frames to one or more destinations. This enables a N_Port to demultiplex multiple Sequences to single or multiple destinations concurrently. (demultiplexing)
- d) A given N_Port may receive consecutive frames from one or more source N_Ports. Each source N_Port is allowed to send consecutive frames for one or more Sequences. (multiplexing)
- e) A destination N_Port shall not provide acknowledgement (ACK) to the source for any valid frame received. (absence of acknowledgement)
- f) The Sequence Initiator shall increment the SEQ_CNT field of each successive frame transmitted within a Sequence. However, the Fabric may not guarantee delivery at the receiver in the same order of transmission (see 18.3.6). (non-sequential delivery)
- g) A N_Port may originate Exchanges and initiate Sequences with one or more destination N_Ports. The N_Port originating an Exchange shall assign an X_ID unique to the Originator called OX_ID and the Responder of the Exchange shall assign an X_ID unique to the Responder called RX_ID. Responder may assign an RX_ID in the first Sequence it transmits. The value of OX_ID or RX_ID is unique to a given N_Port. The Sequence Initiator shall assign a SEQ_Qualifier, for each Sequence it initiates, which is unique to the Sequence Initiator and the respective Sequence Recipient pair (see 18.5). (concurrent Exchanges and Sequences)
- h) The local F_Port exercises buffer-to-buffer flow control with the transmitting N_Port. The remote F_Port exercises buffer to-buffer flow control with the receiving N_Port. R_RDY is used for buffer-to-buffer flow control (see 16.3.5). (buffer to buffer flow control)
- i) If the Fabric is unable to deliver the frame to the destination N_Port, the frame is discarded and the source is not notified. If the destination N_Port is unable to receive the frame, the frame is discarded and the source is not notified. (non-delivery)
- j) The buffer-to-buffer Credit is used for buffer to buffer flow control. End-to-end Credit is not used (see 20.3). Class 3 may share the Credit for Connectionless service with Class 2 and Class 1/SOFc1 frames (see 20.3). (Credit)
- k) Effective transfer rate between any given N_Port pair is dependent upon the number of N_Ports a given N_Port is demultiplexing to and multiplexing from. (bandwidth)
- l) Neither the F_Port nor N_Port shall issue busy or reject to Class 3 frames. (busy/reject)
- m) Frames within a Sequence are tracked on a Sequence_Qualifier and SEQ_CNT basis (see 12.1.1). (tracking)
- n) A N_Port or F_Port shall be able to recognize SOF delimiters of all Classes of service, whether or not the Port supports all Classes, and provide appropriate responses for all Classes with appropriate delimiters. A N_Port, which supports only Class 3, shall issue a P_RJT for Class 1 or Class 2 frames with appropriate Class 1 or Class 2 delimiters respectively while obeying the buffer-to-buffer flow control rules. A F_Port which supports only Class 3 shall issue a F_RJT for Class 1 or Class 2 frames with appropriate Class 1 or Class 2 delimiters respectively, while obeying the buffer-to-buffer flow control rules. (invalid Class)
- o) A N_Port may obtain the delivery status of Class 3 Sequences transferred by using Abort Sequence protocol (see 23.7.1.1) and thus verify the integrity of the delivered Sequences. (Sequence integrity)
- p) The Class 3 PREF field is a Class of Service specific use of the CS_CTL field. Then PREF is zero, the Fabric shall deliver the frame normally. When PREF is one, the Fabric may deliver the frame to the destination N_Port prior to frames that have PREF set to zero. If

the Fabric indicated through login that it guarantees order-of-delivery, the Fabric shall deliver frames with the same PREF value to a destination in the same order received from the source.

16.3.3 Class 3 delimiters

Sequences are initiated by transmitting a frame started by a SOFi3. A SOFn3 starts subsequent frames within a Sequence. A Sequence is terminated with a Data frame ended by EOFt. An EOFn terminates all frames other than the last frame within the Sequence.

16.3.4 Class 3 frame size

The size of each frame transmitted is limited by the smaller value of the maximum Data Field size supported by the Fabric and by the receiving N_Port. Each frame is routed through the Fabric, if present, as a separate entity.

16.3.5 Class 3 flow control

Class 3 uses only buffer to buffer flow control with R_RDY. End-to-end flow control is not supported.

All Class 3 frames shall follow buffer-to-buffer flow control rules (see 20.5.1).

16.3.6 Class 3 Sequence integrity

With a missing Class 3 Data frame, the Sequence Recipient is capable of detecting the error of non-receipt of the frame, but can not communicate it to the Sequence Initiator due to absence of ACK in Class 3. However, using Abort Sequence protocol (see 18.3.11 and 23.7), the Sequence Initiator can verify if one or more transmitted Sequences were received without any Sequence error. This usage of Abort Sequence protocol makes it possible to verify the integrity of Class 3 Sequences delivered.

If a sending ULP relies on the receiving ULP for ensuring Sequence integrity, the Sequence Initiator may not use the Abort Sequence protocol to confirm Sequence delivery.

16.4 Intermix

Intermix is an option of Class 1 service which allows interleaving of Class 2 and Class 3 frames during an established Class 1 Connection between two N_Ports. While engaged in a Class 1 Connection, a N_Port capable of Intermix may transmit and receive Class 2 and Class 3 frames interleaved with Class 1 frames. Class 2 and Class 3 frames may be interchanged with either the N_Port at the other end of the Connection or with any other N_Port. A N_Port, which supports Intermix, shall be capable of both transmitting and receiving intermixed frames. Support for Intermix option of Class 1 service is specified during Login.

In a point-to-point topology, both interconnected N_Ports shall be required to support Intermix if Intermix is to be employed. In the presence of a Fabric, both the N_Port and the Fabric shall be required to support Intermix if Intermix is to be employed in that link. Fabric support for Intermix requires that the full Class 1 bandwidth during a Dedicated Connection be maintained. Intermix permits the potentially unused Class 1 bandwidth to be available for transmission of Class 2 and Class 3 frames.

16.4.1 Fabric management

If a Fabric that supports Intermix receives a Class 2 frame destined to a N_Port engaged in a Dedicated Connection and the N_Port does not support Intermix, the Fabric shall return an F_BSY to the source N_Port of the frame, after ensuring that the frame is not deliverable. The Fabric is allowed to hold a Class 2 frame for a period of time before issuing F_BSY. If the

Fabric receives a Class 3 frame destined to a N_Port engaged in a Dedicated Connection and the N_Port does not support Intermix, the frame is discarded.

If a Fabric that does not support Intermix receives a Class 2 frame from a third N_Port destined to either of the N_Ports engaged in a Dedicated Connection, an F_BSY shall be returned to the source N_Port of the frame. If this Fabric receives a Class 3 frame from a third N_Port destined to either of the N_Ports engaged in a Dedicated Connection, the frame is discarded. The Fabric is allowed to hold a Class 2 frame for a period of time before issuing F_BSY.

If a N_Port, established in a Dedicated Connection and attached to a Fabric that does not support Intermix, transmits a Class 2 or Class 3 frame destined to the other N_Port of the Dedicated Connection or to a third N_Port, the destination of this frame delivery is unpredictable.

During an established Class 1 Connection with Intermix supported, Class 1 frames have priority over Class 2 or Class 3 frames. Class 2 and Class 3 frames shall be interleaved during a Class 1 Connection only if there is no backlog of Class 1 frames en route. Although a Fabric and an attached N_Port both support Intermix, the Fabric may choose to transmit F_BSY to a Class 2 frame or discard a Class 3 frame while the N_Port is engaged in a Class 1 Connection.

The Fabric shall provide adequate buffering for an incoming Class 1 frame while a Class 2 or Class 3 frame is being transmitted during a Class 1 Connection. The extent of buffering needed is dependent on the manner in which a Class 2 or Class 3 frame in transit is managed:

- to successfully complete a Class 2 or 3 frame in transit, an incoming Class 1 frame shall be buffered to the extent of the maximum Class 2 or Class 3 frame size, or
- if a Class 2 or Class 3 frame is being aborted (EOF_a) on receipt of a Class 1 frame, the Class 1 frame shall be buffered to the extent of the time required to append an EOF_a to the Class 2 or 3 frame in transit.

16.4.2 Intermix rules

A N_Port pair shall have a Class 1 Connection established for Intermix rules to be applicable to them. To provide an Intermix service, the Fabric and the receiving and transmitting N_Ports shall obey the following rules:

- a) The Fabric or a N_Port shall provide the Intermix Service Parameter during Login to indicate its Intermix capability to the communicating Port. (Intermix Service Parameter)
- b) If the Fabric supports Intermix, a N_Port supporting Intermix is allowed to transmit Class 2 and Class 3 frames during a Class 1 Connection intermixed with Class 1 frames of that Connection.

If the Fabric does not support Intermix, the N_Port shall not transmit intermixed frames. If the N_Port transmits intermixed frames, delivery of these frames is unpredictable. (transmit Intermix)

- c) An attached N_Port supporting Intermix shall be capable of receiving Class 2 and Class 3 frames during Class 1 Connection intermixed with Class 1 frames of that Connection. If a N_Port supporting Intermix receives a Class 2 or Class 3 frame intermixed, and the Fabric does not support Intermix, the frame shall be discarded. (receive Intermix)
- d) If the Fabric that supports Intermix, receives a Class 2 or Class 3 frame destined to a N_Port engaged in a Dedicated Connection, and the destination N_Port does not support Intermix, the Fabric shall return F_BSY to the Class 2 frame after ensuring that the frame is not deliverable. The Fabric shall discard the Class 3 frame without any response to the sender. (Intermix fabric busy or discard)
- e) If the Fabric that does not support Intermix, receives a Class 2 or Class 3 frame from a third N_Port destined to a N_Port engaged in a Dedicated Connection, regardless of whether the destination N_Port supports Intermix or not, the Fabric shall return F_BSY to

the Class 2 frame after ensuring that the frame is not deliverable. The Fabric shall discard the Class 3 frame without any response to the sender.

- f) If a N_Port attached to the Fabric that does not support Intermix transmits intermixed frames in violation of rule b, the delivery of these intermixed frames is unpredictable. (Non-Intermix fabric busy or discard)
- g) The Fabric shall guarantee full Class 1 bandwidth during a Dedicated Connection. Class 2 and Class 3 frames shall flow on the potentially unused Class 1 bandwidth. (bandwidth sharing)
- h) Class 1 frames have priority over Class 2 and Class 3 frames. Class 2 and Class 3 frames may be intermixed only when there is no backlog of Class 1 frames. The Fabric may issue F_BSY to a Class 2 frame and discard a Class 3 frame. The Fabric may abort (EOFa) a Class 2 or Class 3 frame in progress if its transmission interferes with the transmission of a Class 1 frame. (Class 1 precedence)
- i) The Fabric may cause a delay and displace a Class 1 frame in time due to Intermix. This delay is limited to the maximum Class 2 or 3 frame size. The Fabric shall provide adequate buffering for the incoming Class 1 frame while a Class 2 or a Class 3 frame is in transit and guarantee the integrity and delivery of Class 1 frames. (Class 1 frame skew and integrity)
- j) In a point-to-point topology, a N_Port which supports Intermix may transmit and receive Class 2 and Class 3 frames during Class 1 Connection intermixed with Class 1 frames if the other N_Port supports Intermix. (point-to-point Intermix)

16.4.3 Intermix delimiters

Intermix does not impose any additional delimiters.

16.4.4 Intermix frame size

The size of each Class 1, Class 2, or Class 3 frame is governed by the limitations of each Class individually. Intermix does not impose any additional limitation on the frame size.

16.4.5 Intermix flow control

The flow control for each Class is governed separately by individual Class. Intermix does not impose any additional rules on flow control.

16.5 Class 4

Class 4 is a connection-oriented service that allows circuits to be established between communicating N_Ports. A Class 4 circuit is bi-directional and consists of two unidirectional Virtual Circuits (VC) established between two N_Ports. Each VC is established according to unique Quality of Service (QoS) parameters that include guaranteed bandwidth and latency. A QoS facilitator (QoSF) function provided by the Fabric assists in the establishment, maintenance, and management of VCs and their associated resources

16.5.1 Class 4 functions

Class 4 defines four basic functions. They are:

Circuit Setup

An N_Port requests setup of a Class 4 circuit to another N_Port by sending a Quality of Service request (QoSR) Extended Link Service command to the QoSF. The requesting N_Port becomes the circuit initiator (CTI) and the target N_Port the circuit recipient (CTR). The QoSR identifies the communicating N_Ports and the required QoS parameters.

The QoS_R is granted if the QoS_F can establish a path between the communicating N_Ports with the requested QoS parameters and if the CTR also can accept the QoS parameters. If the QoS_R is accepted then a Class 4 circuit is established in the Pending state when the ACC Sequence is returned to the CTI.

The QoS_R is rejected if the QoS_F is unable to establish a path between the communicating N_Ports with the requested QoS parameters. The CTR is never queried if the QoS_F is unable to establish a path between the communicating N_Ports.

The QoS_R is rejected if the QoS_F can establish a path between the communicating N_Ports with the requested QoS parameters but the CTR is unable or unwilling to accept the QoS_R.

Circuit Activation

A Class 4 circuit activation request is used to activate the two VCs for the Class 4 circuit after which Class 4 frames may be transmitted between the communicating N_Ports. Either N_port may activate the Class 4 circuit. Once activated, the Class 4 circuit may be deactivated with a deactivation request or removed entirely with a removal request by either of the communicating N_Port.

A Class 4 circuit in the Dormant state is activated and enters the Live state when a SOFc4 delimited frame is sent or received.

While in the live state, bandwidth utilization is monitored by the Fabric, and frame transmission is regulated by the Fabric. Once the Class 4 circuit activation frame has been transmitted Class 4 frame flow is possible, only limited by available EE_C4_Credit and VC_Credit.

The Fabric manages buffer-to-buffer flow control and the Fabric regulates frame transmission from each source N_Port. The Fabric regulates each VC independently with the Virtual Circuit Ready (VC_RDY) Primitive Signal. Fabric flow control is accomplished using the VC_RDY Primitive Signal for each Live Class 4 circuit.

Class 4 end-to-end flow control is managed by the communicating N_Ports using ACK frames.

Circuit Deactivation

A Class 4 circuit, in the Live state, is deactivated, and enters the Dormant state, when an EOFdt or EOFdti delimited frame is sent or received. A deactivated Class 4 circuit may later be reactivated to the Live state.

The Fabric shall issue two END Link_Control commands, one on either VC, when the QoS_F determines for any reason that a Class 4 circuit should be deactivated. The Fabric shall use the addresses of the communicating N_Ports, as the S_ID and D_ID, in the generated END Link_Control commands, and the VC_ID shall match with the S_ID field. The END Link_Control command frame delimiters shall be EOFdt and SOFn4. The Fabric shall use this mechanism to deactivate Class 4 circuits when the N_Port or F_Port enters the Link Reset state.

When a Class 4 circuit is deactivated, an N_Port shall reclaim all EE_C4_Credit when the N_Port enters the Dormant state (i.e., set EE_C4_Credit equal to EE_C4_Credit_Limit when using the count down model for credit management).

Circuit Removal

The request to remove a Live Class 4 circuit may be sent by either of the communicating N_Ports.

A Class 4 circuit, in the Live or Dormant state, is removed, and enters the Nonexisting state, when an EOFrt or EOFrti delimited frame is received.

The two N_Ports shall not reactivate the Class 4 circuit when the Class 4 circuit has been removed. However, the VC_IDs associated with the removed Class 4 circuit (CIVC_ID and CRVC_ID) may be reused. The QoS shall return any resources it has allocated to a common pool for use by other Live Class 4 circuits or other Classes of Service when the Class 4 circuit has been removed.

If required for any reason, a Class 4 circuit may be removed by the Fabric from either the Live state or the Dormant state. To achieve this the Fabric shall issue two END Link_Control commands, one on either VC. The Fabric shall use the addresses of the communicating N_Ports, as the S_ID and D_ID, in the generated END Link_Control commands, and the VC_ID shall match with the S_ID field. The END Link_Control command frame delimiters shall be EOFrt and SOFn4. If the Class 4 circuit is alive, then the Fabric shall first deactivate the Class 4 circuit before removing the circuit.

16.5.2 Class 4 rules

To provide a Class 4 circuit, the transmitting and receiving N_Ports and the Fabric, shall obey the following rules:

- a) Except for some Link Service protocols an N_Port shall have performed Fabric Login and N_Port_Login with the ports it intends to communicate with, either explicitly or implicitly (see 28.9.1). To login explicitly, the requesting N_Port shall use the Fabric and N_Port Login protocols.
- b) The Fabric shall be responsible for Class 4 circuit setup. The Fabric shall retain the setup Class 4 circuit until one of the communicating N_Ports in the Class 4 circuit requests removal of the Class 4 circuit unless one of the legal Fabric exceptions occur. Legal Fabric exceptions includes: path failure and Fabric Login.
- c) To activate a Class 4 circuit, the requesting N_Port shall use the Class 4 delimiters as specified in 28.4. A N_Port shall only send an SOFc4 delimited frame on a Class 4 circuit in the Dormant state.
- d) To deactivate a Class 4 circuit, the requesting N_Port shall use the Class 4 delimiters as specified in 28.4. The Class 4 circuit may be re-activated at a later time by using the procedure in 28.2.4.2.
- e) To remove a Class 4 circuit, the requesting N_Port shall use the Class 4 delimiters as specified in 28.4.
- f) A destination N_Port shall acknowledge delivery of each valid Data Frame (see 28.9.4.1).
- g) The Sequence Initiator shall increment the SEQ_CNT field on each successive frame transmitted within a Sequence. The Fabric shall guarantee delivery of the frames at the destination N_Port in the order of transmission by the source N_Port.
- h) Each N_Port of a Live Class 4 circuit may originate multiple Exchanges and initiate multiple Sequences with one or more N_Ports. Each N_Port shall assign a unique X_ID as appropriate for its role in each Exchange. Thus, the value of the OX_ID and RX_ID is unique for each Exchange with an N_Port pair. a Sequence Initiator shall assign a unique Sequence_Qualifier for each Sequence it initiates which is unique for the communicating N_Port pair.
- i) Communicating N_Ports shall be responsible to manage end-to-end flow control without any Fabric involvement. ACK frames shall be used to perform end-to-end flow control (see 28.9.4.1). All Class4 frames, except for the END Link_Control command, shall participate in end-to-end flow control.
- j) The Fabric shall be responsible to regulate the rate at which frames are transmitted by an N_Port. It does so by transmitting the VC_RDY Primitive Signal to an N_Port to request that frames be transmitted. Each VC_RDY shall identify the VC for which a frame is being

- requested. The Fabric shall measure the QoS for bandwidth by measuring all transmission words associated with the VC (SOF delimiters, EOF delimiters, and frame content).
- k) The Fabric may reject a request to activate a Class 4 circuit with a valid reason code. Once the Class 4 circuit is alive the QoSF shall only interfere with a VC to the extent that it supplies the VC_RDY Primitive Signals in a timely manner.
 - l) The Fabric shall respond with an F_BSY Link_Response frame to any received Class 1 connect-request frame (i.e., a SOFc1 delimited frame) if one or more Class 4 circuits are setup with the destination N_Port frame).
 - m) The destination N_Port for a request to establish a VC may respond with a P_RJT with an appropriate reason code.
 - n) An N_Port shall not respond with a P_BSY for any Class 4 frame received.
 - o) The EE_C4_Credit, determined during the circuit setup process shall be honored.
 - p) The Fabric shall guarantee that it provides VC_RDY Primitive Signals to the communicating N_Ports at the correct rate and time intervals to meet or exceed the QoS parameters guaranteed for each VC. The latency guarantee relates only to the frame transit within the Fabric.
 - q) An N_Port shall only transmit Class 4 frames if available VC_Credit permits (see 28.9.4.2)
 - r) N_Ports shall discard all Class 4 frames received on a Nonexisting Class 4 circuit, or if the Class 4 circuit is in the Pending state.
 - s) The N_Port shall discard all SOFi4 and SOFn4 delimited frames received on a Class 4 circuit in the Dormant state.
 - t) An F_Port shall discard all SOFi4 and SOFn4 delimited frames received unless the Class 4 circuit is in the Live state.
 - u) EE_C4_Credit shall remain unaffected by any discarded Class 4 frames.
 - v) If VC_Credit is = 0 for a time of more than E_D_TOV the N_Port may use the RVCS protocol to read the state of the Class 4 circuit or the N_Port shall perform the Online to Offline protocol. If the N_Port obtains the Class 4 circuit state from the ACC to the RVCS request and it shows that the Class 4 circuit is Nonexisting, the N_Port shall set the circuit state to Nonexisting.
 - w) Frames within a Sequence are tracked on a Sequence_Qualifier and SEQ_CNT basis.
 - x) If an N_Port, not supporting Class 4, receives a Class 4 circuit activation frame, the N_Port may return a P_RJT frame with a reason code of "Class not supported", or the N_Port may discard the Class 4 frame. If an F_Port, not supporting Class 4, receives a Class 4 circuit activation frame, the F_Port may return a F_RJT response frame with a reason code of "Class not supported".
 - y) The Fabric shall use the VC_ID from the frame header for routing frames to a destination N_Port (D_ID). The QoSF maintains a translation from S_ID||VC_ID to D_ID from the original QoSR transaction. Prior to routing a frame, if there is a mismatch on S_ID, VC_ID and D_ID, the frame shall be discarded by the Fabric.
 - z) An N_Port supporting Class 4 shall not send Class 1 frames nor attempt to establish a Class 1 Dedicated Connection while a Class 4 circuit exists with the N_Port or while a request to setup a Class 4 circuit has been made by the N_Port.

16.5.3 Class 4 intermix

Intermix is always enabled for service Class 4.

N_Ports may interleave Class 2 or Class 3 frames on the outbound fibre while a Class 4 circuit exists. The same N_Port may receive Class 2 or Class 3 frames on the inbound fibre while a Class 4 circuit exists.

Transmission of Class 4 frames shall have priority over Class 2 and 3, but a Class 2 or Class 3 frame shall not be aborted to send a Class 4 frame.

16.5.4 Class 4 delimiters

A Class 4 circuit in the Dormant state is activated to Live state by using an SOFc4 delimiter for the first frame of a Sequence.

The frames that initiate a Sequence while the Class 4 circuit is in the Live state shall begin with an SOFi4 delimiter.

All other frames for a Class 4 circuit in the Live state shall begin with an SOFn4 delimiter.

All frames other than the last frame of a Sequence shall be terminated with an EOFn delimiter.

Each Sequence other than a Sequence that deactivates or removes a Class 4 circuit shall terminate with an EOFt delimiter.

The EOFni delimiter shall be used by the Fabric to replace either an EOFn or an EOFt on frames for which the Fabric detects a frame error.

The EOFa delimiter shall be used to terminate a partial frame due to a malfunction during transmission. It is also used by the Fabric to replace missing EOF delimiters or to truncate overlength frames.

The EOFdt delimiter shall deactivate the Class 4 circuit.

The EOFdti delimiter shall replace a recognized EOFdt delimiter on a frame with invalid frame content.

The EOFrt delimiter shall remove the Class 4 circuit.

The EOFrti delimiter shall replace a recognized EOFrt delimiter on a frame with invalid frame content.

16.5.5 Class 4 frame size

The maximum frame size used for all frames for a Class 4 circuit shall be the VC Data Field size decided during the QoS Extended Link Service request.

The Fabric may discard or truncate any Class 4 frames exceeding the Data Field Size limit. If the Fabric truncates a Class 4 frame it shall be terminated with an EOFa delimiter.

16.5.6 Class 4 flow control

ACK frames shall be used to perform Class 4 end-to-end flow control. All Class 4 frames shall participate in end-to-end flow control.

Class 4 buffer-to-buffer flow control follows the rules specified in (28.9.4.2).

16.6 Class 6 Multicast Connection

Class 6 is a service which provides Dedicated Connections for Reliable multicast (see clause 25.2) for a Fabric topology. A N_Port for two or more destination N_Ports requests a Class 6 connection. An acknowledgment is returned by the destination N_Ports to a multicast server which returns an acknowledge to the connection initiator, establishing a reliable multicast connection. In general, once a connection is established, it shall be retained and guaranteed by the Fabric until the connection initiator requests removal.

16.6.1 Class 6 function

A Class 6 connection is requested by a N_Port to one or more destination N_Ports via transmission of a frame containing a SOFc1. A Class 6 SOFc1 is identical to a Class 1 SOFc1. The Fabric recognizes the multicast D_ID and establishes circuits between the requesting N_Port and the destination N_Ports. The destination N_Ports each transmit an ACK, indicating acceptance, which are intercepted by the Multicast Server. The Multicast Server, based on the ACKs, RJTs, or BSYs received from the destinations, transmits an ACK, a P_RJT, or a P_BSY to the requesting N_Port indicating acceptance, rejection, or busy for the request. The return of an ACK indicates the establishment of a Multicast Connection. The Fabric retains the allocated circuits between the requesting N_Port and the destination N_Ports until the connection initiator request the Multicast Connection be removed.

Class 1 Delimiters, as specified in 16.1.3, are used to establish and remove the Multicast Connection and to initiate and terminate one or more sequences within the Multicast Connection.

Class 1 service parameters, as specified during Login, shall be used to manage Class 6 connections.

16.6.2 Class 6 rules

The following rules apply to exclusive Class 6 Connections. See 16.4 for additional rules applicable to Intermix. To provide a Class 6 Connection, the transmitting and receiving N_Ports, Fabric, and multicast server shall obey the following rules:

- a) Except for some Link Service Protocols, a N_Port requesting a Class 6 Connection is required to have previously logged in with the Fabric and the N_Ports with which it intends to communicate, either implicitly or explicitly. To Login explicitly, the requesting N_Port shall use Fabric and N_Port_Login protocols.
- b) The Fabric is responsible for establishing a Connection at the request of a N_Port and retaining it until the requesting N_Port explicitly requests removal of the Connection or the requesting N_Port's link participates in a primitive sequence protocol. To establish or remove the Dedicated Connection, the requesting N_Port shall use the Class 1 Delimiters as specified in 16.1.3.
- c) After transmitting a Class 6/SOFc1 frame, the N_Port requesting the Connection shall not transmit additional frames to the destination N_Ports until it receives, from the multicast server, an ACK which shall establish the connection.
- d) Once a Connection is established between multiple N_Ports, only the requesting N_Port (Connection Initiator) shall send data frames. Only the Connection Initiator shall cause the connection to be terminated. All frames shall contain S_ID and D_ID as appropriate (i.e., The D_ID shall be the multicast address).
- e) A destination N_Port shall acknowledge delivery of every valid Data frame with an ACK_1 or the entire Sequence with a single ACK_0 (see 16.1.5).
- f) The Sequence Initiator shall increment the SEQ_CNT field of each successive frame transmitted within a Sequence. The Fabric shall guarantee delivery of the frames at the receivers in the same order of transmission within the Sequence (see 18.3.6).
- g) The Connection Initiator N_Port of an established Connection may originate multiple Exchanges and initiate multiple Sequences within that Connection. This N_Port shall assign a unique X_ID called OX_ID. The Multicast Server for the Exchange shall assign an X_ID unique to the Responders called RX_ID. The value of the RX_ID shall be negotiated during N_Port/Fabric login (see 17.6.8). Thus, within a given Connection, the value of OX_ID or RX_ID is unique to the respective N_Port and multicast server. The Sequence Initiator (Connection Initiator) shall assign a SEQ_Qualifier, for each Sequence it initiates, which is unique to the Sequence Initiator and the respective Sequence Recipients.

- h) Communicating N_Ports and the multicast server shall be responsible for end-to-end flow control, without any fabric involvement. ACK frames are used to perform end-to-end flow control (see 16.1.5). All Class 6 frames, except Class 6/SOFc1, participate only in end-to-end flow control. A Class 6/SOFc1 frame participates in both end-to-end and buffer-to-buffer flow controls.
- i) The Fabric may reject a request for a Class 6 Connection or issue a busy with a valid reason code. After the Dedicated Connection is established, the Fabric shall only issue a reject at the direction of the Multicast Server. The Fabric shall issue a F_BSY to any Class 2 frame or discard any Class 3 frame, transmitted from a third N_Port and destined to one of the N_Ports engaged in the Connection (see 14.3.3.1 and 14.3.3.3).
- j) The destination N_Ports specified in the Class 6/SOFc1 frame may respond with a busy or a reject with a valid reason code to this frame. The multicast server shall interpret these responses and respond with an ACK, busy, or a reject with a valid reason code to the requesting N_Port (see section on multicast server). Once the Dedicated Connection is established, the destination N_Ports shall not issue a busy but may issue a reject (see 14.3.3.2 and 14.3.3.3).
- k) The End-to-end Credit established during the Login protocol by interchanging service Parameters shall be honored (see 20.3). TBD End-to-end credit shall be managed between Connection Initiator and the multicast server. At the beginning of a Connection, the End-to-end Credit_Count is reinitialized to the value determined during Login. A Class 6/SOFc1 frame shall share the End-to-end Credit with Class 2 frames (see 20.3).
- l) The Fabric shall guarantee full bandwidth availability to the connected N_Ports (see clause 3)
- m) Frames within a Sequence are tracked on an Sequence_Qualifier and SEQ_CNT basis (see 12.1.1).
- n) A N_Port or F_Port shall be able to recognize SOF delimiters for all Classes of service, whether or not the Port supports all Classes, and provide appropriate responses for all Classes with appropriate delimiters.
 - If a N_Port which supports only Class 1 receives a Class 2 frame, and
 - is not engaged in a Dedicated Connection, the N_Port shall issue a P_RJT with appropriate Class 2 delimiters and obey buffer-to-buffer flow control rules.
 - is engaged in a Dedicated Connection, the N_Port response is unpredictable.
 - If a N_Port which supports only Class 1 receives a Class 3 frame, and
 - is not engaged in a Dedicated Connection, the N_Port shall discard the frame and obey the buffer-to-buffer flow control rules.
 - is engaged in a Dedicated Connection, the N_Port shall discard the frame and not follow the buffer-to-buffer flow control rules.
 - If a F_Port which supports only Class 1 receives a Class 2 frame, and
 - is not engaged in a Dedicated Connection, the F_Port shall issue a F_RJT with appropriate Class 2 delimiters and obey buffer-to-buffer flow control rules.
 - is engaged in a Dedicated Connection, the F_Port response is unpredictable.
 - If a F_Port which supports only Class 1 receives a Class 3 frame, and
 - is not engaged in a Dedicated Connection, the F_Port shall discard the frame and obey buffer-to-buffer flow control rules.
 - is engaged in a Dedicated Connection, the F_Port response is unpredictable.
- o) If a N_Port does not support Class 6 (i.e., does not support Class 1) and receives a Class 6/SOFc1 frame, the N_Port shall issue a P_RJT with a SOFn1 and EOFdt with a reason code of Class not supported. If a F_Port does not support Class 6 and receives a Class

6/SOF_{c1} frame, the F_Port shall issue an F_RJT with a SOF_{n1} and EOF_{dt} with a reason code of Class not supported.

- p) If a F_Port, not engaged in a Dedicated Connection, receives a frame with a SOF_{i1} or SOF_{n1}, the F_Port response is unpredictable. However, the buffer-to-buffer control shall remain unaffected. (Invalid protocol)
- q) The effect of preemption of one destination N_Port on the remaining N_Ports is implementation-dependent.

16.6.3 Class 6 delimiters

A Dedicated Connection is requested by transmitting a Data frame using an SOF_{c1} delimiter. SOF_{c1} initiates the first Sequence; subsequent Sequences are initiated with a SOF_{i1}. SOF_{n1} starts all frames other than the first within a Sequence.

EOF_n terminates all frames other than the last frame within a Sequence. Each Sequence is terminated using an EOF_t (see 16.1.5). An EOF_{dt} contained in a frame terminates the Sequence in which the frame is sent and it also serves to remove the Dedicated Connection. Other open Sequences in progress are also terminated. Exchanges and Sequences may be left in indeterminate state from the perspective of ULPs.

16.6.4 Class 6 frame size

The size of Data_Field of a frame using the SOF_{c1} delimiter is limited by the smaller value of the maximum Data_Field size supported for frames with SOF_{c1} by the Fabric and the destination N_Port. Subsequent frames, after a Dedicated Connection is established, are limited only by the maximum Data Field size supported by the destination N_Port.

16.6.5 Class 6 flow control

ACK frames are used to perform Class 6 end-to-end flow control. ACK frames are started with SOF_{n1}. Destination N_Ports shall not terminate Sequences or Connections. All ACK frames shall end with EOF_n.

All Class 6 frames shall follow end-to-end flow control rules (see 20.4.1). The Class 6/SOF_{c1} frame shall follow both end-to-end and buffer-to-buffer flow control rules (see 20.5.1).

16.6.6 Stacked Connect-requests

Stacked connect-requests is a feature which may be provided by the Fabric (see 22.5.2).

17 Login and Service Parameters

17.1 Introduction

The Login procedure is a method by which a N_Port establishes its operating environment with a Fabric, if present, and other destination N_Ports with which it communicates. Fabric Login and destination N_Port_Login are both accomplished with a similar procedure using different Destination_Identifiers (D_IDs) and possibly different Source_Identifiers (S_IDs).

Login between a N_Port and the Fabric or between two N_Ports is long-lived. The number of concurrent N_Ports with which a N_Port may be logged in with is a function of the N_Port facilities available. There is no one to one relationship between Login and Class 1 Dedicated Connections.

Explicit Login is accomplished using the Login (FLOGI/PLOGI) Link Service Sequence within a separate Exchange to transfer the Service Parameters (contained in the Payload) of the N_Port initiating the Login Exchange. The Accept (ACC) contains the Service Parameters of the Responder (contained in the Payload).

Implicit Login is a method of defining and specifying the Service Parameters of destination N_Ports by means other than the explicit use of the Login Exchange. Specific methods of implicit Login are not defined in FC-FS.

When Login is referred to throughout other sections of this document, either the explicit or implicit procedure is acceptable. Implicit Login is assumed to provide the same functionality as Explicit Login. Default Login values are specified in 15.1.1. Explicit Login replaces previous Service Parameters and initializes end-to-end or buffer-to-buffer credit, or both. The Login procedure shall follow the Exchange and Sequence management rules as specified in clause 18. Frames within a Sequence shall operate according to R_RDY Primitive, ACK, and Link_Response rules specified in clause 14.

Explicit Fabric Login is performed during the initialization process under the assumption that a Fabric is present. The first explicit Login is directed to the Fabric using the well-known Fabric address (i.e., F_Port at hex 'FF FF FE'). It is mandatory for all Fabric types to support the explicit Login procedure. In its first attempt at explicit Fabric Login, a N_Port shall set the following S_ID:

- Bits 23-0 equal to binary zeroes (denoted as 0), or
- Bits 23-8 equal to binary zeroes and bits 7-0 equal to a model-dependent value (denoted as hex '00 00 yy').

If the N_Port receives an F_RJT with a reason code of Invalid S_ID, it may use its last known native address identifier in the FLOGI Sequence as its S_ID.

Implicit Fabric Login, unlike Explicit Fabric Login, does not require the support, within a N_Port, of the FLOGI Link Service request and reply. Implicit Fabric Login may be supported in a variety of ways, requiring varying levels of support within a N_Port.

Following are some examples of Implicit Fabric Login Support:

- In some environments (e.g., a simple, small Fabric), the fabric parameters may be well known. In this case, a N_Port may only require an internal configuration table to support the requirement of Implicit Fabric Login.
- In some environments (e.g., a point to point topology), a N_Port may require no knowledge of FC-SW parameters. In this case, an N_Port may not even require an internal configuration table to support the requirement of Implicit Login.

The above examples do not preclude other implementations of Implicit Fabric Login that satisfy FC-FS.

Login with the Fabric provides the N_Port with Fabric characteristics for the entire Fabric as defined in the Fabric's Service Parameters. The Service Parameters specified by the N_Port provide the Fabric with information regarding the type of support the N_Port requests. The Service Parameters provided by a N_Port also include a 64-bit N_Port_Name and 64-bit Node_Name. The Service Parameters provided by a F_Port also include a 64-bit F_Port_Name and 64-bit Fabric_Name. During the Fabric Login procedure the Fabric may optionally define the N_Port's native Identifier within a system configuration. If the Fabric does not support native N_Port Identifier assignment, the N_Port shall assign its own native N_Port Identifier by another method not defined in FC-FS.

Destination N_Port_Login (PLOGI) provides each N_Port with the other N_Port's Service Parameters. Knowledge of a destination N_Port's receive and transmit characteristics is required for data Exchanges. Service Parameters of destination N_Ports are saved and used when communication with those N_Ports is initiated. The Service Parameters interchanged between two N_Ports may be asymmetrical. Saving the Service Parameters of destination N_Ports with which a N_Port communicates requires N_Port resources. These resources can be released using the destination N_Port Logout procedure (see 17.5).

A N_Port may transmit the Link Credit Reset command to a destination N_Port in order to reclaim end-to-end Credit for outstanding Data frames in Class 2.

When a N_Port receives a Login from a N_Port, all other Active and Open Sequences with the N_Port performing Rlogin are abnormally terminated prior to transmission of the Accept Sequence to the Rlogin request.

17.2 Applicability

Login with the Fabric is required for all N_Ports, regardless of Class of Service supported. Communication with other N_Ports shall not be accepted until the Fabric Login procedure is complete. For a N_Port, which supports Class 1 or Class 2 Service, a N_Port is required to Login with each N_Port with which it intends to communicate (destination N_Port_Login).

NOTE - It is not required that a N_Port provide the same Login information with each destination N_Port or with the Fabric. However, a N_Port should avoid using contradictory or conflicting parameters with different Login destinations.

17.3 Fabric Login

Fabric Login accomplishes five functions

- It determines the presence or absence of a Fabric.
- If a Fabric is present, it provides a specific set of operating characteristics associated with the entire Fabric.
- If a Fabric is present, it shall optionally assign or shall confirm the native N_Port Identifier of the N_Port that initiated the Login.
- If a Fabric is not present, an Accept with the specification of N_Port in Common Service Parameters indicates that the requesting N_Port is attached in a point-to-point topology.
- If a Fabric is present, it initializes the buffer-to-buffer Credit.

Class 4 is not applicable to Fabric Login and shall not be used for Fabric Login. The Class 4 and 6 N_Port_Login service parameters used during Fabric Login (in FLOGI or ACC to FLOGI) shall be as shown in Table 202. The Fabric shall remove all Class 4 and 6 connections to a N_Port when the N_Port performs a Fabric Login.

17.3.1 Explicit Fabric Login

The explicit Fabric Login procedure shall require transmission of a Login (FLOGI) Link Service Sequence within an Exchange with an assigned OX_ID by a N_Port with a Destination Identifier (D_ID) of the well-known F_Port address (FFFFFE) and a Source Identifier (S_ID) of either 0 or 0000yy.

When S_ID = 0 is used, the F_Port shall either:

- assign a Fabric unique N_Port Identifier to the N_Port in the ACC reply Sequence, or
- return an F_RJT with a reason code indicating Invalid S_ID, if the Fabric does not support N_Port Identifier assignment. The N_Port shall assign its native N_Port Identifier by another method not defined in FC-FS and retry the FLOGI Sequence with an S_ID = X.

When S_ID = 0000yy is used, the F_Port shall either:

- assign a Fabric unique N_Port Identifier to the N_Port in the ACC reply Sequence by setting bits 23-8 and validating bits 7-0, or
- return an F_RJT with a reason code indicating Invalid S_ID, if the value in bits 7-0 does not allow a Fabric unique Identifier to be generated. The N_Port shall assign a different value for yy and retry the FLOGI Sequence with an S_ID = X.

If S_ID = X is used, the F_Port shall either:

- return an ACC reply Sequence with D_ID = X (confirming identifier), or
- return an F_RJT with a reason code indicating Invalid S_ID, if X is invalid.

If the F_Port has rejected both S_ID = 0 or 0000yy and S_ID = X, the N_Port shall attempt to Relogin with another value of X, or determine a valid value of X by a method not defined in FC-FS.

The Payload of this Link Service Sequence contains the Service Parameters of the N_Port transmitting the FLOGI Sequence. The N_Port transmits Service Parameters as specified for F_Port Login which are defined in 17.6.2 and 17.6.7. Only those parameters associated with Fabric Login are specified (see Table 204).

The normal reply Sequence to a FLOGI Link Service Sequence by a F_Port is an Accept (ACC) Link Service Sequence within an Exchange with the OX_ID of the Login request and F_Port assigned RX_ID with a Destination Identifier (D_ID) containing the N_Port Identifier assigned to the N_Port by the Fabric and a Source Identifier (S_ID) of the well-known F_Port address (FFFFFE). The Payload of the ACC contains the Service Parameters of the Fabric. Fabric Service Parameters are defined in 17.7.

The N_Port Common Service Parameters during Fabric Login are specified in 17.6.2. The N_Port Class Service Parameters during Fabric Login are specified in 17.6.7. The F_Port Common Service Parameters specified in the Accept during Fabric Login are specified in 17.7.1. The F_Port Class Service Parameters specified in the Accept during Fabric Login are specified in 17.7.4.

17.3.2 Responses to Fabric Login

The following meanings are associated with Table 197 and Table 198 and:

- the Response/Reply Seq column identifies R_RDY, a Link_Control response, or a Link Service frame transmitted in response to the FLOGI Sequence directed to the well-known F_Port address (FFFFFE). More than one frame is possible in response.
- the D_ID and S_ID columns specify the value of the corresponding field in the response frame received by the N_Port transmitting the FLOGI Sequence.

- the Indication column provides a short summary of the possible conditions associated with a particular response.
- the N_Port Action column specifies the action for the N_Port transmitting the FLOGI Sequence to take based on the response received.

17.3.2.1 FLOGI with S_ID = 0

Table 197 describes the set of possible responses to Fabric Login with an S_ID = 0 and a D_ID of hex 'FF FF FE'. Further description of the response primitive or frame is found in clause 14.

Table 197 - Responses to FLOGI sequence (S_ID = 0 or 0000yy) - Fabric Login				
Response/ Reply Seq	D_ID	S_ID	Indication	N_Port Action
a) R_RDY	N/A	N/A	- Class 1(SOFc1) - Class 2 or 3 - successful frame delivery to F_Port or N_Port	- Wait for frame
b) Last ACK	0 or 0000yy	FFFF FE	FLOGI sequence has been received by N_Port or F_Port	- Wait for Reply Data frame Sequence
c) ACC	X or XXXXyy	FFFF FE	- OX_ID = FLOGI OX_ID - If Common Serv = F_Port, Fabric Login complete	- Perform destination N_Port_Login (see 17.4.2.1) (Fabric Present)
d) ACC	0 or 0000yy	FFFF FE	OX_ID = FLOGI OX_ID - If Common Serv = N_Port no Fabric present	- Perform point-to-point destination N_Port_Login (see 17.4.2.1)
e) F_BSY or P_BSY	0 or 0000yy	FFFF FE	Busy	- retry later
f) F_RJT or P_RJT	0 or 0000yy	FFFF FE	- reason code = Class not supported = Invalid S_ID = Other	- FLOGI-next Class (S_ID = 0 or 0000yy) - FLOGI with S_ID = X or different yy - respond accordingly
g) FLOGI	FFFF FE	0 or 0000yy	- Collision with other N-Port	- respond with ACC Common Serv = N_Port - compare N_Port_Names - if xmit P_Name > rec'd P_Name End Connection if Class 1 then initiate point-to-point destination N_Port_Login - if xmit P_Name < rec'd P_Name End Connection if Class 1 then wait for point-to-point destination N-Port Login
h) LS_RJT	X or XXXXyy	FFFF FE	- Fabric present - Reason code	- Relogin with altered Service Parameters, use D_ID of LS_RJT
i) None			- error	- perform ABTS after Sequence timeout

These responses are characterized by the following:

- a) Response 1 is possible from a N_Port or Fabric.
- b) Response 2 is from a Fabric or a N_Port. The D_ID and S_ID values (in the response to the FLOGI sequence) correspond to the values in the FLOGI fields, respectively, in the FLOGI sequence (also for responses 5 and 6).
- c) Response 3 completes Fabric Login. The N_Port S_ID is assigned as X or XXXXyy.
- d) Response 4 indicates a point-to-point topology with another N_Port, which is determined by examining the Common Service Parameter, which specifies N_Port or F_Port. Based on comparison of Port_Names, either transmit PLOGI, or wait for PLOGI.
- e) Response 5 indicates that either the Fabric or N_Port is busy, retry later.
- f) Response 6 indicates a Fabric or N_Port reject. If Class is not supported, retry Login with another Class with a numerically higher value. If the reason code is invalid S_ID, then retry FLOGI with a value of X (see 17.3.2.2). For other reject reasons, the N_Port shall respond accordingly.
- g) Response 7 indicates a point-to-point attachment and a collision with an FLOGI from the attached N_Port. The N_Port shall respond with an ACC. The Common Service Parameters shall contain the same information as FLOGI but shall indicate that a N_Port is transmitting the Data. Port_Name and Node_Name shall be included, but all Classes of Service shall be indicated as invalid. The N_Port shall compare the Port_Name received to the Port_Name it transmitted. If this N_Port's value is lower, it shall end this Exchange and wait for a PLOGI from the attached N_Port. In Class 1, if this N_Port's value is lower, it shall become the Connection Recipient (see clause 22) for this Connection. In Class 1, if its value is higher, it shall become the Connection Initiator for this Connection. If its value is higher, it shall transmit a PLOGI (see 17.4.2.3) as part of a new Dedicated Connection. The Dedicated Connection associated with FLOGI shall be removed by the normal method to remove a Dedicated Connection in Class 1. See 17.6.4 for a description of N_Port_Names. See 17.4.2.3 for a description of point-to-point destination N_Port_Login.
- h) Response 8 indicates that the Login request is being rejected for a reason specified in the LS_RJT frame. The FLOGI request may be retried if the appropriate corrective action is taken.
- i) Response 9 indicates that a Link error has occurred.

NOTE - When a N_Port originates an Exchange using a N_Port Identifier of unidentified (binary zeros or 0000yy), its N_Port Identifier may change between transmitting a request Sequence (Login) and receiving the reply Sequence (Accept).

17.3.2.2 FLOGI with SID = X

Table 198 describes the set of possible responses to Fabric Login with an S_ID = X. The FLOGI Sequence transmitted contains a D_ID of the well-known F_Port address (FFFFFFE) and an S_ID of X. It is known that a Fabric is present before this Fabric Login is attempted.

Table 198 - Responses to FLOGI sequence (S_ID = X) - Fabric Login				
Response/ Reply Seq	D_ID	S_ID	Indication	N_Port Action
a) R_RDY	N/A	N/A	- Class 1(SOFc1) - Class 2 or 3 successful frame delivery to F_Port	wait for frame
b) ACK_1	X	FFFFFFE	FLOGI sequence has been received	Wait for Reply Data frame Sequence
c) ACC	X	FFFFFFE	- OX_ID = FLOGI OX_ID	Perform destination N_Port_Login

Table 198 - Responses to FLOGI sequence (S_ID = X) - Fabric Login				
Response/ Reply Seq	D_ID	S_ID	Indication	N_Port Action
			- Fabric Login complete - - Address Identifier = X	
d) F_BSY	X	FFFFFFE	- - Fabric Busy	retry later
e) F_RJT	X	FFFFFFE	- Reason code - = Invalid S_ID - = other	FLOGI with S_ID = different X respond accordingly
f) LS_RJT	X	FFFFFFE	- reason code	FLOGI with altered Service Parameters
g) None			- error	perform ABTS after Sequence timeout

These responses are characterized by the following:

- a) Response 1 is possible from a Fabric.
- b) Response 2 is from a Fabric. The D_ID and S_ID values (response to the FLOGI sequence) correspond to the values in the FLOGI fields, respectively, in the FLOGI sequence (also for responses 4 and 5).
- c) Response 3 completes Fabric Login. The N_Port S_ID is confirmed as X.
- d) Response 4 indicates that the Fabric is busy.
- e) Response 5 indicates a Fabric reject. If Class is not supported, retry Login with another Class, with a numerically higher value. If the reason code is S_ID invalid, then retry FLOGI with a different value of X. For other reject reasons, the N_Port shall respond accordingly.
- f) Response 6 indicates that the Login request is being rejected for a reason specified in the LS_RJT frame. The FLOGI request may be retried if the appropriate corrective action is taken.
- g) Response 7 indicates that a Link error has occurred.

17.3.3 SOF delimiters

Since the Fabric may not support all three Classes of Service, the FLOGI Sequence may require retry of the FLOGI Sequence with a different SOF delimiter for each of the following Classes in 1, 2, and 3 order:

- a) Class 1 - SOFc1 (SOFi1)
- b) Class 2 - SOFi2
- c) Class 3 - SOFi3

Selection of the SOF delimiter for the FLOGI Sequence is based on the Classes of Service supported by the originating N_Port. Since the F_Port may not support all Classes of Service, the FLOGI Sequence may have to be retried with SOF delimiters for different Classes of Service. The FLOGI Sequence is transmitted and the appropriate action is specified in Table 197 or Table 198. If a Fabric Reject (F_RJT) with reason code "Class of Service not supported by entity at FFFFFFFE" has been received, another supported delimiter is attempted until the Login procedure is complete or until all supported delimiter types have been attempted. If an Accept Reply Sequence (ACC) is received, the validity bits in the Accept Sequence indicate which Classes of Service are supported by the Fabric.

NOTE - Class 1 communication requires that both N_Ports operate at the same speed. Hence, N_Port_Login should be performed in Class 1 if that Class is supported. Class 2 or 3 communication does not require that both N_Ports operate at the same speed. Link speed is not a Login parameter. A mismatch in Class 1 is indicated by a F_RJT with a reason code of "Fabric path not available."

If all supported delimiter types have been attempted and the Fabric has rejected all or timed out, the Fabric and N_Port are incompatible and outside intervention is required.

17.3.4 Frequency

Login between a N_Port and the Fabric should be long-lived. If Implicit Logout with the Fabric has occurred, it is necessary to Rlogin with the Fabric (see 17.5.3).

17.3.4.1 Login completion - Originator

The Originator of the FLOGI request considers Login to have ended when

- in Class 1, the Originator has transmitted the ACK (EOF_t or EOF_{dt}) to the Accept, or
- in Class 2, the Originator has received the R_RDY in response to transmitting the ACK (EOF_t) to the Accept, or
- in Class 3, the Originator has transmitted the R_RDY in response to the Accept.

When Login is ended, the values of buffer-to-buffer and end-to-end Credit are initialized.

17.3.4.2 Login completion - Responder

The Responder of the FLOGI request considers Login to have ended when

- in Class 1, the Responder has received the ACK (EOF_t or EOF_{dt}) to the Accept, or
- in Class 2, the Responder has transmitted the R_RDY in response to receiving the ACK (EOF_t) to the Accept, or
- in Class 3, the Responder has received the R_RDY in response to the Accept.

When Fabric Login has ended successfully, the values of buffer-to-buffer and end-to-end Credit are initialized.

NOTE - Since the N_Port has not yet performed Fabric Login, it can not be doing anything else. Therefore R_RDY can be assumed to be part of the Fabric Login protocol.

17.4 N_Port_Login

Destination N_Port_Login proceeds following the Fabric Login procedure. Events 4, 7, 8, and 9 in Table 197 indicate a point-to-point attachment and the N_Port shall proceed based on the information provided by the N_Port action for each event. If a Fabric is present, as determined by performing the Fabric Login procedure, a N_Port proceeds with destination N_Port_Login according to 17.4.2.1. If a Fabric is not present, as determined by performing the Fabric Login procedure, a N_Port proceeds with destination N_Port_Login according to 17.4.2.3. Destination N_Port_Login between two N_Ports is complete when each N_Port has received the Service Parameters of the other N_Port. This may be accomplished by either implicit or explicit destination N_Port_Login.

The N_Port Common Service Parameters during N_Port_Login are specified in 17.6.3. The N_Port Class Service Parameters during N_Port_Login are specified in 17.6.8. Both the Common Service Parameters and Class Service Parameters apply to each N_Port during destination N_Port_Login.

NOTE - When a N_Port (A) receives a PLOGI from another N_Port (B), N_Port (A) should verify that it is not already logged in with a N_Port (c) with the same Port_Name but different N_Port native address identifier and Node_Name. If so, it should consider the prior Login to be ended and should follow the Logout rules before accepting the new Login. Such a situation may arise if configuration changes have occurred.

The Class 4 default N_Port_Login service parameters prior to N_Port_Login shall be as shown in Table 202.

End-to-end N_Port_Login shall not affect the state of Class 4 connections.

17.4.1 Address Identifiers

A N_Port determines its own native N_Port Identifier through explicit or implicit Login by

- the Fabric, if present,
- implicit definition,
- assignment in the PLOGI Sequence transmitted to a destination N_Port attached in a point-to-point topology.

Address identifiers of other destination N_Ports with which a N_Port wishes to Login with may be collected from

- the Fabric, if present,
- a name-server function,
- implicit definition, or
- an alternate initialization procedure.

17.4.2 Explicit N_Port_Login

Explicit N_Port_Login accomplishes three functions:

- It provides a specific set of operating characteristics associated with the destination N_Port.
- Initializes the destination end-to-end Credit.
- In point-to-point topology, buffer-to-buffer Credit is initialized.

A well-behaved N_Port shall Logout with another N_Port prior to initiating a Rlogin. However, if a N_Port receives or transmits a Login Link Service request Sequence with another N_Port, it shall respond to any other Exchanges with that N_Port as though a Logout had been previously performed. During the N_Port_Login procedure, other communication with the destination N_Port shall not be initiated or accepted. Once the N_Port_Login procedure has been successfully completed, communication between the N_Ports may be initiated or accepted. That is, if N_Port(A) performs a PLOGI request with N_Port(B) and N_Port(B) transmits the ACC reply Sequence, then either N_Port(A) or N_Port(B) may initiate communication for other protocols. N_Port(B) shall not be required to transmit a PLOGI request Sequence to N_Port(A) unless it wishes to invalidate or alter the existing Login parameters.

The optional support of Class 1 and Buffered Class 1 by N_Ports and F_Ports results in various communication capabilities between them:

- A given N_Port pair may have neither buffered nor unbuffered Class 1 communication capability.
- A given N_Port pair may have the capability to communicate using only unbuffered Class 1.
- A given N_Port pair may have the capability to communicate using only buffered Class 1.
- A given N_Port may have the capability to communicate using Buffered or unbuffered Class 1 with different N_Ports.

The optional support for Class 4 by N_Ports and F_Port means that a pair of N_Ports may be able to communicate using Class 4.

For these reasons, the order of Classes tried for N_Port_Login specified in 17.4.3 is modified as follows:

- If permitted by FLOGI, PLOGI shall be tried first using unbuffered Class 1 Service.
- If unsuccessful, PLOGI shall be attempted, FLOGI permitting, in Buffered Class 1 Service.
- If that is also unsuccessful, PLOGI shall be attempted, FLOGI permitting, in Class 2 Service.
- If unsuccessful, PLOGI shall be attempted, FLOGI permitting, in Class 3 Service.
- If unsuccessful, and permitted by FLOGI and if a Class 4 circuit can be established with the destination N_Port, PLOGI shall be attempted using Class 4 Service. If this attempt is also unsuccessful, then no communication is possible between the two N_Ports.

17.4.2.1 Fabric present

The destination N_Port explicit Login procedure requires transmission of a Login (PLOGI) Link Service Sequence within an Exchange with an assigned OX_ID with a Destination Identifier (D_ID = Y) of the destination N_Port and a Source Identifier (S_ID = X) of originating N_Port. The Payload of this Sequence contains the Service Parameters of the N_Port originating the PLOGI Sequence. N_Port Service Parameters are defined in 17.6.

The normal reply Sequence to a PLOGI Link Service Sequence by a N_Port is an Accept (ACC) Link Service Sequence within the Exchange identified by the OX_ID of the Login Sequence and the RX_ID assigned by the Responder with a Destination Identifier (D_ID) of the originating N_Port (PLOGI Sequence) and a Source Identifier (S_ID) of the responding N_Port. The Payload of the ACC contains the Service Parameters of the responding N_Port.

If a collision occurs, such that N_Port (A) has transmitted a PLOGI to N_Port (B) and N_Port (A) receives a PLOGI from N_Port (B) before receiving the ACC from N_Port (B), N_Port (A) shall respond as though its PLOGI had never been transmitted. There shall be no special processing required.

17.4.2.2 Responses to N_Port_Login (Fabric)

See 17.3.2 for a description of the column meanings. The entries in Table 199 are based on previous Login with a Fabric. Table 199 describes the set of possible responses during destination N_Port_Login with a Fabric present.

Table 199 - Responses to PLOGI frame - N_Port_Login (Fabric present)				
Response/ Reply Seq.	D_ID	S_ID	Indication	N_Port Action
a) R_RDY	N/A	N/A	- Class 1(SOFc1) - Class 2 or 3 only - successful frame delivery to F_Port	wait for frame
b) ACK_1	X	Y	- PLOGI frame has been received	Wait for Reply Data frame Sequence
c) ACC.	X	Y	- OX_ID = PLOGI OX_ID - Login complete	End
d) F_BSY	X	Y	- Fabric present - Fabric busy	retry later
e) F_RJT	X	Y	- Fabric present	

Table 199 - Responses to PLOGI frame - N_Port_Login (Fabric present)				
Response/ Reply Seq.	D_ID	S_ID	Indication	N_Port Action
			- reason code = invalid D_ID = other	- determine Port-ID for Y - reattempt Login - respond accordingly
f) P_BSY	X	Y	- N_Port busy	- retry later
g) P_RJT	X	Y	- reason code = Class not supported = other	- select next Class - respond accordingly
h) LS_RJT	X	Y	- reason code = invalid parameters = other	- correct parameters - respond accordingly
i) None			- error	- perform ABTS after Sequence timeout

These responses are characterized by the following:

- a) Response 1 is possible from a Fabric.
- b) Response 2 is from the N_Port. The D_ID and S_ID values (in the N_Port response to the PLOGI frame) correspond to the values in the PLOGI fields, respectively, in the PLOGI frame (also for responses 6, 7, and 8).
- c) Response 3 completes N_Port_Login.
- d) Response 4 indicates that the Fabric is busy.
- e) Response 5 indicates a Fabric reject. The N_Port shall respond according to the reject reason code specified.
- f) Response 6 indicates that the destination N_Port is busy.
- g) Response 7 indicates a N_Port reject. The N_Port shall respond according to the action and reason code of the P_RJT.
- h) Response 8 indicates that the Login request is being rejected for a reason specified in the LS_RJT frame. The PLOGI request may be retried if the appropriate corrective action is taken.
- i) Response 9 indicates that a Link error has occurred.

17.4.2.3 No Fabric present (point-to-point)

This procedure is based on the assumption that response 4 or 7 in Table 197 was received during an attempted Fabric Login. The destination N_Port explicit Login procedure requires transmission of a Login (PLOGI) Link Service Sequence within an Exchange with an assigned OX_ID.

Since the detection of a point-to-point topology was determined by reception of an ACC or FLOGI with Common Service Parameters indicating a N_Port is directly attached,

- the N_Port waits for a PLOGI from the other N_Port if its N_Port_Name is less than the attached N_Port_Name, or

- the N_Port transmits a PLOGI with S_ID = X and D_ID = Y ($X \neq Y$) and no collision with PLOGI is anticipated if its N_Port_Name is greater than the attached N_Port_Name.

The Payload of this frame contains the Service Parameters of the N_Port originating the PLOGI Sequence. N_Port Service Parameters are defined in 17.6. The Common Service Parameters are specified for point-to-point N_Port_Login.

The normal reply Sequence to a PLOGI Link Service frame by a N_Port is an Accept (ACC) Link Service Sequence with the OX_ID of the Login Sequence and the assigned RX_ID of the Responder. The Payload of the ACC contains the Service Parameters of the responding N_Port.

17.4.2.4 Responses to N_Port_Login (point-to-point)

See 17.3.2 for a description of the column meanings. Table 200 describes the set of possible responses to destination N_Port_Login for a point-to-point configuration.

Table 200 - Responses to PLOGI frame - N_Port_Login (No Fabric, point to point)				
Response/ Reply Seq.	D_ID	S_ID	Indication	N_Port Action
a) R_RDY	N/A	N/A	- Class 1(SOFc1) - Class 2 or 3 only - successful frame delivery to N_Port	- wait for frame
b) ACK_1	X	Y	- PLOGI frame has been received	- Wait for Reply Data frame Sequence
c) ACC	X	Y	- Login complete - OX_ID = PLOGI OX_ID	- End
d) P_BSY	X	Y	- N_Port busy	- retry later
e) P_RJT	X	Y	- reason code = Class not supported = other	- select next Class - respond accordingly
f) LS_RJT	X	Y	- reason code = invalid parameters = others	- correct parameters - respond accordingly
g) PLOGI	R	Z	- Collision with other N_Port	- compare N_Port-Names - if xmit P_Name > rec'd P_Name respond with LS_RJT, - if xmit P_Name < rec'd P_Name process PLOGI from N_Port respond with ACC or LS_RJT
h) None			- Error	- perform ABTS after Sequence timeout

These responses are characterized by the following:

- Response 1 is possible from the N_Port.
- Response 2 is from the N_Port. The D_ID and S_ID values correspond to the values in the PLOGI Sequence (also for responses 4 and 5).
- Response 3 completes N_Port_Login.

- d) Response 4 indicates that the destination N_Port is busy.
- e) Response 5 indicates a N_Port reject. The N_Port shall respond according to the action and reason code of the P_RJT.
- f) Response 6 indicates that the Login request is being rejected for a reason specified in the LS_RJT frame. The PLOGI request may be retried if the appropriate corrective action is taken.
- g) Response 7 indicates that the other N_Port that is attached in a point-to-point attachment has issued a N_Port_Login. After comparing N_Port_Names, the N_Port shall issue an LS_RJT (Login already in progress) and transmit N_Port_Login (if not previously transmitted), or process the N_Port PLOGI.
- h) Response 8 indicates that a Link error has occurred.

Following reception of an ACC to FLOGI indicating a N_Port attached in a point-to-point topology, only one N_Port should be transmitting PLOGI. The S_ID and D_ID values of PLOGI shall be non-zero values that shall be used until implicit or explicit Logout.

17.4.3 SOF delimiters

Since the destination N_Port may support any of three Classes of Service, the PLOGI Sequence may require retransmission with a different SOF for each Class in the same manner described for Fabric Login (see 17.3.3). Login is only valid for the class used to Login and all Classes with a higher numerical value (i.e., Class 1, 2, and 3).

17.4.4 Frequency

The frequency of destination N_Port_Login is installation dependent based on the frequency of configuration changes that may alter the N_Port Identifiers within an installation. Service Parameters of other N_Ports are retained until the next destination N_Port_Login or until N_Port Logout is performed.

17.4.4.1 Login completion - Originator

The Originator of the PLOGI request considers Login to have ended when

- in Class 1, the Originator has transmitted the ACK (EOF_t or EOF_{dt}) to the Accept, or
- in Class 2, the Originator has received the Accept and transmitted the ACK (EOF_t) to the Accept, or
- in Class 3, the Originator has received the Accept.

When N_Port_Login is ended with a Fabric present, the value of end-to-end Credit is initialized. When N_Port_Login is ended in a point-to-point topology, the values of buffer-to-buffer and end-to-end Credit are initialized.

17.4.4.2 Login completion - Responder

The Responder of the PLOGI request considers Login to have ended when

- in Class 1, the Responder has received the ACK (EOF_t or EOF_{dt}) to the Accept, or
- in Class 2, the Responder has received the ACK (EOF_t) to the Accept, or
- in Class 3, the Responder has transmitted the Accept.

When N_Port_Login is ended with a Fabric present, the value of end-to-end Credit is initialized. When N_Port_Login is ended in a point-to-point topology, the values of buffer-to-buffer and end-to-end Credit are initialized.

17.4.5 N_Port_Login frame flow

See Figure D.5, Figure D.6, and Figure D.7 for examples of frame flow for destination N_Port_Login for Classes 1, 2, and 3.

17.5 Logout

17.5.1 Introduction

The destination Logout procedure provides a method for removing service between two N_Ports. Logout releases resources associated with maintaining Service with a destination N_Port. There is no explicit Logout procedure for the Fabric, however, implicit Logout may occur between a N_Port and the Fabric (see 17.5.3).

NOTE - Explicit Fabric Logout is not needed since the Fabric has no resources dedicated to a N_Port, which could be usefully made available.

17.5.2 Explicit N_Port Logout

Logout is accomplished by transmitting a Logout (LOGO) Link Service request Sequence to a destination N_Port. The Logout procedure is complete when the responding N_Port transmits an ACC Link Service reply Sequence.

If a N_Port desires to explicitly Logout, the initiating N_Port shall terminate other Active Sequences that it initiated with the destination N_Port prior to performing Logout, otherwise, the state of other Active Sequences is unpredictable. If a N_Port receives a Logout request while another Sequence is Active which was initiated from the requesting N_Port, it may reject the Logout request using an LS_RJT (Link Service Reject).

After an explicit Logout is performed with a N_Port, the default Login Service Parameters specified in 15.1.1 shall be functional if Login was explicit. After an explicit Logout is performed with a N_Port, the implicit Login Service Parameters shall be functional if Login was implicit.

17.5.3 Implicit Logout

If a N_Port receives or transmits the NOS or OLS Primitive Sequence, it shall be implicitly logged out from the Fabric, if present, or attached N_Port in a point-to-point topology. Fabric Login shall be performed following implicit Logout. Communication with other N_Ports shall not be accepted until the Fabric Login procedure is complete.

During Rlogin with the Fabric, if a N_Port's native Identifier has changed since the last Fabric Login, the N_Port shall not initiate or accept communication with other N_Ports until it has explicitly logged out with each N_Port and performed N_Port_Login.

During Rlogin with the Fabric, if the N_Port detects that the F_Port_Name has changed since the last Fabric Login, the N_Port shall wait an R_A_TOV timeout period before initiating or accepting communication with other N_Ports (i.e., implicit N_Port Logout). The timeout period shall start when the N_Port detects the change in F_Port_Name. After waiting the timeout period, the N_Port shall Logout and Rlogin other N_Ports to which it had been previously Logged-In.

If a N_Port receives OLS from the Fabric, the Fabric may be indicating configuration changes internal to the Fabric using the Online to Offline Protocol.

NOTE - If a N_Port is concerned that a partial Fabric Login may be in process using its link immediately preceding its attempted Fabric Login, it may wait an R_A_TOV in order to ensure that the response it receives from the F_Port during Fabric Login is associated with its Login request.

17.6 N_Port Service Parameters

Table 201 provides the payload format of the FLOGI or PLOGI Extended Link Service commands and their respective ACC replies.

Table 201 - FLOGI, PLOGI, or ACC payload	
Item	Size (Bytes)
LS_Command code	4
Common Service Parameters	16
Port_Name	8
Node or Fabric Name	8
Class 1 Service Parameters	16
Class 2 Service Parameters	16
Class 3 Service Parameters	16
Class 4 Service Parameters	16
Vendor Version Level	16
Services Availability (see note)	8
Reserved (see note)	116
NOTE - These fields are only present when the Payload Length bit (see 17.6.2.3) is set to 1. When the Payload Length bit is set to 0, these fields are not present in the payload, i.e., the payload is 132 bytes long.	

17.6.1 N_Port Common Service Parameters

Table 202 provides a summary of N_Port Common Service Parameters.

Table 202 - N_Port Common Service Parameter applicability										
Service Parameter	Word	Bits	N_Port_Login Class				Fabric Login Class			
			1	2	3	4 ⁵	1	2	3	4 ₅
FC_PH/FS Version	0	31-16								
Highest Version	0	31-24 ¹	y ¹	y	y	y	y	y	y	y
Lowest Version	0	23-16	y	y	y	y	y	y	y	y
Buffer-to-Buffer Credit	0	15-0	y	y	y	n ²	y	y	y	n
Common Features	1	31-16								
Continuously Increasing (C)	1	31	y	y	y	y	n	n	n	n
Random Relative Offset (O)	1	30	y	y	y	y	n	n	n	n
Valid Vendor Version Level (V)	1	29	y	y	y	y	y	y	y	y
N_Port/F_Port (N)	1	28	y	y	y	y	y	y	y	y

Table 202 - N_Port Common Service Parameter applicability

Service Parameter	Word	Bits	N_Port_Login Class				Fabric Login Class			
			1	2	3	4 ⁵	1	2	3	4 ₅
Alternative BB_Credit Management (A) see 20.5.10	1	27	y	y	y	y	y	y	y	y
E_D_TOV Resolution (see 17.6)	1	26	y	y	y	y	n	n	n	n
Clock Synchronization Primitive Capable (Y)	1	20	y	y	y	y	y	y	y	y
Dynamic Half Duplex (H)	1	18	y	y	y	y	y	y	y	y
SEQ_CNT (S) (see 17.6.3.3)	1	17	y	y	y	y	n	n	n	n
Payload Length (P) (see 17.6.2.3)	1	16	y	y	y	y	y	y	y	y
BB_SC Number	1	15-12	y	y	y	n	y	y	y	n
Buffer-to-Buffer Receive Data Field Size	1	11-0	y	y	y	y	y	y	y	y
N-Port Total Concurrent Sequences	2	31-16	y	y	y	y	n	n	n	n
Relative Offset by Info Category	2	15-0	y	y	y	y	n	n	n	n
E_D_TOV ^{3,4}	3	31-0	PTP	PTP	PTP	PTP	n	n	n	n
Notes:										
1) "y" indicates yes, applicable;										
2) "n" indicates no, not applicable										
3) PTP indicates applicable only for Point-to-Point										
4) R_A_TOV and E_D_TOV provided by the F_Port as Fabric Common Service Parameters during Fabric Login are not shown in this table.										
5) Class 4 (see clause 28)										

17.6.2 N_Port Common Service Parameters - Fabric Login

N_Port Common Service Parameters used during Fabric Login are shown in Figure 43.

Word 0	31	16	15		0
	FC-PH/FC-PH-2/FC-PH-3 /FC-FS Version		Buffer-to-buffer Credit (Pt to Pt)		
	HHHH HHHH	LLLL LLLL	BBBB BBBB	BBBB BBBB	
1	31	16	15	12	11
	Common Features		BB_SC Number	0	
	rrVN Arrr	rrrY rHrP	SSSS	Buffer-to-buffer Rcv Data Field size	
					FFFF FFFF FFFF
2	31	16	15		0
	Total Concurrent Sequences Reserved		Relative Offset by Info Category Reserved		
	rrrr rrrr	rrrr rrrr	rrrr rrrr	rrrr rrrr	
3	31	16	15		0
	Reserved		Reserved		
	rrrr rrrr	rrrr rrrr	rrrr rrrr	rrrr rrrr	

Figure 43 - N_Port Common Service Parameters - Fabric Login**17.6.2.1 FC-PH/FS version**

The FC-PH/FS version field provides a method of determining the version of FC-PH or FC-FS that a Port shall be capable of supporting. The FC-PH/FS field is divided into two one-byte fields. Table 203 indicates the hexadecimal values for the low and high FC-PH/FS version levels in word 0 bits 31-16.

Table 203 - FC-PH/FS Version - N_Port	
Hex Value	Version level
00	None
01 - 05	Reserved
06	4.0
07	4.1
08	4.2
09	4.3
0A - 0F	Reserved
10	FC-PH-2
11 - 1F	Reserved
20	FC-PH-3
21-2F	Reserved
30	FC-FS (this document)
31 - FF	Reserved

A version of FC-FS shall be assigned a new binary value when its implementation is not compatible with the previous version specified. A N_Port shall support all versions between the lowest and the highest levels specified.

- **Word 0, Bits 31 - 24 - Highest version (H)**

The binary value of bits 31-24 is encoded to represent the highest or most recent version of FC-PH or FC-FS, which the Port shall be capable of supporting.

- **Word 0, Bits 23 - 16 - Lowest version (L)**

The binary value of bits 23-16 is encoded to represent the lowest or earliest version of FC-PH or FC-FS that the Port shall be capable of supporting.

Where there is overlap in the FC-PH or FC-FS versions supported, each Port shall operate in the highest FC-PH or FC-FS version mutually supported.

17.6.2.2 Buffer-to-buffer Credit

The buffer-to-buffer Credit (B) field (Word 0, bits 15-0) specified shall be associated with the number of buffers available for holding Class 1 connect-request, Class 2, or Class 3 frames received from the F_Port.

The buffer-to-buffer Credit (B) shall be a single value which represents the total buffer-to-buffer Credit available for all Classes. A N_Port tracks buffer-to-buffer Credit as a single entity for all frames subject to buffer-to-buffer flow control (see clause 20).

17.6.2.3 Common features

- **Word 1, bits 31-30 - Reserved**
- **Word 1, bit 29 - Vendor Specific (V)**

0 = Not valid
1 = Valid

Word 1 bit 29 indicates that the Vendor Version Level (fifth 16-byte group) contains valid information. If Word 1 bit 29 = 0, it indicates that the Vendor Version Level field is not meaningful.

- **Word 1, Bit 28 - N_Port/F_Port (N)**

0 = N_Port
1 = F_Port

Word 1 bit 28 shall be set to zero.

- **Word 1, bit 27 - BB_Credit Management (A)**

0 = BB_Credit management specified in 20.5.1 through 20.5.9.
1 = Alternate BB_Credit management (see 20.5)

The BB_Credit Management field specifies the type of BB_Credit Management to be used.

- **Word 1, bits 26-21 – Reserved**
- **Word 1, bit 20 – Clock Synchronization Primitive Capable (Y)**

0 = not Clock Synchronization Primitive Capable
1 = Clock Synchronization Primitive Capable

Word 1, Bit 20 indicates support for the Primitive method of Clock Synchronization (see clause 32.3). If Word 1, Bit 20 = 0, then the N_Port does not support the Primitive method of Clock Synchronization. If Word 1, Bit 20 = 1 for the Clock Synchronization Server well-known address (hex 'FF FF F6'), the Clock Synchronization Server is capable of generating clock synchronization Primitive Signals. If Word 1, Bit 20 = 1 for any other N_Port, the N_Port is capable of receiving the clock synchronization Primitive Signals and acting upon them.

- **Word 1, bit 19 – Reserved**
- **Word 1, bit 18 - Dynamic Half Duplex (H)**

0 = DHD not supported
1 = DHD supported

Word 1, Bit 18 indicates support for Dynamic Half Duplex. If Word 1 Bit 18 = 0, then Dynamic Half Duplex is not supported. If Word 1 Bit 18 = 1, then the N_Port is indicating that it supports the reception of the DHD primitive.

NOTE - DHD is applicable to FC-AL topologies. See FC-AL-2 for a description of DHD behavior.

- **Word 1, bit 17 - Reserved**
- **Word 1, bit 16 - Payload Length (P)**

0 = 116 bytes
1 = 256 bytes

Word 1, Bit 16 indicates the length of the FLOGI payload. If Word 1 Bit 16 = 0, then the payload shall be 116 bytes. If Word 1 Bit 16 = 1, then the payload shall be 256 bytes.

17.6.2.4 BB_SC Number

- Word 1, bits 15-12 - Buffer-to-buffer State Change Number (BB_SC_N) (S)**

The BB_SC_N is a binary value (bits 15-12) which indicates that the sender of the PLOGI or FLOGI frame is requesting $2^{BB_SC_N}$ number of frames to be sent between two consecutive BB_SCs primitives, and $2^{BB_SC_N}$ number of R_RDY primitives to be sent between two consecutive BB_SCr primitives.

When the two ports performing the login specifies different non-zero values of BB_SC_N, the larger value shall be used. If either port specifies a BB_SC_N value of zero, the BB_Credit recovery process shall not be performed, and no BB_SCx primitive shall be sent. See 20.5.12 for a description of the BB_Credit recovery process.

Note - If all frames or R_RDY primitives sent between two BB_SCx primitives are lost, $2^{BB_SC_N}$ number of BB_Credits are lost, and cannot be recovered by the scheme outlined in 20.5.12. Therefore BB_SC_N should be chosen so that the probability of losing $2^{BB_SC_N}$ number of consecutive frames or R_RDY primitives is deemed negligible. The recommended value of BB_SC_N is 8.

17.6.2.5 Buffer-to-buffer Data_Field size

- Word 1, Bits 11 - 0 - Buffer-to-buffer Receive Size (F)**

The buffer-to-buffer Receive Data_Field Size is a binary value (bits 11-0) which specifies the largest Data_Field Size for an FT_1 frame (11.4 that can be received by the N_Port supplying the Service Parameters as a Sequence Recipient for:

- a connect-request (SOF_{c1}),
- a Class 2 Data frame, or
- a Class 3 Data frame.

Values less than 256 or greater than 2¹¹² are invalid. Values shall be a multiple of four bytes.

The buffer-to-buffer Receive Data_Field size is specified by FC-2.

17.6.3 N_Port Common Service Parameters - N_Port_Login

N_Port Common Service Parameters used during N_Port_Login are shown in Figure 44.

Word 0	31	16	15	0
	FC-PH/FS Version		Buffer-to-buffer Credit (Pt to Pt)	
	HHHH HHHH	LLLL LLLL	BBBB BBBB	BBBB BBBB
1	31	16	15	0
	Common Features		15 12 BB_SC Number	11 Buffer-to-buffer Rcv Data Field size
	COVN ARrr	rDrY rHSP	SSSS	FFFF FFFF FFFF
2	31	16	15	0
	Total Concurrent Sequences		Relative Offset by Info Category	
	rrrr rrrr	TTTT TTTT	DDDD DDDD	DDDD DDDD
3	31	16	15	0
	E_D_TOV (Pt-to-Pt)			
	tttt tttt	tttt tttt	tttt tttt	tttt tttt

Figure 44 - N_Port Common Service Parameters - N_Port_Login**17.6.3.1 FC-PH/FS version**

The FC-PH/FS version field provides a method of determining the version of FC-PH/FS, which a N_Port shall be capable of supporting. The FC-PH/FS field is divided into two one-byte fields. Table 203 indicates the hexadecimal values for the low and high FC-PH or FS version levels in word 0 bits 31-24, and bits 23-16.

A version of FC-FS shall be assigned a new binary value when its implementation is not compatible with the previous version specified. A N_Port shall support all versions between the lowest and the highest levels specified.

- **Word 0, Bits 31 - 24 - Highest version supported (H)**

The binary value of bits 31-24 is encoded to represent the highest or most recent version of FC-PH or FS, which the N_Port shall be capable of supporting.

- **Word 0, Bits 23 - 16 - Lowest version supported (L)**

The binary value of bits 23-16 is encoded to represent the lowest or earliest version of FC-PH or FS that the N_Port shall be capable of supporting.

Where there is overlap in the FC-PH versions supported, each N_Port shall operate in the highest FC-PH or FS version mutually supported.

17.6.3.2 Buffer-to-buffer Credit

Word 0, bits 15-0 shall only be meaningful for a N_Port in a point-to-point topology during Login between two N_Ports. In a topology other than point-to-point, word 0, bits 15-0 shall not be meaningful. The buffer-to-buffer Credit (B) field specified shall be associated with the number of buffers available for holding Class 1 connect-request, Class 2, or Class 3 frames received from the N_Port.

The buffer-to-buffer Credit (B) shall be a single value which represents the total buffer-to-buffer Credit available for all Classes. A N_Port tracks buffer-to-buffer Credit as a single entity for all frames subject to buffer-to-buffer flow control (see clause 20).

Values in the buffer-to-buffer Credit field have the same format as in the end-to-end Credit Class Service Parameters.

17.6.3.3 Common features

- **Word 1, Bit 31 - Continuously Increasing Offset (C)**

0 = not supported

1 = supported

Word 1, bit 31 = 1 indicates that the N_Port supplying this parameter shall be capable of supporting Continuously Increasing Relative Offset, if present (F_CTL bit 3), within a Sequence on a frame by frame SEQ_CNT basis. Bit 31 shall only be applicable to those Information Categories in which a N_Port supports Relative Offset (i.e., word 2, bits 15-0). See 3.1 and clause 21 for an explanation of Continuously Increasing Relative Offset.

Word 1, bit 31 support shall be applicable as a Sequence Initiator as well as a Sequence Recipient for all Classes of Service supported by the N_Port. However, support as a Sequence Initiator shall not require that transmitted Sequences use Relative Offset, even if both N_Ports provide such support.

- **Word 1, Bit 30 - Random Relative Offset (O)**

0 = not supported
1 = supported

Word 1, bit 30 = 1 indicates that the N_Port supplying this parameter shall be capable of supporting Random Relative Offset values, if present (F_CTL bit 3). Random values may increase, decrease, or otherwise fluctuate within a Sequence. Bit 30 shall only be applicable to those Information Categories in which a N_Port supports Relative Offset (i.e., word 2, bits 15-0). See 3.1 and clause 21 for an explanation of Random Relative Offset.

Word 1, bit 30 support shall be applicable as a Sequence Initiator as well as a Sequence Recipient for all Classes of Service supported by the N_Port. However, support as a Sequence Initiator shall not require that transmitted Sequences use Relative Offset, even if both N_Ports provide such support.

- **Word 1, Bit 29 - Valid Vendor Version Level (V)**

0 = not valid
1 = valid

Word 1, bit 29 = 1 indicates that the Vendor Version Level (fifth 16-byte group) contains valid information. If Word 1, bit 29 = 0, it indicates that the Vendor Version Level field is not meaningful.

- **Word 1, Bit 28 - N_Port/F_Port (N)**

0 = N_Port
1 = F_Port

- **Word 1 bit 28 shall be set = 0.**

- **Word 1, bit 27 - BB_Credit Management (A)**

0 = BB_Credit management (see 20.5.4)
1 = Alternate BB_Credit management (see 20.5.10)

The BB_Credit Management field specifies the type of BB_Credit Management to be used.

- **Word 1, bit 26 - E_D_TOV Resolution (R)**

0 = 1 ms
1 = 1 ns

Word 1, Bit 26 indicates the resolution of the E_D_TOV timer. If Word 1 Bit 26 = 0, then the timer shall be in increments of 1 ms. If Word 1 Bit 26 = 1, then the timer shall be in increments of 1 ns.

- **Word 1, bit 25-23 - Reserved**

- **Word 1, bit 22 - Dedicated Simplex (D)**

0 = Dedicated Simplex not supported by the N_Port
1 = Dedicated Simplex supported by the N_Port

- **Word 1, bit 21 - Reserved**

- **Word 1, bit 20 - Clock Synchronization Primitive Capable (Y)**

0 = not Clock Synchronization Primitive Capable
1 = Clock Synchronization Primitive Capable

Word 1, Bit 20 indicates support for the Primitive method of Clock Synchronization (see 32.3). If Word 1, Bit 20 = 0, then the N_Port does not support the Primitive method of Clock Synchronization. If Word 1, Bit 20 = 1 for the Clock Synchronization Server well-known address (hex 'FF FF F6'), the Clock Synchronization Server is capable of generating clock synchronization Primitive Signals. If Word 1, Bit 20 = 1 for any other N_Port, the

N_Port is capable of receiving the clock synchronization Primitive Signals and acting upon them.

- **Word 1, bit 19 - Reserved**
- **Word 1, bit 18 - Dynamic Half Duplex (H)**

0 = DHD not supported
1 = DHD supported

Word 1, Bit 18 indicates support for Dynamic Half Duplex. If Word 1 Bit 18 = 0, then Dynamic Half Duplex is not supported. If Word 1 Bit 18 = 1, then the N_Port is indicating that it supports the reception of the DHD primitive.

NOTE - DHD is applicable to FC-AL topologies. (See FC-AL-2).

- **Word 1, bit 17 - SEQ_CNT (S)**

0 = Normal FC-FS rules apply to SEQ_CNT
1 = Continuously Increasing SEQ_CNT shall be used.

Word 1, Bit 17 indicates the requirement on SEQ_CNT. If Word 1 Bit 17 = 0, then normal FC-FS rules shall apply to SEQ_CNT usage. If Word 1 Bit 17 = 1, then the N_Port is guaranteeing that it will transmit all frames within an Exchange using a continuously increasing SEQ_CNT. Each Exchange shall start with SEQ_CNT = 0 in the first frame, and every frame transmitted after that shall increment the previous SEQ_CNT by one, even across transfers of Sequence Initiative. Any frames received from the other N_Port in the Exchange shall have no effect on the transmitted SEQ_CNT.

- **Word 1, bit 16 - Payload Length (P)**

0 = 116 bytes
1 = 256 bytes

Word 1, Bit 16 indicates the length of the PLOGI payload. If Word 1 Bit 16 = 0, then the payload shall be 116 bytes. If Word 1 Bit 16 = 1, then the payload shall be 256 bytes.

- **Word 1, Bits 15 – 12 - Reserved**

17.6.3.4 Buffer-to-buffer Data_Field size

- **Word 1, Bits 11 - 0 - Buffer-to-buffer Receive Size (F)**

The buffer-to-buffer Receive Data_Field Size is a binary value (bits 11-0) which specifies the largest Data_Field Size for an FT_1 frame (11.4) that can be received by the N_Port supplying the Service Parameters as a Sequence Recipient for:

- a connect-request (SOFc1),
- a Class 2 Data frame, or
- a Class 3 Data frame.

Values less than 256 or greater than 2¹¹ are invalid. Values shall be a multiple of four bytes.

The buffer-to-buffer Receive Data_Field size is specified by FC-2.

17.6.3.5 Total Concurrent Sequences

Total Concurrent Sequences is the total number of Concurrent Sequences for all 3 classes that a N_Port is capable of supporting as a Recipient.

Word 2, Bits 23 - 16 - Total Concurrent Sequences (T)

The Total Concurrent Sequences specified by a N_Port shall be less than or equal to the sum of the Concurrent Sequences supported on a Class by Class basis. As an example, a N_Port

may specify that it is capable of supporting ten Concurrent Sequences in Class 2 and ten Concurrent Sequences in Class 3. However, the total number of Concurrent Sequences when both Class 2 and 3 are Active may be fifteen (T).

17.6.3.6 Relative Offset by category

Word 2, bits 15 - 0 shall indicate on a bit-position basis, whether or not Relative Offset shall be supported for the corresponding Information Category. For example, if bit 14 = 1 and bit 2 = 1 and others are = 0, then Information Category 1110 and Information Category 0010 frames shall be capable of using Relative Offset as a Sequence Recipient or a Sequence Initiator.

17.6.3.7 Point-to-point E_D_TOV value

Word 3 shall only be meaningful by a N_Port in a point-to-point topology. In a topology other than point-to-point, word 3 shall not be meaningful. When Word 1, Bit 26 = 0, the E_D_TOV value shall be specified as a count of 1 ms increments. When Word 1, Bit 26 = 1, the E_D_TOV value shall be specified as a count of 1 ns increments. Therefore, based on the setting of Word 1, Bit 26, a value of hex '00 00 00 0A' specifies a time period of either 10 ms or 10 ns.

The E_D_TOV value in the Accept shall be greater than or equal to the value in the PLOGI. The E_D_TOV value in the Accept shall be the value used by each N_Port. R_A_TOV shall be a value twice the E_D_TOV value in a point-to-point topology.

17.6.3.8 BB_SC Number

- **Word 1, bits 15-12 - Buffer-to-buffer State Change Number (BB_SC_N) (S)**

The BB_SC_N is a binary value (bits 15-12) which indicates that the sender of the PLOGI or FLOGI frame is requesting $2^{BB_SC_N}$ number of frames to be sent between two consecutive BB_SCs primitives, and $2^{BB_SC_N}$ number of R_RDY primitives to be sent between two consecutive BB_SCr primitives.

When the two ports performing the login specifies different non-zero values of BB_SC_N, the larger value shall be used. If either port specifies a BB_SC_N value of zero, the BB_Credit recovery process shall not be performed, and no BB_SCx primitive shall be sent. See 20.5.12 for a description of the BB_Credit recovery process.

Note - If all frames or R_RDY primitives sent between two BB_SCx primitives are lost, $2^{BB_SC_N}$ number of BB_Credits are lost, and cannot be recovered by the scheme outlined in 20.5.12. Therefore BB_SC_N should be chosen so that the probability of losing $2^{BB_SC_N}$ number of consecutive frames or R_RDY primitives is deemed negligible. The recommended value of BB_SC_N is 8.

17.6.4 N_Port_Name

The N_Port_Name is an eight-byte field that identifies a N_Port for identification purposes, such as diagnostics, which may be independent and unrelated to network addressing. The format of the name is specified in 13.3. Each N_Port shall provide a unique N_Port_Name within the address domain of the Fabric. Bits 63-60 specify the format of the name. The formats are summarized in Table 28 and Table 29 in 13.3.

17.6.5 Node_Name

The Node_Name is an eight-byte field that identifies a Node for identification purposes, such as diagnostics, which is independent and unrelated to network addressing. The format of the Node_Name is specified in 13.3. Each Node shall provide a unique Node_Name within the address domain of the Fabric. Bits 63-60 specify the format of the name. The formats are summarized in Table 28 and Table 29 in 13.3.

17.6.6 N_Port Class Service Parameters

Each group of sixteen byte Class Service Parameters is divided into the categories as specified in Figure 45.

- Class Validity (V)
- Service Options (E)
- Initiator Control Flags (D)
- Recipient Control Flags (C)
- Receive Data Size (N)
- Concurrent Sequences (L)
- N_Port End-to-end Credit (M)
- Open Sequences per Exchange (X)

Word 0	31	16	15	0
	Service Options		Initiator Control	
	VISS QDCB	PEEE EEEE	DDDD DDDD	DDDD DDDD
1	31	16	15	0
	Recipient Control		Rcv Data Field size	
	CCCC CCCC	CCCC CCCC	rrrr FFFF	FFFF FFFF
2	31	16	15	0
	Concurrent Sequences		N_Port End-to-end Credit	
	rrrr rrrr	LLLL LLLL	0MMM MMMM	MMMM MMMM
3	31	16	15	0
	Open Sequences per Exchange		Reserved	
	rrrr rrrr	xxxx xxxx	rrrr rrrr	rrrr rrrr

Figure 45 - N_Port Class Service Parameters - Fabric Login

The parameters described are located in the Payload of the Login (LOGI) Link Service request as well as the Accept (ACC) Link Service reply Sequence sent in reply to the Login Link Service request.

Applicability

Table 204 identifies N_Port Class Service Parameter fields and specifies the applicability of those parameters for N_Port_Login or Fabric Login by class.

Table 204 - N_Port Class Service Parameter Applicability										
Service Parameter	Word	Bits	N_Port_Login Class				Fabric Login Class			
			1	2	3	4 ³	1	2	3	4 ³
Class Validity	0	31								
Valid = 1	0	31	y ¹	y	y	y	y	y	y	y

Table 204 - N_Port Class Service Parameter Applicability										
Service Parameter	Word	Bits	N_Port_Login Class				Fabric Login Class			
			1	2	3	4 ³	1	2	3	4 ³
Service Options	0	30-16								
Intermix Mode	0	30	y	n ²	n	n	y	n	n	n
Stacked Connect-Requests	0	29-28	n	n	n	n	n	n	n	n
Sequential delivery	0	27	n	n	n	n	n	y	y	n
Dedicated Simplex (see clause 27)	0	26	y	n	n	n	y	n	n	n
Camp on (see clause 29)	0	25	n	n	n	n	y	n	n	n
Buffered Class 1 (see clause 31)	0	24	y	n	n	n	y	n	n	n
Priority (see 12.4.2)	0	23	y	y	y	y	y	y	y	y
Initiator Control	0	15-0								
Initial Responder Process_Associator	0	13-12	y	y	y	y	n	n	n	n
ACK_0 capable	0	11	y	y	n	y	n	n	n	n
ACK generation assistance (see 14.5)	0	9	y	y	n	y	n	n	n	n
Data compression capable - obsolete	0	8								
Data compression history buffer size - obsolete	0	7-6								
Data Encryption Capable - obsolete	0	5								
Clock Synchronization ELS capable	0	4	y	y	y	y	y	y	y	y
Recipient Control	1	31-16								
ACK_0 Capable	1	31	y	y	n	y	n	n	n	n
X_ID interlock	1	29	y	y	n	y	n	n	n	n
Error policy support	1	28-27	y	y	y	y	n	n	n	n
Categories per Sequence	1	25-24	y	y	y	y	n	n	n	n
Data compression capable - obsolete	1	23								
Data compression history buffer size-obsolete	1	22-21								
Data decryption capable – obsolete	1	20								
Clock Synchronization ELS capable	1	19	y	y	y	y	y	y	y	y
Reserved - Fabric Specific	1	18-16	y	y	y	y	y	y	y	y
Receive Data Field Size	1	11-0	y	y	y	y	n	n	n	n
Concurrent Sequences	2	31-16	y	y	y	y	n	n	n	n
N_Port End-to-end Credit	2	14-0	y	y	n	n	n	n	n	n
Open Sequences per Exchange	3	31-16	y	y	y	y	n	n	n	n
Notes:										
1) "y" indicates yes, applicable;										
2) "n" indicates no, not applicable										
3) Class 4 (see clause 28)										

17.6.7 N_Port Class Service Parameters - Fabric Login

17.6.7.1 Class validity (V)

- **Word 0, Bit 31 - Class validity**

0 = Invalid (Class not supported)

1 = Valid (Class supported)

The Class validity bit shall indicate whether this Class is supported or not. If the Class validity bit is zero, it indicates that this group or set of sixteen bytes shall be ignored. If the Class validity bit is one, it indicates that this Class shall be supported. The Class shall be identified based on Class 1 = first sixteen-byte group, Class 2 = second sixteen-byte group, Class 3 = third sixteen-byte group, and Class 4 = fourth sixteen-byte group.

NOTE - Service Parameter options specify FC-2 capability. The Link Service may further limit values supplied during Login as specified by individual Upper Level Protocols.

17.6.7.2 Service options

Service options (E) shall specify optional features of a Class of Service supported by the N_Port supplying the Service Parameters.

- **Word 0, Bit 30 - Intermix Mode**

0 = Intermix not requested

1 = Intermix requested Class 1

Class 1

All N_Ports supporting Class 1 shall support Exclusive Connections. A N_Port supporting Exclusive Connections may only transmit and receive frames from the N_Port to which an existing Class 1 Connection is pending or established. Exclusive Connections require that the Fabric transmit an F_BSY frame in response to Class 2 frames and connect-request Data frames (SOF_{c1}) issued by a third N_Port targeted for one of the two

N_Ports engaged in a Class 1 Connection.: An Intermixed Dedicated Connection specifies that the Fabric may insert or extract Class 2 or Class 3 frames while a Class 1 Connection is established. Support for Intermix is optional by both a N_Port and a Fabric. When a N_Port performs Login with a Fabric, it shall request support for Intermix by specifying bit 30 = 1. If the Fabric supplies, bit 30 = 1 in the Accept reply Sequence, then Intermix shall be functional.

The following set of values specifies the meaning of the combination of Word 0, bit 30:

N_Port	F_Port	
0	0	Neither supports
0	1	Fabric is capable of supporting, Intermix not functional
1	0	N_Port support requested, Fabric does not support
1	1	N_Port requested, Fabric agrees, Intermix is functional

See 16.4 for more discussion on Intermix.

Class 2 and 3

Word 0, bit 30 is reserved.

Class 4

- **Word 0 bit 30 (Intermix Mode)**

has no meaning in Class 4 since the Fabric shall support Intermix on Class 4.

- **Word 0, bit 29-28 - Stacked connect-request**

Class 1

Word 0, bits 29-28 are not meaningful in N_Port_Login. These bits are defined only for Class 1 and Buffered Class 1 operation and are meaningful only as F_Port Service Parameters.

Class 2, 3 and 4

Word 0, bits 29-28 are reserved.

- **Word 0, Bit 27 - Sequential delivery**

0 = Out of order delivery

1 = Sequential delivery requested

This bit is only meaningful in class 2 and 3. Out of order frame delivery in class 2 and 3 is the default function by a Fabric.

If bit 27 is set to one by a N_Port, then it is requesting that all frames delivered to the N_Port requesting this function be delivered in the same order in which the frames were transmitted by the source N_Port. If bit 27 is set to one by the F_Port in the Accept reply Sequence, the F_Port is indicating that sequential delivery is functional.

A Fabric supporting the sequential delivery feature routes Class 2 and 3 frames via a fixed route through the Fabric to provide in-order frame delivery. This feature does not imply any other alteration to the normal class of service functions. For example, F_BSY responses are still possible in Class 2, and Class 3 frames may still be discarded by the Fabric based on normal Class 2 and 3 rules.

Class 1

Bit 27 has no meaning in Class 1 since the Fabric shall deliver frames in the order transmitted based on class of service.

Class 2

In Class 2, if bit 27 = 1, the Fabric guarantees the order of delivery of both Data and Link_Control frames to the N_Port requesting this feature in the same order in which the frames were transmitted.

Class 3

In Class 3, if bit 27 = 1, the Fabric guarantees the order of delivery of Data frames to the N_Port requesting this feature in the same order in which the frames were transmitted.

N_Port	F_Port	
0	0	Neither supports
0	1	Fabric is capable of supporting, Sequential delivery is functional
1	0	N_Port support requested, Fabric does not support
1	1	N_Port requested, Fabric agrees, Sequential delivery is functional

Class 4

Bit 27 has no meaning in Class 4 since the Fabric shall deliver frames in the order transmitted based on class of service.

- **Word 0, Bit 26 Dedicated Simplex (D)**

Class 1

0=Dedicated Simplex not supported

1=Dedicated Simplex supported

See clause 27.

Class 2, 3 and 4

Word 0, bit 26 is reserved.

- **Word 0, Bit 25 Camp-On (C)**

Class 1

0 = Camp-On not supported

1 = Camp-On supported

See clause 29.

Class 2, 3 and 4

Word 0, bit 25 is reserved.

- **Word 0, Bit 24 Buffered Class 1 (B)**

0 = Buffered Class 1 is not requested or Buffered Class 1 is not supported

1 = Buffered Class 1 is requested or Buffered Class 1 is supported

Buffered Class 1

Support for Buffered Class 1 Service is negotiated during Fabric Login. If a N_Port requests Buffered Class 1 Service and the attached F_Port supports it, only then is Buffered Class 1 Service enabled and available for use by the N_Port and Fabric. Table 205 summarizes the meaning of the combination of Word 0, bit 24.

Table 205 - Buffered Class 1 Login Service parameters		
N_Port	F_Port	Description
0	0	Neither supports Buffered Class 1
0	1	Fabric is capable of supporting Buffered Class 1, Buffered Class 1 is nonfunctional
1	0	N_Port requests use of Buffered Class 1, Buffered Class 1 is nonfunctional
1	1	Fabric is capable of supporting Buffered Class 1 and N_Port request use of Buffered Class 1, Buffered Class 1 is functional (enabled)

Class 1, 2, 3 and 4

Word 0, bit 24 is reserved.

- **Word 0, Bit 23 - Priority/Preemption (P)**

0 = Priority/Preemption not supported

1 = Priority/Preemption supported

See clause 12.4.

Class 1, 2, 3, and 4

When a N_Port performs Login with a Fabric, it shall request support for Priority/Preemption by specifying bit 23 = 1. If the Accept reply from the Fabric specifies bit 23 = 1, then both the N_Port and Fabric have agreed that Priority/Preemption is available for use.

The set of values specified in Table 206 give the meaning of the combination of Word 0, bit 23.

Table 206 – Priority/Preemption bit definition		
N_Port	F_Port	Description
0	0	Neither supports Priority/Preemption
0	1	Fabric is capable of supporting Priority/Preemption
1	0	N_Port support requested, Fabric does not support Priority/Preemption
1	1	N_Port requested, Fabric agrees, Priority/Preemption is available for use

- **Word 0, Bit 22-16**

Reserved

17.6.7.3 Initiator control

- **Word 0, bits 15-5 - Reserved**

- **Word 0, bit 4 - Clock synchronization ELS capable (Y)**

Class 1,2,3, and 4 (see clause 32.3)

0 = Initiator does not have clock synchronization ELS capability

1 = Initiator has clock synchronization ELS capability

Word 0, bit 4 is only meaningful from the Clock Synchronization Server well-known address (hex 'FF FF F6'). This bit indicates support for the ELS method of Clock Synchronization. If Word 0, bit 4 = 0 then the N_Port does not support the ELS method of Clock Synchronization. If Word 0, bit 4 = 1, the N_Port is capable of generating the CSU ELS frames.

- **Word 0, bits 3-0 - Reserved**

17.6.7.4 Recipient control

- **Word 1 bits 31 - 20**

These bits are not meaningful during Fabric Login.

- **Word 1, bit 19 - Clock synchronization capable (Y)**

Class 1,2,3, and 4 (see 32.3)

0 = Recipient does not have clock synchronization ELS capability

1 = Recipient has clock synchronization ELS capability

Word 1, Bit 19 indicates support for the ELS method of Clock Synchronization. If Word 1, Bit 19 = 0, then the N_Port does not support the ELS method of Clock Synchronization. If Word 1, Bit 19 = 1, the N_Port is capable of receiving and acting on the CSU ELS frames.

- **Word 1 bits 18 - 16**

Reserved for Fabric Specific use.

17.6.7.5 Receive Data_Field Size

This field is not meaningful during Fabric Login.

17.6.7.6 Concurrent Sequences

This field is not meaningful during Fabric Login.

17.6.7.7 N_Port End-to-end Credit

This field is not meaningful during Fabric Login.

17.6.7.8 Open Sequences per Exchange

This field is not meaningful during Fabric Login.

17.6.8 N_Port Class Service Parameters - N_Port_Login

N_Port Class Service Parameters using during N_Port_Login is illustrated in Figure 46.

Word 0	31	16	15	0
	Service Options		Initiator Control	
	VISS	QDCB	PEEE	EEEE
	XXPP	ZNGC	CCED	YDDD
1	31	16	15	0
	Recipient Control		Rcv Data Field size	
	ZNXL	LrSS	CCCE	Yrrr
	rrrr	NNNN	NNNN	NNNN
2	31	16	15	0
	Concurrent Sequences		N_Port End-to-end Credit	
	rrrr	rrrr	LLLL	LLLL
	0MMM	MMMM	MMMM	MMMM
3	31	16	15	0
	Open Sequences per Exchange		Class 6 Multicast RX_ID	
	rrrr	rrrr	xxxx	xxxx
	xxxx	xxxx	xxxx	xxxx

Figure 46 - N_Port Class Service Parameters - N_Port_Login

17.6.8.1 Class validity

- **Word 0, Bit 31 - Class validity (V)**

0 = Invalid (Class not supported)

1 = Valid (Class supported)

The Class validity bit shall indicate whether this Class is supported or not. If the Class validity bit is zero, it indicates that this group or set of sixteen bytes shall be ignored. If the Class validity bit is one, it indicates that this Class shall be supported. The Class shall be identified based on Class 1 = first sixteen-byte group, Class 2 = second sixteen-byte group, Class 3 = third sixteen-byte group, and Class 4 = fourth sixteen-byte group.

NOTE - Service Parameter options specify FC-2 capability. The Link Service may further limit values supplied during Login as specified by individual Upper Level Protocols.

17.6.8.2 Service options

Service Options (E) shall specify optional features of a Class of Service supported by the N_Port supplying the Service Parameters (see Table 204).

- **Word 0, Bit 30 - Intermix mode**

0 = Intermix not functional
1 = Intermix is functional

Class 1

When Word 0 bit 30 is set = 1 in N_Port_Login with another N_Port, it indicates that Intermix is functional between the N_Port supplying this parameter and the Port to which it is attached. In a point-to-point topology if both N_Ports indicate Intermix support, then Intermix is functional. Otherwise, Class 1 Dedicated Connections shall be removed before transmission of Class 2 or Class 3 frames.

Class 2 and 3

Word 0, Bit 30 is reserved.

Class 4

Word 0 bit 30 has no meaning in Class 4 since the Fabric shall support Intermix on Class 4.

- **Word 0, Bits 29-28 - Stacked connect-requests**

Class 1, 2 and 3

Word 0, bits 29-28 are not meaningful in N_Port_Login.

Class 4

Word 0, bits 29 and 28 are reserved.

- **Word 0, Bit 27 - Sequential delivery**

Class 1, 2 and 3

This bit is not meaningful in N_Port_Login.

Class 4

Bit 27 has no meaning in Class 4 since the Fabric shall deliver frames in the order transmitted based on class of service.

- **Word 0, Bit 26 Dedicated Simplex (D)**

Class 1

See clause 27.

Class 2, 3 and 4

Word 0, bit 26 is reserved.

- **Word 0, Bit 25 Camp-On (C)**

Class 1

Not meaningful in N_Port_Login.

Class 2, 3 and 4

Word 0, bit 25 is reserved.

- **Word 0, Bit 24 - Buffered Class 1 (B)**

Word 0, Bit 24 is not meaningful for N_Port_Login. See clause 31.

Class 1, 2, 3 and 4

Word 0, bit 24 is reserved.

- **Word 0, Bit 23 - Priority (P)**

0 = Priority not supported

1 = Priority supported

See clause 12.4.2.

Class 1, 2, 3 and 4

When a N_Port performs Login with another N_Port, it shall indicate support for Priority by specifying bit 23 = 1. The other N_Port indicates support for Priority by specifying bit 23 = 1 in the ACC sent as a reply. Support of Priority as an Exchange Originator means that the Priority bit (F_CTL bit 17) may be set to indicate that Word 1, Bits 31-24 of the Frame Header specifies the desired Priority. Support of Priority as an Exchange Responder means that the Priority bit (F_CTL bit 17), if set, will be interpreted as indicating that Word 1, Bits 31 - 24 of the Frame Header contains the desired Priority.

- **Word 0, Bit 22 – Preference**

0 = non-zero CS_CTL may be tolerated

1 = non-zero CS_CTL shall be tolerated

When a N_Port performs Login with another N_Port, it shall indicate tolerance for non-zero CS_CTL within the Class of Service by specifying bit 22 = 1. The other N_Port indicates tolerance for non-zero CS_CTL by specifying bit 22 = 1 in the ACC sent as a reply. A N_Port that tolerates a non-zero CS_CTL shall not reject or otherwise deprecate a frame solely because the CS_CTL field is non-zero.

NOTE – Even if a N_Port never intends to set the PREF field to any value other than zero, the N_Port may still have reason to set the bit to one. Setting the bit to one indicates to the other N_Port that the N_Port will accept frames with a non-zero CS_CTL field value.

Class 1

Bit 22 has no meaning in Class 1.

Class 2

In Class 2, if bit 22 = 1, the N_Port shall tolerate the PREF field in the CS_CTL field of the Frame Header. Tolerance for CS_CTL as a Sequence Initiator means that the PREF field may specify Preference to the Fabric. Tolerance for CS_CTL as a Sequence Recipient means that the N_Port shall ignore the PREF field. (see 16.2.2)

Class 3

In Class 3, if bit 22 = 1, the N_Port shall tolerate the PREF field in the CS_CTL field of the Frame Header. Tolerance for CS_CTL as a Sequence Initiator means that the PREF field may specify Preference to the Fabric. Tolerance for CS_CTL as a Sequence Recipient means that the PREF field shall be ignored by the N_Port. (see 16.3.2)

Class 4

Bit 22 has no meaning in Class 4.

- **Word 0, Bit 21-16 - Reserved**

17.6.8.3 Initiator control

Initiator Control Flags (D) specify which protocols, policies, or functions the Sequence Initiator function in the N_Port supplying the Service Parameters requests of the Recipient or is capable of as a Sequence Initiator (see Table 204).

- **Word 0, Bits 13-12 - Initial Process_Associator**

- 0 0 = Initial Process_Associator not supported
- 0 1 = Initial Process_Associator supported
- 1 0 = Reserved
- 1 1 = Initial Process_Associator required (and supported)

Initial Process_Associator required indicates that the N_Port supplying this parameter requires an Association_Header at certain Sequence boundaries (see 13.4) which contains a specific initial value in the Process_Associator field. A N_Port which supports Initial Process_Associator shall supply an Association_Header with an initial Responder Process_Associator value at certain Sequence boundaries, such as when it originates an Exchange (see 19.2.1 If an Initial Process_Associator is required or supported, then X_ID interlock also is required in Class 1 and 2.

If the Responder N_Port to the PLOGI request requires an Initial Process_Associator and the Originator of the PLOGI request does not support an Initial Process_Associator, the Responder shall transmit an LS_RJT indicating the Initiator Control bits are in conflict. If the Responder N_Port to the PLOGI request does not support an Initial Process_Associator and the Originator of the PLOGI request has indicated that Initial Process_Associator is required, the Responder shall transmit an LS_RJT indicating the Initiator Control bits are in conflict. In either of these cases, the N_Ports are unable to communicate.

- **Word 0, Bit 11 - ACK_0 capability**

- 0 = ACK_0 incapable
- 1 = ACK_0 capable

Bit 11 specifies that the N_Port supplying these Class Service Parameters is or is not capable of support for ACK_0 as a Sequence Initiator for acknowledgement of an entire Sequence in either Discard or Process Exchange Error Policies. As a Sequence Initiator a N_Port receives and processes ACK frames in response to Data frame transmission. ACK_0 support is applicable to Acknowledged Class Sequences.

The conditions under which ACK_0 is support is defined in Table 207 and described in the following text.

Table 207 – ACK_0 Support Conditions		
N_Port A Word 0, Bit 11	N_Port B Word 1, Bit 31	N_Port A as Sequence Initiator
0	0	ACK_0 not supported
0	1	ACK_0 not supported
1	0	ACK_0 not supported
1	1	ACK_0 supported

If one N_Port (N_Port A, for example) is capable of receiving ACK_0 as a Sequence Initiator (Word 0, Bit 11 = 1) and the other N_Port (N_Port B, for example) is capable of transmitting ACK_0 as a Sequence Recipient (Word 1, Bit 31 = 1), then ACK_0 is supported when N_Port A is the Sequence Initiator and N_Port B is the Sequence Recipient. Otherwise, ACK_0 shall not be supported while N_Port A is the Sequence Initiator and N_Port B is the Sequence Recipient. ACK_0 usage shall take precedence over ACK_1.

If ACK_0 is supported, as indicated, N_Port A shall transmit Data frames to N_Port B with infinite buffering in effect and shall be able to receive ACK_0 for Sequences transmitted. If ACK_0 is supported, as indicated, N_Port B shall support infinite buffering and be capable of transmitting ACK_0 to acknowledge an entire Sequence (see 14.3.3.2 and 20.4.3.1).

ACK_0 capability may be asymmetrical for a single N_Port. That is, a N_Port may be capable processing ACK_0 as a Sequence Initiator, but not be capable of ACK_0 transmission as a Sequence Recipient. Similarly, a N_Port may be capable of generating ACK_0 as a Sequence Recipient, but not be capable of ACK_0 reception as a Sequence Initiator.

ACK_0 may be used for all forms of Discard Error Policies for Exchanges. ACK_0 support shall be required for Process Policy with infinite buffers Exchange Error Policy. However, any requirements regarding symmetrical or asymmetrical ACK_0 support shall be specified by the FC-4 or upper level and is outside the scope of FC-FS. For example, if an FC-4 only transmitted Video_Data from N_Port A to N_Port B (a frame buffer), then N_Port A need only support ACK_0 as a Sequence Initiator, whereas, N_Port B need only support ACK_0 as a Sequence Recipient. See 17.6.8.4 for additional discussion regarding the use of ACK_0.

- **Word 0, Bit 10 - reserved**
- **Word 0, bit 9 - ACK generation assistance (G)**

Class 1, 2 and 4 (see 14.5)

0 = No ACK generation assistance is provided to Sequence Recipient.
1 = ACK generation assistance is provided to Sequence Recipient.

Class 3

Same function as Class 1 and 2.

- **Word 0, bit 8 - Data compression capable(C)**
This bit is obsolete in this standard.
- **Word 0, bit 7 -6 - Data compression History buffer size (CC)**
This field is obsolete in this standard.
- **Word 0, bit 5 - Data encryption capable (E)**
This bit is obsolete in this standard.

Class 4

Same function as Class 1 and 2.

- **Word 0, bit 4 - Clock synchronization capable (Y)**

Class 1,2,3, and 4 (see 32.2)

0 = Initiator does not have clock synchronization ELS capability
1 = Initiator has clock synchronization ELS capability

Word 0, bit 4 is only meaningful from the Clock Synchronization Server well-known address (hex 'FF FF F6'). This bit indicates support for the ELS method of Clock Synchronization. If Word 0, bit 4 = 0 then the N_Port does not support the ELS method of Clock Synchronization. If Word 0, bit 4 = 1, the N_Port is capable of generating the CSU ELS frames.

17.6.8.4 Recipient control

Recipient Control Flags (C) shall specify which functions are supported by the N_Port supplying the Service Parameters when acting as a receiver of Data frames. Recipient Control Flags specify the Recipient functions supported by the N_Port (see Table 204).

- **Word 1, Bit 31 - ACK_0 capability**

0 = ACK_0 incapable

1 = ACK_0 capable

Bit 31 specifies that the N_Port supplying these Class Service Parameters is or is not capable of support for ACK_0 as a Sequence Recipient for acknowledgement of an entire Sequence in either Discard or Process Exchange Error Policies. As a Sequence Recipient a N_Port shall support infinite buffering and be capable of transmitting ACK_0 frames in response to Data frame transmission. ACK_0 support is applicable to Acknowledged Class Sequences.

N_Port A	N_Port B	
Word 0, Bit 11	Word 1, Bit 31	N_Port A as Sequence Recipient:
0	0	ACK_0 not supported
0	1	ACK_0 not supported
1	0	ACK_0 not supported
1	1	ACK_0 supported

If one N_Port (N_Port A, for example) is capable of receiving ACK_0 as a Sequence Initiator (Word 0, Bit 11 = 1) and the other N_Port (N_Port B, for example) is capable of transmitting ACK_0 as a Sequence Recipient (Word 1, Bit 31 = 1), then ACK_0 may be used when N_Port A is the Sequence Initiator and N_Port B is the Sequence Recipient. Otherwise, ACK_0 shall not be supported while N_Port A is the Sequence Initiator and N_Port B is the Sequence Recipient. If ACK_0 is supported, as indicated, N_Port A shall transmit Data frames to N_Port B with infinite buffering in effect and shall be able to receive ACK_0 for Sequences transmitted. If ACK_0 is supported, as indicated, N_Port B shall support infinite buffering and be capable of transmitting ACK_0 to acknowledge an entire Sequence (see 14.3.2.2 and 20.4.3.1).

ACK_0 capability may be asymmetrical for a single N_Port. That is, a N_Port may be capable processing ACK_0 as a Sequence Initiator, but not be capable of ACK_0 transmission as a Sequence Recipient. Similarly, a N_Port may be capable of generating ACK_0 as a Sequence Recipient, but not be capable of ACK_0 reception as a Sequence Initiator. If a N_Port sets both Word 0, bit 11 and Word 1, bit 31 to one, then it is capable of ACK_0 support as either a Sequence Initiator or a Sequence Recipient.

ACK_0 may be used for all forms of Discard Error Policies for Exchanges. ACK_0 support shall be required for Process Policy with infinite buffers Exchange Error Policy. However, any requirements regarding symmetrical or asymmetrical ACK_0 support shall be specified by the FC-4 or upper level and is outside the scope of FC-FS. For example, if an FC-4 only transmitted Video_Data from N_Port A to N_Port B (a frame buffer), then N_Port A need only support ACK_0 as a Sequence Initiator, whereas, N_Port B need only support ACK_0 as a Sequence Recipient. See 17.6.8.4 for additional discussion regarding the use of ACK_0.

- **Word 1, Bit 30 - reserved**
- **Word 1, Bit 29 - X_ID interlock**

0 = X_ID interlock not required

1 = X_ID interlock required

X_ID interlock only applies to Class 1 and Class 2. This bit indicates that the N_Port supplying this parameter requires that an interlock be used during X_ID assignment or reassignment in Class 1 and 2. In X_ID assignment, the Sequence Initiator shall set the Recipient X_ID value to hex 'FF FF' in the first Data frame of a Sequence and the Recipient shall supply its X_ID in the ACK frame corresponding to the first Data frame of a Sequence. The Sequence Initiator shall not transmit additional frames until the corresponding ACK is received. Following reception of the ACK, the Sequence Initiator continues transmission of the Sequence using both assigned X_ID values.

- **Word 1, Bits 28-27 - Error policy supported**

0 0 = Only discard supported

0 1 = Reserved

1 0 = Both discard and process supported

1 1 = Reserved

These bits are set to specify the types of support possible for missing frame conditions processed by the Receiver N_Port. The policy used for a given Exchange shall be specified as discard or process by the Exchange Originator (see Abort Sequence Condition bits in 12.6 and 23.6.1.1).

In either Discard or Process policy, when a missing frame error is detected, the expected sequence count is saved in the Error SEQ_CNT field of the appropriate Sequence Status Block and a Sequence error is posted in the S_STAT field in the same Sequence Status Block for a given Exchange (OX_ID, RX_ID). Only the first error is saved.

NOTE - The error status is reported by FC-2 to FC-4.

Discard policy support specifies that the N_Port shall be able to discard Data frames received following detection of a missing frame error condition including the frame at which the error is detected.

Except for Class 3, when the missing frame condition is detected, the Sequence Recipient shall notify the Sequence Initiator using the Abort Sequence Condition bits in F_CTL on the ACK corresponding to the frame on which the error was detected. All subsequent frames continue to be discarded and Abort Sequence indicated in the ACK frame (see 18.3.10). In one Discard Policy, only a single Sequence shall be discarded, whereas in the other Discard Policy, multiple Sequences shall be discarded (see Abort Sequence Condition bits in 12.6 and 23.6.1.1).

Process policy support specifies that the N_Port is to be able to continue processing valid Data frames following a detected missing frame error condition in the normal manner, including the frame at which the error is detected (see 23.6.1.1).

- **Word 1, Bits 25-24 - Categories per Sequence**

0 0 = 1 Category/Sequence

0 1 = 2 Categories/Sequence

1 0 = Reserved

1 1 = More than 2 Categories/Sequence

The setting of bits 25-24 shall specify that the Recipient is capable of processing one, two, or more than two Information Categories (R_CTL bits 27-24) in a single Sequence. Bits 25-24 are applicable to each Class of Service since support for an individual Class may offer different capabilities in the same N_Port.

When a N_Port is acting as a Sequence Initiator, it shall restrict the number of Information Categories per Sequence based on the Sequence Recipient's capability as specified during N_Port_Login. A N_Port's capability for processing Information Categories in a single Sequence may prohibit that N_Port from communicating in certain FC-4 protocols.

Each FC-4 should allow the ability to communicate using only one Information Category per Sequence but always provide the ability to communicate using multiple Information Categories per Sequence where possible, and when performance may be enhanced.

- **Word 1, bit 23- Data compression capable (C)**

This bit is obsolete in this standard.

- **Word 1, bit 22 -21- Data compression History buffer size (CC)**

This field is obsolete in this standard.

- **Word 1, bit 20 - Data decryption capable (E)**

This bit is obsolete in this standard.

Class 4

Same function as Class 1 and 2.

- **Word 1, bit 19 - Clock synchronization capable (Y)**

Class 1,2,3, and 4 (see 32.2)

0 = Recipient does not have clock synchronization ELS capability

1 = Recipient has clock synchronization ELS capability

Word 1, Bit 19 indicates support for the ELS method of Clock Synchronization. If Word 1, Bit 19 = 0, then the N_Port does not support the ELS method of Clock Synchronization. If Word 1, Bit 19 = 1, the N_Port is capable of receiving CSU ELS frames and acting upon them.

- **Word 1, Bit 18-12 - Reserved**

17.6.8.5 Receive Data_Field Size

- **Word 1, Bits 11-0 Receive Data_Field Size (N)**

The Receive Data_Field Size is a binary value (bits 11-0) which specifies the largest Data_Field Size for an FT_1 frame (11.4) that can be received by the N_Port supplying the Service Parameters as a Sequence Recipient. Values less than 128 or greater than 2 112 are invalid. Values shall be a multiple of four bytes. A N_Port shall support a Data Field size of at least 128 bytes, however, a minimum of 256 bytes is recommended.

Class 1

In Class 1 the Receive Data_Field size represents the largest Data_Field size that a N_Port is able to receive after a Dedicated Connection is established. (The connect-request Data_Field size is specified in the Buffer-to-Buffer Receive Data_Field size in Common Service Parameters.)

Class 2 and 3

The Receive Data_Field size for Class 2 and 3 shall be equal to or less than the Buffer-to-Buffer Receive Data_Field size specified in the Common Service Parameters.

17.6.8.6 Concurrent Sequences

- **Word 2, Bits 31-16 Concurrent Sequences (L)**

Concurrent Sequences shall specify the number of Sequence Status Blocks available in the N_Port supplying the Service Parameters for tracking the progress of a Sequence as a Sequence Recipient. The maximum number of Concurrent Sequences that can be specified is 255 per N_Port as a Recipient, which may be allocated across all three Classes. The total number of Concurrent Recipient Sequences which may be Open or Active across all three Classes by a single N_Port is specified in the Common Service Parameter field (see 17.6.3). The number of concurrent sequences open is specified in the following Table:

Bits

23	22	21	20	19	18	17	16	
0	0	0	0	0	0	0	0	= Reserved
0	0	0	0	0	0	0	1	= 1 Sequence Status Register
...	...	...	...	...	...	...	...	...
1	1	1	1	1	1	1	1	= 255 Sequence Status Registers

NOTE - If each N_Port specified 255 Concurrent Sequences, then the maximum number of Open Sequences between the two N_Ports is 510 (255 as Recipient and 255 as Initiator, for each N_Port).

Class 1

The SEQ_ID values shall range from 0 to L, inclusively, where L is the value of the Concurrent Sequence field. During a Class 1 Connection a N_Port shall support the maximum number of Concurrent Sequences specified during Login.

Class 2 and 3

The SEQ_ID values shall range from 0 to 255. In Class 2 a N_Port is allowed to respond with a P_BSY to a frame initiating a new Sequence if N_Port resources are not available.

17.6.8.7 End-to-end Credit

- **Word 2, Bits 14-0 End-to-end Credit (M)**

End-to-end Credit is the maximum number of Data frames, which can be transmitted by a N_Port without receipt of accompanying ACK, or Link_Response frames. The minimum value of end-to-end Credit is one. The end-to-end Credit field specified is associated with the number of buffers available for holding the Data_Field of a frame and processing the contents of that Data_Field by the N_Port supplying the Service Parameters. End-to-end Credit is not applicable to Class 3 since ACK frames are not used.

NOTE - In order to ensure frame identification integrity, end-to-end Credit is defined as a 15-bit field while sequence count is a 16-bit field. This ensures that end-to-end Credit can never exceed one-half of the maximum sequence count. Bit 15 shall be set = 0.

Values in the end-to-end Credit Field have the following meanings:

Bits

14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	= Reserved
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	= 1 Buffer

...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..	
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	= 32 766 Buff- ers
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	= 32 767 Buff- ers

17.6.8.8 Open Sequences per Exchange

- **Word 3, Bits 31-16 Open Sequences / Exchange (X)**

The value of Open Sequences per Exchange shall specify the maximum number of Sequences that can be Open at the Recipient at one time between a pair of N_Ports for one Exchange. The value of X+2 specifies the number of instances of Sequence Status that shall be maintained by the Recipient for a single Exchange in the Exchange Status Block. This value is used for Exchange and Sequence tracking. The value of X limits the link facility resources required for error detection and recovery (see clause 23). The value of X is specified in bits 23-16.

NOTE - The number of SSBs specified at X+2 to be retained in the ESB ensures that if Sequence streaming rules are followed, the ESB shall contain at least one "good" Sequence that ended normally. Another SSB position was allocated in order to allow for any race or timing conditions that might impact that "good" Sequence.

17.6.8.9 Class 6 Multicast RX_ID

When a N_Port performs Login with the Multicast Server, it shall specify the RX_ID value to be used by the Multicast Server when acknowledging the Connection Initiator. Word 3, bits 15-0 shall be the RX_ID value used by the Multicast Server.

17.6.9 Vendor Version Level

Vendor Version Level is a 16-byte field which specifies vendor-specific information. If Word 1, bit 29 = 1 in the Common Service Parameters, the Vendor Version Level field contains valid information. If Word 1, bit 29 = 0, the Vendor Version Level field is not meaningful.

17.6.10 Services Availability

This field is reserved.

17.7 F_Port Service Parameters

The first sixteen bytes of the Payload following the LS_Command Code in the Accept Sequence shall specify F_Port Service Parameters common to all Classes. The next eight bytes shall contain the F_Port_Name of the F_Port transmitting the Service Parameters. The next eight bytes shall contain the Fabric_Name of the Fabric, which controls the F_Port. The 64 bytes of the Payload following the Fabric_Name shall specify four sets of Class Service Parameters as shown in Table 201. The first 16-byte group shall specify Service Parameters for Class 1 and Class 6. The second 16-byte group shall specify the Service Parameters for Class 2. The third 16-byte group shall specify the Service Parameters for Class 3. The fourth 16-byte group shall specify the Service Parameters for Class 4. The next two 16-byte fields shall be reserved. If the Payload Length bit is set to 1, the next 8-byte field shall specify Services Availability, and the final 116 bytes of the Payload shall be reserved.

F_Port Service Parameters in the Accept Link Service Sequence have different interpretations than Login data supplied by a N_Port.

17.7.1 F_Port Common Service Parameters

F_Port Common Service Parameters represent Parameters, which are common across all Classes supported by a F_Port, and are shown in Figure 47.

Word 0	31	16	15	0
	FC-PH/FS Version HHHH HHHH LLLL LLLL		Buffer-to-buffer Credit BBBB BBBB BBBB BBBB	
1	31	16	15 12 BB_SC Number	11 0
	Common Features rrrN rRMB GDrY rHrP		SSSS	Buffer-to-buffer Rcv Data Field size FFFF FFFF FFFF
2	31	16	15	0
	R_A_TOV tttt tttt tttt tttt tttt tttt tttt tttt			
3	31	16	15	0
	E_D_TOV tttt tttt tttt tttt tttt tttt tttt tttt			

Figure 47 - F_Port Common Service Parameters - Fabric Login

17.7.1.1 FC-PH/FS version

The FC-PH/FS version field provides a method of determining the version of FC-PH/FS, which a F_Port shall be capable of supporting. The FC-PH/FS field is divided into two one-byte fields. Table 203 indicates the hexadecimal values for the low and high FC-PH or FS version levels in word 0 bits 31-24, and bits 23-16.

A version of FC-PH shall be assigned a new binary value when its implementation is not compatible with the previous version specified. A F_Port shall support all versions between the lowest and the highest levels specified.

- **Word 0, Bits 31 - 24 - Highest version supported (H)**
The binary value of bits 31-24 is encoded to represent the highest or most recent version of FC-PH or FS, which the F_Port shall be capable of supporting.
- **Word 0, Bits 23 - 16 - Lowest version supported (L)**
The binary value of bits 23-16 is encoded to represent the lowest or earliest version of FC-PH or FS that the F_Port shall be capable of supporting.
Where there is overlap in the FC-PH versions supported, each F_Port shall operate in the highest FC-PH or FS version mutually supported.

17.7.1.2 Buffer-to-buffer (F_Port) Credit

The buffer-to-buffer Credit (B) field (Word 0, bits 15-0) specified is associated with the number of buffers available for holding Class 1 connect-request, Class 2, or Class 3 frames received from the N_Port.

The buffer-to-buffer Credit (B) shall be a single value which represents the total buffer-to-buffer Credit available for all Classes. A N_Port or F_Port tracks buffer-to-buffer Credit as a single entity for all frames subject to buffer-to-buffer flow control (see clause 20).

Values in the buffer-to-buffer Credit field have the same format as in the end-to-end Credit Class Service Parameters.

17.7.1.3 Common features

- **Word 1, bits 31-29 - Reserved**
- **Word 1, Bit 28 - N_Port/F_Port (N)**

0 = N_Port

1 = F_Port

Word 1 bit 28 shall be set = 1.

- **Word 1, bit 27 - Reserved**
- **Word 1, bit 26 - E_D_TOV Resolution (R)**

0 = 1 ms

1 = 1 ns

Word 1, Bit 26 indicates the resolution of the E_D_TOV timer. If Word 1 Bit 26 = 0, then the timer shall be in increments of 1 ms. If Word 1 Bit 26 = 1, then the timer shall be in increments of 1 ns.

- **Word 1, bit 25 Multicast (M)**
- 0 = Multicast not supported by the Fabric
1 = Multicast supported by the Fabric

- **Word 1, bit 24 Broadcast (B)**
- 0 = Broadcast not supported by the Fabric
1 = Broadcast supported by the Fabric

- **Word 1, bit 23 - Hunt Group (G)**
- 0 = Hunt Groups not supported.
1 = Hunt Groups supported.

Word 1, Bit 23 indicates whether or not the Fabric supports Hunt Group routing. If Word 1 Bit 23 = 0, then the Fabric shall not support Hunt Group routing. If Word 1 Bit 23 = 1, then the Fabric shall support Hunt Group routing.

- **Word 1, bit 22 Dedicated Simplex (D)**
- 0 = Dedicated Simplex not supported by the N_Port
1 = Dedicated Simplex supported by the N_Port

- **Word 1, bit 21 – Reserved**

- **Word 1, bit 20 - Clock Synchronization Primitive Capable (Y)**

0 = not Clock Synchronization Primitive Capable

1 = Clock Synchronization Primitive Capable

Word 1, Bit 20 indicates support for the Primitive method of Clock Synchronization (see clause 32.3). If Word 1, Bit 20 = 0, the Fabric does not support the Primitive method of Clock Synchronization. If Word 1, Bit 20 = 1, the Fabric is capable of receiving the clock synchronization Primitive Signals and acting upon them.

- **Word 1, bit 19 - Reserved**
- **Word 1, bit 18 - Dynamic Half Duplex (Y)**

0 = DHD not supported
1 = DHD supported

Word 1, Bit 18 indicates the support for Dynamic Half Duplex. If Word 1 Bit 18 = 0, then the Dynamic Half Duplex is not supported. If Word 1 Bit 18 = 1, then the N_Port is indicating that it supports the reception of the DHD primitive.

NOTE - DHD is applicable to FC-AL topologies. See FC-AL-2.

- **Word 1, bit 17 - Reserved**
- **Word 1, bit 16- Payload Length (P)**

0 = 132 bytes
1 = 256 bytes

Word 1, Bit 16 indicates the length of the ACC payload. If Word 1 Bit 16 = 0, then the payload shall be 132 bytes. If Word 1 Bit 16 = 1, then the payload shall be 256 bytes.

17.7.1.4 BB_SC Number

- **Word 1, bits 15-12 - Buffer-to-buffer State Change Number (BB_SC_N) (S)**

The BB_SC_N is a binary value (bits 15-12) which indicates that the sender of the PLOGI or FLOGI frame is requesting $2^{BB_SC_N}$ number of frames to be sent between two consecutive BB_SCs primitives, and $2^{BB_SC_N}$ number of R_RDY primitives to be sent between two consecutive BB_SCr primitives.

When the two ports performing the login specifies different non-zero values of BB_SC_N, the larger value shall be used. If either port specifies a BB_SC_N value of zero, the BB_Credit recovery process shall not be performed, and no BB_SCx primitive shall be sent. See 20.5.12 for a description of the BB_Credit recovery process.

Note - If all frames or R_RDY primitives sent between two BB_SCx primitives are lost, $2^{BB_SC_N}$ number of BB_Credits are lost, and cannot be recovered by the scheme outlined in 20.5.12. Therefore BB_SC_N should be chosen so that the probability of losing $2^{BB_SC_N}$ number of consecutive frames or R_RDY primitives is deemed negligible. The recommended value of BB_SC_N is 8.

17.7.1.5 Buffer-to-buffer Data_Field size

- **Word 1, Bits 11 - 0 Buffer-to-buffer Receive Size (F)**

The buffer-to-buffer Receive Data_Field Size is a binary value (bits 11-0) which specifies the largest Data_Field Size for an FT_1 frame (11.4) that can be received by the F_Port supplying the Service Parameters as a Sequence Recipient for:

- a connect-request (SOFc1),
- a Class 2 Data frame, or
- a Class 3 Data frame.

Values less than 128 or greater than 2¹¹ are invalid. Values shall be a multiple of four bytes. A F_Port shall support a Data Field size of at least 128 bytes, however, a minimum of 256 bytes is recommended.

17.7.1.6 E_D_TOV

When Word 1, Bit 26 = 0, the E_D_TOV value shall be specified as a count of 1 ms increments. When Word 1, Bit 26 = 1, the E_D_TOV value shall be specified as a count of 1 ns increments. Therefore, based on the setting of Word 1, Bit 26, a value of hex '00 00 00 0A' specifies a time period of either 10 ms or 10 ns.

17.7.1.7 R_A_TOV

The R_A_TOV value shall be specified as a count of 1 ms increments. Therefore, a value of hex '0000000A' specifies a time period of 10 ms.

17.7.2 F_Port_Name

The F_Port_Name is an eight-byte field. The format of the F_Port_Name is specified in 13.3. Each F_Port shall provide a unique F_Port_Name within the address domain of the Fabric and associated N_Ports. Bits 63-60 specify the format of the F_Port_Name. The formats are summarized in Table 28 and Table 29 in 13.3.

17.7.3 Fabric_Name

The Fabric_Name is an eight-byte field. The format of the Fabric_Name is specified in 13.3. Each Fabric shall provide a unique Fabric_Name. Bits 63-60 specify the format of the Fabric_Name. The formats are summarized in Table 28 and Table 29 in 13.3.

17.7.4 F_Port Class Service Parameters

The 64 bytes of the Payload following the Fabric_Name shall specify four sets of Class Service Parameters as shown in Table 201. The first 16-byte group shall specify Service Parameters for Class 1 and Class 6. The second 16-byte group shall specify the Service Parameters for Class 2. The third 16-byte group shall specify the Service Parameters for Class 3. The fourth 16-byte group shall specify the Service Parameters for Class 4.

F_Port Class Service Parameters used during F_Port Login are illustrated in Figure 48.

Word	31	16	15	0
0	Service Options		Initiator Control	
	VISS	QDCB	PEEE	EEEE
	Not meaningful		Not meaningful	
1	31	16	15	0
	Recipient Control		Rcv Data Field size	
	rrrr	rrrr	rrrr	Yrrr
	Not meaningful		Not meaningful	
2	31	16	15	0
	Concurrent Sequences		N_Port End-to-end Credit	
	Not meaningful		Not meaningful	
3	31	16	15	0
	CR_TOV			
	tttt	tttt	tttt	tttt

Figure 48 - F_Port Class Service Parameters - Fabric Login

17.7.4.1 Class validity (V)

- **Word 0, Bit 31 - Class validity**

0 = Invalid (Class not supported)

1 = Valid (Class supported)

The Class validity bit shall indicate whether this Class is supported or not. If the Class validity bit is zero, it indicates that this group or set of sixteen bytes shall be ignored. If the Class va-

lidity bit is one, it indicates that this Class shall be supported. The Class shall be identified based on Class 1 = first sixteen-byte group, Class 2 = second sixteen-byte group, Class 3 = third sixteen-byte group, and Class 4 = fourth sixteen-byte group.

17.7.4.2 Service options

Service Options (E) shall specify Class of Service capabilities supported or required by the Fabric supplying the Service Parameters.

Service options shall specify optional features of a Class of Service supported by the N_Port supplying the Service Parameters.

• Word 0, Bit 30 - Intermix mode

- 0 = Intermix not supported
- 1 = Intermix supported

Class 1

All Fabrics supporting Class 1 shall support Exclusive Connections. A N_Port supporting Exclusive Connections may only transmit and receive frames from the N_Port to which an existing Class 1 Connection is established. Exclusive Connections require that the Fabric transmit an F_BSY frame, as appropriate, in response to frames issued by a third N_Port targeted for one of the two N_Ports engaged in a Class 1 Connection.

An Intermixed Dedicated Connection specifies that the Fabric may insert or extract Class 2 or Class 3 frames while a Class 1 Connection is established. Support for Intermix is optional by both a N_Port and a Fabric. If the N_Port supports intermixing of Class 2 and 3 frames during a Class 1 Connection, it shall request Fabric support for Intermix during Login with the Fabric. See 16.4 for more discussion on Intermix.

If both the N_Port and the F_Port have indicated support for Intermix Connections, then Intermix mode shall be functional.

N_Port		F_Port	
Word	Bit	Word	Bit
0	30	0	30
0		0	
0		1	
1		0	
1		1	
Neither supports			
Fabric is capable of supporting, Intermix not functional			
N_Port support requested, Fabric does not support			
N_Port requested, Fabric agrees, Intermix is functional			

Class 2 and 3

Word 0 bit 30 is reserved for Class 2 and 3.

Class 4

Word 0 bit 30 (Intermix Mode) has no meaning in Class 4 since the Fabric shall support Intermix on Class 4.

- **Word 0, Bits 29-28 - Stacked Connect-requests**

Support for stacked connect-requests is optional in both a Fabric and N_Port. Both a N_Port's and F_Port's behavior change if stacked connect-requests are functional (see 22.5.2 and 30).

Class 1

Word 0, bits 29 and 28 of the F_Port Class Service Parameters are used in the Fabric Login procedure. The ability to support the Stacked Connect-request function by a Fabric is indicated by bits 29 and 28 of word 0 of the F_Port service parameters during Fabric Login. Table 208 specifies the meaning of the combination of word 0 bits 29 and 28.

Table 208 - Stacked Connect-request support Login bit		
Service Parameter Word 0		Meaning
Bit 29	Bit 28	
0	0	Stacked Connect-request not supported
0	1	Lock-Down Mode
1	0	Transparent Mode
1	1	Reserved

Class 2 and 3

Word 0, bits 29 and 28 are reserved.

Class 4

Word 0, bits 29 and 28 (Stacked connect-request - Transparent and Lock-down modes) are reserved.

- **Word 0, Bit 27 - Sequential delivery**

0 = Out of order delivery

1 = Sequential delivery functional

If bit 27 is set to one by a N_Port, then it is requesting that all frames delivered to the N_Port requesting this function be delivered in the same order in which the frames were transmitted by the source N_Port. If bit 27 is set to one by the F_Port, then the F_Port is indicating that it shall deliver Class 2 and 3 frames in the same order as transmitted from any one N_Port.

A Fabric supporting the sequential delivery feature routes Class 2 and 3 frames via a fixed route through the Fabric in order to provide in-order frame delivery. This feature does not imply any other alteration to the normal class of service functions such as F_BSY.

If the N_Port has requested this support and the Fabric responds with bit 27 = 1, then in-order delivery is being supported by the Fabric. If the N_Port has not requested this support and the Fabric responds with bit 27 = 1, then in-order delivery is functional by the Fabric as default operation.

Class 1

Bit 27 has no meaning in Class 1 since the Fabric is required to deliver frames in the order transmitted based on class of service.

Class 2

In Class 2, if bit 27 = 1, the Fabric guarantees the order of delivery of both Data and Link_Control frames to correspond to the order in which the frames are transmitted.

Class 3

In Class 3, if bit 27 = 1, the Fabric guarantees the order of delivery of Data frames to correspond to the order in which the frames are transmitted.

N_Port	F_Port	
Word 0, Bit 27	Word 0, Bit 27	
0	0	Neither supports
0	1	Sequential delivery by the Fabric is functional
1	0	N_Port support requested, Fabric does not support
1	1	N_Port requested, Fabric agrees, Sequential delivery is functional

Class 4

Bit 27 has no meaning in Class 4 since the Fabric shall deliver frames in the order transmitted based on class of service.

- **Word 0, Bit 26 Dedicated Simplex (D)**

Class 1 (see clause 27)

0=Dedicated Simplex not supported

1=Dedicated Simplex supported

Class 2, 3 and 4

Word 0, bit 26 is reserved.

- **Word 0, Bit 25 Camp-On (C)**

Class 1 (see clause 29)

0=Camp-On not supported

1=Camp-On supported

Class 2, 3 and 4

Word 0, bit 25 is reserved.

- **Word 0, Bit 24 - Buffered Class 1 (B)**

0 = Buffered Class 1 not requested or Buffered Class 1 is not supported

1 = Buffered Class 1 requested or Buffered Class 1 is supported

See clause 31.

Buffered Class 1

Support for Buffered Class 1 Service is negotiated during Fabric Login. If a N_Port requests Buffered Class 1 Service and the attached F_Port supports it, only then is Buffered Class 1 Service enabled and available for use by the N_Port and Fabric. Table 205 summarizes the meaning of the combination of Word 0, bit 24.

Class 1, 2, 3 and 4

Word 0, bit 24 is reserved.

- **Word 0, Bit 23 - Priority (P)**

Class 1, 2, 3 and 4 (see 12.4.2)

0 = Priority/Preemption not supported

1 = Priority/Preemption supported

- **Word 0, Bit 22 – Preference**

0 = Normal delivery

1 = Preferred delivery functional

If bit 22 is set to one by a N_Port, then it is requested that all frames transmitted by the N_Port requesting this function be delivered according to the setting of the PREF field in the CS_CTL field of the Frame Header (see 16.2.2 and 16.3.2). If bit 22 is set to one by the F_Port, then the F_Port is indicating that it shall deliver Class 2 and 3 frames transmitted by the requesting N_Port according to the setting of the PREF field.

If the N_Port has requested this support and the Fabric responds with bit 22 = 1, then the Fabric shall support Preferred delivery. If the N_Port has not requested this support and the Fabric responds with bit 22 = 1, then Preferred delivery shall be functional by the Fabric as default operation. If the Fabric responds with bit 22 = 0, then Preferred delivery may be functional by the Fabric as default operation.

NOTE – A F_Port that responds with bit 22 = 0 may not itself support Preferred delivery, but other Fabric Elements in the path to the destination may support it. A N_Port may attempt Preferred delivery even if the F_Port does not indicate support.

Class 1

Bit 22 has no meaning in Class 1.

Class 2

In Class 2, if bit 22 = 1, the Fabric shall deliver both Data and Link_Control frames according to the setting of the PREF field in the CS_CTL field of the frame header.

Class 3

In Class 3, if bit 22 = 1, the Fabric shall deliver Data frames according to the setting of the PREF field in the CS_CTL field of the frame header.

The following summarizes the function of bit 22 for both Class 2 and Class 3:

N_Port		F_Port		
Word	0, Bit	Word	0, Bit	
	22		22	
	0		0	Preferred delivery by the Fabric may be functional
	0		1	Preferred delivery by the Fabric shall be functional
	1		0	N_Port support requested, Preferred delivery by the Fabric may be functional
	1		1	N_Port requested, Fabric agrees, Preferred delivery by the Fabric shall be functional

The following table summarizes the relationship between Preferred delivery and sequential delivery for both Class 2 and Class 3.

Preference Functional	Sequential delivery Functional	
0	0	Frames may be delivered in any order
0	1	Frames shall be delivered to a destination in the same order received from the source, PREF is ignored
1	0	Frames may be delivered in any order, but frames with PREF set to one may be delivered prior to frames with PREF set to zero
1	1	Frames with PREF set to one shall be delivered to a destination in the same order received from the source relative to each other, and may be delivered prior to frames with PREF set to zero; frames with PREF set to zero shall also be delivered to a destination in the same order received from the source relative to each other

Class 4

Bit 22 has no meaning in Class 4.

17.7.4.3 Initiator control

This field is not meaningful.

17.7.4.4 Recipient control

- **Word 1, Bits 31-20 Reserved**

These bits are reserved for specific use for Fabric features.

- **Word 1, bit 19 - Clock synchronization ELS capable (Y)**

Class 1,2,3, and 4 (see clause 32.2)

0 = Recipient does not have clock synchronization ELS capability

1= Recipient has clock synchronization ELS capability

Word 1, Bit 19 indicates support for the ELS method of Clock Synchronization. If Word 1, Bit 19 = 0, then the Fabric does not support the ELS method of Clock Synchronization. If Word 1, Bit 19 = 1, the Fabric is capable of receiving CSU ELS frames and acting upon them.

- **Word 1, Bits 18-16 Reserved**

These bits are reserved for specific use for Fabric features.

17.7.4.5 Receive Data_Field Size

This field is not meaningful.

17.7.4.6 Concurrent Sequences

This field is not meaningful.

17.7.4.7 N_Port End-to-end Credit

This field is not meaningful.

17.7.4.8 Open Sequences per Exchange

This field is not meaningful.

17.7.4.9 CR_TOV

The CR_TOV value shall be specified as a count of 1 ms increments. Therefore a value of hex '00 00 00 0A' specifies a time period of 10 ms. See 30.8.1 for CR_TOV usage. CR_TOV is applicable to Class 1. CR_TOV is not applicable to other Classes.

17.7.5 Vendor Version Level

This field is reserved

17.7.6 Services Availability

This field returns information regarding the Fabric's ability to route to the defined well-known addresses.

- **Word 0, bits 31-11 - Reserved**

- **Word 0, bit 10 - Multicast Server**

When set to 1, this bit shall indicate that the Fabric supports routing to the well-known Multicast Server address identifier (hex 'FF FF F5'). When set to 0, this bit shall indicate that the Fabric does not support routing to the well-known Multicast Server address identifier.

- **Word 0, bit 9 - Clock Synchronization Server**

When set to 1, this bit shall indicate that the Fabric supports routing to the well-known Clock Synchronization Server address identifier (hex 'FF FF F6'). When set to 0, this bit shall indicate that the Fabric does not support routing to the well-known Clock Synchronization Server address identifier.

- **Word 0, bit 8 - Security Key Distribution Server**

When set to 1, this bit shall indicate that the Fabric supports routing to the well-known Key Distribution Server address identifier (hex 'FF FF F7'). When set to 0, this bit shall indicate that the Fabric does not support routing to the well-known Key Distribution Server address identifier.

- **Word 0, bit 7 - Alias Server**

When set to 1, this bit shall indicate that the Fabric supports routing to the well-known Alias Server address identifier (hex 'FF FF F8'). When set to 0, this bit shall indicate that the Fabric does not support routing to the well-known Alias Server address identifier.

- **Word 0, bit 6 - Quality of Service Facilitator**

When set to 1, this bit shall indicate that the Fabric supports routing to the well-known Quality of Service Facilitator address identifier (hex 'FF FF F9'). When set to 0, this bit shall indicate that the Fabric does not support routing to the well-known Quality of Service Facilitator address identifier.

- **Word 0, bit 5- Management Server**

When set to 1, this bit shall indicate that the Fabric supports routing to the well-known Management Server address identifier (hex 'FF FF FA'). When set to 0, this bit shall indicate that the Fabric does not support routing to the well-known Management Server address identifier.

- **Word 0, bit 4- Time Server**

When set to 1, this bit shall indicate that the Fabric supports routing to the well-known Time Server address identifier (hex 'FF FF FB'). When set to 0, this bit shall indicate that the Fabric does not support routing to the well-known Time Server address identifier.

- **Word 0, bit 3 - Directory Server**

When set to 1, this bit shall indicate that the Fabric supports routing to the well-known Directory Server address identifier (hex 'FF FF FC'). When set to 0, this bit shall indicate that the Fabric does not support routing to the well-known Directory Server address identifier.

- **Word 0, bits 2-0 - Reserved**

- **Words 1-7 - Reserved**

17.8 Procedure to estimate end-to-end Credit

17.8.1 Introduction

An estimate of the minimum end-to-end Credit between a N_Port pair for a given distance helps achieve the maximum bandwidth utilization of the channel, by continuously streaming data. The procedure to estimate end-to-end Credit is defined to accomplish this purpose.

Link Service Sequences which support this procedure are optional. This procedure shall be performed after Login between this N_Port pair. Login determines a number of Service Parameters such as the maximum frame size that can be received by each N_Port.

Applicability

The procedure is applicable to both Class 1 and Class 2. In Class 2, the procedure and the continuous streaming function may also be limited by the buffer-to-buffer Credit.

Users

The procedure shall be invoked by the Link Service support of the source N_Port and responded to by the Link Service support of the destination N_Port. Since the Extended Link Service requests used to perform this procedure are optional, LS_RJT may be received to any request with a reason code of Command Not Supported

Prerequisite

To perform this procedure for Class 1 or Class 1 Intermix, a Class 1 Connection shall have been established before the procedure is performed.

17.8.2 Procedure steps

This procedure is optional and consists of following three request Sequences.

1. Establish Streaming Sequence
2. Estimate Credit Sequence
3. Advise Credit Sequence

The procedure is illustrated with these request Sequences and their respective reply Sequences in Figure 49.

The procedure shall be performed for Class 1 or Class 2 with respective delimiters, as defined in clause 16.

17.8.2.1 Establish Streaming Sequence

This Sequence shall be used to obtain an end-to-end Credit large enough to perform continuous streaming from a source N_Port to a destination N_Port. This Sequence provides an opportunity for the destination N_Port to communicate the maximum end-to-end Credit it can provide for the purposes of streaming. This temporary allocation is termed Streaming Credit (L).

This Sequence shall be used between a N_Port pair after the N_Port pair have logged in with each other. This Sequence shall be initiated as a new Exchange. The Originator of the Exchange shall initiate the Sequence.

- The source shall transmit the Establish Streaming (ESTS) frame.
- The destination shall reply with an ACC frame.
- The Class Validity bit = 1 shall identify the Class which is requesting streaming Credit.
- The Payload of ACC shall have the same format as the Service Parameters for N_Port_Login. The Payload shall contain Streaming Credit (L) allocated in the end-to-end Credit field of the appropriate Class of the Service Parameters for which the Estimate Credit procedure is being performed (word 2, bits 14-0 of each class group). The receiver shall ignore the other fields.

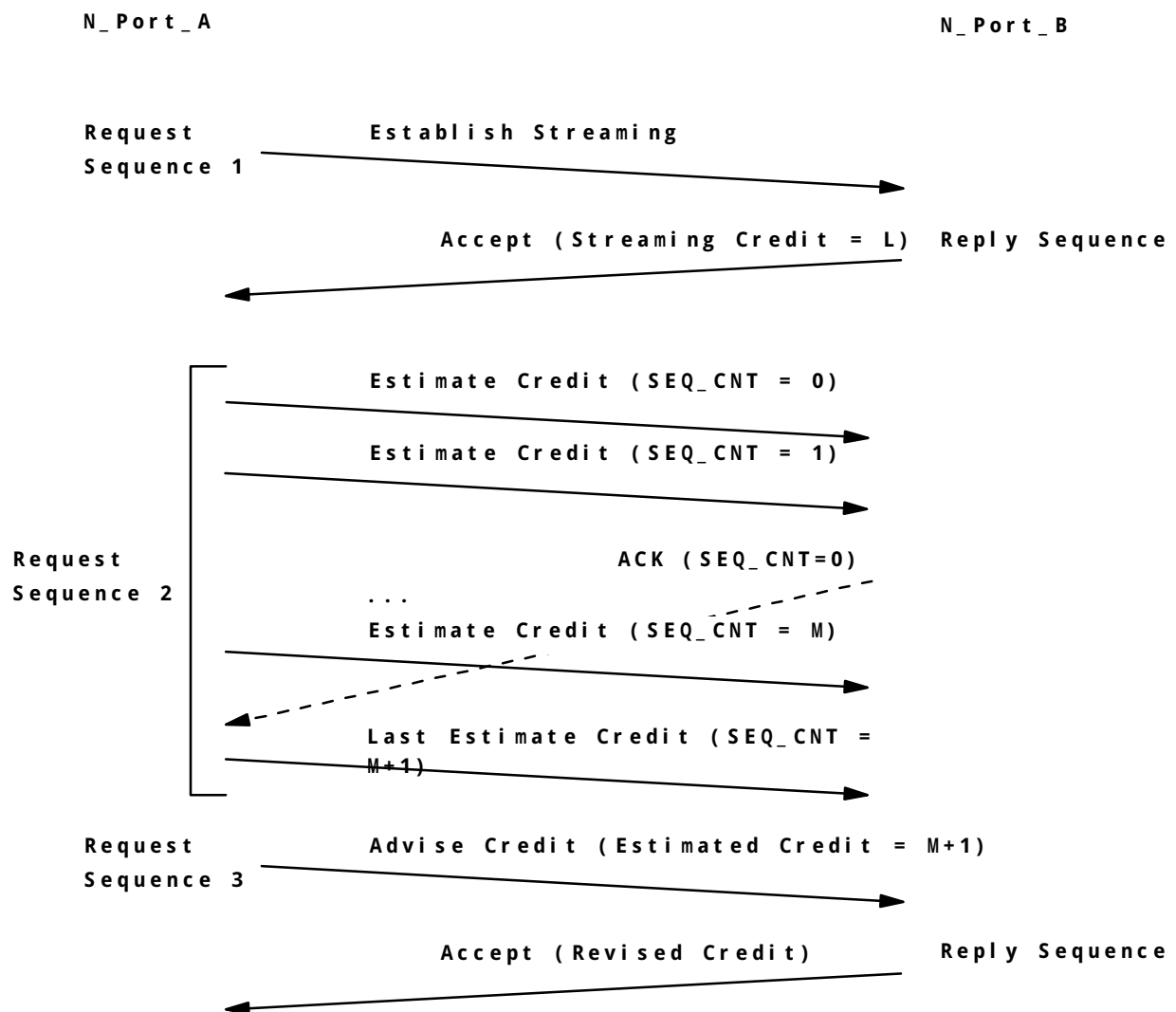


Figure 49 - Procedure to estimate end-to-end Credit

17.8.2.2 Estimate Credit Sequence

This Sequence shall be performed immediately following the completion of the Establish Streaming Sequence. This Sequence requires the use of ACK_ and may not be executed by all N_Ports.

- a) The source N_Port shall stream Estimate Credit (ESTC) frames consecutively until it receives the first ACK (ACK_1) from the destination N_Port which shall set the Abort Sequence bits (F_CTL bits 5-4) = 1 0. The source shall not exceed the Streaming Credit obtained during the Establish Streaming Sequence.
- b) If the source does not receive ACK (ACK_1) after it has reached the limit imposed by the Streaming Credit value, it shall stop streaming and wait for the first ACK to be received with the Abort Sequence bits (F_CTL bits 5-4) = 1 0.
- c) The size of the Data Field of the ESTC frame shall be the normal size frames transmitted by an FC-4 based on the Service Parameters from N_Port_Login.
- d) The Payload shall contain valid data bytes.
- e) The SEQ_CNT shall follow the normal rules for Sequence transmission.
- f) The destination N_Port shall respond with ACK for Data frames received.
- g) If the highest SEQ_CNT transmitted by the source N_Port at the time it receives the first ACK is M, the number of outstanding frames (i.e., Credit estimated for continuous streaming) shall equal M+1. If ACK is received within the Streaming Credit limit ($L > M$), this value of M+1 represents the minimum Credit required to utilize the maximum bandwidth of the Fibre. If the ACK is received after reaching the Streaming Credit limit (L), this value is less than the optimal Credit required to utilize the maximum bandwidth of the Fibre.
- h) The source N_Port shall follow all the rules in closing the Sequence, by sending the last Data frame of the Sequence and waiting for corresponding ACK to be received.

17.8.2.3 Advise Credit Sequence

This Sequence shall be performed immediately following completion of the Estimate Credit Sequence. The source N_Port that performed the Estimate Credit Sequence shall advise the destination N_Port of the Estimated Credit in ADVC Data Field. The destination N_Port shall reply using an ACC frame, with a revised end-to-end Credit value in its Payload. This value is determined by the destination N_Port based on its buffering scheme, buffer management, buffer availability and N_Port processing time. This is the final value to be used by the source N_Port for revised end-to-end Credit.

This Sequence provides a complementary function to Login. In contrast to the Login frame, the ADVC frame contains the end-to-end Credit it would like to be allocated for continuous streaming.

If the Estimated Credit (M+1) is less than or equal to the Streaming Credit (L), the destination may choose to reallocate the estimated end-to-end Credit. If the Streaming Credit (L) is smaller than needed for continuous streaming, the source N_Port is bound to run short of end-to-end Credit and the source N_Port may advise that value as the Estimated Credit.

- a) The source N_Port shall transmit Advise Credit frame with the Estimated Credit (M+1).
- b) The Payload of the ADVC shall have the same format as the Service Parameters for Login. The Payload shall contain the Estimated Credit (M+1) in end-to-end Credit field of the appropriate Class of the Service Parameters. The appropriate Class is identified by the Class Validity bit = 1. The receiver shall ignore the other fields.
- c) The destination N_Port shall determine the revised end-to-end Credit value. The destination shall determine the value based on its buffer management, buffer availability and port processing time and may add a factor to the Estimated Credit value. This is the final value to be used by the source N_Port for end-to-end Credit.

- d) The destination N_Port replies with an ACC frame that successfully completes the Protocol. The ACC Sequence shall contain the end-to-end Credit allocated to the source N_Port. The Payload of ACC shall have the same format as the Service Parameters for Login. The Payload shall contain the final end-to-end Credit in end-to-end Credit field of the appropriate Class of the Service Parameters. The appropriate Class is identified by Class Validity bit = 1. The receiver shall ignore the other fields.

Asymmetry

Since the maximum frame size is permitted to be unequal in forward and reverse directions, the Estimate Credit procedure may be performed separately for each direction of transfer. Credit modification applies only to the direction of the transfer of Estimate Credit frames.

Frequency

The Estimate Credit procedure provides an approximation of the distance involved on a single path. If there are concerns that in a Fabric in which the length (and time) of the paths assigned can vary, the procedure may be repeated several times to improve the likelihood that the Estimated end-to-end Credit value is valid.

Alternatively, a source may accept the Estimated end-to-end Credit value. If, at a later time, data transfers are unable to stream continuously, the source may re-initiate the Estimate Credit Procedure, or arbitrarily request an increase in Estimated end-to-end Credit by using an ADVN Link request Sequence.

18 Exchange, Sequence, and sequence count management

18.1 Introduction

An Exchange is the fundamental mechanism for coordinating the interchange of information and data between two N_Ports or nodes. All Data transmission shall be part of an Exchange. This clause discusses Exchanges between N_Ports. FC-FS does not address the means to manage Exchanges across multiple N_Ports within a Node.

Data frame transfer

Transfer of information between two N_Ports is based on transmission of a Data frame by a source N_Port with an ACK response frame from the destination N_Port or N_Ports receiving the Data frame to acknowledge Data frame delivery as appropriate (Class 1, 2 and 6).

Sequence

A Sequence is a set of one or more related Data frames transmitted unidirectionally from one N_Port to another N_Port within an Exchange. In Class 1, 2 and 6, an ACK_1 frame is transmitted in response to each Data frame or a single ACK_0 is transmitted for all Data frames of a Sequence. A Sequence is assigned a SEQ_ID (Sequence_Identifier) by the Sequence Initiator. A Sequence shall only be initiated when a N_Port holds the Sequence Initiative for a given Exchange.

Streamed Sequences

FC-FS allows a N_Port to initiate a new Sequence for the same Exchange while it already has Sequences Open for that Exchange. The new Sequence is termed a streamed Sequence. See 12.7 for more information regarding the assignment of SEQ_IDs for additional rules when streaming Sequences.

Sequence count

Each frame within a Sequence contains a sequence count (SEQ_CNT) which represents the sequential number of each Data frame within one or multiple Sequences transmitted by an Exchange Originator or Responder. An ACK response frame contains a sequence count that is set equal to the Data frame sequence count to which it is responding (Class 1 and 2).

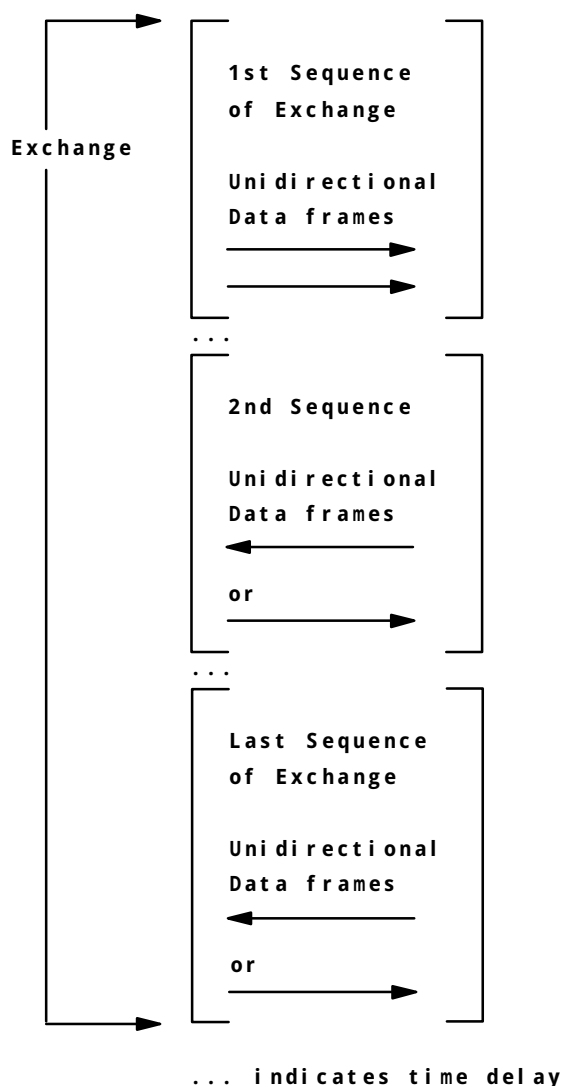


Figure 50 - Exchange - Sequence relationship

Exchange

An Exchange is a set of one or more related Sequences. Sequences for the same Exchange may flow in the same or opposite direction between a pair of N_Ports but not simultaneously (i.e., Data flows in one direction at a time within an Exchange for a single N_Port pair). Therefore, an Exchange may be unidirectional or bi-directional. Within a single Exchange only one Sequence shall be Active at any given time for a single initiating N_Port. That is, a Sequence Initiator shall complete transmission of Data frames for a Sequence of an Exchange before initiating another Sequence for the same Exchange. In Class 1, an Exchange may span multiple or successive Dedicated Connections at Sequence boundaries.

An Exchange may utilize multiple Classes of Service. Separate Sequences for the same Exchange may be transmitted in both Class 1 and 2. Class 3 Sequences shall not be transmitted in the same Exchange with Class 1 or 2 Sequences. A Sequence Initiator shall not stream Sequences that are in different Classes of Service.

NOTE - In Class 2, when Sequences are streamed, a Recipient N_Port may see multiple Active Sequences for the same Initiator because of out of order delivery.

The Sequence Initiator shall use continuously increasing SEQ_CNT if Sequences are streamed. If the Sequence Initiator does not stream Sequences, it may also use continuously increasing SEQ_CNT to allow the Sequence Recipient to track delivery order.

In the Discard multiple Exchange Policy, the Sequence Recipient shall deliver consecutive Sequences within an Exchange in the order transmitted. The Sequence Recipient shall preserve transmission order from one Sequence to the next even if the Sequence Initiator does not use continuously increasing SEQ_CNT. Should frames arrive out of order, the Sequence Recipient may delay transmission of the last ACK until the order is re-established.

An Exchange is assigned an Originator Exchange_ID (OX_ID) by the Originator and a Responder Exchange_ID (RX_ID) by the Responder facilities within the N_Ports specified by the N_Port Identifiers involved in the Exchange. When an Exchange is originated, there is a binding of resources in both the Originator and Responder.

An Exchange Status Block exists throughout the life of an Exchange and is located by using one or more fields of the Sequence_Qualifier, such as a N_Port's X_ID. Since an Exchange utilizes link resources, a N_Port may choose to invalidate and, subsequently, reassign its X_ID value at certain Sequence boundaries during an Exchange.

Sequence Initiative

The Exchange Originator is the Initiator of the first Sequence of the Exchange and holds the initiative to transmit Sequences. At the end of each Sequence of the Exchange, the Initiator of the Sequence may transfer the initiative to transmit the next Sequence to the other N_Port, or it may retain the initiative to transmit the next Sequence.

18.2 Applicability

Class 1 and 2:

For Class 1 and 2, FC-2 manages:

- Activation and deactivation of Exchanges,
- Initiation and termination of Sequences,
- Assignment of X_IDs,
- Sequence Initiative,
- Assignment of SEQ_IDs,
- Segmentation and Reassembly,
- Sequences,
- Sequence count of frames,
- Detection of frame Sequence errors, and
- Notification of frame Sequence errors.

For Class 1, the Sequence Initiator shall assign SEQ_IDs from 0 to L where L is the number of Concurrent Sequences supplied by the Recipient N_Port.

NOTE - The Sequence Recipient may assign the SEQ_ID directly to a Recipient Sequence Status Block on an indexed basis.

A Class 1 Sequence requires a Class 1 Dedicated Connection for the duration of the Sequence. When the Connection is terminated, all Open Class 1 Sequences are terminated. A Sequence,

which has been abnormally terminated, is left in an indeterminate state at the Upper Level Protocol.

For Class 2, the Sequence Initiator shall assign SEQ_IDs from 0 to 255. The Sequence Recipient assigns the SEQ_ID to an available Recipient Sequence Status Block.

For Class 3, the Sequence Initiator shall assign SEQ_IDs from 0 to 255. FC-2 manages the same functions as Class 1 and 2 excluding notification of frame Sequence errors to the Sequence Initiator.

18.3 Summary rules

These rules summarize the guidelines for Exchange and Sequence management. The sections that follow elaborate on these summary rules and take precedence over these guidelines.

18.3.1 Exchange management

- a) Over the life of an Exchange, the Sequence Recipient shall deliver Data to FC-4 or an upper level on a Sequence basis.
- b) In the Discard multiple Sequences Error Policy, each Sequence shall be delivered in the order in which the Sequence was transmitted relative to other Sequences transmitted for the Exchange.
- c) In the Discard multiple Sequences Error Policy, a Sequence shall be deliverable if the Sequence completes normally and the previous Sequence, if any, is deliverable.
- d) In the Discard multiple Sequences with retransmission Error Policy, each Sequence shall be delivered in the order in which the Sequence was transmitted relative to other Sequences transmitted for the Exchange.
- e) In the Discard multiple Sequences with retransmission Error Policy, a Sequence shall be deliverable if the Sequence completes normally and the previous Sequence, if any, is deliverable.
- f) In the Discard a single Sequence Error Policy, each Sequence shall be delivered in the order in which the Sequence was received relative to other Sequences received for the Exchange.
- g) In the Discard a single Sequence Error Policy, a Sequence shall be deliverable if the Sequence completes normally without regard to the deliverability of other Sequences within the same Exchange.
- h) In all Discard policies, a Sequence is complete with regard to Data content if all valid Data frames for the Sequence were received without rejectable errors being detected.
- i) In Process policy with infinite buffers under Class 1 operation, a Sequence is complete with regard to Data content if at least the first and last Data frames were received as valid frames without rejectable errors being detected. For Class 3 Process policy with infinite buffers, a sequence is complete if a frame of another sequence is received or E_ D_TOV expires before the last frame of the current sequence is received.
- j) The ordering relationship and deliverability of Sequences between two separate Exchanges is outside the scope of FC-FS. Certain specific cases of Basic Link Service and Extended Link Service do, however, specify collision cases such as FLOGI, PLOGI, ABTX, and RSI.

18.3.2 Exchange origination

- a) An Exchange being originated for Extended Link Services before Login is complete or for the purpose of Login shall follow default Login parameters and special Extended Link Services rules specified in 15.1.1 and 15.1.2 as well as clause 17.
- b) A new Exchange, other than Extended Link Services, may be originated if three conditions are met:

350

- 1) The originating N_Port has performed Login with the destination N_Port.
- 2) The originating N_Port has an Originator_Exchange_Identifier (OX_ID) and Exchange resources available for use.
- 3) The originating N_Port is able to initiate a new Sequence.
- c) Each frame within the first Sequence of an Exchange shall set the First_Sequence F_CTL bit to one.
- d) The first frame of the first Sequence of the Exchange shall specify the Error Policy for the Exchange in F_CTL bits 5-4 of the Frame_Header. The Exchange Error Policy shall be consistent with the error policies supported by both the Originator and Responder.
- e) The Originator shall transmit the first Data frame of the Exchange with its assigned OX_ID and an unassigned RX_ID of hex 'FF FF'.
 - 1) If the Responder requires X_ID interlock (Login), the Originator (and Sequence Initiator) shall not transmit additional Data frames for this Exchange until the ACK to the first frame of the Exchange is received. The RX_ID in the ACK frame shall be used in subsequent frames of the Exchange.
 - 2) If the Responder (Login) does not require X_ID interlock, the Originator may transmit additional frames of the Sequence. In Class 1 and 2, the Responder shall return its X_ID no later than in the ACK corresponding to the last Data frame of the Sequence. The next Sequence of the Exchange shall contain both the OX_ID and RX_ID assigned in the previous Sequence.
- f) In Class 1 and 2, the Sequence Initiator shall receive at least one ACK from the Recipient before the Initiator attempts to initiate subsequent Sequences for the Exchange.
- g) The rules specified in Sequence initiation and termination specify the method for assigning X_IDs.

18.3.3 Sequence delimiters

For a more complete description of Data frame and Link_Control delimiters see Table 33 and Table 35. The following rules summarize the management of frame delimiters within a Sequence:

- A Sequence shall be initiated by transmitting the first frame with an SOFix, or SOFc1.
- Intermediate frames within a Sequence shall be transmitted with SOFnx and EOFn.
- The Sequence shall be complete when an EOFt (or EOFdt) has been transmitted or received for the appropriate SEQ_ID and all previous Data frames and ACKs (if any) have been accounted for by the Initiator and Recipient, respectively.

18.3.4 Sequence initiation

- a) A new Sequence may be initiated if three conditions are met:
 - 1) The initiating N_Port holds the initiative to transmit (Sequence Initiative).
 - 2) The initiating N_Port has a SEQ_ID available for use.
 - 3) The total number of Active Sequences initiated by the initiating N_Port with the Recipient N_Port does not exceed any of the following:
 - total concurrent Sequences (see 17.6.3.5),
 - concurrent Sequences per Class (see 17.6.8.6),
 - Open Sequences per Exchange (see 17.6.8.8).
- b) A SOFix (or a SOFc1 for the first frame establishing a Class 1 Connection) shall start the first Data frame of the Sequence.

- c) The Sequence Initiator shall assign a SEQ_ID that is unique between the Initiator and Recipient N_Port Identifier pair while the Sequence is Open. The SEQ_ID shall not match the last SEQ_ID transmitted by the Sequence Initiator for this Exchange for the current Sequence Initiative.

For streamed Sequences for the same Exchange, the Sequence Initiator shall use X+1 different subsequent SEQ_IDs where X is the number of Open Sequences per Exchange so that the Exchange Status Block contains status of the last deliverable Sequence.

- d) The Sequence Initiator shall not initiate the (X+1)th streamed Sequence until the first Sequence status is known. For example, if X=3 and the Sequence Initiator transmits SEQ_ID = 3, then 4, then 7, it shall not initiate another Sequence for the same Exchange until it resolves the completion status of SEQ_ID = 3, regardless of the completion status of SEQ_ID = 4 or 7.
- e) The Sequence_Qualifier shall be unique until an Open Sequence is ended normally or until the Recovery_Qualifier is determined by the Abort Sequence Protocol (ABTS).
- f) In Class 2 and 3 each Data frame of the Sequence shall be limited in size to the lesser of the F_Port and destination N_Port capabilities as specified by Login.
- g) In Class 1 the connect-request frame shall be limited in size to the lesser of the F_Port and destination N_Port capabilities as specified by Login.
- h) The Sequence Recipient shall supply the status of the Sequence in the format of the Sequence Status Block defined in 18.8.2 in response to a Read Sequence Status Extended Link Service request while the Sequence is Active. Sequence status shall be associated with the Exchange in which the Sequence is being transmitted.
- i) Frame transmission shall follow Flow Control Rules as specified in clause 20 and Connection rules as specified in clause 22.

18.3.5 Sequence management

The Sequence Recipient and the Sequence Initiator should verify that frames received for a Sequence adhere to the items listed. If the Sequence Recipient determines that one of the following conditions is not met in Class 1 or Class 2, it shall transmit a P_RJT. If the Sequence Initiator determines that one of the following conditions is not met, it shall abort the Sequence (Abort Sequence Protocol).

- a) Each frame shall contain the assigned SEQ_ID and assigned or reassigned OX_ID and RX_ID values.
- b) Hex 'FF FF' shall be used for unassigned X_ID values.
- c) Each frame shall indicate the Exchange Context.
- d) Each frame shall indicate the Sequence Context.
- e) Each frame shall contain a SEQ_CNT, which follows the Sequence Count rules as defined (see 18.3.6).
- f) Frame transmission shall follow Flow Control Rules as specified in clause 20.
- g) The Data Field size of each frame of the Sequence shall be less than or equal to:
 - 1) the maximum size specified by the Fabric (SOFc1, Class 2, or 3), if present, or
 - 2) the maximum size specified by the destination N_Port in the Service Parameters defined during Login, whichever is smaller.
- h) A Sequence shall be transmitted in one Class.
- i) Each Data frame in a Sequence shall be transmitted within an E_D_TOV timeout period of the previous Data frame transmitted within the same Sequence. Otherwise, a Sequence timeout shall be detected.

18.3.6 Sequence count

Within a Data frame Sequence, SEQ_CNT is used to identify each Data frame for verification of delivery and transmission order. The following rules specify the sequence count of each frame of a Sequence:

- a) The sequence count of the first Data frame of the first Sequence of the Exchange transmitted by either the Originator or Responder shall be binary zero.
- b) The sequence count of each subsequent Data frame within the Sequence shall be incremented by one from the previous Data frame.
- c) The sequence count of the first Data frame of a streamed Sequence shall be incremented by one from the last Data frame of the previous Sequence.
- d) The sequence count of the first Data frame of a non-streamed Sequence may be incremented by one from the last Data frame of the previous Sequence or may be binary zero.
- e) The sequence count of each Link_Response shall be set to the sequence count of the Data frame to which it is responding (Class 1 and 2).
- f) The sequence count of each ACK_1 frame shall be set to the sequence count of the Data frame to which it is responding (Class 1 and 2). See 20.4.3.2 for ACK_1 rules.
- g) The sequence count of each ACK_0 frame shall be set to the sequence count of the last Data frame transmitted (End_Sequence = 1) for the Sequence (Class 1 and 2). See 20.4.3.1 for ACK_0 rules.
- h) If infinite buffers and ACK_0 are being used for Sequences in which the SEQ_CNT is allowed to wrap, frame uniqueness is not being ensured (see 17.6.8.7 and 20.4.3.1 for ACK_0 rules).
- i) Within an acknowledged class of service, the Sequence Count of any frame shall not be reused until that frame is acknowledged.

18.3.7 Normal ACK processing

- a) Based on N_Port_Login parameters (Initiator support indicated in Initiator Control and Recipient support in Recipient Control in Class Service Parameters), if both N_Ports support multiple ACK forms, ACK_0 usage shall take precedence over ACK_1. ACK_0 use may be asymmetrical between two N_Ports (see 17.6.8.3 and 17.6.8.4).
- b) Mixing ACK forms in a Sequence is not allowed.
- c) ACK_0 may be used for both Discard and Process Error Policies. A single ACK_0 per Sequence shall be used to indicate successful Sequence delivery or to set Abort Sequence Condition bits to a value other than 0 0. An additional ACK_0 shall be used within a Sequence to
 - 1) perform X_ID interlock or
 - 2) respond a connect-request
 ACK_0 shall not participate in end-to-end Credit management.
- d) ACK frames may be transmitted in the order in which the Data frames are processed and need not be transmitted in SEQ_CNT order, however, the History bit (bit 16) of the Parameter Field shall indicate transmission status of previous ACK frame transmission for the current Sequence.
- e) The final ACK (EOft) of a Sequence shall be transmitted according to the rules for normal Sequence completion (see 18.3.8) in the absence of detected errors.
- f) ACK_1 or an ACK_0 frame, shall be transmitted during X_ID interlock, (see 18.5.4) and in response to a connect-request (see 22.4.1).
- g) If a Sequence Recipient receives a Data frame in Class 2 which falls within the SEQ_CNT range of a Recovery_Qualifier, it shall discard the Data Field of the frame and shall not de-

liver the Payload. The Sequence Recipient may transmit an ACK for the corresponding Data frame.

- h) If a Sequence Initiator receives an ACK for a Data frame in Class 2 which falls within the SEQ_CNT range of a Recovery_Qualifier, it shall discard and ignore the ACK frame.
- i) See 20.4.3.1 and 20.4.3.2 for the role of acknowledgement frames (ACK) in flow control.
- j) Each ACK shall be transmitted within an E_D_TOV timeout period of the event, which prompts the initiative to transmit an ACK frame. For example, when using ACK_1, it shall be transmitted within E_D_TOV of the Data frame reception. Whereas, when using ACK_0, it shall be transmitted within E_D_TOV of receiving the last Data frame of the Sequence.

18.3.8 Normal Sequence completion

- a) The Last Data frame of a Sequence shall be indicated by setting the F_CTL End_Sequence bit (F_CTL Bit 19) to one.
- b) An Exchange Event shall be defined when the End_Sequence bit (Bit 19) is set = 1, and any of the following F_CTL bits are set as indicated:
 - 1) Sequence Initiative (Bit 16) = 1,
 - 2) Continue Sequence Condition (Bits 7-6) = 1 1,
 - 3) Last Sequence (Bit 20) = 1,
 - 4) Chained Sequence (Bit 17) = 1, or
 - 5) End_Connection (Bit 18) = 1.
- c) A Sequence Event shall be defined when the End_Sequence bit (Bit 19) is set = 1, and the Continue Sequence Condition bits (Bits 7-6) in F_CTL are set to 0 0, 0 1, or 1 0 in the absence of an Exchange Event.
- d) In Class 1 the Sequence Initiator shall consider a Sequence as deliverable and complete when it receives the final ACK for the Sequence (ACK with EOFt or EOFdt delimiter).
- e) In Class 2 the Sequence Initiator shall consider a Sequence as deliverable (to the ULP) and complete when it receives the final ACK for the Sequence (ACK with EOFt delimiter). However, the Sequence Initiator shall account for all ACKs before reusing the SEQ_ID for this Exchange.
- f) For Class 3 Sequences, FC-FS provides no deliverability guarantees.
- g) A Class 1 or 2 Sequence shall be considered complete by the Sequence Recipient if
 - 1) all Data frames are received,
 - 2) no Sequence errors are detected, and
 - 3) acknowledgements, if any, prior to acknowledgment of the last Data frame received have been transmitted.
- h) A Class 3 Sequence shall be considered complete by the Sequence Recipient if
 - 1) all Data frames are received,
 - 2) no Sequence errors are detected, and
 - 3) An EOFt terminates the last Data frame.
- i) For a normally completed Sequence in Class 1, the Sequence Recipient shall transmit an ACK frame (ACK_1 or ACK_0) with EOFt (or EOFdt) in response to the last Data frame of the Sequence (End_Sequence bit in F_CTL = 1) when the Sequence is deliverable. The End_Sequence bit in F_CTL of the ACK shall be set to one.
 - 1) In Discard multiple Sequences Error Policy, the Sequence is deliverable when all preceding ACK frames have been transmitted and the previous Sequence, if any, is deliverable.

- 2) In Discard a single Sequence Error Policy, the Sequence is deliverable when all preceding ACK frames have been transmitted without regard to a previous Sequence.
- j) In Class 2, if the last Data frame (End_Sequence = 1) transmitted is the last Data frame received for the Sequence, or if the last Data frame (End_Sequence = 1) received indicates an Exchange event (item b), the Sequence Recipient shall operate in the same manner as in Class 1 as specified in item 9.
- k) In Class 2, if the last Data frame (End_Sequence = 1) transmitted is not the last Data frame received and the last Data frame (End_Sequence = 1) transmitted indicates a Sequence Event, the Sequence Recipient may either:
 - 1) withhold transmission of the ACK corresponding to the last Data frame transmitted until all previous ACKs have been transmitted and the Sequence is deliverable, or
 - 2) save the value of End_Sequence (Bit 19) and Continue Sequence Condition (Bits 7-6), and transmit the ACK corresponding to the last Data frame with EOF_n with End_Sequence and Continue Sequence bits set to zeros. When the last missing Data frame of the Sequence is received and the Sequence is deliverable, the Sequence Recipient shall transmit an ACK with EOF_t with the value of End_Sequence and Continue Sequence Condition bits saved from the Data frame containing the End_Sequence bit = 1 with the SEQ_CNT and other fields which match the corresponding Data frame.

NOTE - When Sequences are being streamed in Class 2 with out of order delivery, transmission of ACK (EOF_n) in response to the last Data frame of the Sequence (End_Sequence = 1) avoids costing the Initiator an extra Credit of one for the last Data frame of the Sequence while the Sequence Recipient waits for the last frame to be delivered.

- l) In Class 1 or 2 the Sequence Initiator shall transmit the last Data frame with an EOF_n.
- m) In Class 3 the Sequence Initiator shall transmit the last Data frame with an EOF_t.
- n) In the last Data frame of a Sequence with the Sequence Initiative held, the Sequence Initiator may set the Continue Sequence bits in F_CTL to indicate that the next Sequence will be transmitted immediately (0 1), soon (1 0), or delayed (1 1). If Continue Sequence bits are set to 1 1, then the Sequence Initiator shall wait until the final ACK (EOF_t) is received before initiating a new Sequence for this Exchange.
- o) In the last Data frame of a Sequence, the Sequence Initiator shall set the
 - 1) Sequence Initiative bit in F_CTL to 0 to hold Sequence Initiative.
 - 2) Sequence Initiative bit in F_CTL to 1 to transfer Sequence Initiative.
- p) In Class 1 and 2, the Sequence Initiative is considered to be passed to the Sequence Recipient when the Sequence Initiator receives the final ACK (EOF_t) of the Sequence with the Sequence Initiative bit = 1.
- q) In the last Data frame of a Class 1 Sequence, the Sequence Initiator may indicate its desire to end the Dedicated Connection by setting the End_Connection (bit 18) = 1 in order to begin the procedure to remove the Dedicated Connection (see 22.7.2).
- r) In the last Data frame of a Class 1 Sequence, the Sequence Initiator may indicate that it requires a reply Sequence be transmitted within the existing Dedicated Connection by setting the Sequence Initiative bit to one and Chained_Sequence bit to one in F_CTL.
- s) After a Sequence is complete, the Sequence Recipient shall supply the status of the Sequence in the format of the Exchange Status Block defined in 18.8.1 in response to a Read Exchange Status Link Service command. Sequence status in the Exchange Status Block is available until X+2 Sequences have been completed (where X is the number of Open Sequences per Exchange supported by the Sequence Recipient as specified during Login) or the Exchange is terminated.

18.3.9 Detection of missing frames

The following methods of detecting missing frames apply to a non-streamed Sequence or multiple streamed Sequences with continuously increasing SEQ_CNT.

- a) In Class 1 a missing Data frame error is detected if a frame is received with a SEQ_CNT that is not one greater than the previously received frame, except when a SEQ_CNT wrap to zero occurs (e.g., SEQ_CNT = 3 received, then SEQ_CNT = 5 received; a missing frame error (SEQ_CNT = 4) is detected).
- b) In Class 1 a missing Data frame error due to timeout is detected by the Sequence Recipient if a partial Sequence has been received and the next expected Data frame (current SEQ_CNT+1, except when a wrap to zero occurs) is not received within an E_D_TOV timeout period.
- c) In Class 2 and 3, with out of order delivery, a potentially missing Data frame is detected if a frame is received with a SEQ_CNT that is not one greater than the previously received frame, except when a SEQ_CNT wrap to zero occurs. If the potentially missing Data frame is not received within the E_D_TOV timeout period, a missing frame error is detected.
- d) In Class 2, with in order delivery, a potentially missing Data frame is detected if a frame is received with a SEQ_CNT that is not one greater than the previously received frame, except when a SEQ_CNT wrap to zero occurs. If the potentially missing Data frame is not received within the E_D_TOV timeout period, a missing frame error is detected.

NOTE - With in order delivery, a Class 2 frame may be delivered with its SEQ_CNT that is not one greater than the previously received frame, if a Class 2 frame which was transmitted earlier has been issued F_BSY or F_RJT. This frame is potentially missing, since it may be retransmitted.

- e) In Class 3, with in order delivery, a missing Data frame is detected if a frame is received with a SEQ_CNT that is not one greater than the previously received frame, except when a SEQ_CNT wrap to zero occurs.
- f) A Sequence Recipient may also detect missing Class 2 or 3 Data frames through the use of a missing frame window. The size of the missing frame window, W, is set by the Sequence Recipient and is not specified by FC-FS. A frame is considered missing by a Sequence Recipient if its SEQ_CNT is less than the highest SEQ_CNT received for that Sequence minus W. It is suggested that W be at least equal to End-to-end Credit.

NOTE - Fabric characteristics should be taken into account when attempting to establish a missing frame window - W. Too small a value may give false errors, whereas too large a value may create out of Credit conditions.

18.3.10 Sequence errors - Class 1 and 2

Either the Sequence Initiator or the Sequence Recipient may detect errors within a Sequence.

18.3.10.1 Rules common to all Discard policies

In discard policy, the Recipient shall discard the Data Field portion of Data frames (FC-2 Header is processed) received after the point at which the error is detected and including the frame in which the error was detected. In all cases except the Stop Sequence condition, the Sequence Recipient shall discard the entire Sequence. The following rules apply.

- a) The types of Sequence errors that shall be detected by a N_Port include:
 - 1) detection of a missing frame based on SEQ_CNT,
 - 2) detection of a missing frame based on a timeout (E_D_TOV),
 - 3) detection of an error within a frame (P_RJT),
 - 4) reception of a Reject frame (F_RJT or P_RJT) or
 - 5) detection of an internal malfunction.

- b) If a Recipient receives a Data frame for a Sequence that the Recipient ULP wishes to stop receiving, the Recipient shall indicate the Stop Sequence condition to the Initiator by using the Abort Sequence Condition bits (1 0) in F_CTL. See 23.7.2.
- c) If a Recipient detects an error within a valid frame of a Sequence, it shall indicate that error to the Initiator by transmitting a P_RJT with a reason code.
- d) If a Recipient receives a Data frame for an Active Sequence which has previously had one or more Data frames rejected, the Recipient shall indicate that previous error to the Initiator on subsequent ACK frames using the Abort Sequence Condition bits (0 1) in F_CTL in the same manner as it would if a missing frame were detected.
- e) If the Recipient has transmitted an ACK with the Abort Sequence Condition bits set, or a P_RJT in response to a Data frame, it shall post that information in the Sequence Status.
- f) If an Initiator receives an ACK with the Abort Sequence Condition bits in F_CTL requesting Stop Sequence (1 0), it shall end the Sequence by transmitting the End_Sequence bit set to 1 in the next Data frame. If the last Data frame has already been transmitted, the Sequence Initiator shall not respond to the Stop Sequence request but shall notify the FC-4.
- g) If an Initiator detects a missing frame, internal error, or receives an ACK with a detected rejectable condition, it shall abort the Sequence by transmitting an Abort Sequence (ABTS) Basic Link Service command (see 15.2.1).
- h) If an Initiator receives an ACK with the Abort Sequence Condition bits (0 1) in F_CTL requesting that the Sequence be aborted, it shall abort the Sequence by transmitting an Abort Sequence (ABTS) Basic Link Service command (see 15.2.1).
- i) If an Initiator receives a Reject frame (F_RJT, or P_RJT), it shall abort the Sequence by transmitting an Abort Sequence (ABTS) Basic Link Service command (see 15.2.1) if the Sequence has not been terminated by the Sequence Recipient or Fabric using an EOFt (EOFdt) on the RJT.
- j) In Class 1, if the Sequence Initiator detects a Sequence timeout (see 23.2.4), it shall:
 - 1) abort the Sequence using ABTS if end-to-end Credit is available,
 - 2) perform the Link Reset Protocol if all Active Sequences have timed out or no end-to-end Credit is available, or
 - 3) read status from the Recipient if the current state of a Sequence is uncertain in order to determine the existence of an error condition.
- k) In Class 2, if the Sequence Initiator detects a Sequence timeout (see 23.2.4), it shall:
 - 1) transmit Link Credit Reset to the Recipient if no end-to-end Credit is available,
 - 2) abort the Sequence using ABTS when end-to-end Credit is available, or
 - 3) read status from the Recipient if the current state of a Sequence is uncertain in order to determine the existence of an error condition.

18.3.10.2 Discard multiple Sequences Error Policy

These rules apply to Discard multiple Sequences Error Policy and Discard multiple Sequences with Retransmission Error Policy.

- a) In Class 1, if a Sequence Recipient detects a missing frame error, or detects an internal malfunction for a Sequence within an Exchange which requested Discard multiple Sequences Error Policy, it shall request that the Sequence be aborted by setting the Abort Sequence Condition bits (0 1) in F_CTL on the ACK corresponding to the Data frame during which the missing frame error was detected. For errors detected other than missing frame, the Abort Sequence Condition bits (0 1) in F_CTL shall be transmitted for any subsequent ACKs transmitted. The Sequence Recipient may continue to transmit ACKs for subsequent frames of the Sequence and any subsequent streamed Sequences until the ABTS frame is received. If an ACK is transmitted for the last Data frame of a Sequence, F_CTL bits 19,

18, 17, 16, and 14 settings on the Data frame shall be ignored and those bits shall be set to zero in the ACK frame in addition to bits 5-4 (0 1). See 23.7.1.

- b) In Class 1, if a Sequence Recipient detects a missing frame error, or detects an internal malfunction for a Sequence within an Exchange which requested Discard multiple Sequences with Retransmission Error Policy, it requests that the Sequence be aborted and immediately retransmitted by setting the Abort Sequence Condition bits (1 1) in F_CTL on the ACK corresponding to the Data frame during which the missing frame error was detected.
- c) If the Sequence Recipient is unable to transmit an ACK with the same SEQ_ID as the Sequence which requires retransmission, the Sequence Recipient shall follow the rules for Discard multiple Sequences Error Policy (bits 5-4 set to 0 1).
- d) For errors detected other than missing frame, the Abort Sequence Condition bits (1 1) in F_CTL shall be transmitted for any subsequent ACKs transmitted. The Sequence Recipient may continue to transmit ACKs for subsequent frames of the Sequence and any subsequent streamed Sequences until it receives a new Sequence (SOFix) with the F_CTL Retransmission bit set = 1, or an ABTS frame is received. If an ACK is transmitted for the last Data frame of a Sequence, F_CTL bits 19, 18, 17, 16, and 14 settings on the Data frame shall be ignored and those bits shall be set to zero in the ACK frame in addition to bits 5-4 (1 1).
- e) If the Sequence Recipient is unable to support Discard multiple Sequences with Retransmission Error Policy, it shall follow the rules for Discard multiple Sequences Error Policy (bits 5-4 set to 0 1). If the Sequence Initiator is unable to determine the correct Sequence boundary to begin retransmission, it shall either transmit the ABTS frame or Read Exchange Status (RES). See 23.7.1.
- f) In Class 2, if a Sequence Recipient detects a missing frame error, transmits a P_RJT, or detects an internal malfunction for a Sequence within an Exchange which requested Discard multiple Sequences Error Policy, it shall request that the Sequence be aborted by setting the Abort Sequence Condition bits (0 1) in F_CTL on the ACK corresponding to the Data frame during which the missing frame error was detected. For detected errors other than missing frame, the Abort Sequence Condition bits (0 1) in F_CTL shall be transmitted for any subsequent ACKs transmitted. The Sequence Recipient may continue to transmit ACKs for subsequent frames of the Sequence and any subsequent streamed Sequences until the ABTS frame is received. Any ACKs transmitted for frames in this Sequence or any subsequent Sequences shall continue to set the Abort Sequence Condition bits to (0 1) (see 23.7.1). The last ACK for the Sequence in error shall be transmitted as described under normal Sequence completion.
- g) In Class 1, if a Sequence Initiator receives an ACK with the Abort Sequence Condition bits (1 1) in F_CTL requesting that the Sequence be retransmitted, it shall begin retransmission of the first non-deliverable Sequence by starting a new Sequence and setting the F_CTL Retransmission bit (F_CTL bit 9) set to 1 until it has received at least one ACK which indicates that the retransmitted Sequence has been successfully initiated by the Recipient. If the Sequence Initiator is unable to determine the correct Sequence boundary to begin retransmission, it shall either transmit the ABTS frame to abort the Sequence or Read Exchange Status using the RES Extended Link Services request. See 23.7.1.2.

18.3.10.3 Discard a single Sequence Error Policy

In Class 1 and 2, if a Sequence Recipient detects a missing frame error, or detects an internal malfunction for a Sequence within an Exchange which requested Discard a single Sequence Error Policy, it shall request that the Sequence be aborted by setting the Abort Sequence Condition bits (0 1) in F_CTL on the ACK corresponding to the Data frame during which the missing frame error was detected. For errors detected other than missing frame, the Abort Sequence Condition bits (0 1) in F_CTL shall be transmitted for any subsequent ACKs transmitted for this Sequence.

The Sequence Recipient may continue to transmit ACKs for subsequent frames of the Sequence until the ABTS frame is received. However, it shall not continue to set the Abort Sequence Condition bits in any subsequent streamed Sequences. If the final ACK (EOF_t) to the Sequence is transmitted, F_CTL bits 19, 18, 17, 16, and 14 settings on the Data frame shall be ignored and those bits shall be set to zero in the ACK frame in addition to bits 5-4 (0 1) (see 23.7.1).

18.3.10.4 Process with infinite buffers Error Policy

In process policy, the Recipient shall ignore errors detected on intermediate frames, or timeout errors such that ABTS is not requested. However, such errors shall be reported to an upper level.

If a Recipient detects an internal error related to a Sequence, or it detects that the first or last frame of a Sequence is missing, it shall request that the Sequence be aborted by setting the Abort Sequence Condition bits (0 1) in F_CTL on subsequent ACK frames. The Recipient shall continue to respond in the same manner as defined under Discard a single Sequence Error Policy.

NOTE - Missing last Data frame is detected by the Sequence timeout.

If a Sequence Recipient detects an error within a valid frame of a Sequence, it shall indicate that error to the Initiator by transmitting a P_RJT with a reason code.

18.3.11 Sequence errors - Class 3

Errors within a Sequence may only be detected by the Sequence Recipient.

18.3.11.1 Rules common to all discard policies

In both Discard policies, the Sequence Recipient shall discard Sequences in the same manner as in Class 1 and 2 with the exception that an ACK indication of Abort Sequence shall not be transmitted. In discard policy, the Recipient shall discard frames received after the point at which the error is detected. Individual FC-4s or upper levels may recover the entire Sequence or only that portion after which the error is detected.

- a) The types of errors that shall be detected by a N_Port include:
 - 1) detection of a missing frame based on timeout, or
 - 2) detection of an internal malfunction.
- b) If a Recipient detects an internal error, it shall abnormally terminate the Sequence, post the appropriate status, and notify the FC-4 or upper level. One or more Sequences shall not be delivered based on single or multiple Sequence discard Error Policy.
- c) If a Recipient detects a missing frame, it shall abnormally terminate the Sequence, post the appropriate status, and notify the FC-4 or upper level. One or more Sequences shall not be delivered based on single or multiple Sequence discard Error Policy.
- d) In the Discard multiple Sequences Error Policy in Class 3, the Sequence Recipient shall not be required to utilize a timeout value of R_A_TOV following detection of a missing frame. Therefore, frames may be discarded for an Exchange forever if the Sequence Initiator does not utilize other detection mechanisms.
- e) Notification of the Sequence error condition to the Initiator is the responsibility of the FC-4 or upper level.

18.3.11.2 Process with infinite buffers Error Policy

In process Policy, the Recipient shall ignore errors detected on all frames, or timeout errors. However, such errors shall be reported to an upper level.

NOTE - Ignoring an error on the first frame of a Sequence or an Exchange may cause the frame to be delivered to the wrong Recipient.

18.3.12 Sequence Status Rules

The following rules summarize handling of Sequence Status Block processing:

- a) The Sequence Initiator shall consider a Sequence Open and Active after transmission of the first frame of the Sequence. The Sequence shall remain Active until the Sequence Initiator has transmitted the last frame of the Sequence. The Sequence Initiator shall consider the Sequence Open until
 - 1) it receives the ACK (EOF_t),
 - 2) Basic Accept is received to an ABTS frame,
 - 3) A Read Sequence Status or Read Exchange Status indicates normal or abnormal completion,
 - 4) the Link Reset Primitive Sequence is received or transmitted during a Dedicated Connection,
 - 5) the Exchange is aborted using the ABTX Link Service request, or
 - 6) a Logout Link Service request is completed.
- b) A Sequence shall be considered Open and Active, and an SSB Opened, by the Sequence Recipient when any frame in a Sequence is received for the first Sequence of a new Exchange as indicated in F_CTL bit 21. An Exchange Status Block is Opened at the same time and the Exchange becomes Active.
- c) A Sequence shall be considered Open and Active, and an SSB Opened, by the Sequence Recipient when any frame in a Sequence is received for an Open Exchange.
- d) If the Sequence Recipient transmits an ACK frame with the Abort Sequence Condition bits other than 0 0, it shall post that status in the Sequence Status Block status.
- e) If a Sequence completes normally and is deliverable, its status shall be posted in the Sequence Status Block.
- f) If a Sequence completes abnormally by the Abort Sequence Protocol, its status shall be posted in the Sequence Status Block.
- g) The Exchange Status Block shall be updated with Sequence Status information when the Sequence becomes abnormally complete, or normally complete.

18.3.13 Exchange termination

- a) The last Sequence of the Exchange shall be indicated by setting the F_CTL Last_Sequence bit = 1 in the last Data frame of a Sequence. The Last_Sequence bit may be set to 1 prior to the last Data frame; once it has been set = 1, it shall remain set for the remainder of the Sequence.
- b) The Exchange shall be terminated when the last Sequence is completed by normal Sequence completion rules.
- c) An Exchange may be abnormally terminated using the ABTX Extended Link Services request. A Recovery_Qualifier range timeout may be required in Class 2 and 3.
- d) An Exchange shall be abnormally terminated following Logout with the other N_Port involved in the Exchange (either Originator or Responder). A Recovery_Qualifier range timeout may be required in Class 2 and 3.

18.3.14 Exchange Status Rules

The following rules summarize handling of Exchange Status Block processing:

- a) An Exchange shall be considered Active, and an ESB Opened, by the Originator after transmission of the first frame of the first Sequence.
- b) An Exchange shall be considered Active, and an ESB Opened, by the Sequence Recipient when any frame in the first Sequence is received.
- c) An Exchange shall remain Open until
 - 1) the last Sequence of the Exchange completes normally,
 - 2) a timeout period of E_D_TOV has elapsed since the last Sequence of the Exchange completed abnormally, or
 - 3) the Exchange is successfully aborted with ABTX (which includes a Recovery_Qualifier timeout, if necessary).
- d) When an Exchange is no longer Open, it shall be complete and the Exchange resources associated with the Exchange, including the Exchange Status Block, are available for re-use. An upper level may choose to complete an Exchange with an interlocked protocol in order to ensure that both the Originator and Responder agree that the Exchange is complete. Such a protocol is outside the scope of FC-FS.
- e) The contents of an Exchange Status Block may be made available (RES request) until the resource is allocated to another Exchange.

18.4 Exchange management

An Exchange is managed as a series of unidirectional Data frame Sequences. The Originator of the Exchange shall transmit the initial Sequence. F_CTL bits within the Frame Header identify control and intermixing of Sequences within an Exchange.

Following the initial Sequence, subsequent Sequences may be transmitted by either the Originator or the Responder facilities based on which facility holds the Sequence Initiative.

18.5 Exchange origination

The key facilities, functions, and events involved in the origination of an Exchange by both the Originator and Responder are diagrammed in Figure 51. An Exchange for Data transfer may be originated with a destination N_Port following destination N_Port_Login. Login provides information necessary for managing an Exchange and Sequences such as: Class, the number of Concurrent Sequences, Credit, and Receive Data Field Size. Login and Service Parameters are discussed in 23.6. An Exchange is originated through the initiation of a Sequence. The rules in 18.3.2 specify the requirements for originating an Exchange.

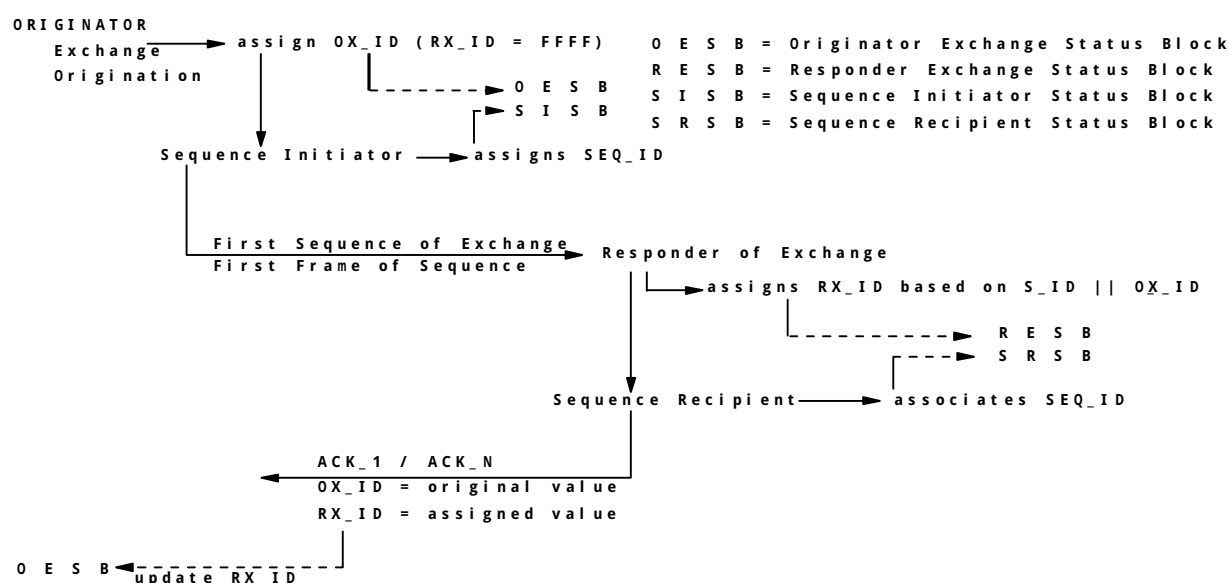


Figure 51 - Exchange origination

18.5.1 Exchange Originator

When an Exchange is originated by a N_Port, that N_Port shall assign an Originator Exchange_ID (OX_ID) unique to that N_Port Identifier. An Originator Exchange Status Block associated with the OX_ID is allocated and bound to the Exchange and other link facilities in that N_Port for the duration of the Exchange. All frames associated with that Exchange contain the assigned OX_ID. The Originator in the Originator Exchange Status Block shall track the status of the Exchange. See 18.6.2 for more information on unique Sequence_Qualifiers.

Each frame within the Exchange transmitted by the Originator shall be identified with an F_CTL bit designating the frame as Originator generated (Exchange Context). The Originator Exchange_ID provides the mechanism for tracking Sequences for multiple concurrent Exchanges which may be Active at the same time.

NOTE - Since the Originator assigns the OX_ID, assignment may be organized to provide efficient processing within the N_Port.

18.5.2 Exchange Responder

The destination N_Port shall be designated as the Responder for the duration of the Exchange. When the destination N_Port receives the first Sequence of the Exchange, that N_Port shall assign a Responder Exchange_ID (RX_ID) to the newly established Exchange. This RX_ID is associated with the OX_ID from a given S_ID to a Responder Exchange Status Block (S_ID||OX_ID). See 18.6.2 for more information on unique Sequence_Qualifiers.

In Class 1 and 2, the assigned RX_ID shall be transmitted to the Originator on the ACK frame responding to the last Data frame of the Sequence or earlier, if possible. In a Class 3 bi-directional Exchange, the assigned RX_ID shall be transmitted to the Originator in the first Data frame transmitted by the Responder. If the Sequence Recipient has specified X_ID interlock during Login, the RX_ID shall be assigned in the ACK to the first Data frame of the Sequence. The Originator shall withhold additional frame transmission for the Exchange until the ACK is received. The Responder Exchange_ID provides the mechanism for tracking Sequences for multiple concurrent Exchanges from multiple S_IDs or the same S_ID.

NOTE - Since the Responder assigns the RX_ID, assignment may be organized to provide efficient processing within the N_Port.

Each frame within the Exchange transmitted by the Responder is identified with an F_CTL bit designating the frame as Responder generated (Exchange Context). Each frame within the Exchange transmitted by the Responder is identified with the assigned RX_ID. The status of the Exchange shall be tracked by the Responder in the Responder Exchange Status Block.

18.5.3 X_ID assignment

In the first frame of an Exchange, the Originator shall set the OX_ID to an assigned value and the RX_ID value to hex 'FF FF' (unassigned). When the Responder receives the first Sequence of an Exchange, it shall assign an RX_ID and shall return the RX_ID in the ACK frame sent in response to the last Data frame in the Sequence, or in an earlier ACK (Class 1 and 2). In a Class 3 bi-directional Exchange, the Responder shall assign an RX_ID in the first Data frame transmitted.

For all remaining frames within the Exchange, the OX_ID and RX_ID fields retain these assigned values.

18.5.4 X_ID interlock

X_ID interlock is only applicable to Classes 1 and 2. When a N_Port initiates a Sequence with a N_Port which has specified during Login that X_ID interlock is required and the Recipient's X_ID is invalid or unassigned, the initiating N_Port shall transmit the first frame of the Sequence with the Recipient's X_ID set to hex 'FF FF' and shall withhold transmission of additional frames until the corresponding ACK with an assigned X_ID has been received from the Recipient. The assigned X_ID is then used in all subsequent frames in the Sequence.

18.6 Sequence management

18.6.1 Active and Open Sequence

From the standpoint of the Sequence Initiator, a Sequence is Active for the period of time from the transmission of the Data frame initiating the Sequence until the end of the last Data frame of the Sequence is transmitted. In Class 1, the Sequence Initiator considers the Sequence Open until the ACK with EOF_t (EOF_{dt}) is received, the Sequence is aborted by performing the ABTS Protocol, or the Sequence is abnormally terminated. In Class 2, the Sequence Initiator considers the Sequence Open until the ACK with EOF_t is received, the Sequence is aborted by performing the ABTS Protocol, or the Sequence is abnormally terminated. In Class 3, the Sequence Initiator considers the Sequence Active until the EOF_t is transmitted. In Class 3, the Sequence Initiator considers the Sequence Open until the deliverability is confirmed or an R_A_TOV timeout period has expired. The determination of deliverability of Class 3 Sequences is beyond the scope of FC-FS, which provides no deliverability guarantees for Class 3 Sequences.

In Class 1 and 2, from the standpoint of the Sequence Recipient, a Sequence is Active and Open from the time any Data frame is received until the EOF_t (EOF_{dt}) is transmitted in the ACK to the last Data frame, or abnormal termination of the Sequence. In Class 3, from the standpoint of the Sequence Recipient, a Sequence is Active and Open from the time the initiating Data frame is received until all Data frames up to the frame containing EOF_t have been received.

See 18.6.2 for more information regarding reuse of SEQ_ID and Sequence_Qualifiers.

18.6.2 Sequence_Qualifier management

The Sequence Initiator assigns a SEQ_ID that is unique between the N_Port pair of the Initiator and the Recipient at the time it is assigned and until the Sequence is complete. When the Se-

quence completes normally or abnormally, the SEQ_ID is reusable by the Sequence Initiator for any Sequence_Qualifier, including the same Recipient and Exchange providing that Sequence rules are followed (see 18.3.4). If a Sequence is aborted using the Abort Sequence Protocol or the ABTX Link Service request, a Recovery_Qualifier range may be specified by the Sequence Recipient (see 23.7.1.1), however, SEQ_ID shall not be included in the Recovery_Qualifier.

18.6.3 Sequence initiative and termination

When a Sequence is terminated in a normal manner, the last Data frame transmitted by the Sequence Initiator is used to identify two conditions:

- Sequence initiative, and
- Sequence termination.

18.6.4 Transfer of Sequence Initiative

The Sequence Initiator controls which N_Port shall be allowed to initiate the next Sequence for the Exchange. The Sequence Initiator may hold the initiative to transmit the next Sequence of the Exchange or the Sequence Initiator may transfer the initiative to transmit the next Sequence of the Exchange. The decision to hold or transfer initiative shall be indicated by Sequence_Initiative bit in F_CTL.

In Class 1 and 2, the Sequence Recipient shall not consider Sequence Initiative to have been passed until the Sequence which passes the Sequence Initiative is completed successfully and the ACK (EOF_t) has been transmitted with the Sequence Initiative bit (F_CTL bit 16) = 1.

In Class 1 and 2, when a Sequence Initiator detects a Data frame from the Recipient for an Exchange in which it holds the Sequence Initiative, it shall transmit a P_RJT with a reason code of Exchange error (excluding the ABTS frame). In Class 3, when a Sequence Initiator detects a Data frame (excluding the ABTS frame) from the Recipient for an Exchange in which it holds the Sequence Initiative, it shall abnormally terminate the Exchange and discard all frames for the Exchange.

End_Sequence

When the Sequence Initiator is ending the current Sequence, it shall set the End_Sequence bit in F_CTL to one on the last Data frame of the Sequence.

18.6.5 Sequence termination

Setting the End_Sequence bit in F_CTL to one indicates the last Data frame transmitted by the Sequence Initiator. The Sequence is terminated by either the Initiator or the Recipient transmitting a frame terminated by EOF_t. The Sequence Initiator is in control of terminating the Sequence. Transmission of the EOF_t may occur in two ways:

- In Class 1 and 2, the Sequence Recipient transmits an ACK frame of ACK_1 or ACK_0 with EOF_t in response to the last Data frame received for the Sequence.
- In Class 3, the Sequence Initiator transmits the last Data frame of the Sequence with EOF_t.

Class 1

When EOF_t has been transmitted or received by each N_Port, the appropriate Exchange Status Block associated with the Sequence shall be updated in each N_Port to indicate that the Sequence was completed and whether the Originator or Responder facility holds the Sequence Initiative. Link facilities associated with the Sequence (including Sequence Status Block) are released and available for other use.

Class 2

Since Class 2 frames may be delivered out of order, Sequence processing is only completed after all frames (both Data and ACK) have been received, accounted for, and processed by the respective N_Ports.

When the Sequence is completed by each N_Port, the appropriate Exchange Status Block associated with the Sequence shall be updated in each N_Port to indicate that the Sequence was completed and whether the Originator or Responder facility holds the Sequence Initiative. Link facilities associated with the Sequence (including the Sequence Status Block) are released and available for other use.

NOTE - Since ACKs may arrive out of order, the Sequence Initiator may receive the ACK which contains EOFt before ACKs for the same Sequence. The Sequence Initiator shall not consider the Sequence normally terminated until it has received the final ACK (see 23.7.3).

Class 3

The Sequence Initiator shall terminate the last Data frame of the Sequence with EOFt. Acknowledgment of Sequence completion is the responsibility of the Upper Level Protocol.

When the Sequence is completed by each N_Port, the appropriate Exchange Status Block associated with the Sequence shall be updated in each N_Port to indicate that the Sequence was completed and whether the Originator or Responder facility holds the Sequence Initiative. Link facilities associated with the Sequence (including Sequence Status Block) are released and available for other use.

Continue Sequence Condition

When a N_Port terminates a Sequence yet holds the Sequence Initiative, the N_Port may include additional information regarding the timing of the next Sequence. The N_Port uses the Continue Sequence Condition bits in F_CTL to indicate that the next Sequence is being transmitted immediately (0 1), soon (1 0), or delayed (1 1). The estimate of time is based on the time to remove and reestablish a Class 1 Connection, regardless of which Class of Service is being used.

Chained_Sequence (C_S)

Certain existing system architectures require immediate feedback during specific phases of an I/O operation. The Chained_Sequence (C_S) bit in F_CTL may be set to one on the last Data frame of a Sequence when the Sequence Initiative is being transferred to indicate that a reply Sequence is required from the Sequence Recipient before a Dedicated Connection is removed. The presence of the C_S bit, End_Sequence bit, and transfer of Sequence Initiative overrides the E_C bit if previously set by the Sequence Recipient (see 22.7.1 and 22.7.2).

End_Sequence

When the Sequence Initiator is ending the current Sequence, it shall set the End_Sequence bit in F_CTL to one on the last Data frame of the Sequence.

18.7 Exchange termination

18.7.1 Normal termination

Either the Originator or the Responder may terminate an Exchange. The facility terminating the Exchange shall indicate Exchange termination on the last Sequence of the Exchange by setting the Last_Sequence bit in F_CTL on the last frame, or earlier, if possible, of the last Sequence of the Exchange.

The Sequence shall be terminated according to normal Sequence termination rules. When the last Sequence of the Exchange is terminated normally, the Exchange shall also be terminated. The OX_ID and RX_ID and associated Exchange Status Blocks are released and available for reuse.

18.7.2 Abnormal termination

Either the Originator or the Responder may abnormally terminate an Exchange by using the Abort Exchange Protocol or Sequence timeout of the last Sequence of the Exchange. In general, reception of a reject frame with an action code of 2 is not recoverable at the Sequence level and aborting of the Exchange is probable. Other reasons to abort an Exchange are FC-4 protocol dependent and not defined within FC-FS.

18.8 Status blocks

18.8.1 Exchange Status Block

An Exchange Status Block (see Table 209) is a logical construct used to associate the OX_ID, RX_ID, D_ID and S_ID of an Exchange. The Status Block is used throughout the Exchange to track the progress of the Exchange and identify which N_Port holds the initiative to transmit Sequences.

The Exchange Status Block shall record Exchange status information and Sequence Status for a number of Sequences received as a Sequence Recipient. The Exchange Status Block is supplied in the RES Link Services request. Equivalent information to track transmitted Sequences is required by the Sequence Initiator for internal tracking of Exchange progress but is not required to be supplied to any other N_Port. The Sequence Status is stored in the Exchange Status Block in the oldest to newest order. The oldest Sequence is dropped out of the ESB when new Sequence status is added.

Table 209 - Exchange Status Block	
Item	Size - Bytes
Priority (if enabled) (MSb is preemption flag)	1
OX_ID	2
RX_ID	2
Originator Address Identifier (High order byte - reserved)	4
Responder Address Identifier (High order byte - reserved)	4
E_STAT	4
reserved	4
Service Parameters	112
Oldest Sequence Status (SSB format-first 8 bytes)	8
Intermediate Sequence Status (SSB format-first 8 bytes)	X*8
Newest Sequence Status (SSB format-first 8 bytes)	8
Note - LSB is the Preemption Flag	

E_STAT

- **Bit 31 - ESB owner**

0 = Originator

1 = Responder

- **Bit 30 - Sequence Initiative**

0 = Other Port holds initiative

1 = This Port holds initiative

- **Bit 29 - Completion**

0 = Open

1 = complete

- **Bit 28 - Ending Condition**

0 = normal

1 = abnormal

- **Bit 27 - Error type**

0 = Exchange aborted with ABTX

1 = Exchange abnormally terminated

Error type is only valid when bit 28 is set to one.

- **Bit 26 - Recovery_Qualifier**

0 = none

1 = Active

- **Bits 25-24 - Exchange Error Policy**

0 0 = Abort, Discard multiple Sequences

0 1 = Abort, Discard a single Sequence

1 0 = Process with infinite buffers

1 1 = Discard multiple Sequences with immediate retransmission

- **Bit 23 - Originator X_ID invalid**

0 = Originator X_ID valid

1 = Originator X_ID invalid

X_ID validity status reflects the completion condition of the newest Sequence Status Block contained in the ESB.

- **Bit 22 - Responder X_ID invalid**

0 = Responder X_ID valid

1 = Responder X_ID invalid

X_ID validity status reflects the completion condition of the newest Sequence Status Block contained in the ESB.

- **Bit 21 - Priority in Use**

0 = Priority/Preemption not used for this exchange

1 = Priority/Preemption in use for this exchange

Priority/Preemption not enabled reflects the condition set F_CTL for SOFc1 and SOFix frames.

- **Bits 20-0 - Reserved**

18.8.2 Sequence Status Block

A Sequence Status Block (see Table 210) is a logical construct used to track the progress of a single Sequence by a N_Port on a frame by frame basis. A Sequence Status Block shall be Opened and maintained by the Sequence Recipient for each Sequence received in order to support the RSS Link Service request. Information equivalent to an SSB is required for the Sequence Initiator to track Sequence progress internally, but is not required to be supplied to any other N_Port.

Table 210 - Sequence Status Block	
Item	Size - Bytes
SEQ_ID	1
reserved	1
Lowest SEQ_CNT	2
Highest SEQ_CNT	2
S_STAT	2
Error SEQ_CNT	2
Frame header Word 1, bits 31-24	1
Frame header Word 4, bits 31-16	2
RX_ID	2
reserved	2

- **Frame header Word 1, Bits 31-24**

When priority is in use, this field shall contain the current priority. When priority is not in use, this field shall contain the CS_CTL field.

- **Frame header Word 4, Bits 31-16**

This field shall contain the OX_ID.

S_STAT

- **Bit 15 - Sequence context**

0 = Initiator
1 = Recipient

- **Bit 14 - Open**

0 = not Open
1 = Open

- **Bit 13 - Active**

0 = not Active
1 = Active

- **Bit 12 - Ending Condition**

0 = normal
1 = abnormal

Bits 11 - 7 clarify the reason for an abnormal ending condition.

- **Bits 11 - 10 ACK, Abort Sequence condition**
 - 0 0 = continue
 - 0 1 = Abort Sequence requested
 - 1 0 = Stop Sequence requested
 - 1 1 = Abort with Retransmission requested
- **Bit 9 - ABTS protocol performed**
 - 0 = ABTS not completed
 - 1 = ABTS completed by Recipient
- **Bit 8 - Retransmission performed**
 - 0 = Retransmission not completed
 - 1 = Retransmission completed by Recipient
- **Bit 7 - Sequence time-out**
 - 0 = Sequence not timed-out
 - 1 = Sequence timed-out by Recipient (E_D_TOV)
- **Bit 6 - P_RJT transmitted**
 - 0 = P_RJT not transmitted
 - 1 = P_RJT transmitted
- **Bits 5-4 Class**
 - 0 0 = reserved
 - 0 1 = Class 1
 - 1 0 = Class 2
 - 1 1 = Class 3
- **Bit 3 - ACK (EOF_t) transmitted**
 - 0 = ACK (EOF_t) not transmitted
 - 1 = ACK (EOF_t or EOF_{dt}) transmitted
- **Bits 2-0 - Reserved**

19 Association_Header management and usage

19.1 Introduction

FC_FS provides the Exchange facility as the fundamental mechanism for coordinating the interchange of information and data between two N_Ports or nodes. All Data transmission shall be part of an Exchange. Certain system architectures may use an Association_Header to provide two additional, but separate, functions.

A Process_Associator may be used to provide one or more of the following functions:

- provide a method to direct an Exchange to a single system image where multiple system images share a N_Port facility,
- provide a method to direct an Exchange to a single process where multiple processes share a N_Port facility,
- identify a source or destination process, or a source or destination system image that shall be retained beyond the life of an Exchange for certain FC-4s. In this instance, the Process_Associator in combination with the N_Port Identifier is used to identify a process or image related to a previous Exchange.

The N_Port_Login option that may be required, supported, or not supported by a N_Port in Class 1 and 2 is Initial Process_Associator.

Login only completes successfully (Accept) if each N_Port's requirements are satisfied. If a N_Port either supports or requires an Initial Process_Associator, then it shall also require X_ID interlock (Class 1 and 2), as specified in Login. The following list specifies the need for an Association_Header for a specific Exchange:

If neither N_Port requires an Initial Process_Associator, then no Association_Header is required.

If both N_Ports require an Initial Process_Associator, then an Association_Header is required.

If only one N_Port requires an Initial Process_Associator and the other N_Port supports it, then an Association_Header may be required.

19.2 Establishing the final Association_Header

The rules which specify the procedures for Association_Header management are in addition to all the rules and behavior described in other clauses of FC_FS. The Association_Header is composed of information from the Originator and from the Responder. The method by which this composite information is obtained is by transmission of a first Association_Header by the Originator that is followed by transmission of the final Association_Header by the Responder in a later Sequence to reflect one or both of the Responder Associators. The procedure is simplified by following the first and final Association_Header rules. The procedure begins with the origination of the Exchange. The content of the Association_Header is specified in 13.4.

19.2.1 Exchange Origination

If the Responder requires an Initial Process_Associator, the Originator of an Exchange shall transmit an Association_Header in the first Data frame of the Exchange.

If the Responder does not require an Initial Process_Associator, then an Association_Header is not required for the Exchange.

Meaningful values (validity byte in Word 0 of Association_Header) are implementation dependent and have no constraints placed on their values other than those defined in 13.4.

The Association_Header to be transmitted by the Originator shall be composed as follows:

- the Originator Process_Associator shall be set to
 - a meaningful value if the Originator requires an Initial Process_Associator, or
 - validity indicated as not meaningful.
- the Responder Process_Associator shall be set to
 - a meaningful value if the Responder requires an Initial Process_Associator (obtained from a Name_Server or by other means outside the scope of FC_FS), or
 - validity indicated as not meaningful.

19.2.2 First transfer rules

The following rules apply to the transmission of the first Association_Header transmitted by the Originator:

- a) the Originator shall assign a value to OX_ID and set RX_ID to hex 'FF FF' and include the Association_Header on the first Data frame of the Exchange.
- b) X_ID interlock is required on the first Data frame of the Exchange.
- c) the RX_ID shall be specified in the ACK to the first Data frame of the Exchange.
- d) the first Association_Header transfer is confirmed by receiving the ACK to the first Data frame of the Exchange.
- e) if the first Association_Header transfer is not confirmed, the Originator shall transmit an ABTS frame to either Abort the Sequence or confirm transfer.
- f) The first Association_Header transmitted by the Originator shall be the final Association_Header.

19.2.3 Final transfer rules

The Association_Header transmitted by the Responder becomes the final Association_Header for the Exchange. That is, whenever an Association_Header is required to be transferred, the Association_Header designated as final shall be transferred. The Association_Header shall not change its content throughout the remainder of the Exchange.

The following rules apply to the Association_Header transmitted by the Responder.

- a) the Responder shall use the RX_ID value assigned when the Exchange was originated.
- b) if the OX_ID is valid, the Responder shall use that value.
- c) the final Association_Header transfer is confirmed by receiving an ACK to the first Data frame of the first Sequence Initiative transmitted by the Responder, or by confirming that the first Data frame was successfully received.
- d) if the final Association_Header transfer is not confirmed, the Responder shall transmit an ABTS frame to either Abort the Sequence or confirm transfer.
- e) both the Originator and Responder shall use the final Association_Header whenever an Association_Header is required to be transmitted for the duration of the Exchange.

19.3 Extended Link Services

If a N_Port performs any of the following Extended Link Service requests relating to an Exchange, which has an Association_Header, the Payload shall contain the current (first or final) Association_Header:

- Abort Exchange (ABTX)

- Read Exchange Status (RES)
- Reinstall Recovery Qualifier (RRQ)
- Request Sequence Initiative (RSI)

The Association_Header may be required to locate the Exchange Status Block since the state of X_ID validity may be unknown by the N_Port transmitting the request.

19.4 Error Recovery and other effects

When a N_Port uses a Process_Associator in order to better manage Exchange facilities, there are instances that restrict that management. Additional factors in error recovery are introduced as a result of use of the Association_Header. The additional factor that arises is confirmation of Association_Header transfer

Both the confirmation of Association_Header transfer may be handled by performing the Abort Sequence Protocol. If the ABTS is rejected (BA_RJT) for an Invalid RX_ID-OX_ID combination, the N_Port transmitting the ABTS frame shall perform a Read Exchange Status request in order to determine the status of the X_IDs for the Exchange identified by the Association_Header. If the ABTS is accepted, recovery proceeds normally.

20 Flow control management

20.1 Introduction

Flow control is the FC-2 control process to pace the flow of frames between N_Ports and between a N_Port and the Fabric to prevent overrun at the receiver. Flow control is managed using end-to-end Credit, end-to-end Credit_CNT, ACK (ACK_1), buffer-to-buffer Credit, buffer-to-buffer Credit_CNT, and R_RDY along with other frames.

Flow control is managed between N_Ports (end-to-end) and between N_Port and F_Port (buffer-to-buffer). Flow control management has variations dependent upon the Class of Service.

Applicability

Class 1 frames use end-to-end flow control. Class 1/SOF_{c1} frame and Class 2 frames use both end-to-end and buffer-to-buffer flow controls. Class 3 uses only buffer-to-buffer flow control. Table 211 shows the applicability of the flow control mechanisms to each Class.

20.2 Physical flow control model

The physical flow control model is illustrated in Figure 52. The model consists of following physical components:

- Each N_Port with Class 1 and Connectionless receive buffers.
- Each F_Port to which a N_Port is attached, with its receive buffers for Connectionless service. (Class 1 buffers internal to Fabric used for Class 1 service and Intermix are transparent to FC-2 flow control.)

Buffer participation

Buffering and transmission of Class 1 frames through the Fabric is transparent to FC-2. Class 1 buffering requirements during Intermix are specified in 16.4. The use of Class 1 buffers by the Fabric during Intermix is transparent to flow control. Class 2, Class 3 and Class 1/SOF_{c1} frames shall share connectionless buffers. Table 211 summarizes the use of buffers for end-to-end and buffer-to-buffer flow controls.

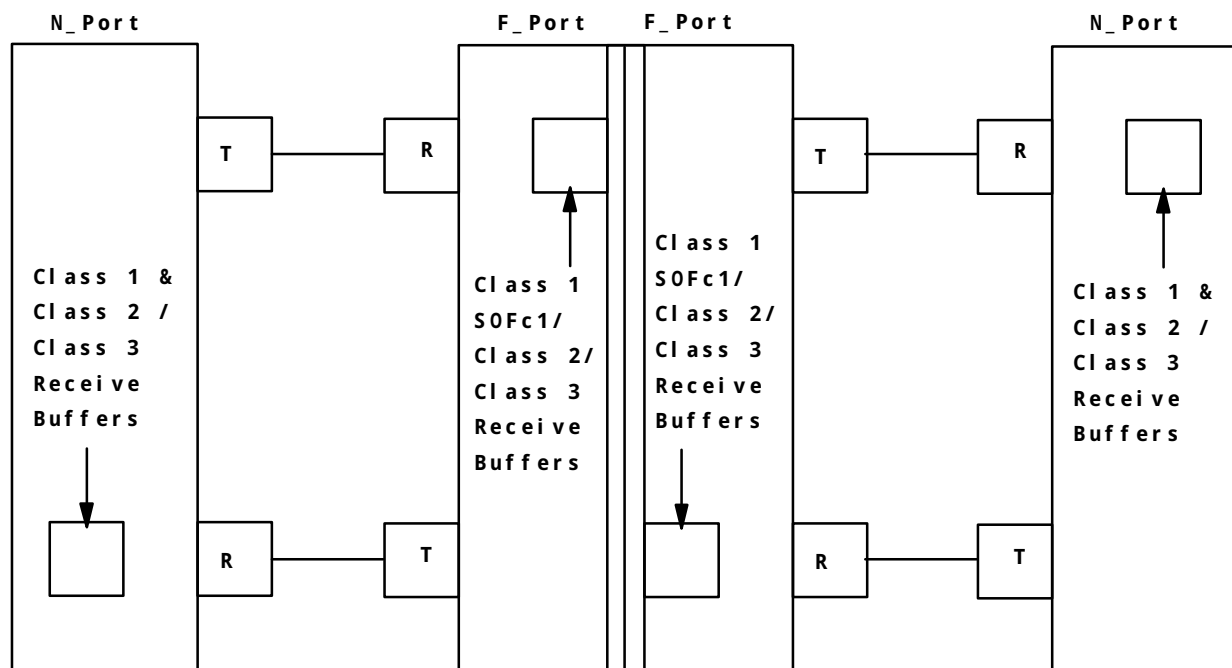


Figure 52 - Physical flow control model

Table 211 - Flow control applicability				
Flow Control methodology and mechanism	Class 1 without SOFc ₁	Connect-request frame with SOFc ₁	Class 2	Class 3
End-to-end	Yes	Yes	Yes	No
Buffer-to-buffer	No	Yes	Yes	Yes
ACK_1	Yes	Yes	Yes	No
ACK_0	One per Sequence	Yes	One per Sequence	No
R_RDY	No	Yes	Yes	Yes
F_BSY (DF)	No	Yes	Yes	No
F_BSY (LC)	No	No	Yes	No
F_RJT	No	Yes	Yes	No
P_BSY	No	Yes	Yes	No
P_RJT	Yes	Yes	Yes	No
Note - F_BSY (DF) indicates F_BSY response to a Data frame F_BSY (LC) indicates F_BSY response to a Link_Control frame except P_BSY				

Table 212 - Buffer participation		
Participating buffers	End to end flow control	Buffer to buffer flow control
Class 1 buffers	Class 1 frames	-
Connectionless buffers	Class 2 frames	Class 2, Class 3 and Class 1/SOFC1 frames
Note - Participation of Class 1 buffers in the Fabric during Intermix is transparent to flow control		

20.3 Credit and Credit_Count

Credit is the number of receive buffers allocated to a transmitting Port (a N_Port or a F_Port). Two types of Credits used in flow control are:

- End-to-end Credit (EE_Credit)
- Buffer-to-buffer Credit (BB_Credit)

The Sequence Initiator manages the Credit_Count.

Credit_Count management

The Credit_Count management is internal to a Port and is transparent to the other Port. The Credit_Count may therefore be managed by the Port either by increasing or decreasing starting with an appropriate value.

Management by increasing the Credit_Count

This is the method specified in FC_FS. In this method, the Credit_Count is initialized to a 0 (zero) value and increased by a specified value (see Table 213, Table 214, and Table 215) to a maximum value given by the Credit. The Credit_Count in this case represents the number of outstanding Data frames that have not been acknowledged by the Sequence Recipient. The Credit_Count is a positive integer and shall not exceed the value of Credit to avoid the possibility of overflow at the receiver. The Credit_Count shall not be decreased below 0 (zero).

Management by decreasing the Credit_Count

This method is not specified in this document but may be used in a given implementation. In this method, the Credit_Count is initialized to the value of the Credit and decreased to the minimum value of 0 (zero). The Credit_Count in this case represents the number of Data frames that may be sent without the possibility of causing overflow at the receiver. The Credit_Count is a positive integer and shall not be decreased below 0 (zero).

Credit_Count types

Corresponding to two types of Credits listed above, two types of Credit_Counts used are:

- End-to-end Credit_Count (EE_Credit_CNT)
- Buffer-to-buffer Credit_Count (BB_Credit_CNT)

Usage

The N_Port transmitting Class 1 or Class 2 Data frames shall use the end-to-end Credit allocated by the receiving N_Port for end-to-end flow control and manage the corresponding end-to-end Credit_Count. Class 3 Data frames do not participate in end-to-end flow control. When a Port (a N_Port or a F_Port) is transmitting Data frames or Link_Control frames to the attached Port, the transmitting Port shall use BB_Credit allocated by the receiving Port for buffer-to-buffer flow control and manage the corresponding BB_Credit_Count.

20.4 End-to-end flow control

End-to-end flow control is a FC-2 control process to pace the flow of frames between N_Ports. A N_Port pair in Class 1 or Class 2 uses end-to-end flow control.

End-to-end flow control is performed with EE_Credit_CNT with EE_Credit as the controlling parameter.

20.4.1 End-to-end management rules summary

End-to-end management rules are summarized for those cases where no error occurs. Management of EE_Credit_CNT is summarized in Table 213. The EE_Credit recovery is specified in 20.4.9.

20.4.2 Sequence Initiator

- a) The Sequence Initiator is responsible for managing EE_Credit_CNT across all active Sequences.
- b) The Sequence Initiator shall not transmit a Data frame unless the allocated EE_Credit is greater than zero and the EE_Credit_CNT is less than this EE_Credit.
- c) In Class 1 or Class 2, the value of the EE_Credit_CNT is set to zero (0) at the end of N_Port_Login, N_Port re-Login, or Link Credit Reset (see 14.3.4).
- d) In Class 1, the EE_Credit_CNT is set to one (1), on transmitting Class 1/SOFc1 frame. It is incremented by one (1) for each subsequent Class 1 Data frame transmitted. In the case of ACK_0 usage, EE_Credit_CNT management is not applicable.
- e) The EE_Credit_CNT is incremented by one (1) for each Class 2 Data frame transmitted. In the case of ACK_0 usage, EE_Credit_CNT management is not applicable.
- f) The Sequence Initiator decrements the EE_Credit_CNT by a value of one for each ACK_1 (parameter field: History bit = 1, ACK_CNT = 1), F_BSY(DF), F_RJT, P_BSY, or P_RJT received.
- g) For an ACK_1 (parameter field: History bit = 0, ACK_CNT = 1) received, the Sequence Initiator decrements the EE_Credit_CNT by a value of one for the current SEQ_CNT in the ACK_1 plus all unacknowledged Data frames with lower SEQ_CNTs. If any of these ACKs with lower SEQ_CNT is received later, it is ignored and Credit_Count is not decremented.
- h) For an ACK_0 (parameter field: History bit = 0, ACK_CNT = 0) received, the Sequence Initiator recognizes that the Sequence has been received successfully or unsuccessfully, or that the interlock is being completed (see 14.3.2 and 14.3.2.2), but does not perform any EE_Credit_CNT management.
- i) For an ACK_1 with EOFt or EOFdt received, even if the History bit contains a value of 1, the Sequence Initiator shall recover the Credit for the Sequence by decrementing the EE_Credit_CNT by an additional value equal to all unacknowledged Data frames with lower SEQ_CNT of the Sequence. If any of these ACKs with lower SEQ_CNT is received later, it is ignored and Credit_Count is not decremented.

20.4.3 Sequence Recipient

- a) The Sequence Recipient is responsible for acknowledging valid Data frames received (see 14.3.2.2).
- b) The Sequence Recipient is allowed to use ACK_0 or ACK_1 as determined during N_Port_Login (see 17.6). The Sequence Recipient rules for using ACK_0 or ACK_1 are different and are listed for a non-streamed Sequence first, followed by additional rules for streamed Sequences.

20.4.3.1 ACK_0

If ACK_0 is used (see 14.3.2 and 14.3.2.2), the following rules apply to the Sequence Recipient:

- a) ACK_0 shall not participate in end-to-end flow control.
- b) A single ACK_0 per Sequence shall be used to indicate successful or unsuccessful Sequence delivery at the end of the Sequence except under specified conditions (see 14.3.2 and 14.3.2.2).
- c) Both the History bit and the ACK_CNT of the Parameter field shall be set to 0 (see 14.3.2.2).
- d) The ACK_0 used at the end of a Sequence shall have the End_Sequence bit set to 1. The ACK_0 used at the end of a Sequence shall be ended with EOF_t or EOF_{dt} in Class 1 and with EOF_t in Class 2.

20.4.3.2 ACK_1

If ACK_1 is used, the following rules apply to the Sequence Recipient:

- a) For each valid Data frame acknowledged an ACK_1 shall be sent with ACK_CNT set to 1.
- b) The History bit of the Parameter field shall be set to 1 if at least one ACK is pending for a previous SEQ_CNT for the Sequence, or shall be set to 0 if no ACK is pending for any previous SEQ_CNT for the Sequence (see 14.3.2.2).
- c) In Class 1, the Sequence Recipient shall withhold transmission of the last ACK_1 until all preceding ACK_1s corresponding to all Data frames with previous SEQ_CNTs have been transmitted. The last ACK_1 in Class 1 shall have the End_Sequence bit set to 1, History bit set to 0 and shall contain EOF_t or EOF_{dt}.
- d) In Class 2, the last ACK_1 shall be issued by the Sequence Recipient in one of the two ways specified.
 - 1) In Class 2 the Sequence Recipient shall withhold transmission of the last ACK_1 until all preceding Data frames with lower SEQ_CNTs have been received, processed, and corresponding ACK_1s transmitted (see 18.3.10). In this case, the last ACK_1 transmitted by the Sequence Recipient shall have the End_Sequence bit set to 1, History bit set to 0 and shall contain EOF_t.
 - 2) In Class 2, in response to the last Data frame (End_Sequence bit = 1) transmitted by the Sequence Initiator, if any of the Data frame is pending for the Sequence, the Sequence Recipient is allowed to transmit ACK_1 (with End_Sequence bit set to 0) but with EOF_n in lieu of EOF_t. In this case, the last ACK_1 transmitted by the Sequence Recipient shall have the End_Sequence bit set to 1, History bit set to 1 and shall contain EOF_t.

20.4.3.3 Last ACK timeout

If a Sequence error is detected or the E_D_TOV expires when the Sequence Recipient is withholding the last ACK for a Sequence and waiting to send other ACKs for that Sequence, the Sequence Recipient supporting discard policy shall set Abort Sequence bits and transmit the last ACK (see 18.6.5). The Sequence Recipient supporting the Process Policy shall transmit the last ACK without setting the Abort Sequence bits (see 18.3.10.4).

20.4.3.4 Streamed Sequences

Each of the streamed Sequences shall follow all the rules for a non-streamed Sequence.

In addition, in the case of multiple Sequence discard policy, the last ACK for the succeeding Sequence shall be withheld until all the previous Sequences are complete and deliverable. This additional withholding, for previous Sequences to complete and be deliverable, is not applicable to the case of Single Sequence discard policy.

20.4.4 EE_Credit

EE_Credit is the number of receive buffers in the Sequence Recipient that have been allocated to a given Sequence Initiator. EE_Credit represents the maximum number of unacknowledged or outstanding frames that can be transmitted without the possibility of overrunning the receiver at the Sequence Recipient. EE_Credit is defined per Class per Sequence Recipient and managed by the Sequence Initiator. Class 1 EE_Credit represents the number of Class 1 receive buffers and Class 2 EE_Credit represents the number of Class 2 buffers allocated to the Sequence Initiator. EE_Credit is not applicable to Class 3. The value of EE_Credit allocated to the Sequence Initiator is conveyed to this N_Port through the EE_Credit field of the Service Parameters. The minimum or default value of EE_Credit is one (1).

EE_Credit is used as a controlling parameter in end-to-end flow control.

20.4.5 EE_Credit_Count

EE_Credit_CNT is defined as the number of unacknowledged or outstanding frames awaiting a response and represents the number of receive buffers that are occupied at the Sequence Recipient. To track the number of frames transmitted and outstanding, the Sequence Initiator uses the above variable.

20.4.6 EE_Credit management

EE_Credit management involves a N_Port establishing and revising EE_Credit with the other N_Port it intends to communicate with, for Class 1 or Class 2 or both.

N_Port_Login is used to establish and optionally revise these EE_Credit values. The Estimate Credit procedure may be used to estimate and revise End-to-End Credit for streaming. The Advise Credit Sequence and associated Accept Sequence may also be used as a stand-alone procedure to revise the EE_Credit (see 16.4). The Service Parameters interchanged during N_Port_Login provide the Class 1 or Class 2 EE_Credit separately in their respective Credit fields.

EE_Credit is obtained by a Sequence Initiator during N_Port_Login from the N_Port to which it is logging in. EE_Credit allocated by the Sequence Recipient forms the maximum limit for the EE_Credit_CNT value. The EE_Credit_CNT value is set at zero (0), at the end of initialization, Login or re-Login. The EE_Credit_CNT is incremented, decremented or left unaltered as specified by the flow control management rules (see 20.4.1). The EE_Credit_CNT shall not exceed the EE_Credit value to avoid possible overflow at the receiver.

The Sequence Initiator shall allocate the total N_Port Credit associated with a Sequence Recipient among all active Sequences associated with that Sequence Recipient. The Sequence Initiator function may dynamically alter the Credit associated with each active Sequence as long as the total N_Port Credit specified for the Sequence Recipient is not exceeded. In the event of an abnormal termination of a Sequence using the Abort Sequence Protocol, the Sequence Initiator may reclaim the Sequence Credit allocation when the Accept response has been received to the Abort Sequence frame.

The N_Port is responsible for managing EE_Credit_CNT using EE_Credit as the upper bound on a per N_Port basis.

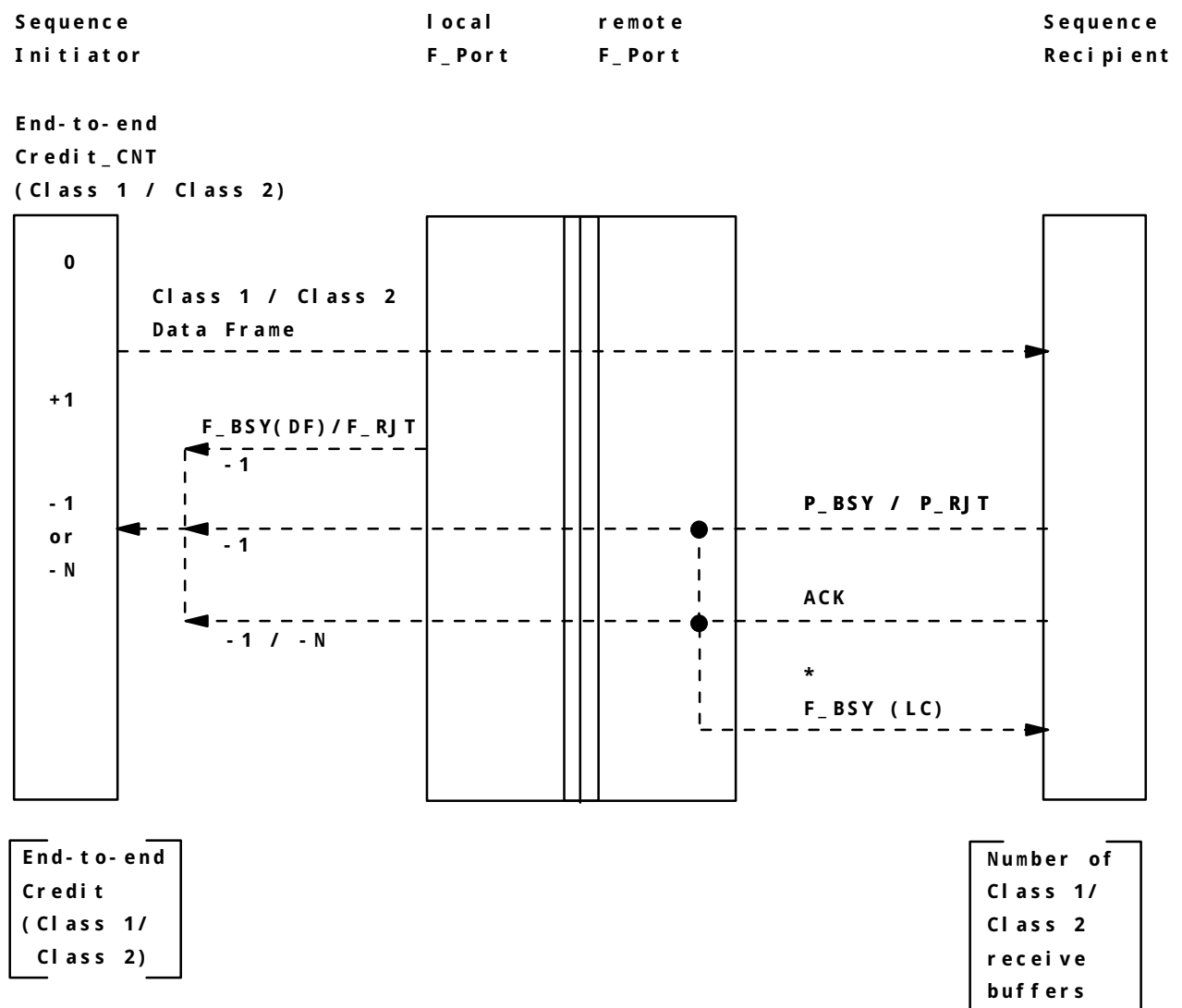
20.4.7 End-to-end flow control model

The end-to-end flow control model is illustrated in Figure 53. The model includes flow control parameters, control variables and resources for a Data frame from a Sequence Initiator and ACK_1 or BSY/RJT in response from the Sequence Recipient.

- The Sequence Recipient provides a number of Class 1 and/or Class 2 receive buffers.

- The Sequence Initiator obtains the allocation of Class 1 and/or Class 2 receive buffers, as Class 1 and Class 2 EE_Credits, respectively. That allocation is distributed among all the active Sequences for a specific Sequence Recipient.
- The Sequence Initiator manages the end-to-end flow by managing Class 1 and Class 2 end-to-end Credit-CNT(s). (That management is distributed among all the active Sequences for a specific Sequence Recipient.)

The model illustrates all possible replies to the Data frame. The EE_Credit_CNT is decremented by one or N depending upon the type of Link_Control frame received.



Legend:

N = number of valid Data frames acknowledged collectively

* F_BSY(LC) is a valid response only to Class 2

Figure 53 - End-to-end flow control model

20.4.8 End-to-end Class dependency

Allocation of EE_Credit and management of EE_Credit_CNT have some variations dependent upon Class of Service.

End-to-end Credit allocation

- Each Sequence Recipient may allocate the same Class 1 N_Port Credit value to each N_Port it is logging into. This Class 1 Credit value may be the maximum supportable by the Sequence Recipient.
- Each Sequence Recipient allocates some number of its receive buffers for Class 2 Service to N_Ports it is logging into. The sum of allocated Class 2 buffers may exceed the total number of Class 2 buffers supported at the Sequence Recipient. This excess buffer allocation shall not result in overrun. Class 2 EE_Credit allocation depends upon system requirements, which are outside the scope of FC-FS.

EE_Credit_CNT management

- Since Class 2 supports demultiplexing to multiple Sequence Recipient, the Sequence Initiator manages a Connectionless EE_Credit_CNT for each Sequence Recipient currently active, with that Sequence Recipient's EE_Credit as the upper bound.
- A Class 3 N_Port does not perform EE_Credit_CNT management.

20.4.9 EE_Credit recovery

In Class 1 and Class 2, the Sequence Initiator can recover EE_Credit when a Sequence is terminated, by the number of unacknowledged Data frames associated with the Sequence being terminated. Termination may be normal or abnormal.

In Class 1 and Class 2, EE_Credit may be recovered by the Sequence Initiator within the Sequence by detection of SEQ_CNT discontinuity in ACK, if the ACK received contains zero in the History bit of the Parameter field. Otherwise, the Sequence Initiator at the termination of the Sequence can recover EE_Credit.

If an ACK is received which contains a zero in the History bit of the parameter field, then EE_Credit for any outstanding ACK with lower SEQ_CNT shall be reclaimed. If any of these outstanding ACKs arrive later due to misordering, they shall not affect the EE_Credit_Count.

Class 1 EE_Credit is also recovered when a Dedicated Connection is removed by either EOFdt or by the Link Reset Protocol.

In Class 1 and Class 2, EE_Credit may also be recovered by a N_Port through re-Login.

Class 2 EE_Credit may also be reinitialized to N_Port_Login value using Link Credit Reset (see 14.3.4).

20.5 Buffer-to-buffer flow control

Buffer-to-buffer flow control is an FC-2 staged control process to pace the flow of frames. The buffer-to-buffer control occurs in both directions between

- Sequence Initiator and local F_Port
- remote F_Port and Sequence Recipient N_Port
- Sequence Initiator and Sequence Recipient N_Ports in point to-point topology

20.5.1 Buffer-to-buffer management rules summary

Buffer-to-buffer flow control rules are summarized. Managing BB_Credit_CNT at a N_Port or a F_Port is summarized in Table 214.

- a) Each Port (N_Port or F_Port) is responsible for managing the BB_Credit_CNT.
- b) The sending N_Port or F_Port shall not transmit a Class 2, Class 3, or Class 1/SOFc1 frame unless the allocated BB_Credit is greater than zero and the BB_Credit_CNT is less than this BB_Credit. To avoid possible overrun at the receiver, each port manages BB_Credit_CNT less than BB_Credit.
- c) Each Port sets BB_Credit_CNT value to zero (0) at the end of Fabric Login, or Fabric re-Login.
- d) Each Port increments BB_Credit_CNT by one (1) for each Class 2, Class 3, or Class 1/SOFc1 frame transmitted and decrements by one (1) for each R_RDY received.
- e) Each Port issues an R_RDY for each Class 2, Class 3, or Class 1 frame with SOFc1 received (see 14.3.2.1).

20.5.2 BB_Credit

BB_Credit represents the number of receive buffers supported by a Port (N_Port or F_Port) for receiving Class 1/SOFc1, Class 2, or Class 3 frames. BB_Credit values of the attached ports are mutually conveyed to each other during the Fabric Login through the Credit field of common Service Parameters (see 17.6). The minimum or default value of BB_Credit is one (1).

BB_Credit is used as the controlling parameter in buffer-to-buffer flow control.

20.5.3 BB_Credit_Count

BB_Credit_CNT is defined as the number of unacknowledged or outstanding frames awaiting R_RDY responses from the directly attached port. It represents the number of receive buffers that are occupied at the attached port. To track the number of frames transmitted for which R_RDY responses are outstanding, the transmitting Port uses the above variable.

20.5.4 BB_Credit management

BB_Credit management involves a Port receiving the BB_Credit value from the Port to which it is directly attached. Fabric Login is used to accomplish this. The common Service Parameters interchanged during Fabric Login provide these values (see 17.6 and 17.7).

The transmitting Port is responsible to manage BB_Credit_CNT with BB_Credit as its upper bound.

20.5.5 Buffer-to-buffer flow control model

The Buffer-to-buffer flow control model is illustrated in Figure 54. The model includes flow control parameters, control variables for a Class 2, Class 3, or Class 1/SOFc1 frame and R_RDY as its response, and the resources for Connectionless service. All possible responses to a Class 2 or Class 3 Data frame are illustrated.

- Each N_Port and F_Port provides a number of receive buffers for Connectionless service.
- Each N_Port obtains the allocation of receive buffers from the F_Port (or N_Port in case of point-to-point topology) to which it is attached, as BB_Credit. Each F_Port obtains the allocation of receive buffers from the N_Port to which it is attached, as total BB_Credit for Connectionless service.
- Each Port manages the buffer-to-buffer flow by managing the BB_Credit_CNT for the Connectionless service, with BB_Credit as the upper limit.

20.5.6 Buffer-to-buffer Class dependency

Allocation of BB_Credit and management of BB_Credit_CNT for Connectionless service are described.

BB_Credit allocation

Each Port allocates the total number of Connectionless buffers to the Port to which it is directly attached.

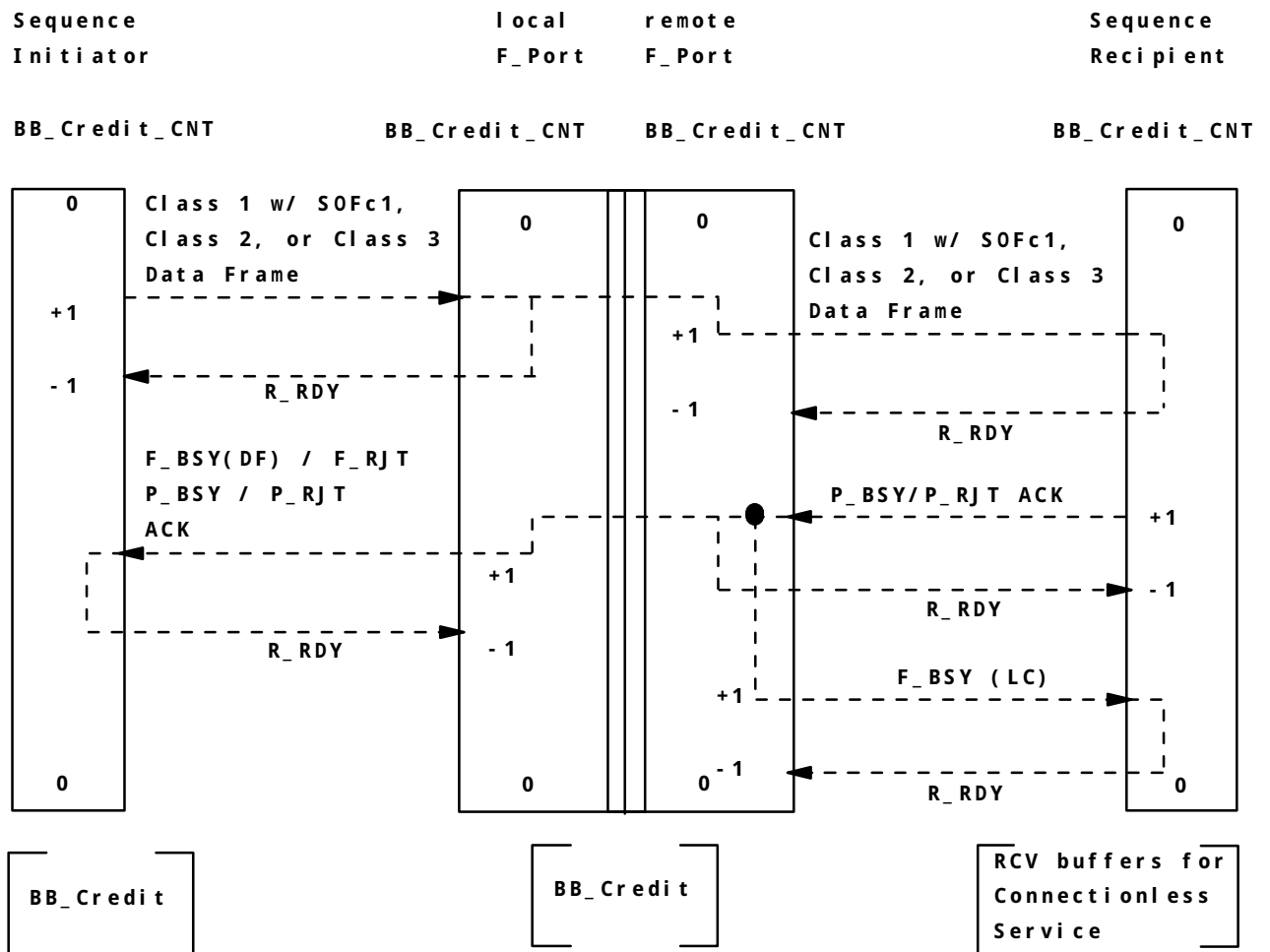
BB_Credit_CNT management

A Port manages the BB_Credit_CNT with BB_Credit as the upper bound.

20.5.7 Class dependent frame flow

The Class dependent flow of frames accomplishing buffer-to-buffer flow control for the following cases are illustrated:

- Class 1/SOFc1 frame with delivery or non-delivery to the Fabric (see Figure 55).
- Possible responses from the F_Port or a N_Port within the Fabric to a Class 1/SOFc1 frame are illustrated.
- Class 1/SOFc1 frame with delivery or non-delivery to a N_Port (see Figure 56).
- Possible responses from the F_Port and the destination N_Port to a Class 1/SOFc1 frame are illustrated.
- Class 2 with delivery or non-delivery to the Fabric (see Figure 57).
- Possible responses from the F_Port or a N_Port within the Fabric to a Class 2 are illustrated.
- Class 2 with delivery or non-delivery to a N_Port (see Figure 58).
- Possible responses from the F_Port and the destination N_Port to a Class 2 Data frame are illustrated.
- Class 3 (see Figure 59).
- Possible responses from the F_Port and the destination N_Port to a Class 3 Data frame are illustrated.



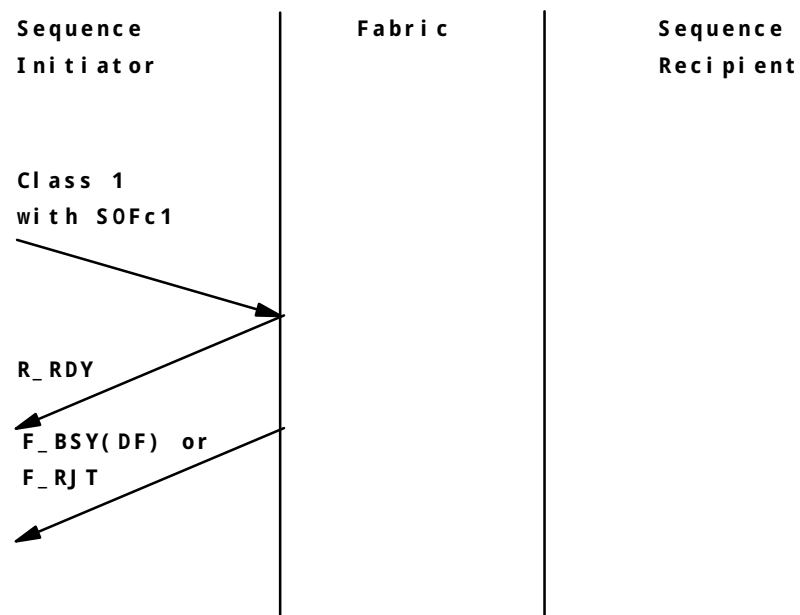


Figure 55 - Class1/SOFc1 frame flow with delivery or non-delivery to the fabric

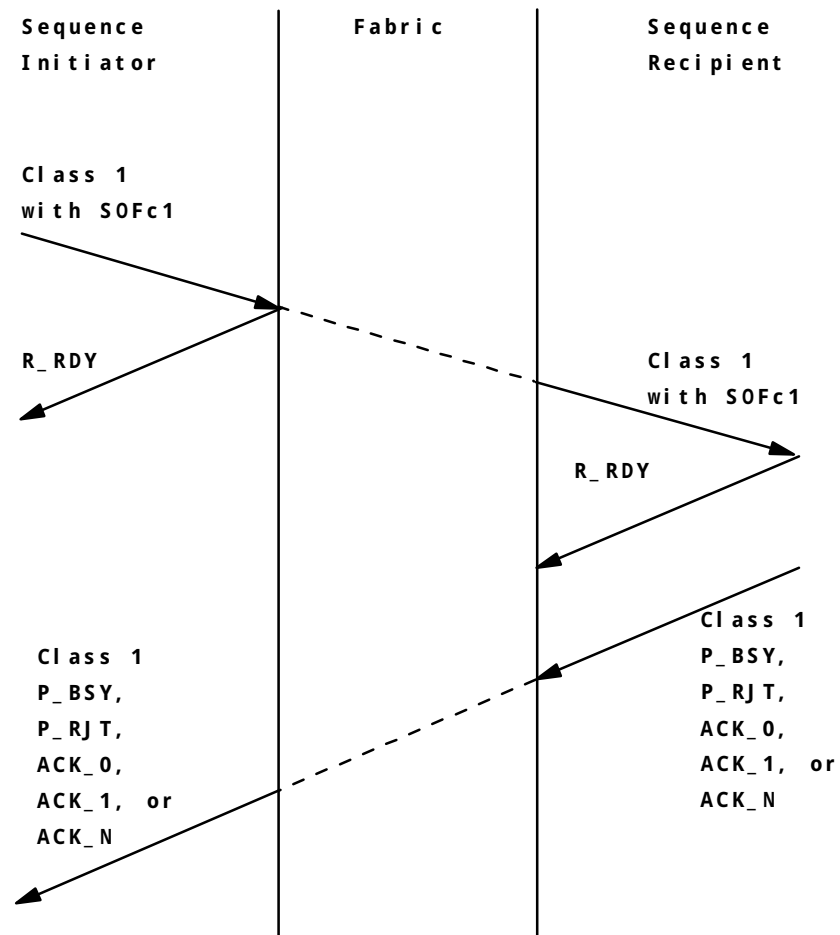


Figure 56 - Class1/SOFc1 frame flow with delivery or non-delivery to a N_Port

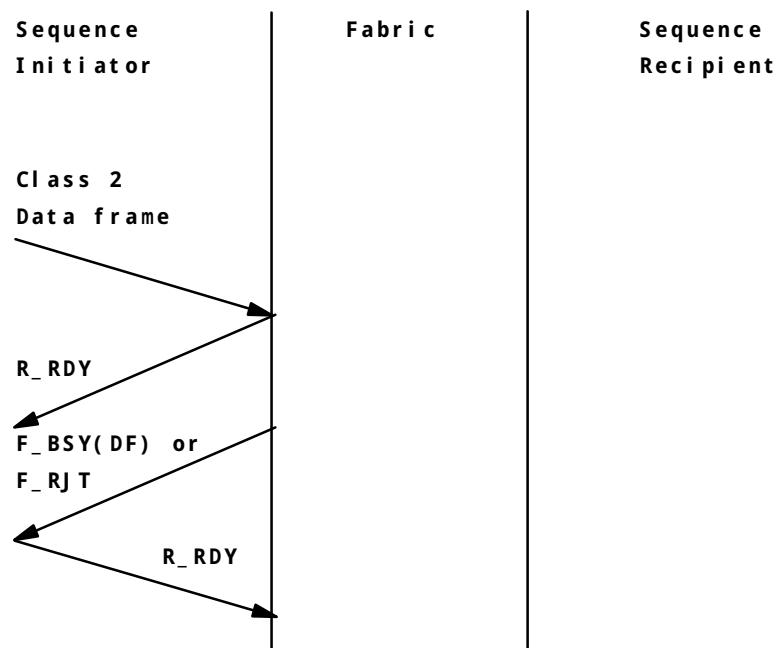


Figure 57 - Buffer-to-buffer - Class2 frame flow with delivery or non-delivery to a Fabric

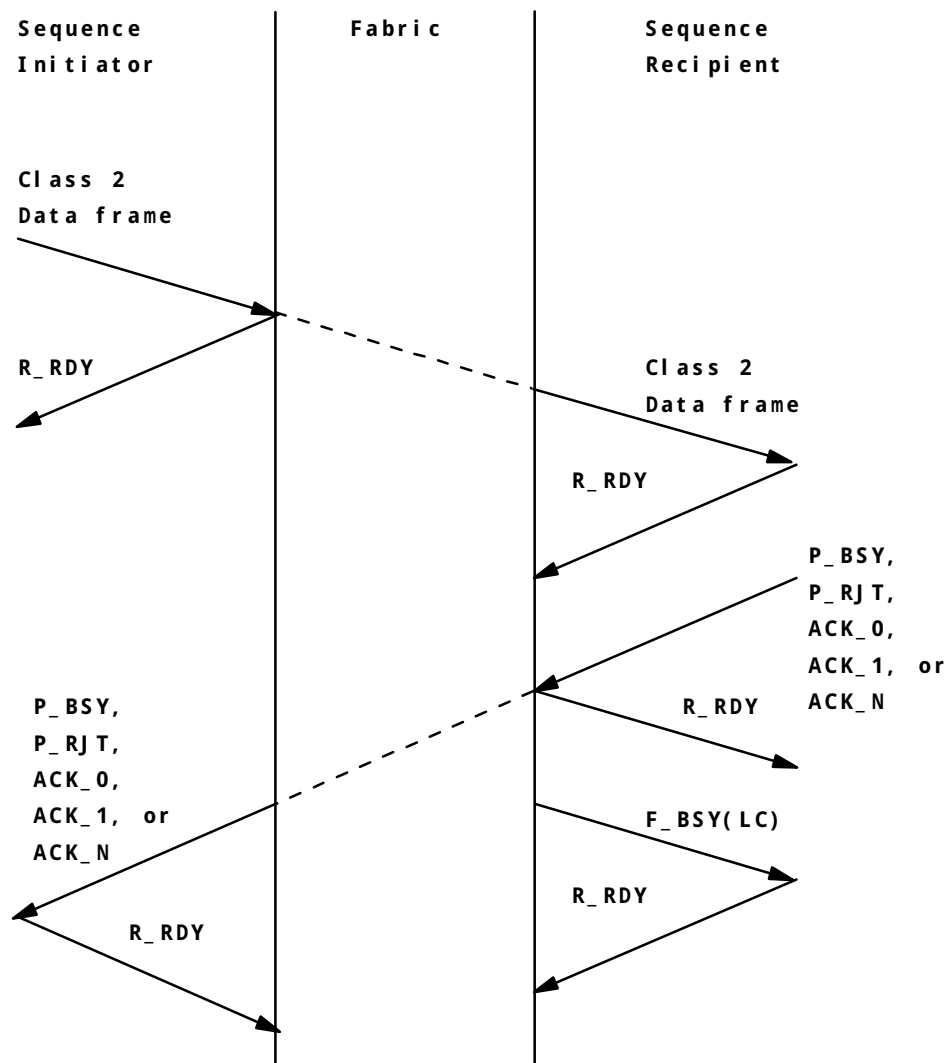


Figure 58 - Buffer-to-buffer - Class2 frame flow with delivery or non-delivery to a N_Port

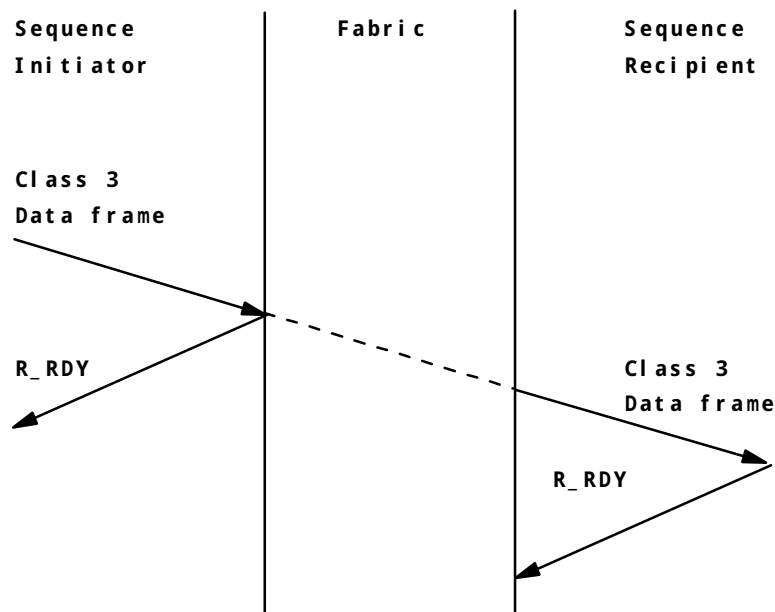


Figure 59 - Buffer-to-buffer - Class3 frame flow

20.5.8 R_RDY

R_RDY is the pacing mechanism exclusively used for buffer-to-buffer flow control. For any Class 2, Class 3, or Class 1/SOF₀₁ frame received at a F_Port or at a N_Port, each Port issues a R_RDY primitive.

20.5.9 BB_Credit_Count reset

BB_Credit_Count is reinitialized to login value for both N_Port and F_Port on Fabric re-Login or after the Link Reset Protocol has been performed.

20.5.10 Alternate buffer-to-buffer Credit management

An alternate buffer-to-buffer credit management may be used instead of the one described in 20.5.1 through 20.5.9.

The use of alternate buffer-to-buffer Credit management shall be indicated by the N_Port through a N_Port_Login Service Parameter during Fabric Login and N_Port_Login (see 17.6.2.3 and 17.6.3.3).

20.5.11 Alternate BB_Credit management rules summary

Alternate BB_Credit management rules are summarized:

- a) Each Port is responsible for managing the Alternate BB_Credit.
- b) During Login, BB_Credit shall be set to a value that represents the number of receive buffers that a Port shall guarantee to have available as soon as a circuit is established. If this value is greater than zero, the Port may start transmitting a frame without waiting for R_RDYs. If this value is equal to zero, the sending Port has to wait to receive at least one R_RDY, before transmitting a frame.
- c) The receiving Port shall transmit at least one R_RDY, representing the number of additional receive buffers available, before, during, or after the reception of frames.

- d) The transmitting Port shall decrement BB_Credit by one for each frame transmitted and increment by one for each R_RDY received.
- e) For transmitting frames, the Available Credit shall be greater than zero. The Available Credit at any given time is expressed by the following equation:

Equation 5 - Available Credit

Available Credit = Login_BB_Credit + (Number of R_RDYs received - Login_BB_Credit)* -
Number of frames transmitted

where

- Number of R_RDYs received $\geq$ Login_BB_Credit.
- The parenthetical expression is applicable only if it is positive, otherwise it is zero.

20.5.12 BB_Credit Recovery

BB_Credit recovery as described in this section shall only be performed when two ports, not operating in arbitrated loop mode, have logged in with each other and have agreed to a non-zero BB_SC_N value. See 17.3 and **Error! Reference source not found.** for a description of the login process.

A port which can support BB_Credit Recovery must maintain the following BB_Credit Recovery counters.

- BB_SC_N is the $\log_2$ of BB_Credit Recovery modulus.
- BB_RDY_N counts the number of R_RDY primitives received modulo $2^{BB_SC_N}$.
- BB_FRM_N counts the number of frames received modulo $2^{BB_SC_N}$;

To initialize the counters, each port capable of supporting BB_Credit Recovery shall perform the following steps:

- Set all counters for all ports to zero before fabric login.
- After having sent or received the appropriate login (whether the first login after reset or a re-login), if the BB_SC_N value communicated by the other port in the login is zero, discard or set to zero all counters. Otherwise, set BB_SC_N to the greater of BB_SC_N from the other port's login parameters or its own BB_SC_N value, and set to zero all other BB_Credit Recovery counters.
- After explicit or implicit logout, discard or set to zero all counters.

To recover any lost BB_Credit, each port shall perform the following steps (Each step of initialization and operation should be processed atomically, that is, it must be completed before any other BB_Credit recovery operation):

- a) Send a BB_SCs primitive if $2^{BB_SC_N}$ number of frames that requires BB_Credit have been sent since the start of communication, or since the last time a BB_SCs primitive was sent.
- b) Send a BB_SCr primitive if $2^{BB_SC_N}$ number of R_RDY primitives have been sent since the start of communication, or since the last time a BB_SCr primitive was sent.
- c) After receiving each R_RDY, increment BB_RDY_N by one. If BB_RDY_N equals $2^{BB_SC_N}$, set BB_RDY_N to zero.
- d) After receiving each frame, increment BB_FRM_N by one. If BB_FRM_N equals $2^{BB_SC_N}$, set BB_FRM_N to zero.
- e) When a BB_SCr primitive is received, the number of BB_Credits lost can be calculated using the following equation:

$$(2^{BB_SC_N} - BB_RDY_N) \text{ modulo } 2^{BB_SC_N}.$$

The BB_Credit_CNT shall then be decremented by the number of BB_Credits lost. BB_RDY_N is then reset to zero, before the next R_RDY is received.

- f) When a BB_SCs primitive is received, the number of BB_Credits the other port has lost can be calculated using the following equation:

$$(2^{BB_SC_N} - BB_FRM_N) \text{ modulo } 2^{BB_SC_N}.$$

One R_RDY shall be resent for each BB_Credit that is lost. BB_FRM_N is then reset to zero, before the next frame is received.

20.6 BSY / RJT in flow control

In Class 1 end-to-end flow control, neither F_BSY, F_RJT, nor P_BSY occurs, except for a Class 1/SOF_{c1}. In Class 2 end-to-end flow control, F_BSY, F_RJT, P_BSY or P_RJT may occur for any Data frame. Each of these responses contributes to end-to-end and buffer-to-buffer flow controls.

End-to-end Class 2 buffers at the Sequence Recipient N_Port are shared by multiple source N_Ports which may be multiplexing Data frames. This Class 2 multiplexing requires the distribution of Class 2 Credit to each source N_Port to be honored to prevent BSY or RJT. Unless an adequate number of end-to-end Class 2 buffers are available and Credit distribution is honored, a BSY or RJT may occur in Class 2. If buffer-to-buffer flow control rules are not obeyed by the transmitter, the results are unpredictable e.g., if a Class 2 frame is received without Credit and the receiver does not have a buffer to receive it, the receiver may discard the frame without issuing a P_BSY or P_RJT.

20.7 LCR in flow control

LCR does not need EE_Credit and does not participate in end-to-end flow control. LCR participates only in buffer-to-buffer flow control as Class 2. (see Figure 60). In response to an LCR, the Fabric shall issue an R_RDY and may issue a F_BSY or F_RJT. The destination N_Port shall issue an R_RDY and may issue a P_RJT (see 14.3.4). The destination N_Port shall not issue a P_BSY to an LCR.

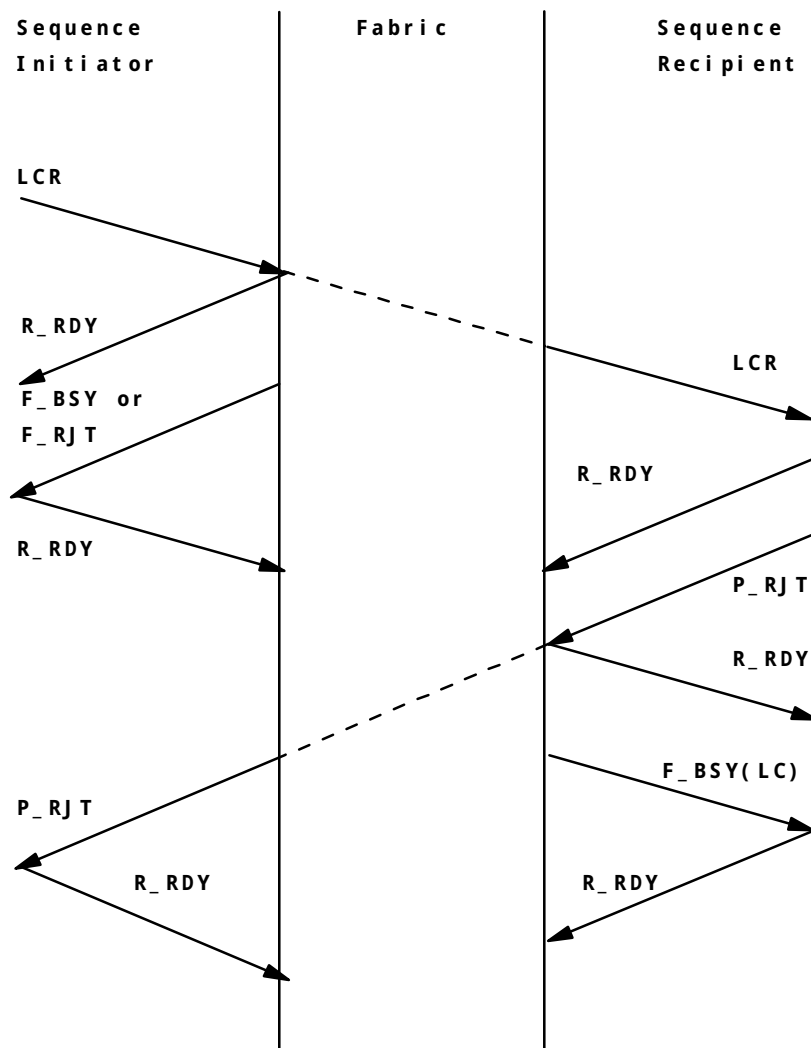


Figure 60 - LCR frame flow and possible responses

Flow control model for an LCR frame is illustrated in Figure 61, which covers the buffer-to-buffer flow control for all possible responses to an LCR.

EE_Credit recovery

After issuing the LCR, the Sequence Initiator shall reinitialize its EE_Credit to N_Port_Login value for the destination N_Port. The Sequence Initiator shall not wait for any response before reinitializing this Credit (see 14.3.4).

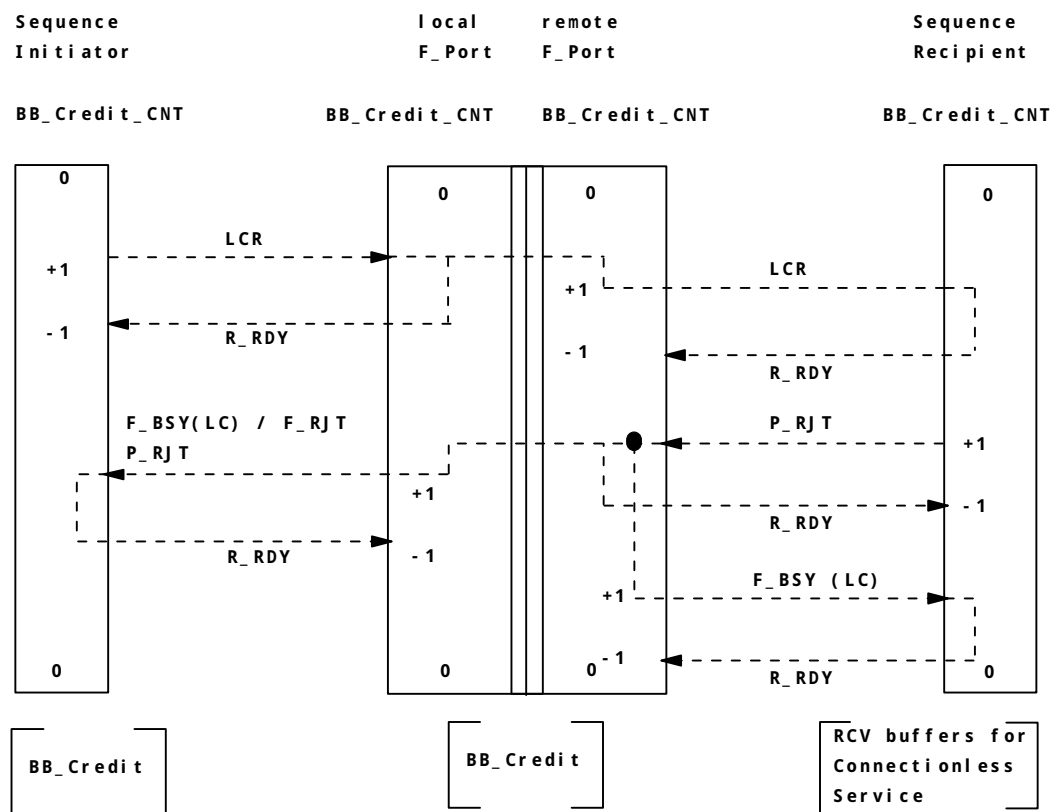


Figure 61 - LCR flow control model

20.8 Integrated Class 2 flow control

Integrated buffer-to-buffer and end-to-end flow controls applicable to Class 2 is illustrated in Figure 62 for a Data frame from the Sequence Initiator and its response from the Sequence Recipient.

Management

Integrated Class 2 flow control management is summarized in Table 215.

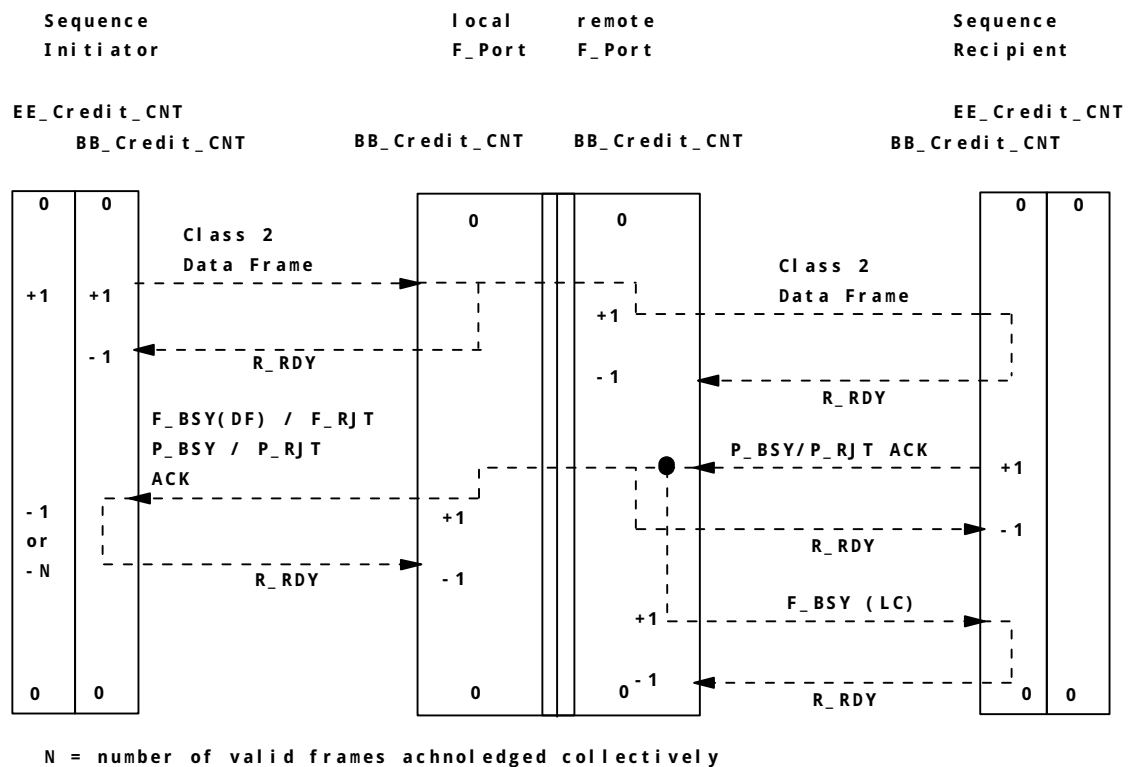


Figure 62 - Integrated Class 2 flow control

20.9 Intermix

The flow control model for Intermix is represented by the superposition of end-to-end and Buffer-to-buffer flow controls (Figure 53 and Figure 54).

20.10 Point-to-point topology

All the flow control models specified above in this clause apply to Fabric topology. The flow control model for Point-to-point topology is represented by the corresponding model for the Fabric topology, without the flow of F_BSY(DF), F_BSY(LC), and F_RJT.

Table 213 - End-to-end flow control management	
Activity	EE_Credit_CNT (N_Port only)
N_Port transmits a Class 1 or Class 2 Data frame	+1
N_Port transmits an LCR	And reinitializes End-to-end Credit for the destination N_Port to its Login value.
N_Port receives F_BSY (DF), F_RJT, P_BSY, or P_RJT	-1
N_Port receives F_BSY (LC)	NA
N_Port receives ACK_1 (Parameter field: History bit = 1, ACK_CNT = 1_	-1
N_Port receives ACK_1 (Parameter field: History bit = 0, ACK_CNT = 1)	subtract 1 for current SEQ_CNT of the ACK_1 and also subtract all unacknowledged lower SEQ_CNTs (see 14.3.2.2)

N_Port receives ACK_0 (Parameter field: History bit = 0, ACK_CNT = 0)	NA (see 14.3.2.2)
N_Port receives Data frame (Class 1, Class 2 or Class 3)	NA
N_Port receives an LCR	NA*
N_Port transmits a Class 3 Data frame	NA
N_Port transmits P_BSY or P_RJT	NA
N_Port transmits ACK	NA
Notes:	
NA = Not applicable	
N - number of valid frames acknowledged collectively	
* On receipt of LCR, the Sequence Recipient rests buffer (see 14.3.4)	

Table 214 - Buffer-to-buffer flow control management	
Activity	BB_Credit_CNT (N_Port or F_Port)
N_Port or F_Port transmits any Class 2, Class 3, Class 1 frame with SOFc1 (including F_BSY(DF), F_BSY(LC), F_RJT, P_BSY, P_RJT or LCR_)	+1
N_Port or F_Port receives R_RDY	-1
N_Port or F_Port receives any Class 2, Class 3, or Class 1 frame with SOFc1 (including F_BSY(DF), F_BSY(LC), F_RJT, P_BSY, P_RJT or LCR)	NA
N_Port or F_Port transmits R_RDY	NA
Note - NA = Not applicable	

Table 215 - Integrated Class 2 flow control management			
Activity	N_Port		F_Port
	EE_Credit_CNT	BB_Credit_CNT	BB_Credit_CNT
Port transmits a Class 2 Data frame	+ 1	+1	+1
Port transmits an LCR	Reinitializes to Login value.	+1	+1
Port receives R_RDY	NA	-1	-1
Port receives F_BSY(DF), F_RJT, P_BSY, or P_RJT	-1	NA	NA
Port receives F_BSY(LC)	NA	NA	NA
Port receives ACK_1 (Parameter field: History bit = 1, ACK_CNT = 1)	-1	NA	NA
Port receives ACK_1 (Parameter field: History bit = 0, ACK_CNT = 1)	subtract 1 for current SEQ_CNT of the ACK_1 and also subtract all unacknowledged lower SEQ_CNTs (see 14.3.2.2)	NA	NA
Port receives ACK_0 (Parameter field: History bit = 0, ACK_CNT = 0)	NA (see 14.3.2.2)	NA	NA
Port receives an LCR	NA*	NA	NA
Port receives a Class 2 Data frame	NA	NA	NA

Table 215 - Integrated Class 2 flow control management			
Activity	N_Port		F_Port
	EE_Credit_CNT	BB_Credit_CNT	BB_Credit_CNT
Port transmits R_RDY	NA	NA	NA
Port transmits F_BSY, F_RJT, P_BSY, P_RJT, or ACK	NA	+1	+1
Note - NA - Not Applicable N - number of valid frames acknowledged collectively * On receipt of LCR, the Sequence Recipient rests buffer (see 14.3.4)			

21 Segmentation and reassembly

21.1 Introduction

Segmentation and reassembly are the FC-2 functions provided to subdivide application data corresponding to an Information Category to be transferred into Payloads, compute Relative Offset values, embed each Payload and the corresponding Relative Offset value in an individual frame, transfer these frames over the link, and reassemble at the receiving end the application data of the Information Category as sent by the sending end.

Application data mapping

Mapping application data to Upper Level Protocols (ULPs) is outside the scope of FC-FS. ULPs maintain the status of application data transferred.

21.2 Sending end

An upper level (a level above FC-2) at the sending end shall define a Relative Offset space for each Information Category for which Relative Offset usage is supported. An upper level specifies a block or a collection of subblocks to be transferred within a Sequence.

21.2.1 Relative Offset space

The Relative Offset space shall start from zero, representing an upper level-defined origin, and extend to its highest value. Application data for an Information Category transferred may be mapped to all or portions of Relative Offset space.

21.2.2 Block

A block is an upper level construct of application data related to a single Information Category and transferred within a single Sequence. If Continuously Increasing Relative Offset is used, a block shall be specified. A block is allowed to map to all or part of Relative Offset space of the Information Category. An upper level shall specify an Initial Relative Offset (IRO₀) for each block. The Initial Relative Offset value is allowed to be zero or non-zero.

21.2.3 Subblock

A subblock is an upper level construct that contains partial data for a single Information Category. A collection of subblocks is specified for a given Information Category to be transferred within a Sequence. The subblocks shall be specified, only if the Sequence Recipient supports Random Relative Offset. An upper level shall specify an Initial Relative Offset for each subblock. The Initial Relative Offset value is allowed to be zero or non-zero. The Initial Relative Offset values of sub-blocks of a given Information Category transmitted consecutively are allowed to be random.

The collection of subblocks for a single Information Category is allowed to map to portions or all of Relative Offset space for the Information Category.

21.2.4 Sequence

The blocks to be transferred within a single Sequence and the Information Category for each block shall be specified to FC-2 by an upper level. The collections of subblocks to be transferred within a Sequence and the information Category for each subblock shall be specified to FC-2 by an upper level.

21.2.5 Relationship between Sequences

An upper level shall specify the Relative Offset relationship between multiple Sequences of a given Information Category. This relationship is transparent to FC-2.

21.3 FC-2

Mechanisms

FC-2 mechanisms to support this function are

- a) Relative Offset or SEQ_CNT
- b) Sequence
- c) F_CTL bit indicating the presence of Relative Offset in the parameter field (see 12.2)

Relative Offset

The Parameter field in the Frame_Header may be used to hold the value of Relative Offset which is the relative displacement of first byte of Payload related to an upper level-defined origin for a given Information Category.

Sequence Count (SEQ_CNT)

If Relative Offset is not used for reassembly, the Information Category shall be reassembled using the SEQ_CNT. The reassembly shall be performed by joining or concatenating the Payloads of the Data frames in the continuously increasing order of SEQ_CNTs starting from the SEQ_CNT of the first Data frame to that of the last Data frame of a given Sequence.

For an Information Category i within a given Sequence, the reassembly is expressed as:

Equation 6 - Information category

Application data $_i$ = PL $_i$ (m) || PL $_i$ (m+1) || ... || PL $_i$ (m+n) where

PL (m) represents Payload of a frame with SEQ_CNT m
 n is the total frame count within the Sequence
 m is the lowest SEQ_CNT (zero or non-zero) and
 m+n is the highest SEQ_CNT.

If the SEQ_CNT wraps to zero from hex 'FF FF' within a Sequence, the reassembly shall be continued according to modulo 65536 arithmetic (i.e., SEQ_CNT = 0 follows SEQ_CNT = hex 'FF FF').

21.4 Login

The following Common Service Parameters related to segmentation and reassembly are interchanged during N_Port_Login (see Figure 44):

- a) Relative Offset by Information Category
- b) Continuously Increasing or Random Relative Offset

Through the interchange of these Login parameters, the Sequence Recipient indicates its Relative Offset requirements to the Sequence Initiator. The Sequence Recipient indicates Relative Offset support or non-support for each Information Category. For the Relative Offset supported Information Categories, the Sequence Recipient collectively indicates Continuously Increasing or Random Relative Offset requirement.

The Sequence Initiator shall follow the Relative Offset requirements of the Sequence Recipient, for Information Categories supported and not supported.

21.5 Segmentation rules summary

Segmentation summary rules are listed and referred to Table 216.

- a) The Sequence Initiator shall be responsible for segmentation. The Sequence Initiator shall follow the Relative Offset requirements of the Sequence Recipient for Information Categories.
- b) An upper level at the sending end shall specify to the sending FC-2 one or more blocks to be transferred as a Sequence, the Information Category for each block, an Relative Offset space, and the Initial Relative Offset for each Information Category. The Initial Relative Offset value may be zero or non-zero.
- c) The Sequence Initiator shall use the specified Relative Offset space for each Information Category and transfer one or more blocks specified in a single Sequence.
- d) If the Sequence Recipient does not support Relative Offset for one or more Information Categories, the Sequence Initiator shall transmit each of these Information Categories as a contiguous block. The Sequence Initiator shall set the Relative Offset present bit in F_CTL to zero, indicating that the parameter field is not meaningful.
- e) If the Sequence Recipient supports Relative Offset for one or more Information Categories and has specified during Login this support as Continuously Increasing Relative Offset, the Sequence Initiator shall transmit each of these Information Categories with Continuously Increasing Relative Offset.
 - 1) The Sequence Initiator shall set the Relative Offset present F_CTL bit to one.
 - 2) The Sequence Initiator shall use the Initial Relative Offset specified by the upper level for the Relative Offset RO_i in the first frame of the block, namely, $RO_i(0) = \text{Initial Relative Offset (IRO}_i\text{)}$ for the Information Category i
 - 3) The Sequence Initiator shall use for all other frames of the block, the Relative Offset computed as follows: $RO_i(n+1) = RO_i(n) + \text{Length of Payload}_i(n)$ where n is ≥ 0 (zero) and represents the consecutive frame count of frames for a given Information Category within a single Sequence.
 - 4) Above steps 1 through 3 shall be repeated for each block within the Sequence.
- f) If the Sequence Recipient supports Relative Offset for one or more Information Categories and has specified during Login this support as Random Relative Offset, the Sequence Initiator is permitted to transmit each of these Information Categories with Random Relative Offset.
 - 1) The Sequence Initiator shall set the Relative Offset present F_CTL bit to one.
 - 2) The Sequence Initiator shall use the Initial Relative Offset specified by the upper level for the Relative Offset RO_i in the first frame of the subblock, namely, $RO_i(0) = \text{Initial Relative Offset (IRO}_i\text{)}$ for the Information Category i
 - 3) The Sequence Initiator shall use for all other frames of the subblock, the Relative Offset computed as follows: $RO_i(n+1) = RO_i(n) + \text{Length of Payload}_i(n)$ where n is ≥ 0 (zero) and represents the consecutive frame count of frames for the subblock for a given Information Category within a single Sequence.
 - 4) Above steps 1 through 3 shall be repeated for each subblock within the Sequence.

21.6 Reassembly rules summary

Reassembly summary rules are listed and referred to Table 217.

- a) The Sequence Recipient shall be responsible for reassembly of each Information Category received within the Sequence. The Sequence Recipient shall use Relative Offset or SEQ_CNT field, as specified, to perform the reassembly and make the blocks available to the receiving upper level as sent by the sending upper level.

- b) The Sequence Recipient shall reassemble each Information Category within its Relative Offset space specified by the sending upper level.
- c) If the Sequence Recipient has specified during Login non support of Relative Offset for one or more Information Categories, the Sequence Recipient shall verify Relative Offset present F_CTL bit for zero value and reassemble each of these Information Categories using SEQ_CNT.
- d) If this F_CTL bit is set to one for any of these Information Categories, the Sequence Recipient shall issue a P_RJT with a reason code of "Relative Offset not supported."
- e) If the Sequence Recipient has indicated during Login Relative Offset support for one or more Information Categories and specified this support as Continuously Increasing Relative Offset, the Sequence Recipient shall verify Relative Offset present F_CTL bit for one and assemble each of these Information Categories using Relative Offset.
- f) If the Sequence Initiator lacks the Relative Offset capability and has set the bit to zero, the Sequence Recipient shall use SEQ_CNT for reassembly.
- g) If the Sequence Recipient has indicated during Login Relative Offset support for one or more Information Categories and specified this support as Random Relative Offset, the Sequence Recipient shall verify Relative Offset present F_CTL bit for one and assemble each of these Information Categories using Relative Offset.
- h) If the Sequence Initiator lacks the Relative Offset capability and has set the bit to zero, the Sequence Recipient shall use SEQ_CNT for reassembly.
- i) If the Sequence Recipient supports Continuously Increasing Relative Offset and detects random Relative Offsets, the Sequence Recipient shall issue P_RJT with the reason code of "Relative Offset out of bounds".

Table 216 - Segmentation and reassembly rules summary			
Case	Relative Offset support by Sequence Recipient	Sequence Initiator action (Segmentation)	Sequence Recipient action (Reassembly)
1	Not supported	F_CTL Relative Offset present bit = 0 Parameter field meaning is protocol and IU specific	Relative Offset shall not be used (ignore parameter field) SEQ_CNT shall be used
		F_CTL Relative Offset present bit = 1 Parameter field = Relative Offset	Use P_RJT Reason code = Relative Offset not supported
2	Continuously Increasing Relative Offset supported	F_CTL Relative Offset present bit = 1 Parameter field = Relative Offset First frame of a block: $RO_i(0) = IRO_i$ for the block specified All other frames of the block: $RO_i(n+1) = RO_i(n) + \text{Length of Payload}_i(n)$	Relative Offset shall be used
		F_CTL Relative Offset present bit = 0 Parameter field meaning is protocol and IU specific	Ignore parameter field SEQ_CNT shall be used

Table 216 - Segmentation and reassembly rules summary			
Case	Relative Offset support by Sequence Recipient	Sequence Initiator action (Segmentation)	Sequence Recipient action (Reassembly)
3	Random Relative Offset supported	F_CTL Relative Offset present bit = 1 Parameter field = Relative Offset Initial Relative Offset for subblocks permitted to be random First frame of a subblock: $RO_i(0) = IRO_i$ for the subblock specified All other frames of the subblock: $RO_i(n + 1) = RO_i(n) + \text{Length of Payload}_i(n)$	Relative Offset shall be used
		F_CTL Relative Offset present bit = 0 Parameter field meaning is protocol and IU specific	Ignore parameter field SEQ_CNT shall be used
Note: If RO value in the Parameter field is out of bounds, the Sequence Recipient shall issue a P_RJT with a reason code of "Invalid Parameter field".			

22 Connection management

The procedures for establishing and removing Class 1 (see 16.1.1) and Class 6 (see 16.6.1) Dedicated Connections are specified in this clause.

22.1 Introduction

Class 1 Service is based on establishing a Dedicated Connection between a source N_Port and a destination N_Port. The Dedicated Connection guarantees that the full bandwidth of the Link is available to each N_Port.

Establishing a Connection

In order to establish a Class 1 Dedicated Connection, the source N_Port shall transmit a Data frame to a destination N_Port with a SOF_{c1} delimiter. The Data field size of the connect-request frame is limited by the maximum buffer-to-buffer Receive Data_Field size specified by the Common Service Parameters of the Fabric (or point-to-point N_Port) or by the maximum Receive Data_Field size specified by the destination N_Port, whichever is smaller.

The N_Port shall receive an R_RDY Primitive Signal to indicate that the connect-request frame was received successfully and a buffer in the F_Port or N_Port is available. No additional frames shall be transmitted for the pending Connection until the ACK frame has been received. When the N_Port transmitting the connect-request receives an ACK (ACK_0 or ACK_1) with an SOF_{n1} and with the appropriate S_ID and D_ID fields of the connect-request frame, a Dedicated Connection is established.

NOTES

1. A Connection is established from the N_Port's perspective (Connection Initiator) at the time the N_Port receives the ACK frame. It has no relation to the method or timing by which the Fabric actually forms the Dedicated Connection.
2. It is recommended that a Fabric not introduce a phase discontinuity (see FC-PI) while establishing a Class 1 Connection. No frame errors are introduced by such a discontinuity.

When a Dedicated Connection is established:

- Both N_Ports shall reinitialize their end-to-end Class 1 Credit to the Login values.
- the N_Port initiating the connection shall be known as the Connection Initiator and the N_Port responding to the connect-request shall be known as the Connection Recipient.
- the Connection Initiator may continue the initial Sequence, and
- the Connection Recipient may initiate new Sequences with an SOF_{i1} delimiter when a Dedicated Connection is not unidirectional, as indicated by the setting of F_CTL bit 8 (see 22.5.3).

Removing a Connection

Removing a Dedicated Connection is accomplished by either N_Port transmitting a frame terminated by an EOF_{dt} or the transmission or reception of the Link Reset Primitive Sequence. Normally, removing a Dedicated Connection shall be negotiated between the two N_Ports involved. Negotiation is required in order to avoid breaking the Dedicated Connection while frames are still flowing between the N_Ports.

In urgent situations which do not require the use of the Link Reset Primitive Sequence, a N_Port may request the immediate removal of a Dedicated Connection by transmitting the Remove Connection Basic Link Service frame. The ACK frame transmitted in response shall end with an EOF_{dt} delimiter.

The Fabric terminates a Dedicated Connection after an EOF_{dt} or EOF_{dti} passes through the F_Port in either direction. Any frames flowing in either direction at the time the Connection is removed may be corrupted.

End_Connection (E_C) is the control bit in F_CTL that is used to perform the negotiation.

Preempting a Dedicated Connection

Interrupting and removing a Class 1 or Class 6 Dedicated Connection between N_Ports and establishing a new Class 1 or Class 6 Dedicated Connection between an unconnected N_Port and one or more of the previously connected N_Ports is Preemption. A preemption is initiated by a N_Port sourcing a Preemption Connection Request frame (an SOFc1 with the Preemption flag set, Word 1 bit 24 along with F_CTL bit 17 = 1) to the Fabric.

The preempting N_Port then receives

- a) a reject Link_Response frame (F_RJT) with either a "Preemption Request Rejected" or a "Preemption Not Enabled" reason code, or
- b) a connection acknowledges (ACK_1 or ACK_0) with an SOFn1 having the appropriate S_ID and D_ID fields of the preemption request frame.

When the ACK is received a dedicated connection has been established. If an F_RJT is received no connection has been established and the preemption request process has ended.

NOTE - Fabric resolution of a preemption request may be based on the priority of the connection request (established when the connection request was processed by the Fabric) and the priority of the preemption request as provided to the Fabric in Word 1, bits 31-25 of the preemption request SOFc1 header. If priority is used for preemption request resolution, a preemption request will be rejected with a "Preemption Request Rejected" reason code if the preemption request priority is equal to or less than the connection being preempted.

Once a preemption request results in a Dedicated Connection all Dedicated Connection and disconnection rules defined in FC-FS shall be followed.

Frame processing precedence

There are a number of different fields within a frame which affect the processing for establishing and removing Dedicated Connections. The general precedence hierarchy follows:

- a) Frame delimiters (SOFc1, EOFdt),
- b) R_CTL
 - Data frame, or
 - Link_Control frame.
- c) F_CTL
 - Unidirectional Connection, when meaningful (bit 8),
 - End_Sequence (bit 19),
 - other bits when meaningful (for example, if bit 19=1, check End_Connection, Chained_Sequence, and so forth) as in Table 23 and Table 24.

22.2 Applicability

Connection management applies to Class 1 Service. A N_Port supporting Class 1 Service may also support Class 2 and Class 3. Depending on the options supported by the Fabric, multiple class support by a N_Port may be complex. Because Class 1 involves Dedicated Connections, managing Class 1 usually overrides Class 2 or 3 management. FC_FS specifies the allowable responses on a Class by Class basis.

NOTE - A N_Port engaged in Class 2 communication with another N_Port may experience Sequence timeouts if the other N_Port supports Class 1 and Class 2 without functioning in Intermix mode and a Class 1 Connection is created to the other N_Port.

22.3 Topology models

A N_Port may be attached directly to another N_Port through a point to point connection or through a Fabric. The topology may be determined using the explicit Login procedure. Topology may also be determined by a method not defined by FC_FS.

22.3.1 Fabric model

The N_Port behavior described in this clause is based on a Fabric model in which a F_Port acts as the control point for establishing and removing Class 1 connections on behalf of the locally attached N_Port. The N_Port relies on specific behavioral characteristics in order to base its operation.

The following terminology is used in the discussion of Connection management. N_Port (A), (B), or (X) describe three separate N_Port Identifiers where $A \neq B \neq X$. The side of the F_Port directly attached to the N_Port side is termed its "Link" side, whereas the side of the F_Port attached internally to the Fabric is termed its "internal" side.

The following F_Port characteristics are required behavior:

- a) When a F_Port is not connected, it may receive a connect-request and begin processing that request. The process of acting on that request is termed accepting the connect-request.
- b) After a F_Port has accepted a connect-request from the Link, it reserves Fabric resources as it attempts to establish the requested Dedicated Connection.
 - the F_Port is Busy to other connect-requests from its internal side as destination N_Port.
 - the F_Port returns an F_BSY with EOF_{dt} to the Link if a busy condition is encountered. (If a Fabric supports stacked connect-requests, the period of time before issuing an F_BSY may be extended.)
 - the F_Port returns an F_RJT with EOF_{dt} to the Link if a reject condition is satisfied.
- c) After a F_Port has accepted a connect-request from the internal side:
 - it passes the connect-request to the Link as the destination N_Port.
 - it monitors its Link side for a proper confirmation response frame expected in response to the delivered connect-request.
 - it discards connect-request frames (SOF_{c1}) received from its Link side. (If the Fabric supports stacked connect-requests, the connect-request received from its Link side shall be stacked for establishing a Dedicated Connection at a later time.)
- d) If a F_Port encounters a collision case wherein connect-requests from both the internal side and the Link side arrive simultaneously, the F_Port accepts the connect-request from the internal side and proceeds as in step c above.
- e) If a failure is detected within a Fabric such that a Dedicated Connection can no longer be used, the Fabric may notify each of the F_Ports attached to the N_Ports involved in an established or pending Dedicated Connection and each of the F_Ports may then transmit the Link Reset Primitive Sequence to its locally attached N_Port (i.e., initiate Link Reset Protocol).
- f) See 10.4 for information on the effect of Primitive Sequences on F_Ports.

22.3.2 Point to Point model

A point to point topology is indicated during the Login procedure. Two N_Ports arranged in a point to point connection may choose to:

- establish one Dedicated Connection for the duration of an operating period, or
- establish and remove Dedicated Connections dynamically, as the need to communicate arises.

22.4 Connect / disconnect rules

22.4.1 Connect-request rules

The following sections specify the rules governing the behavior of the source and destination of the connect-request.

22.4.1.1 Source of connect-request

The following rules specify the connect-request procedure as the source (A) of the connect-request:

- a) A N_Port (A) shall initiate a connect-request using a Data (Device_Data or Link_Data) frame with an SOFc1 delimiter directed to destination N_Port (B). The connect-request frame is formed as follows:
 - an SOFc1 delimiter
 - a Data (Device_Data or Link_Data) frame
 - an S_ID of (A) and a D_ID of (B)
 - the E_C bit (F_CTL bit 18) shall be set = 0
 - the Unidirectional bit (F_CTL bit 8) shall be set = 0 for a bi-directional Connection, and = 1 for a unidirectional Connection
 - an EOFn ending delimiter
- b) The Data Field of the connect-request shall be limited to the smaller of
 - the maximum buffer-to-buffer Receive_Data_Field size specified by the Fabric, if present, or
 - the maximum Receive_Data_Field size specified by the destination N_Port.
- c) After N_Port (A) transmits the connect-request frame, N_Port (A) shall wait for a response frame before transmitting another frame for this Sequence. Additional Sequences for this Connection shall not be initiated until the Dedicated Connection is established.
- d) A Dedicated Connection is established when the connect-request frame has been responded to by an ACK frame. A proper response frame consists of:
 - an ACK_1 or ACK_0 frame with
 - an SOFn1 delimiter
 - an S_ID of (B), and a D_ID of (A)
 - an EOFn, or EOFt delimiter
- e) An alternate response frame is also possible from the destination N_Port:
 - a P_BSY or P_RJT frame with
 - an SOFn1 delimiter,
 - an S_ID of (B), and a D_ID of (A), and
 - an EOFdt delimiter.

- f) An alternate response frame is also possible from the Fabric:
 - an F_BSY or F_RJT frame with
 - an SOF_{n1} delimiter,
 - an S_ID of (B), and a D_ID of (A), and
 - an EOF_{dt} ending delimiter.
- g) After a Dedicated Connection is established, N_Port (A) shall be the Connection Initiator and N_Port (B) shall be the Connection Recipient.
- h) After a Dedicated Connection is established (i.e., the ACK to the connect-request has been received), the Connection Initiator, N_Port (A), may continue transmitting its initial Sequence and initiate other Sequences with SOF_{i1} up to N_Port (B)'s ability to support Concurrent Sequences.

22.4.1.2 Destination of connect-request

The following rules specify the connect-request procedure at the destination (B) of the connect-request:

- a) If a Data frame started by SOF_{c1} is received when N_Port (B) is not connected and N_Port (B) is busy, N_Port (B) responds with P_BSY with an EOF_{dt} delimiter as specified in 22.4.1.1 item number 5.
- b) If a Data frame started by SOF_{c1} is received when N_Port (B) is not connected and N_Port (B) rejects the connect-request, N_Port (B) responds with P_RJT with an EOF_{dt} delimiter as specified in 22.4.1.1 item number 5.
- c) If a Data frame started by SOF_{c1} is received when N_Port (B) is not connected and not busy, N_Port (B) responds with the proper response frame. A proper response frame is defined in 22.4.1.1 item number 4.

A Dedicated Connection is established with N_Port (A). N_Port (B) shall be the Connection Recipient and N_Port (A) shall be the Connection Initiator.

- d) With a Fabric present, if N_Port (B) receives a connect-request frame from N_Port (X) after a connect-request has been transmitted by N_Port (B), N_Port (B) requeues its own request for transmission at a later time and responds with a proper response frame to N_Port (X).

NOTE - N_Port (B) requeues its original connect-request because the Fabric has discarded it. N_Port (B) needs to adjust its end-to-end Credit_CNT to account for the discarded connect-request.

If stacked connect-requests are being employed, the connect-request shall not be requeued by N_Port (B).

A Dedicated Connection is established with N_Port (X) with N_Port (B) as Connection Recipient.

- e) Without a Fabric present, if N_Port (B) accepts a connect-request frame from N_Port (A) after a connect-request has been transmitted by N_Port (B), N_Port (B) shall respond as follows:
 - 1) if $A > B$, in value,
 - N_Port (B) requeues its own request for transmission at a later time,
 - responds to (A) with a proper response frame,
 - a Dedicated Connection is established with N_Port (A) with N_Port (B) as Connection Recipient, and
 - N_Port (B) may reinitiate its connect-request Sequence using SOF_{i1}.
- f) if $A < B$, in value,
 - N_Port (B) discards connect-request from (A), and

- waits for a proper response frame.

g) After a Dedicated Connection is established (i.e., the ACK to the connect-request is transmitted), N_Port (B) may begin initiating Sequences with SOFi1 up to the destination N_Port's ability to support Concurrent Sequences when the Connection is bi-directional.

22.4.2 Connection Rules

- If a bi-directional Connection is established (F_CTL bit 8 = 0 on connect-request), it shall remain bi-directional for the life of the Connection.
- If a unidirectional Connection is established (F_CTL bit 8 = 1), it shall remain unidirectional until the Connection Initiator sets F_CTL bit 8 = 0, if ever, on the first or last frame of a Sequence (see 12.6) subsequent to the connect-request frame.
- If a unidirectional Connection is made bi-directional, it shall remain a bi-directional Connection for the life of the Connection.
- The Connection Recipient may request that a unidirectional Connection be made bi-directional by setting F_CTL bit 8 = 0 on an ACK in response to a Data frame. Once F_CTL bit 8 is set to 0 on an ACK, it shall remain set to 0 for the life of the Connection.

22.4.3 Remove Connection rules

- An Active Sequence is complete when the corresponding ACK response to the last Data frame has been transmitted by the Sequence Recipient from the Recipient's perspective and has been received by the Sequence Initiator from the Initiator's perspective.
- If a N_Port detects an abnormal condition which requires immediate removal of an existing Connection, the N_Port shall use the Remove Connection (RMC) Basic Link Service frame with the appropriate F_CTL bit settings which includes setting the E_C bit = 1 in order to remove the Dedicated Connection. All Open and Active Class 1 Sequences are abnormally terminated and left in an indeterminate state relative to the Upper Level Protocol. RMC shall not be used in place of Link Reset in protocols, which require Link Reset.
- A N_Port shall transmit the E_C bit in F_CTL set to one on the last Data frame of a Sequence to indicate:
 - it is ready to end the Connection,
 - it shall not initiate any new Sequences, and
 - it requests the other N_Port to complete its Active Sequences and not initiate any new Sequences.
- If a N_Port has transmitted the E_C bit in F_CTL set to one and it receives a Data frame initiating a new Sequence, it shall respond as though the Sequence had been initiated before the E_C bit had been transmitted as one.
- If a N_Port has transmitted the E_C bit in F_CTL set to one and it receives the last Data frame of a Sequence with the Chained_Sequence (C_S) bit in F_CTL set to one, it shall respond as though it had not previously transmitted the E_C bit (it shall set the E_C bit on a future Sequence).
- If either the Connection Initiator or Connection Recipient has completed its last Active Sequence of the existing Connection and it receives a Data frame with E_C set to one, it shall transmit the corresponding ACK with an EOFdt delimiter.
- If either the Connection Initiator or Connection Recipient receives a Data frame with the E_C bit set to one and it has not completed all of its Active Sequences, it
 - does not initiate any new Sequences (unless C_S bit is received),
 - completes Active Sequences,

- transmits the E_C bit set to one on the last Data frame of its last Active Sequence for the Connection.
- If a N_Port encounters a collision case wherein a Data frame has been transmitted with E_C set to one and a Data frame is received with E_C set to one before receiving its ACK,
 - the Connection Recipient shall respond with an ACK with an EOF_t delimiter, whereas
 - the Connection Initiator shall withhold transmitting ACK until after its Sequence is complete.
 - the Connection Initiator shall transmit the ACK with the EOF_{dt} delimiter.

22.5 Establishing a Connection

A Dedicated Connection is established with a N_Port as the source of a connect-request (Connection Initiator) or as the destination N_Port (Connection Recipient) of a connect-request from another Class 1 N_Port.

22.5.1 Connection Initiator

When FC-2 receives a request from an FC-4 or upper level to initiate a Class 1 Sequence when a Dedicated Connection does not exist, the N_Port shall also establish a Class 1 Connection with the destination N_Port as part of the Sequence initiation.

The N_Port (A) initiates the connect-request using a Data (Device_Data or Link_Data) frame with a SOF_{c1} delimiter. The Data Field size is limited to the smaller of:

- the maximum buffer-to-buffer Receive_Data_Field size specified by the Fabric, if present, or
- the maximum Receive_Data_Field size specified by the destination N_Port.

After the N_Port transmits the connect-request frame, no additional frames shall be transmitted for any Sequence for the pending Connection until a response frame has been received. The N_Port receives an R_RDY Primitive in response to the connect-request to indicate successful delivery to the F_Port or N_Port and that a buffer is available for a connectionless frame. If a N_Port is not operating in Intermix mode, the N_Port shall not transmit Class 2 or 3 frames until the pending Dedicated Connection is removed. A Dedicated Connection is not established until a proper ACK frame is received from the destination N_Port. A proper ACK frame is defined in 22.4.1.1 item number d.

Table 217 defines Event 1 as the connect-request and events 2 through 9 define the possible responses.

Table 217 - Responses to connect-request (SOF _{c1})						
Event	SOF	D_ID	S_ID	Frame	EOF	N_Port Action
1.	SOF _{c1}	B	A	Data Frame	EOF _n	Transmit connect-request Wait for confirmation frame
2.	SOF _{n1}	A	B	F_BSY	EOF _{dt}	Connection failed, Busy in Fabric
3.	SOF _{n1}	A	B	P_BSY	EOF _{dt}	Connection failed, Busy in N_Port
4.	SOF _{n1}	A	B	F_RJT	EOF _{dt}	Connection failed, Fabric Reject
5.	SOF _{n1}	A	B	P_RJT	EOF _{dt}	Connection failed, Port Reject
6.	SOF _{n1}	A	B	ACK_1 or ACK_0	EOF _n or	Dedicated Connection established

Table 217 - Responses to connect-request (SOFc1)						
Event	SOF	D_ID	S_ID	Frame	EOF	N_Port Action
					EOFTd	Continue transmitting Sequence Sequence ended, Connection established
7.	SOFc1	A	B	Data frame	EOFn	N_Port A responds as follows PTP If A > B in value, discard B's frame and wait for ACK response. Dedicated connection established with B.
						PTP: If A < B in value, respond with SOFn1 on ACK Dedicated Connection established with B. retransmit request associated with event 1 with SOFi1
8.	SOFc1	A	X	Data frame	EOFn	Fabric is present. Requeue request associated with event 1 (unless stacked connect-requests used) Respond with SOFn1 on ACK_0 or ACK_1 Dedicated Connection established with X.
9.						Timeout, no response frame. Perform Link Reset Protocol. (see 10.6.5)

Event 1

A connect-request is transmitted by N_Port (A) with an SOFc1 delimiter with a destination of N_Port (B).

Event 2

An F_BSY indicates that the Fabric is unable to access the destination N_Port due to a busy condition internal to the Fabric. Try again later.

Event 3

A P_BSY indicates that the destination N_Port link facility is temporarily occupied with other activity and unable to accept the connect-request. Try again later.

Event 4

An F_RJT indicates that the Fabric is unable to establish the Dedicated Connection. The reason code specifies the cause.

Event 5

A P_RJT indicates that the destination N_Port is unable to establish the Dedicated Connection. The reason code specifies the cause.

Event 6

A Dedicated Connection has been established. N_Port (A) is Connected to N_Port (B).

a) N_Port (A) is the Connection Initiator and N_Port (B) is the Connection Recipient.

- b) N_Port (A) may continue transmitting the Sequence initiated (EOF_n), or the Sequence that initiated the Connection may be complete (EOF_t).
- c) N_Port (A) may initiate other Sequences with the same destination N_Port (B) up to the maximum number of Sequences defined by the Service Parameters obtained from (B) during Login.
- d) The connected N_Port (B) may initiate Sequences using SOF_{i1} to start each Sequence when the Connection is bi-directional. The number of Active Sequences is limited by the Service Parameters provided by N_Port (A) during Login.

Event 7

In the case of a point to point (PTP) topology, if (A) is greater than (>) (B) in absolute value, then N_Port (A) discards the connect-request and waits for a response (N_Port (B) requeues request with SOF_{i1}).

In the case of a PTP topology where (A) < (B), N_Port (A) terminates its own previous connect-request with the intent of retransmission at a later time (i.e., requeues Event 1 with SOF_{i1}). N_Port (A) responds as the destination of the connect-request with an appropriate ACK frame and becomes the Connection Recipient. N_Port (B) is the Connection Initiator.

Event 8

In the case of a Fabric present, N_Port (A) terminates its own previous connect-request with the intent of retransmission at a later time (i.e., requeues Event 1 with SOF_{i1} or SOF_{c1}, as appropriate). N_Port (A) responds as the destination of the connect-request with an appropriate ACK frame and becomes the Connection Recipient. N_Port (X) is the Connection Initiator. If stacked connect-requests are functional, then it is unnecessary to requeue its own previous connect-request.

Event 9

If a frame response is not received within the timeout period (E_D_TOV), a Link timeout shall be detected and the Link Reset Protocol shall be performed (see 10.6.5).

See 22.4.1 for the rules that a source N_Port of a connect-request shall follow.

22.5.2 Stacked connect-requests

Stacked connect-requests is a feature which may be provided by a Fabric (see Clause 30). A N_Port during Login with the Fabric determines support for Stacked connect-requests. Intermix shall also be functional in order to use this feature. If Stacked connect-requests are functional, a N_Port may transmit one or more connect-requests (SOF_{c1}) without regard as to whether a Connection is pending, established, or complete.

The Fabric may process multiple connect-requests in any order and shall provide the status of the Stacked request via the Read Connection Status Extended Link Service command. Stacked connect-requests allow the Fabric to work on establishing a Connection while a N_Port is busy servicing another Connection and may enhance system performance. A N_Port uses the CR_TOV timeout period after a connect-request has been transmitted (part of Sequence timeout) whether the connect-request is a normal request or a Stacked request. That is, an ACK response shall be received within an CR_TOV timeout period or an F_BSY shall be returned from the Fabric to the N_Port. If either condition is not met within CR_TOV, the N_Port detects a Sequence timeout and Connection Recovery (see 22.8) is performed. A Fabric that supports stacked connect-requests shall deal with any associated race conditions which occur during the establishment and removal of connections.

Due to the timing relationships involved, there are two methods of implementing stacked connect-requests by a Fabric:

1. transparent mode, or
2. lock-down mode.

In transparent mode, when the SOF_{c1} Data frame is delivered to the destination N_Port, the return path of the bi-directional circuit is established in the same manner as Exclusive Dedicated Connections. This means that the destination N_Port of the SOF_{c1} is able to transmit Data frames immediately following transmission of the ACK frame in response to the SOF_{c1} frame.

In lock-down mode, when the SOF_{c1} Data frame is delivered to the destination N_Port, the return path of the bi-directional circuit is not necessarily established to the source N_Port of the SOF_{c1}. Therefore, the SOF_{c1} Data frame shall also set F_CTL bit 8 = 1 (Unidirectional Transmit) in order to inhibit the destination N_Port of the SOF_{c1} from sending any Data frames after the ACK frame is transmitted in response to the connect-request.

The ability of the Fabric to support Stacked Connect_requests is indicated by Word 0, bits 29 and 28 of the F_Port service parameters (see Table 208).

22.5.3 Unidirectional Dedicated Connection

When a N_Port transmits a connect-request, it may set F_CTL bit 8 = 1 to indicate that a unidirectional Dedicated Connection is being established. Unidirectional means that only the N_Port transmitting the connect-request (Connection Initiator) shall be able to transmit Data frames. The destination of the connect-request (Connection Recipient) shall only transmit ACK or Link_Response frames (P_BSY, P_RJT).

F_CTL bit 8 is meaningful on the first Data frame of a Sequence and the last Data frame of a Sequence. The Connection Initiator may reset F_CTL bit 8 = 0 at the end of the first Sequence (if multi-frame), or at the start or end of any subsequent Sequence. If F_CTL bit 8 is reset to = 0 on a meaningful Data frame (first or last) of a subsequent Sequence, the Connection becomes bi-directional for the remainder of the Connection.

If the Connection Recipient of a unidirectional Connection desires that the Connection become bi-directional, it may request that change by setting F_CTL bit 8 to 0 in the ACK response to any Data frame in a Sequence. Once the Connection Recipient has transmitted F_CTL bit 8 = 0 on an ACK, it shall continue to do so for the remainder of the Connection (if set to 1, it shall be ignored). The request for a bi-directional Connection may not be honored by the Connection Initiator.

The unidirectional transmit bit (F_CTL bit 8) has three primary uses:

- a) lock-down stacked connect-requests,
- b) Connection Initiator with temporary receive buffer allocation problems (i.e., is unable to receive as much data as the end to-end Credit promised at Login due to buffering for transmit), or
- c) an implementation that may only receive or transmit Data frames at any instant in time.

22.5.4 Destination of connect-request

When N_Port(B) is not connected, but is available, and it receives a connect-request as the destination N_Port, N_Port (B) transmits the appropriate ACK frame (ACK_1 or ACK_0) to N_Port (A) which is requesting the connection. After the ACK frame has been transmitted with SOF_{n1}, a Dedicated Connection is established with N_Port (A) as the Connection Initiator and N_Port (B) as the Connection Recipient. After a Connection has been established, N_Port (B) may initiate Sequences with N_Port (A) using an SOF_{i1} delimiter.

If N_Port (B) is not connected, but is busy, and it receives a connect-request as the destination N_Port from N_Port (A), it responds with P_BSY with an EOF_{dt} delimiter.

See 22.4.1 for the rules that a destination N_Port of a connect-request shall follow.

22.6 Connected

When a N_Port is in a Dedicated Connection, it may receive Sequences as the Sequence Recipient as well as initiate Sequences as the Sequence Initiator.

When a N_Port has acted as a Connection Initiator and is in a Unidirectional Connection, it may only initiate Sequences and shall not receive Sequences. When a N_Port has been the Connection Recipient and is in a Unidirectional Connection, it may only receive Sequences and shall not initiate Sequences.

22.7 Removing a Connection

Class 1 Connections may be removed in four ways:

- using the remove connection procedure which uses the E_C bit in F_CTL for negotiation,
- the Remove Connection command, which forces the recipient N_Port to transmit EOFdt,
- the Link Reset Protocol, or
- transmission or reception of any Primitive Sequence.

The first two methods use the EOFdt delimiter to remove the Connection. The third method uses the Link Reset Primitive Sequence to remove the Connection.

22.7.1 When to remove a Connection

FC_FS does not specify the method to be employed in determining when to end a connection. Normally, a N_Port chooses to remove a Dedicated Connection when it has no Sequences to transmit to the Connected N_Port and it has a request to transmit Sequences to a different N_Port. In addition, F_CTL bits offer one alternative to suggest that removal might be appropriate (Continue Sequence condition bits) and one alternative that restricts a N_Port from removing a Connection while using the E_C bit in the normal remove connection procedure (Chained_Sequence).

NOTE - In order to maximize system utilization, a N_Port should remove a Dedicated Connection when it has no Data to transmit in order to be available for Connection to another N_Port.

Continue Sequence condition bits

Two information bits are available in F_CTL to indicate when the next Sequence of an Exchange is estimated to be transmitted: immediately (0 1), soon (1 0), or delayed (1 1). The indication of delayed transmission may be used as one condition to assist in determining the appropriate time to remove a Dedicated Connection.

Chained_Sequence (C_S)

The C_S bit in F_CTL shall be used in the last Data frame of a Sequence in conjunction with Sequence Initiative transfer to indicate that the transmitter of the C_S = 1 requires a reply Sequence before the existing Dedicated Connection is removed.

22.7.2 End_Connection bit

An E_C bit in F_CTL is used during the remove connection procedure. By monitoring both transmission of the E_C bit as well as reception of the E_C bit, each N_Port is able to determine the appropriate circumstances to transmit an EOFdt in order to remove the connection. The E_C bit in F_CTL shall be set to zero on a connect-request frame (SOFc1) in order to avoid ambiguous error scenarios where the ACK (EOFdt) is not properly returned to the Connection Initiator.

E_C shall be transmitted as zero in a frame to indicate:

- This N_Port wishes to maintain the existing Dedicated Connection.
- This N_Port may transmit another Sequence within this Connection.
- This N_Port may wait for a reply Sequence within this Connection.

E_C shall be transmitted as one on a Data frame to indicate:

- This N_Port is ready to remove the existing Dedicated Connection.
- This N_Port has completed all Active Sequences and agrees not to initiate another Sequence unless it receives a Chained_Sequence indication on the last Data frame of a Sequence being normally terminated.
- This N_Port is requesting the destination N_Port to complete Active Sequences in progress and not to initiate any new Sequences.

E_C shall be transmitted set to one on the last Data frame of the last Active Sequence for this Connection to indicate that this N_Port requests the receiving N_Port to transmit EOF_{dt} on the ACK response frame if the receiving N_Port has completed all Active Sequences, otherwise, transmit an ACK frame with EOF_t. When the N_Port that received the E_C bit set to one is completing its last Sequence, it shall transmit the E_C bit set to one on the last Data frame of its last Sequence.

22.7.3 EOF_{dt} transmission

EOF_{dt} is transmitted by either the Connection Initiator or the Connection Recipient on an ACK frame corresponding to the last Data frame of a Sequence with the E_C bit set to one if the N_Port receiving the Data frame has completed its last Active Sequence of the existing Connection.

If both N_Ports have transmitted their last Data frame of a Sequence with E_C set to one but have not received the ACK frame to complete the Sequence, the Connection Initiator delays transmitting ACK until it has received the ACK for its last Sequence. After the Connection Initiator has completed its last Sequence, it transmits the final ACK to the Connection Recipient with the EOF_{dt} delimiter to remove the Connection.

The EOF_{dt} is also used on the ACK responding to the RMC Basic Link Service command in order to immediately remove an existing Connection.

22.8 Connection Recovery

In case of link errors, the state of the existing Dedicated Connection may not be known with certainty. When the state of the Dedicated Connection is unknown, the Link Reset Protocol shall be used to remove the Connection and establish a known state (see 10.6.5). Errors within a specific Sequence are processed according to the rules for handling Sequence errors.

For example, the following events provide instances when a N_Port is unable to determine the state of an existing Dedicated Connection and should perform the Link Reset Protocol. N_Port (A) believes that a Dedicated Connection exists with N_Port (B):

- N_Port (A) receives an SOFc1 from N_Port (C),
- N_Port (A) receives an SOFi1 from N_Port (C), or
- N_Port (A) receives an SOFn1 from N_Port (C).

22.8.1 Link timeout

When the last Active Sequence during a Class 1 Dedicated Connection has detected a Sequence timeout, a Link Timeout is detected. For more discussion on Link Timeout see 23.2.3. The Link Reset Protocol is performed as described in 10.6.5.

22.8.2 Corrupted connect-request

If a N_Port is not engaged in a Class 1 Dedicated Connection, and it receives a frame started by SOFc1 and the frame is detected as invalid, the N_Port discards the frame and performs the Link Reset Protocol.

22.9 Connection Preemption

The procedure for preempting a Dedicated Connection is specified in this subclause.

22.9.1 Applicability

This preemption process applies only to Class 1 and Class 6 Service. Other preemption processes may be defined that apply to other connection oriented classes of service.

22.9.2 Topology Model

A N_Port that uses preemption shall have the ability to be connected to another N_Port through a Fabric.

22.9.3 Rules for Preemption

The following subsections specify the rules governing the behavior of the preemptor (P), preempted source (PS) or destination (PD), and the preemption destination (PnD).

22.9.3.1 Preemptor (P)

The following rules specify the procedure for preempting a Dedicated Connection with respect to the Preemptor.

- a) A N_Port shall initiate a preemption request using a Data (Device_Data or Link_Data) frame directed to the desired destination N_Port. The SOFc1 preemption request frame shall be formed as follows:
 - An SOFc1 delimiter
 - a Data (Device_Data or Link_Data) frame
 - an S_ID of the P N_Port and a D_ID of the PnD N_Port(s)
 - the E_C bit (F_CTL bit 18) shall be set = 0
 - the Priority (F_CTL bit 17) shall be set = 1, and the Priority field (Word 1, bits 31-25) shall contain an appropriate priority value.
 - the Unidirectional bit (F_CTL bit 8) shall be set = 0 for bi-directional Connection, and = 1 for a unidirectional Connection
 - the preemption flag bit (Word 1, bit 31) shall be set = 1.
 - an EOFn or EOFt ending delimiter
- b) The Data Field of the SOFc1 preemption request frame shall be limited to the smaller of
 - the maximum buffer-to-buffer Receive_Data_Field size specified by the Fabric, or
 - the maximum Receive_Data_Field size specified by the destination N_Port.

- c) After the P N_Port transmits the SOFc1 frame it shall wait for the response frame before transmitting another frame for this sequence. Additional sequences for this connection shall not be initiated until the Dedicated Connection is established.
- d) A Dedicated Connection is established when the SOFc1 frame has been responded to by an ACK (ACK_1 or ACK_0) frame. A proper response frame consists of:
 - an ACK_1 or ACK_0 frame with:
 - an SOFn1 delimiter, and
 - an S_ID of the PnD and a D_ID of the P, and
 - an EOFn, or EOFt delimiter
- e) An alternate response frame is also possible from the PnD N_Port:
 - a P_BSY or P_RJT frame with:
 - an SOFn1 delimiter, and
 - an S_ID of the PnD and a D_ID of the P, and
 - an EOFdt delimiter
- f) An alternate response frame is also possible from the Fabric F_Port:
 - a F_BSY or F_RJT frame with:
 - an SOFn1 delimiter, and
 - an S_ID of the PnD and a D_ID of the P, and
 - an EOFdt delimiter
- g) After a Dedicated Connection is established, the P N_Port shall be the Connection Initiator and the PnD(s) shall be the Connection Recipient(s).
- h) After a Dedicated Connection is established (i.e., the ACK to the SOFc1 preemption request has been received), the Connection Initiator may continue transmitting its initial Sequence and initiate other Sequences with SOFi1 up to the Connection Recipients ability to support Concurrent Sequences.

22.9.3.2 Preempted Source (PS)

The following rules specify the procedure for a preempted source. The PS is the connection initiator N_Port for a Class 1 or Class 6 connection.

If a basic link service frame consisting of a PRMT command with:

- an SOFc1 delimiter, and
- an S_ID of the P and a D_ID of the PS, and
- an EOFdt delimiter

is received by a connection initiator N_Port, that N_Port's Dedicated Connection has been removed. This PS N_Port should notify its host that its Dedicated Connection has been preempted.

22.9.3.3 Preempted Destination(s)(PD)

The following rules specify the procedure for a preempted destination. The PD is the connection recipient N_Port for a Class 1 or Class 6 connection.

If a basic link service frame consisting of a PRMT command with:

- an SOFn1 delimiter, and
- an S_ID of the P and a D_ID of the PD, and

- an EOFdt delimiter

is received by a connection recipient N_Port, that N_Ports Dedicated Connection has been removed. This PD N_Port should notify its host that its Dedicated Connection has been preempted.

22.9.3.4 Preemption Destination(s) (PnD)

The following rules specify the procedure for a preemption destination (PnD). A PnD is any N_Port, connected or not, that is addressed by the D_ID in a SOFc1 preemption request frame.

- a) If a N_Port not connected receives an SOFc1 preemption request frame, and this N_Port is busy, the N_Port shall respond with P_BSY with an EOFdt delimiter as specified in 22.4.1.1 item 5.
- b) If a N_Port not connected receives an SOFc1 preemption request frame, and this N_Port rejects the preemption request, the N_Port shall respond with a P_RJT with an EOFdt delimiter as specified in 22.4.1.1 item 5.
- c) If an SOFc1 preemption request frame is received by a N_Port previously connected (i.e., connected prior to being preempted) or not connected and not busy, this N_Port shall respond with the proper response frame. A proper response frame shall be an ACK_1 or ACK_0 frame with:
 - an SOFn1 delimiter, and
 - an S_ID of the PnD N_Port and a D_ID of the P, and
 - an EOFn, or EOFt delimiter

A dedicated connection is established with the PS N_Port. The PnD N_Port(s) shall be the Connection Recipient(s) and the PS N_Port shall be the Connection Initiator.

- d) If a PnD N_Port receives a preemption request frame after it has transmitted a connection-request frame it should re-queue its own request for transmission at a later time and respond with the proper response frame to the PS N_Port.

NOTE - The preemption destination N_Port requeues its original connection-request because the Fabric has discarded it. The PnD N_Port needs to adjust its end-to-end Credit_CNT to account for the discarded connection-request.

If stacked connection-requests are being employed, the connection-request shall not be re-queued by the PnD N_Port.

22.9.4 Connection Rules

Once a connection is established as a result of a preemption request, all rules specified for normal Class 1 or Class 6 connections shall be followed.

22.9.5 Remove Connection Rules

All rules to remove Class 1 or Class 6 connections shall be followed.

22.10 Establishing a Connection Using Preemption

A Dedicated Connection is established between a preemption requesting (Connection Initiator) N_Port and preemption destination(s) (Connection Recipient(s)) N_Port(s) following the successful completion of the Preemption process specified in sub-clause 22.9.

22.10.1 Connection Initiator

When the FC-2 protocol layer receives a request from an FC-4 or upper level to initiate a Class 1 or Class 6 sequence using preemption, the N_Port shall attempt to establish a Class 1 or Class 6 connection with the destination N_Port(s) using an SOFc1 preemption request frame (preemption flag set = 1) as part of a Sequence initiation.

The N_Port initiates a preemption request using the rules defined in sub-clause 22.9.

Table 218 defines event 1 as the preemption request and events 2 through 8 as possible responses. Event 9 defines the response of the preempted source or destination.

Table 218 - Responses to Preemption Requests						
Event	SOF	D_ID	S_ID	Frame	EOF	N_Port Action
1)	SOFc1	B, ...	A	Data Frame	EOFn	<ul style="list-style-type: none"> - Transmit Preemption Request - Wait for confirmation frame
2)	SOFn1	A	B	F_RJT	EOFdt	<ul style="list-style-type: none"> - Preemption Failed. Fabric Rejected request.
3)	SOFn1	A	B	F_BSY	EOFdt	<ul style="list-style-type: none"> - Preemption Failed, Fabric Busy.
4)	SOFn1	A	B	P_BSY	EOFdt	<ul style="list-style-type: none"> - Busy N_Port, Connection Removed.
5)	SOFn1	A	B	P_RJT	EOFdt	<ul style="list-style-type: none"> - N_Port Rejected, Connection Removed.
6)	SOFn1	A	B	ACK_1/ ACK_0	EOFn/ EOFdt	<ul style="list-style-type: none"> - Dedicated connection Established - Continue Transmitting Sequence - Sequence/Connection Ended.
7)	SOFc1	A	Any	Connect-request Frame	EOFn	<ul style="list-style-type: none"> - Requeue Preemption Request Associated with Event 1. - Respond with SOFn1 on ACK_1, ACK_0, or P_BSY.
8)						<ul style="list-style-type: none"> - Time-out, no response frame - Perform Link Reset Protocol (see 10.6.5)
9)	SOFn1	PS or PD	A	PRMT	EOFdt	<ul style="list-style-type: none"> - Dedicated Connection Preempted - Notify Host, Preempted.

Event 1

A preemption Request is transmitted by N_Port (A) with an SOFc1 delimiter with a destination address of N_Port (B) and the preemption flag set = 1 (word 1, bit 31).

Event 2

An F_RJT indicates that the Fabric is unable to establish the Dedicated Connection or has rejected the request. A dedicated connection has not been established. The reason code specifies the cause.

Event 3

An F_BSY indicates that the Fabric is busy and was unable to service the preemption request. A Dedicated connection has not been established.

Event 4

A P_BSY indicates that the destination N_Port link facility is temporarily occupied with other activities and is unable to accept the preemption request. Try again later.

Event 5

A P_RJT indicates that the destination N_Port is unable to establish a Dedicated Connection. The reason code specifies the cause.

Event 6

An ACK_1 or ACK_0 indicates that the preemption request has been serviced and a Dedicated connection has been established. N_Port (A) is connected to N_Port (B).

- a) N_Port (A) is the Connection Initiator and N_Port (B) is the Connection Recipient.
- b) N_Port (A) may continue transmitting the Sequence initiated (EOFn), or the Sequence that initiated the Connection may be complete (EOft).
- c) N_Port (A) may initiate other Sequences with the same destination N_Port (B) up to the maximum number of Sequences defined by the Service Parameters obtained from (B) during Login.
- d) The connected N_Port (B) may initiate Sequences using SOFi1 to start each Sequence when the Connection is bi-directional. The number of active Sequences is limited by the Service parameters provided by N_Port (A) during Login.

Event 7

N_Port (A) shall requeue the SOFc1 sent to the Fabric in Event 1 and may respond to the SOFc1 received with an ACK_1 or ACK_0, or P_BSY. Once this connection is completed N_Port (A) may re-send the SOFc1 preemption request frame to the Fabric.

Event 8

If a frame response is not received within the time-out period (E_D_TOV), a link time-out shall be detected and the Link Reset Protocol shall be performed (see 10.6.5)

See 22.4.1 for the rules that a source N_Port of a connect-request shall follow.

Event 9

When a connected source (i.e., Connection Initiator) or destination (connection recipient), receives a PRMT, the dedicated connection has been preempted and no longer exists. The N_Port should notify its host that the Sequence(s) were abnormally terminated due to the connection being preempted.

22.10.2 Preemption Destination

If an SOFc1 preemption request frame is received by a N_Port previously connected (i.e., connected prior to being preempted), it shall transmit the appropriate ACK frame (ACK_1 or ACK_0) to the P N_Port. After the ACK frame has been transmitted with the SOFn1, a Dedicated Connection is established with the P N_Port as the Connection Initiator and this destination N_Port(s) as Connection Recipient(s). After a Connection has been established, this Destination N_Port may initiate Sequences with the preemptor N_Port using an SOFi1 delimiter, if the Connection is bi-directional (Note - Class 6 dedicated connections are uni-directional).

When an unconnected N_Port receives a preemption request it shall respond normally for a Class 1 connection request, as specified in 22.5.4.

23 Error detection/recovery

23.1 Introduction

Link integrity and Sequence integrity are the two fundamental levels of error detection in FC-FS. Link integrity focuses on the inherent quality of the received transmission signal. When the integrity of the link is in question, a hierarchy of Primitive Sequences is used to reestablish link integrity. When Primitive Sequence Protocols are performed, additional data recovery on a Sequence basis may be required.

A Sequence within an Exchange provides the basis for ensuring the integrity of the block of data transmitted and received. Exchange management ensures that Sequences are delivered in the manner specified by the Exchange Error Policy (see 23.6.1.1). Each frame within a Sequence is tracked on the basis of Exchange_ID, Sequence_ID, and a sequence count within the Sequence. Each frame is verified for validity during reception. Sequence retransmission may be used to recover from any frame errors within the Sequence and requires involvement, guidance, or authorization from an upper level.

Credit loss is an indirect result of frame loss or errors. Credit loss is discussed in regard to methods available to reclaim apparent lost Credit resulting from other errors. See clause 20 for a more complete discussion on flow control, buffer-to-buffer Credit, and end-to-end Credit.

23.2 Timeouts

23.2.1 Timeout periods

Three timeout values are specified in FC-FS. The actual value used for a timeout shall not be less than the specified value and shall not exceed the specified value by more than 20%.

23.2.1.1 R_T_TOV

The Receiver_Transmitter timeout value (R_T_TOV) is used by the receiver logic to detect Loss of Synchronization (see 6.1.3). The value for R_T_TOV is 100 ms.

23.2.1.2 E_D_TOV

A short timeout value is known as the Error_Detect_Timeout Value (E_D_TOV). The E_D_TOV is used as the timeout value for detecting an error condition. The value of E_D_TOV represents a reasonable timeout value for detection of a response to a timed event. For example, during Data frame transmission it represents a timeout value for a Data frame to be delivered, the destination N_Port to transmit a Link_Response, and the Link_Response to be delivered to the Sequence Initiator. The E_D_TOV value selected should consider configuration and N_Port processing parameters. A default value is 2 seconds. However, a valid E_D_TOV value shall also adhere to the proper relationship to the R_A_TOV value. When a N_Port performs Fabric Login, the Common Service Parameters provided by the F_Port specify the proper value for E_D_TOV.

When a N_Port performs N_Port_Login in a point-to-point topology, the Common Service Parameters provided by each N_Port specify a value for E_D_TOV. If the two values differ, each N_Port shall use the longer time. A N_Port may determine another N_Port's value for E_D_TOV via the Read Timeout Value (RTV) Link Service command (see 15.5.13). Timeout values as specified in the Payload of the Accept are counts of either 1 ms or 1 ns increments, depending on the resolution specified. Therefore, a value of hex '00 00 00 0A' specifies a time period of either 10 ms or 10 ns.

There are three cases when E_D_TOV is used as an upper limit, that is, an action shall be performed in less than an E_D_TOV timeout period:

- transmission of consecutive Data frames within a single Sequence,
- retransmission of a Class 2 Data frame in response to an F_BSY or P_BSY, and
- transmission of ACK frames from the point in time that the event which prompted the acknowledgment action.

For all other cases, E_D_TOV shall expire before an action is taken. Two such examples include:

- Link timeout (see 23.2.3), and
- Sequence timeout (see 23.2.4).

23.2.1.3 R_A_TOV

A long timeout value is known as the Resource_Allocation_Timeout Value (R_A_TOV). The R_A_TOV is used as the timeout value for determining when to Reinstall a Recovery_Qualifier. The value of R_A_TOV represents E_D_TOV plus twice the maximum time that a frame may be delayed within a Fabric and still be delivered. The default value of R_A_TOV is 10 seconds.

When a N_Port performs Fabric Login, the Common Service Parameters provided by the F_Port specify the proper value for R_A_TOV. When a N_Port performs N_Port_Login in a point-to-point topology, the Common Service Parameters provided by each N_Port only specify a value for E_D_TOV. R_A_TOV shall be set to twice the E_D_TOV value in a point-to-point topology. A N_Port may determine another N_Port's value for R_A_TOV via the Read Timeout Value (RTV) Link Service command (see 15.5.13).

When R_A_TOV is used to determine when to reuse a N_Port resource such as a Recovery_Qualifier, the resource shall not be reused until R_A_TOV has expired for all frames previously transmitted which fall within the SEQ_CNT range of the Recovery_Qualifier. This may be accomplished by restarting the R_A_TOV timer as each Data frame of a Sequence is transmitted. Other techniques may also be employed in order to accomplish equivalent behavior.

From the Fabric viewpoint, the maximum time that a frame may be delayed within the Fabric and still be delivered is in the range of 1 to $2^{31} - 1$ ms. The Fabric shall ensure delivery within the maximum delivery time by requiring each Fabric Element to timeout frames stored in receive buffers within the Element. Individual Elements may use different timeout values, possibly one for each service class. The maximum Fabric delivery time is variable and is the cumulative timeout value for elements along the path or paths joining the source and destination N_Ports.

23.2.2 Link Failure timeouts

A Link Failure is detected under the following timeout conditions:

- Loss of Signal (see 6.1.1.2),
- Loss of Synchronization > timeout period (R_T_TOV) (see 6.1.1.2),
- Link Reset Protocol timeout (> R_T_TOV) (see 10.6.5).

23.2.3 Link timeout

In Class 1, a Link timeout error is detected during a Dedicated Connection if Sequence timeouts have been detected for all Active Sequences.

In Class 1 (SOF_{c1}), Class 2, or Class 3, a Link timeout error shall be detected if one or more R_RDY Primitive Signals are not received within the timeout period (E_D_TOV) after the buffer-to-buffer Credit_CNT has reached zero.

Recovery from Link timeout is accomplished by performing the Link Reset Protocol (see 10.6.5).

Link timeout values need to take Fabric characteristics into consideration. In Class 1, the concern is the time required by the Fabric to process a connect-request. In Class 2 and 3, the concern is the maximum time required for frame delivery by the worst case route with any associated delays within the Fabric.

23.2.4 Sequence timeout

The basic mechanism for detecting errors within a Sequence is the Sequence timeout. Other mechanisms that detect frame errors within a Sequence are performance enhancements in order to detect an error sooner than the timeout period. Since an Active Sequence utilizes N_Port resources, Sequence timeout is applicable to both discard and process error policies. The Sequence timeout mechanism varies by class.

23.2.4.1 Class 1 and 2

Both the Sequence Initiator and the Sequence Recipient use a timer facility with a timeout period (E_D_TOV) between expected events. The expected event for the Initiator to Data frame transmission is an ACK response (or alternate). The expected event for the Recipient is another Data frame for the same Sequence that is Active and incomplete. Other events such as Link Credit Reset and Abort Exchange shall also stop the Sequence timer. When a Sequence Recipient receives the last Data frame transmitted for the Sequence, it shall verify that all frames have been received before transmitting the final ACK (EOF_t or EOF_{dt}) for the Sequence.

If the timeout period (E_D_TOV) expires for an expected event before the Sequence is complete, a Sequence timeout is detected. Timeouts are detectable by both the Sequence Initiator and the Sequence Recipient. If a Sequence Initiator detects a Sequence timeout, it shall transmit the ABTS frame to perform the Abort Sequence Protocol. If a timeout is detected by the Sequence Initiator before the last Data frame is transmitted, ABTS notifies the Sequence Recipient that an error was detected by the Sequence Initiator (see 23.7.1.1). Detection of a Sequence timeout by the Sequence Initiator may also result in aborting the Exchange (see 15.2.1 and 15.5.2).

If a Sequence Recipient detects a Sequence timeout, it shall set the Abort Sequence Condition (F_CTL bits 5-4) in an ACK to an 0 1, or 1 1 value to indicate a missing frame error condition. The Sequence Recipient shall also post the detected condition in the Exchange Status Block associated with the Sequence. A Sequence timeout results in either aborting the Sequence (see 15.2.1) by the Sequence Initiator, abnormal termination of a Sequence (see 23.7.1) by the Sequence Recipient, or aborting the Exchange by either the Sequence Initiator or Sequence Recipient (see 15.5.2).

In Class 2, if a Sequence has been aborted and the Sequence Recipient supplies the Recovery_Qualifier (OX_ID, RX_ID, and a SEQ_CNT range, low and high SEQ_CNT values), the Sequence Initiator shall not transmit any Data frames within that range within a timeout period of R_A_TOV. Both the Sequence Initiator and Sequence Recipient discard frames within that range. After R_A_TOV has expired, the Sequence Initiator shall reinstate the Recovery_Qualifier using a Reinstate Recovery Qualifier Link Service request.

23.2.4.2 Class 3

In Class 3, the expected event for the Recipient is another Data frame for the same Sequence. Other events such as Abort Exchange shall also stop the Sequence timer. When a Sequence Recipient receives the last Data frame transmitted for the Sequence, it shall verify that all frames have been received.

If the Sequence Initiator periodically transmits an ABTS frame, the Sequence Recipient is able to inform the Sequence Initiator of the last deliverable Sequence. If the Sequence Initiator does not transmit ABTS frames, in Discard multiple Sequences Exchange Error Policy following an error in a Single Sequence, the Sequence Recipient shall continue to abnormally terminate subsequent Sequences and not deliver them to the FC-4 or upper level due to the requirement of in-order delivery of Sequences in the order transmitted. For bi-directional Exchanges, it is possible to infer proper Sequence delivery without the use of ABTS, if Sequence Initiative has been transferred and the reply Sequence for the same Exchange is received.

23.2.4.3 End-to-end Class 2 Credit loss

In Class 2 it is possible to reduce end-to-end Credit to zero as a result of one or more Sequence timeouts. The Link Credit Reset (LCR) Link_Control frame shall be transmitted by the N_Port, which has no end-to-end Credit (Sequence Initiator) to the destination N_Port (Sequence Recipient). The Sequence Initiator (S_ID of LCR frame) shall perform normal recovery for the Sequence that timed out (see 18.6.2).

The Fabric may return F_BSY if unable to deliver the LCR frame. A Reject may also be returned if either the S_ID or D_ID is invalid or an invalid delimiter is used.

When a N_Port receives LCR, it shall discard the Data in its buffers associated with the S_ID of the LCR frame and abnormally terminate the Sequences associated with any discarded frames.

23.2.5 OLS transmit timeout

When a Port is performing the Online to Offline Protocol (see 10.6.3), the Port shall timeout the transmission of OLS (5 ms) before reception of a Primitive Sequence in response. The timeout value of 5 ms allows a Port to enter the Offline State in the absence of an appropriate response from the attached Port.

23.2.6 Timer Enhancements

FC-FS defined two timers, E_D_TOV and R_A_TOV, that are used for normal frame flow. In general, E_D_TOV is used by both the Sequence Recipient (SR) and the Sequence Initiator (SI) to time FC2 inter-frame arrival of data and ACK frames respectively to determine if an error has occurred. R_A_TOV is used as the maximum transit time within a Fabric to guarantee that a lost frame will never emerge from the Fabric after an R_A_TOV time-out.

In general, it is desirable to keep these timers small to minimize the interval required to recognize time-out errors. However, under certain conditions, it is desirable that these time-out values be much larger to allow frames to remain outstanding for an extended time without reporting an error. Specifically, connect-request frames under Stacked Connect-request are required to remain active within the Fabric for an extended period of time to accomplish their intended functions. The time-out value requirements for these connect-request frames and all other frames are considerably different.

This section defines a new timer called Connection Request time-out Value CR_TOV) that is used by the Connection Initiator (CI), the Connection Recipient (CR), and the Fabric to control the connect-request process when Stacked Connect-request is invoked.

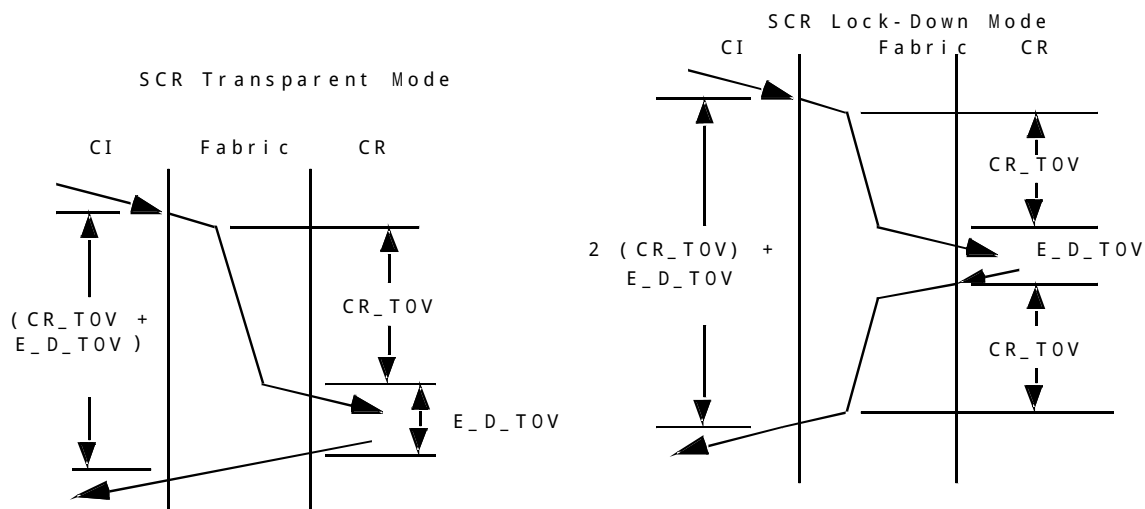


Figure 63 - Frame Flow Timers

23.2.6.1 Terms

The following new term is used within this section:

CR_TOV Connection Request Time-out value

23.2.6.2 Applicability

The CI, CR, and the Fabric to time connect-request frames that are transmitted under Stacked Connect-request shall use the CR_TOV timer.

23.2.6.3 Login

CR_TOV is specified by the F_Port during Fabric Login as part of F_Port Class Service Parameters (see 17.7.4).

23.2.6.4 Value

The minimum CR_TOV value shall be RA_TOV. The Fabric sets the maximum value of CR_TOV.

23.2.6.5 Stacked Connect-request

The CR_TOV timer shall be used by the Fabric to define the maximum time period that the Fabric shall hold a Stacked Connect-request frame.

23.2.6.6 Rules

Connection Initiator

- Following the transmission of a connect-request frame invoking Stacked Connect-request, if the CI receives an F_BSY in reply to that connect-request, the CI must retry that connect-request.
- The CI shall wait a minimum time period of $CR_TOV + E_D_TOV$ in Stacked Connect-request Transparent Mode following the transmission of a connect-request frame invoking

Stacked Connect-request before it can declare the connect-request frame to be lost and initiate error recovery.

- The CI shall wait a minimum time period of $2 (CR_TOV) + E_D_TOV$ in Stacked Connect-request Lock_Down Mode following the transmission of a connect-request frame invoking Stacked Connect-request before it can declare the connect-request frame to be lost and initiate error recovery.

Connection Recipient

- The CR shall wait a minimum time period of E_D_TOV following receipt of a connect-request frame that invoked Stacked Connect-request Transparent mode, before considering the ACK to the CR to have been lost.
- The CR shall wait a minimum time period of $CR_TOV + E_D_TOV$ following receipt of a connect-request frame which invoked Stacked Connect-request Lock Down mode, before considering the ACK to the CR to have been lost.

Fabric

- The Fabric shall start the CR_TOV timer on receipt of a connect-request frame, which has invoked Stacked Connect-request.
- If the CR_TOV timer expires on a connect-request frame which has invoked either Stacked Connect-request or Camp-On before the Fabric can forward it to the CR, the Fabric shall discard the connect-request frame and Fabric transmit an F_BSY frame to the CI.

23.2.7 Bit-Error-Rate Thresholding

The bit-error-rate thresholding process is designed to detect an increased error rate before performance degradation becomes serious. When the specified bit-error-rate threshold is reached, a Registered Link Incident Record (RLIR) ELS shall be generated as required by the RLIR ELS. (see 15.5.33.2)

The bit-error rate is measured during frame, idle, and primitive sequence reception. The bit-error rate is not calculated during times when Transmission-Word synchronization has been lost, when in the offline state, or when in a link-failure state.

23.2.7.1 Types of Link Errors Caused by Bit Errors

Bit errors are not detected directly, however they usually result in the recognition of invalid Transmission Words, primitive sequence protocol errors, CRC errors, or other events. Only recognition of invalid Transmission words are counted toward the bit-error-rate threshold.

23.2.7.2 Error Bursts

A single error may result in several related errors occurring closely together which in turn may result in multiple counts. A character might have a single bit error in it that causes a code-violation error. A disparity error might occur on a following character, caused by the same single error. To prevent multiple error counts from a single cause, the following concept of an error burst is introduced:

- An “error burst” is a time period 1.5 ± 0.5 seconds long during which one or more invalid Transmission Words are recognized. This time may be exceeded due to infrequent unusual conditions.
- Only one error in an error burst is counted toward the bit-error threshold.

23.2.7.3 Bit-Error-Rate-Thresholding Measurement

Measurement of bit-error-rate thresholding shall be accomplished by counting the number of error bursts that occur in a 5-minute period. When the count equals a specific number, the threshold is exceeded. This specific number is called the "threshold error count."

A bit-error-rate threshold shall be detected when 15 error bursts occur in a 5-minute period.

The required accuracy of the 5-minute period mentioned above is +/- 0.3 seconds.

The bit-error-counting process shall be restarted when Active state is entered and when a model-dependent amount of time has elapsed after a bit-error-rate threshold is detected. After a bit-error-rate threshold is detected, at least 15 additional error bursts shall occur before the next bit-error-rate threshold is detected. In addition, the bit-error-counting process may be restarted whenever the 5-minute time period has expired even though a bit-error-rate threshold is not reached. The bit-error-counting process may also be reset and restarted when an initialization procedure occurs.

NOTE – The delay in restarting bit-error counting after a threshold is detected is to prevent threshold-exceeded conditions from being reported at an excessive frequency.

23.3 Link error detection

Link errors are detected at three levels.

23.3.1 Link Failure

The first level of link error detection is at the receiver. Link Failure is detected under the following conditions:

- Link Failure conditions (see 23.2.2), or
- Reception of the NOS Primitive Sequence (see 10.4.2).

Recovery from Link Failure is accomplished by performing the Link Failure Protocol (see 10.6.4).

23.3.2 Code violations

Code violations are link errors that result from an invalid transmission code point or disparity error. If a code violation occurs during Idles, it is recorded in the Link Error Status Block by incrementing the Invalid Transmission Word count by one. If a code violation occurs during frame reception (see 11.8), the Link Error Status Block shall also be updated by incrementing the Invalid Transmission Word count by one and the frame identified as invalid. The Data Field of the invalid frame may be discarded or processed based on the Exchange Error Policy.

NOTE - When a code violation is detected, the actual location of the error may precede the location at which the code violation is detected (see Figure 17). In particular, even if the code violation is detected following the Frame_Header, fields in the header may not be valid.

23.3.3 Primitive Sequence protocol error

If a N_Port is in the Active State and it receives the Link Reset Response Primitive Sequence (LRR), it shall detect a Primitive Sequence protocol error which is counted in the Link Error Status Block (LESB).

23.4 Link error recovery

The Link recovery hierarchy is shown in Figure 64.

The recovery protocols are nested and organized from the most serious to least serious link action.

- Link Failure Protocol (see 10.6.4)
- Link Initialization Protocol (see 10.6.2)
- Link Reset Protocol (see 10.6.5)

Primitive Sequence Protocols provide two basic functions. The first function is to notify the other end of the link that a specific type of link error has occurred. The second function is to reset the link to a known state at both ends.

Link initialization and shutdown

Link initialization is accomplished by performing the Link Initialization Protocol. When this protocol is complete, the Port is synchronized, transmitting and receiving Idles (see 10.6.2) and is in the Active State.

Shutdown prior to power-off is accomplished by the Online to Offline Protocol. This Protocol provides an attached Port with a graceful indication prior to loss of signal. This avoids logging an error event for a normal system function (see 10.6.3).

Link Reset Protocol

Link Reset Protocol may be performed when any of the following conditions are detected:

- Connection Recovery (see 22.8),
- Link timeout (see 23.2.3),
- Buffer-to-buffer overrun (i.e., a N_Port receives a frame subject to buffer-to-buffer flow control without a buffer available).

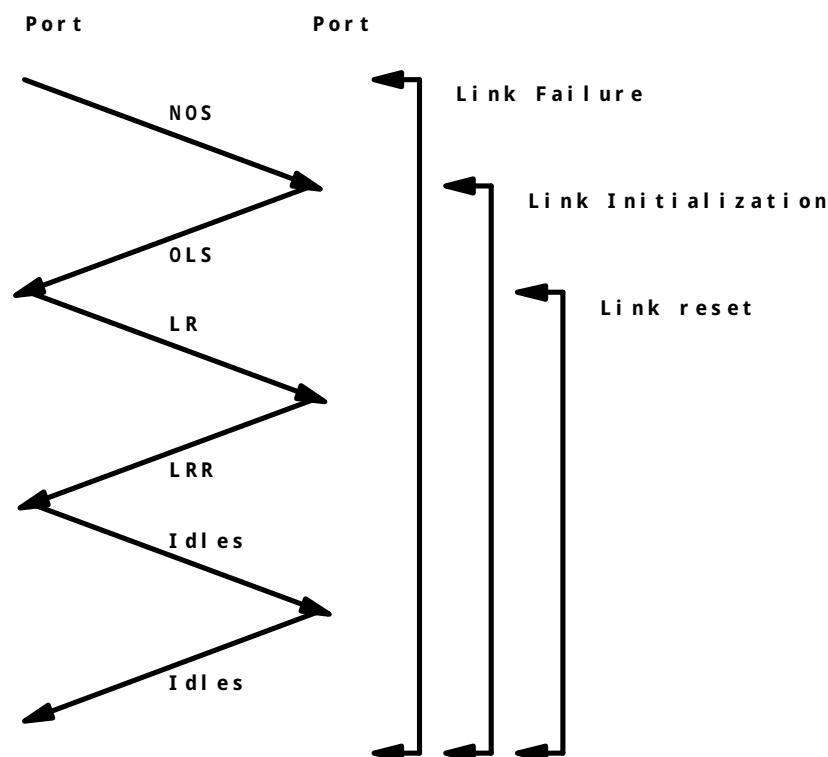


Figure 64 - Link recovery hierarchy

23.5 Link recovery - secondary effects

If the recovery action from an error involving link integrity requires transmission of a Primitive Sequence, Sequences within an Exchange may be adversely affected based on Class of Service.

Class 1

During a Class 1 Dedicated Connection, transmission or reception of Primitive Sequences removes the connection. While a N_Port is transmitting a Primitive Sequence, it shall discard any subsequent Class 1 frame received. For intermix, see the discussion regarding Class 2 and Class 3. After receiving the first Primitive Sequence of any Primitive Sequence Protocol, the end-to-end Credit and buffer-to-buffer Credit shall be reset to their original Login values in both N_Ports and the F_Ports involved.

After leaving the Active State to perform a Primitive Sequence Protocol, all frame Sequences that were Active are terminated abnormally. Recovery on a Sequence by Sequence basis is required of the Sequence Initiator at the discretion of the FC-4 or upper level. The Exchange is not aborted or abnormally terminated, unless either the Originator or Responder explicitly performs ABTX or LOGO. When Sequences are abnormally terminated, the current status of each Sequence is posted in the corresponding Exchange Status Block and the Sequence Status Blocks are reset and released. Link resources associated with the Active Sequences are released. If the Sequence Initiative is in doubt, the previous Initiator N_Port shall interrogate the other N_Port's Exchange Status Block to determine which N_Port has the Sequence Initiative (see 23.7.1).

Class 2

When Primitive Sequences are transmitted or received, the F_Port may discard any Class 2 frames held in its buffers. While a N_Port is transmitting a Primitive Sequence, it may discard any subsequent Class 2 frame received. After receiving the first Primitive Sequence of any Primitive Sequence Protocol, the buffer-to-buffer Credit shall be reset in the N_Port and the F_Port. Both the N_Port and F_Port may begin transmitting frames after entering the Active State.

Active Sequences within an Exchange are not necessarily affected. Therefore, normal processing continues and Sequence recovery is performed as required.

Class 3

When Primitive Sequences are transmitted or received, the F_Port may discard any Class 3 frames held in its buffers. While a N_Port is transmitting a Primitive Sequence, it may discard any subsequent Class 3 frame received. After receiving the first Primitive Sequence of any Primitive Sequence Protocol, the buffer-to-buffer Credit shall be reset in the N_Port and the F_Port. Both the N_Port and F_Port may begin transmitting frames after entering the Active State.

Active Sequences within an Exchange are not necessarily affected. Therefore, normal processing continues and Sequence recovery is performed as required.

23.6 Exchange Integrity

Proper Exchange and Sequence management accomplishes Exchange Integrity.

Applicability

Since Class 3 does not use ACK or Link_Response frames, Sequence integrity is verified at the Sequence Recipient on a Sequence by Sequence basis. In Class 3, only the Recipient is aware of a missing frame condition and communication of that information to the Initiator is the responsibility of the FC-4 or upper level.

The remaining discussion concentrates on Class 1 and 2. Items applicable to Class 3 shall be specified explicitly.

23.6.1 Exchange management

An Exchange is managed according to the rules specified in 18.3.1. When an Exchange is originated, the Originator specifies the Exchange Error Policy for the duration of the Exchange. Delivery of Data within a Sequence from the Originator to the Responder or from the Responder to the Originator shall be in the same order as transmitted. The discarding of Sequences, the delivery order of Sequences, and the recovery of Sequences varies based on the Exchange Error Policy identified by the Originator (see 12.6 Abort Sequence Condition bits (F_CTL bits 5-4)).

23.6.1.1 Exchange Error Policies

There are two fundamental Exchange Error Policies - discard and process. Discard policy means that a Sequence is delivered in its entirety or it is not delivered at all. There are three variations of discard policy which relate to the deliverability of ordered Sequences and one method of retransmission at the FC-2 level. Process policy allows an incomplete Sequence to be deliverable. Process policy allows the Data portion of invalid frames to be delivered if the Sequence Recipient has reason to believe that it is part of the proper Exchange. See 18.3.10 for rules that discuss detailed requirements for each type of Discard and Process policy.

Discard multiple Sequences

The Discard multiple Sequences Error Policy requires that for a Sequence to be deliverable, it shall be complete (all Data frames received and accounted for) and any previous Sequences, if any, for the same Exchange from the Sequence Initiator are also deliverable. This policy is useful if the ordering of Sequence delivery (Sequence A followed by Sequence D, followed by Sequence T, and so forth) is important to the FC-4 or upper level. Sequences shall be delivered to the FC-4 or upper level on a Sequence basis in the same order as transmitted.

Discard a single Sequence

The Discard a single Sequence Error Policy requires that for a Sequence to be deliverable, it shall be complete (all Data frames received and accounted for). There shall be no requirement on the deliverability of previous Sequences for the Exchange. This policy is useful if the Payload of the Sequences delivered contains sufficient FC-4 or upper level information to process the Sequence independently of other Sequences within the Exchange. Sequences shall be delivered to the FC-4 or upper level on a Sequence basis in the same order as received.

Process with infinite buffering

The Process with infinite buffering Error Policy does not require that a Sequence be complete or that any previous Sequences be deliverable. Process policy allows a N_Port to utilize the Data Field of invalid frames under certain restrictive conditions (see 11.8.1 and 11.8.2). The Process with infinite buffering Error Policy is intended for applications such as a video frame buffer in which loss of a single frame has minimal effect or no effect on the Sequence being delivered. Frames shall be delivered to the FC-4 or upper level in the same order as received.

The above shall also apply to Process with infinite buffering in Class 3.

Discard multiple Sequences with retransmission

The Discard multiple Sequences with immediate retransmission is a special case of the Discard multiple Sequences Error Policy. This policy is only applicable to an Exchange where all transmission is in Class 1. It provides a method by which the Sequence Recipient may request immediate Sequence retransmission at the FC-2 level (see 23.7.1.2) under the guidance or authorization of the FC-4 or upper level. Sequences shall be delivered to the FC-4 or upper level on a Sequence basis in the same order as transmitted.

23.6.2 Sequence integrity

Sequence management and integrity involves

- proper initiation of Sequences (see 18.3.4),
- proper control of the ordering of Sequences (SEQ_ID usage) with continuously increasing SEQ_CNT (see 18.3.6),
- proper control of Data frames by SEQ_CNT within single Sequence (see 18.3.6), and
- proper completion of a Sequence (see 18.3.8).

Frame identification and Sequence identification (see 12.1.1 and 12.1.2) provide the appropriate uniqueness to ensure the integrity of the Data delivered. A Sequence Recipient shall not reassemble and deliver the Data frames of a single Sequence unless all of the Data frames adhere to the Sequence management rules specified in 18.3.5.

23.6.3 Sequence error detection

Sequence errors are detected in three ways

- a) detection of a missing frame (see 18.3.10 and 18.3.11),
- b) detection of a Sequence timeout (see 23.2.4), and
- c) detection of rejectable condition within a frame (see 14.3.3.3).

In Class 1 and 2, detection of Sequence errors by the Recipient is conveyed in the Abort Sequence Condition bits (F_CTL 5-4) in an ACK frame or by a P_RJT frame. Otherwise, either the Sequence Initiator or Sequence Recipient or both detect a Sequence timeout.

Exchange and Sequence status are tracked in the Exchange Status Block (see 18.8.1 and 18.3.14) and the Sequence Status Block (see 18.8.2 and 18.3.12), respectively.

23.6.4 X_ID processing

During certain periods in an Exchange, one or both X_ID fields may be unassigned. If an X_ID is unassigned, special error recovery for both the Sequence Initiator and the Sequence Recipient may be required (see 19.4).

23.7 Sequence recovery

Sequence recovery is under control of the individual FC-4 or upper level as well as options within a specific implementation. Such control may be exercised in the form of guidance, authorization to automatically perform recovery, a requirement to inform the upper level prior to recovery, or no Sequence recovery within the Exchange encountering a Sequence error. FC-FS specifies procedures to terminate or abort a Sequence, recover end-to-end Credit, handle missing or delayed frames, and synchronize both N_Ports with respect to Sequence and Exchange status. FC-FS does not require Sequence retransmission within the same Exchange in which a Sequence error is detected.

23.7.1 Abnormal Sequence termination

There are multiple methods by which a Sequence completes abnormally and one method by which a Sequence completes but is only partially received by the Sequence Recipient. When a Sequence completes abnormally, it shall not be delivered to the FC-4 or upper level. The rules for normal Sequence completion are specified in 18.3.8. The methods by which a Sequence completes abnormally include:

- the Sequence Initiator aborts the Sequence with an ABTS frame utilizing the Abort Sequence Protocol. At the time the ABTS frame is received, one or more Sequences are incomplete.
- a Class 1 Dedicated Connection is broken by transmission or reception of a Primitive Sequence while the Sequence is Open.
- if the Exchange of which a Sequence is a member is abnormally terminated, each Open Sequence shall also be abnormally completed (see 18.3.13).
- Logout (see 15.5.8).

A Sequence may complete normally with only a part of the Sequence being received by the Sequence Recipient in the Stop Sequence Protocol.

23.7.1.1 Abort Sequence Protocol

The Sequence Initiator shall begin the Abort Sequence Protocol (ABTS Protocol) by transmitting the ABTS Basic Link Services frame. The SEQ_ID shall match the SEQ_ID of the last Sequence transmitted even if the last Data frame has been transmitted. The ABTS frame may be transmitted without regard to whether transfer of Sequence Initiative has already been attempted or completed. The SEQ_CNT of the ABTS frame shall be one greater than the SEQ_CNT of the last frame transmitted for this Exchange by the Sequence Initiator of the ABTS frame.

The Sequence Recipient shall accept an ABTS frame even if the Sequence Initiative has been previously transferred. The Recipient determines the last deliverable Sequence as the Recipient for this Exchange and it includes that SEQ_ID in the Basic Accept Payload along with a valid indication (see 15.2.1). The Payload of the Basic Accept also includes the current OX_ID and RX_ID for the Exchange in progress. Low and high SEQ_CNT values are also returned. The low SEQ_CNT value is equal to the SEQ_CNT of the last Data frame (End_Sequence = 1) of the last deliverable Sequence. If there was no deliverable Sequence, the low value is zero.

The high SEQ_CNT value shall match the SEQ_CNT of the ABTS frame. The combination of OX_ID, RX_ID, low SEQ_CNT and high SEQ_CNT defines a Recovery_Qualifier range. This range indicates a range of Data frames that were not and shall never be delivered to the FC-4 or upper level in the Discard multiple Sequences Error Policy. In the Discard a single Sequence Error Policy, the Recovery_Qualifier may contain Sequences, which have been delivered.

If the ABTS frame is transmitted in Class 2 or 3, the Recovery_Qualifier shall be timed out by the Sequence Initiator of the ABTS frame for a timeout period of R_A_TOV. After the R_A_TOV timeout has expired, the Sequence Initiator of the ABTS frame shall issue a Reinstall Recovery Qualifier Link Service request in order to purge the Recovery_Qualifier. Timing out the Recovery_Qualifier range for R_A_TOV allows both N_Ports to discard frames received in this range. This ensures the Data integrity of the Exchange.

NOTE - Since streaming Sequences across multiple Classes for the same Exchange is not allowed, and since ABTS shall be transmitted as part of the last Sequence transmitted, the Abort Sequence Protocol is performed in the same Class of Service in which the error, if any, occurred. Therefore, there is no ambiguity between the Sequence Initiator and the Sequence Recipient as to whether a Recovery_Qualifier is needed. If ABTS is transmitted in Class 1, then none is needed. If ABTS is transmitted in Class 2 or Class 3, then a Recovery_Qualifier is established. For example, if a Se-

quence Initiator has transmitted a Sequence in Class 1, but Link Recovery is performed before all acknowledgements are received, then the Sequence Initiator is required to reestablish a Class 1 Connection in order to transmit ABTS.

Transmission of BA_ACC by the Sequence Recipient is a synchronizing, atomic event. The Sequence Recipient shall discard any frames received within the Recovery_Qualifier range, if timeout is required, the instant that the Basic Accept is transmitted and thereafter, until it receives a Reinstate Recovery Qualifier (RRQ) Extended Link Services request. The Sequence Initiative F_CTL bit setting in the BA_ACC shall indicate whether the Sequence Initiative is held or transferred to the Sequence Initiator of the ABTS frame. If the Sequence Recipient of the ABTS frame holds Sequence Initiative, it shall withhold Sequence Initiative transfer until the ACK to the Basic Accept is received.

In like manner, after the Sequence Initiator has received the Basic Accept to the ABTS frame, it shall discard any frames received for the Recovery_Qualifier range. In Class 2 and 3, the Sequence Initiator shall retire the SEQ_CNT range within the Recovery Qualifier until R_A_TOV has expired, or it shall abort the Exchange (the Recovery_Qualifier for the Exchange times out R_A_TOV).

When the Sequence Initiator has received the Basic Accept, it may reclaim any outstanding end-to-end Credit for all unacknowledged Data frames within the SEQ_CNT range of the Recovery_Qualifier. After the Sequence Initiator of the ABTS frame has received the BA_ACC with Sequence Initiative transferred to the Initiator, it may retransmit the Sequences that it determines as non-deliverable by the Sequence Recipient (see 18.3.8 and 18.3.11).

If a Recovery_Qualifier exists and is being timed out (R_A_TOV) and another Sequence error occurs which would cause transmission of the ABTS frame, the Exchange shall be aborted using the ABTX request. However, if the Reinstate Recovery Qualifier request has been completed such that the Recovery_Qualifier has been purged, the ABTS Protocol may be utilized and a new Recovery_Qualifier may be established.

Special case - new Exchange

If a Sequence Initiator of the ABTS frame attempts to originate a new Exchange and a Sequence timeout occurs, the Sequence Initiator shall transmit the ABTS frame as in the ABTS protocol defined. If the Sequence Recipient receives an ABTS frame for an Exchange which is unknown, it shall Open an Exchange Status Block, with OX_ID = value of ABTS, RX_ID = hex 'FF FF', and post the SEQ_ID of the ABTS frame. The Basic Accept Payload shall indicate invalid SEQ_ID, a low SEQ_CNT set to zero, and a high SEQ_CNT set to SEQ_CNT of the ABTS frame.

The Sequence Initiator of the ABTS frame shall timeout the Recovery_Qualifier, if required, and transmit the Reinstate Recovery Qualifier in the normal manner.

Special case - Class 3

If a Sequence Initiator desires to determine the status of an Open Exchange in Class 3, it is possible to transmit an ABTS frame on a periodic basis in order to determine successful status (see 18.3.8). Deliverable Sequences are not affected, yet the Sequence Initiator detects undeliverable Sequences. However, it should be noted that if the ABTS frame arrives prior to a frame in flight for the Active Sequence, an otherwise deliverable Sequence becomes undeliverable once a Recovery_Qualifier is established. If the ABTS Protocol is performed, the normal rules are applicable.

23.7.1.2 Class 1 Sequence retransmission

In an asynchronous environment, Sequence retransmission is made difficult due to a possible difference in error information available in the Sequence Recipient compared to the Sequence Initiator. Therefore, there are restrictions placed on its use:

- Class 1 only (in-order delivery, with no frames delayed in the Fabric).
- the Sequence Initiator shall request Discard multiple Sequences with immediate retransmission as the Error Policy on the first Data frame of the Exchange.
- the Sequence Recipient shall not be required to support Sequence retransmission and may reply with Abort Sequence Condition bits set to (0 1) if a missing frame error is detected.
 - if the Sequence Recipient does not support the Discard multiple Sequences with immediate retransmission Error Policy, it shall set the Abort Sequence Condition bits (F_CTL 5-4) to (0 1) to indicate a missing frame error.
 - if the Sequence Recipient does not support the Discard multiple Sequences with immediate retransmission Error Policy, it shall transmit an ACK with the Abort Sequence Condition bits (F_CTL bits 5-4) set to (0 1) if a Data frame is received with the retransmit bit (F_CTL bit (9) set = 1).
- if Discard multiple Sequences with immediate retransmission is supported by the Sequence Recipient, the Sequence Recipient shall set the Abort Sequence Condition bits (F_CTL 5-4) to (1-1) to indicate a missing Data frame error when:
 - the Sequence Recipient determines that a missing frame error has occurred in a Sequence such that an ACK may be transmitted for the same SEQ_ID of the first non-deliverable Sequence with the retransmit bit in F_CTL set = 0.
 - the Sequence Recipient shall continue to transmit ACKs for the same SEQ_ID with the Abort Sequence Condition bits set = (1 1).
 - for any subsequent streamed Sequences, the Sequence Recipient shall set the Abort Sequence Condition bits to (0 1) and the Sequences shall be non-deliverable. The Sequence Recipient shall discard those subsequent Sequences with no Sequence Status Blocks opened, and no information saved.
 - the Sequence Recipient shall save the SEQ_ID and starting SEQ_CNT for that SEQ_ID for comparison to a retransmitted Sequence (i.e., the SEQ_CNT of the SOFi1 frame).
- if the Sequence Recipient detects a missing Data frame error, but is unable to determine what the SEQ_ID of the first non-deliverable Sequence should be, or it is unable to determine the starting SEQ_CNT, it shall set the Abort Sequence Condition bits (F_CTL bits 5-4) to (0 1) in the corresponding ACK frame.
- when the Sequence Initiator receives an ACK with Abort Sequence Condition bits set = (1 1), it shall make a determination as to whether the SEQ_ID of the ACK frame identifies the first non-deliverable Sequence based on its ACK frame history of prior Sequences. If the Sequence Initiator is unable to make such a determination, it shall either Abort the Sequence with ABTS, Read Exchange Status, or Read Sequence Status to determine what Sequence was in error.
- if the Sequence Initiator is able to determine that the first non-deliverable Sequence in error matches the SEQ_ID of the ACK frame with the Abort Sequence Condition bits set to (1 1), it shall begin Sequence retransmission with the following restrictions:
 - F_CTL bit 9 (retransmission bit) shall be set to one on each Data frame of the Sequence being retransmitted as well as any subsequent streamed Sequences until the first retransmitted Sequence is determined by the Sequence Initiator to be deliverable.
 - the SEQ_ID shall match the SEQ_ID of an ACK frame with the Abort Sequence Condition bits set to (1 1),
 - the SEQ_ID being retransmitted is the first non-deliverable Sequence as determined by the Sequence Initiator,
 - the SEQ_CNT shall match the SEQ_CNT of the first Data frame (SOFi1) of the original non-deliverable Sequence transmitted as determined by the Sequence Initiator.

- the Sequence Recipient shall check for an ABTS frame and perform the normal Abort Sequence Protocol if received, independently of the retransmit bit (F_CTL bit 9) setting.
- the Sequence Recipient shall also check for an SOFi1, with F_CTL bit 9 = 1, SEQ_ID = first non-deliverable Sequence, and SEQ_CNT set to first SEQ_CNT of the non-deliverable Sequence.
- the Sequence Recipient shall set the retransmit bit (F_CTL bit 9) = 1 in the ACK frame corresponding to a Data frame in which F_CTL bit 9 = 1.
- if the Sequence Recipient detects an error in the restart point of the Sequence retransmission (either incorrect SEQ_ID, or incorrect SEQ_CNT of SOFi1.), it shall transmit an ACK with the retransmission bit (F_CTL bit 9) set to 1, the Abort Sequence Condition bits set to (0 1), requesting ABTS, and the Sequence being retransmitted shall be non-deliverable.
- after the Sequence Initiator determines that the retransmitted Sequence is successfully delivered (ACK EOFt), it shall transmit subsequent Sequences with F_CTL bit 9 = 0.

23.7.1.3 Recipient abnormal termination

If no Data frames are being received for a Sequence in error, the Sequence Recipient shall timeout the Sequence and abnormally terminate the Sequence by setting status in the Sequence Status Block to indicate Sequence timed-out by Recipient, update the Sequence status in the Exchange Status Block, and release link facilities associated with the Sequence. If an ABTS frame for the abnormally terminated Sequence is received, the Abort Sequence Protocol shall be performed and completed.

The Sequence Recipient may timeout the Exchange in a system dependent manner and timeout period.

23.7.1.4 End_Sequence

If the last Data frame of a Sequence has been transmitted and the Sequence Initiator detects a Sequence timeout, the Initiator may initiate an Exchange with a Read Exchange Status Link Service request to determine Sequence and Exchange status. If the Initiator is in the process of timing out a Sequence for a missing EOFt with Sequence Initiative transferred and it receives a new Sequence initiated by the Recipient (new Initiator), it shall assume that the previous Sequence ended successfully. In order to make such an assumption, the N_Port Identifier, OX_ID, and RX_ID shall be the same for the new Sequence as the Sequence that transferred Sequence Initiative.

From a Recipient view, if the last Data frame is lost, the Recipient abnormally terminates the Sequence when a Sequence timeout is detected.

23.7.2 Stop Sequence Protocol

The Stop Sequence Protocol shall be used when a Sequence Recipient wishes to terminate a Sequence without invoking a drastic recovery protocol. To cause a Sequence to be terminated by the Initiator, the Sequence Recipient shall set the Abort Sequence Condition bits in F_CTL to a 1 0 value in the ACK to each Data frame received after the Recipient recognizes the need to terminate the Sequence. When the Sequence Initiator receives the first ACK with the Stop Sequence Condition indicated, it shall terminate the Sequence by transmitting a Data frame of the Sequence with End_Sequence = 1. If the Sequence Initiator has already transmitted the last Data frame of the Sequence, no further action is required of it except that which may be required by the FC-4 or upper level.

Once the Sequence Recipient has indicated the Stop Sequence condition, it shall not report Sequence errors related to Data frames with a SEQ_CNT greater than that of the Data frame at which the Stop Sequence condition was recognized. However, it shall observe the normal Se-

quence timeout protocols before transmitting the ACK to the frame with the End_Sequence bit set and shall recover Credit in the normal manner.

NOTES

1. When the Sequence Initiator stops the Sequence, or if it sent the last Data frame before receiving the Stop-Sequence indication, it may either hold or pass Sequence Initiative as determined by the Upper Level Protocol. It is the responsibility of the Upper Level Protocol to define the protocol for indicating to the Sequence Initiator why the Sequence was stopped, if such an indication is needed, and the protocol for transferring Sequence Initiative if needed.
2. A common use of this protocol is to signal that there is no more room in the Upper Level Protocol buffer for the Data being received. To terminate the Sequence when the Upper Level Protocol buffer is exhausted, the Sequence Recipient stores as much data as possible from the first frame whose Payload cannot be completely stored and indicates the Stop Sequence condition in the Abort Sequence Condition bits in F_CTL in the ACK to that Data frame and in each subsequent ACK until the end of the Sequence. When the Sequence ends, the ULP protocol may send a message from the Sequence Recipient to the Initiator which includes the count of the number of bytes in the Sequence which were stored before the ULP buffer was exhausted.

23.7.3 End-to-end Credit loss

FC-FS does not specify the method to be employed for Credit allocation to a destination N_Port. If destination N_Port end-to-end Credit is allocated on a Sequence basis, then that portion of end-to-end Credit is reclaimed when the Sequence is aborted or abnormally terminated. When a Sequence is aborted, any end-to-end Credit for outstanding ACK frames associated with that Sequence may be reclaimed. This applies to both Class 1 and Class 2.

23.8 Link Error Status Block

The errors shown in Figure 65 are accumulated over time within a N_Port. The format shown is the format in which the LESB information shall be supplied in response to an RLS Link Service command. It does not require any specific hardware or software implementation. The errors accumulated provide a coarse measure of the integrity of the N_Port link over time. No means is provided to reset the LESB, it merely is allowed to overflow and thus reset. The counts shall be 32 bit binary integers.

A N_Port may choose to log these events as well as other errors that occur on a N_Port specific basis, which is not defined within FC-FS.

NOTE - It is recommended that F_Ports also maintain an LESB and accumulate error events in a manner, which is not defined in FC-FS.

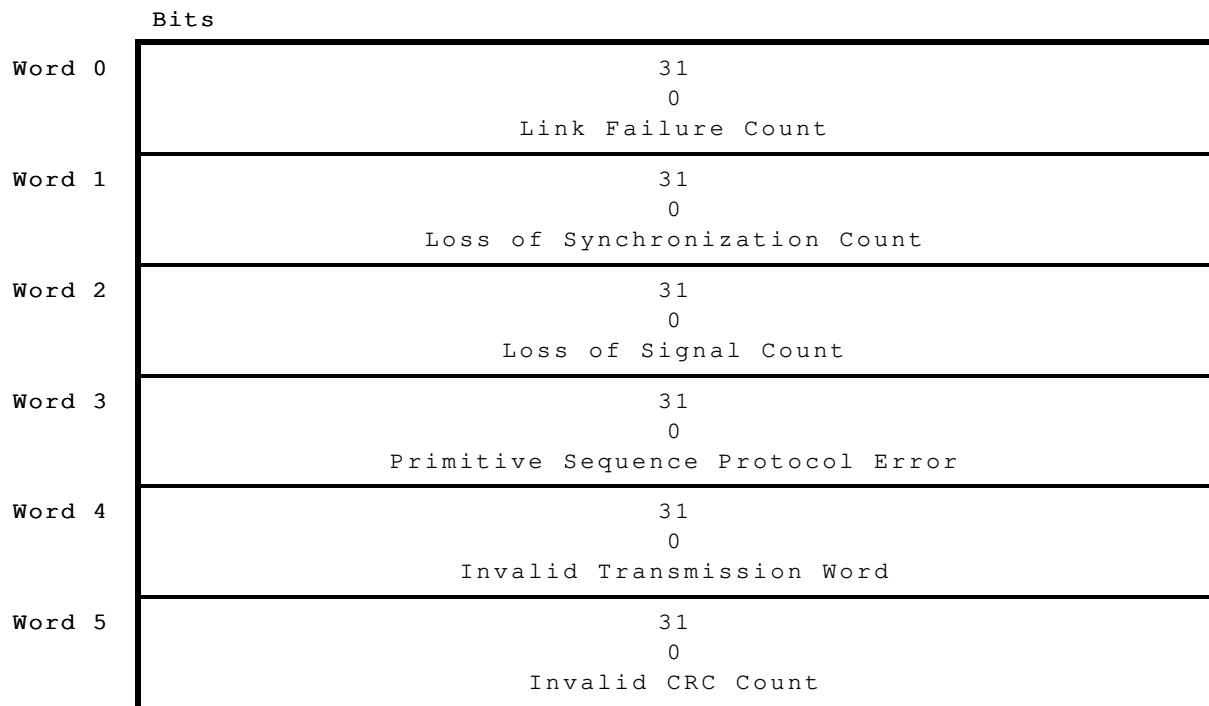


Figure 65 - Link Error Status Block format for RLS command

NOTE - Informative guideline to manage LESB is provided in Annex H.

23.9 Detailed error detection / actions

23.9.1 Errors detected

Table 219 lists 10 categories of errors that are detectable. The categories specified relate directly to link integrity or data integrity as previously discussed. This list is representative of the types of errors which may be detected. It is not an exhaustive list. Column 1 of Table 219 specifies a general error category. Column 2 identifies specific errors within that general category. Column 3 identifies the action that the Sequence Initiator takes on ACK frame errors detected for Sequences being transmitted or link integrity errors (ACK frame reception is only applicable to Class 1 and 2). Column 4 identifies the action that the Sequence Recipient takes on Data frame errors detected for the Sequences being received or link integrity errors.

Table 219 - Detailed errors and actions			
Error Category	Specific Error	Seq Init Action	Seq Recp Action
Link Failure	Loss of Signal	12	12
	Loss of Sync> timeout period	12	12
Link Errors	Invalid Transmission Word during frame reception	1,11	1,11
	Invalid Transmission Word outside of frame reception	11	11
	Loss of Sync	11	11
Link Timeout	Class 1: all Sequences timed out	6	6
	missing R_RDYs (buffer-to-buffer credit = 0)	6	6

Table 219 - Detailed errors and actions			
Error Category	Specific Error	Seq Init Action	Seq Recp Action
Link Reset protocol timeout	missing LRR response to LR transmission	12	12
	missing Idle response to LRR transmission	12	12
Sequence timeout	timeout during Sequence	8, 10	9
	timeout at end of Sequence	8, 14, 10	9
Delimiter Errors	Class not supported	2	2
	Delimiter usage error (SOFct while connected)	2	2
	Abnormal frame termination	1	1
	EOFa received	1	1
	Incorrect SOF or EOF (see Table 33 and Table 35)	1	1
Address Identifier errors	incorrect D_ID	2	2
	incorrect S_ID	2	2
Frame_Content errors	CRC	1	1
	Busy frame received	5	5
	Reject frame received	3	3
	TYPE not supported	2	2
	Invalid Link_Control	2	2
	Invalid R_CTL	2	2
	Invalid F_CTL	2	2
	Invalid OX_ID	2	2
	Invalid RX_ID	2	2
	Invalid SEQ_ID	2	2
	Invalid SEQ_CNT	2	2
	Invalid DF_CTL	2	2
	Exchange Error	2	2
	Protocol Error	2	2
	Incorrect length	2	2
	Unexpected Link_Continue	2	2
	Unexpected Link_Response	2	2
	Login Required	2	2
	Excessive Sequences attempted	2	2
	Unable to Establish Exchange	2	2
	RO out of bounds	NA	2
Data Frame errors	buffer not available - Class 1	NA	2.7
	buffer not available - Class 2	NA	4
	buffer not available - Class 3	NA	1
	ABTS frame received	NA	8
	missing frame error detection	NA	13,7
ACK_1 or ACK_0 frame errors	ABTS frame received	13,8	NA
	missing frame error detection	8,10	NA

23.9.2 Actions by Initiator or Recipient

1. Discard frame

In both discard and process policy, SOF_{c1}, EOF_{dt}, and EOF_{dti} delimiters shall be processed for both invalid and valid frames. In both discard and process policy, a frame that is terminated with an EOF_a shall be discarded.

2. Discard policy

If an invalid frame is detected, the entire invalid frame shall be discarded. If a valid frame is received and a rejectable or busy condition in Class 3 is detected, the entire frame shall be discarded.

3. Process policy

If a N_Port is able to determine that an invalid frame is associated with an Exchange which is designated as operating under Process policy, the N_Port may process and use the Data Field at its discretion, otherwise, the entire invalid frame shall be discarded.

4. Transmit P_RJT frame

If a valid Data frame is received which contains information in the Frame_Header, which is invalid or incorrect, a P_RJT frame shall be transmitted with the appropriate reason code as specified in 14.3.3.3. Reason codes are defined such that the first error detected is returned as the reason code. In Class 2, R_RDY is transmitted in response to the discarded frame.

During a Class 1 Dedicated Connection, if a Data frame is received and no buffer is available, this indicates that the transmitter has an end-to-end Credit tracking problem. The reason code in P_RJT shall indicate protocol error.

5. Process Reject

When a P_RJT or F_RJT frame is received in response to a frame transmission, the reject information shall be passed to the appropriate Upper Level Protocol in order to process. Certain errors are recoverable by taking an appropriate action, such as Login. The Sequence shall be aborted using the Abort Sequence Protocol, regardless of possible recovery actions. For typical non-retryable errors the Exchange shall also be aborted.

If a P_RJT or F_RJT frame is received which contains information in the Frame_Header, which is invalid or incorrect, the frame shall be discarded.

6. Transmit P_BSY frame

A N_Port shall track the status of its buffers such that if a Class 2 Data frame is received and no EE_buffer is available, a P_BSY shall be returned to the transmitter of the frame. If a Class 2 Data frame is received and no BB_buffer is available, the Recipient may discard the frame without issuing a P_BSY or P_RJT. Portions of the frame other than the Frame_Header are discarded. The Frame_Header is captured in order to generate a proper P_BSY Link_Response frame.

If a N_Port receives a Class 1 connect-request frame and its internal link facilities are busy or unavailable, the N_Port responds by transmitting a P_BSY frame with the appropriate reason code (see 14.3.3.2) with EOF_{dt}. Portions of the frame other than the Frame_Header are discarded. The Frame_Header is captured in order to generate a proper P_BSY Link_Response frame.

An R_RDY is transmitted in response to a Class 2 frame or a Class 1 connect-request regardless of the disposition of the received frame.

7. Process Busy

When an F_BSY or P_BSY is received in response to a Class 1 connect-request, the N_Port may:

- attempt a connect-request to a different destination N_Port if such a request is pending,
- delay a period of time before reissuing the connect-request, or
- immediately reissue the connect-request.

The Fabric shall not queue a connect-request, if the destination N_Port has responded with a P_BSY to the connect-request.

NOTE - The decision as to which action to take is dictated based on the conditions in the N_Port and the period of time lost if another busy condition is returned.

When an F_BSY or P_BSY is received in response to a Class 2 Data frame, and if the N_Port has the capability to retransmit, the N_Port shall retransmit the Class 2 Data frame within E_D_TOV of the last Data frame transmission. In order to avoid reissuing a frame for an extended number of retries a N_Port may choose to count the number of retries and decide to shutdown communication with a specific N_Port.

When an F_BSY is received in response to an ACK frame (Class 2), the N_Port shall not retransmit the ACK frame.

8. Perform Link Reset Protocol

When a N_Port has reached a buffer-to-buffer Credit_CNT of zero and has exceeded the Link timeout period (E_D_TOV), a Link timeout is detected. In Class 1, if a N_Port has timed out all Active Sequences (E_D_TOV), a Link timeout is detected. When a Link timeout is detected, the N_Port or F_Port begins the Link Reset Protocol.

In addition, in Class 1 a N_Port may not know whether a Dedicated Connection exists or not. A N_Port may initiate the Link Reset Protocol in order to reach a known state.

9. Set Abort Sequence Bits

When a Sequence Recipient detects a Sequence error by missing frame detection or other internal processing errors, the Recipient sets the appropriate Abort Sequence in F_CTL bits 5-4.

- 0 0 = Continue Sequence
- 0 1 = Abort, perform ABTS
- 1 0 = Stop Sequence
- 1 1 = Immediate Sequence retransmission requested

The SEQ_CNT of the first missing frame is saved in the Sequence Status Block. Only the first error is saved in the Sequence Status Block. This information may or may not be required by the Sequence Initiator for recovery purposes.

10. Perform Abort Sequence Protocol

When a Sequence Initiator detects a Sequence error or receives an appropriate Abort Sequence Condition (0 1) in F_CTL bits 5-4 in an ACK for an Open Sequence, the Initiator shall transmit an Abort Sequence Link Service request to the Recipient and transfers Sequence Initiative in order to complete Abort Sequence processing (see 23.7.1).

When a Sequence Recipient receives an ABTS frame, it shall respond as specified in 23.7.1.1 and 15.2.1.

11. Abnormally terminate Sequence

When a Sequence Recipient detects a Sequence timeout (E_D_TOV) and no Data frames are being received for the Sequence, the Recipient shall terminate the Sequence and update the Exchange Status Block.

When a Class 1 Dedicated Connection is removed by an EOF delimiter or a Primitive Sequence, the N_Port shall abnormally terminate any Active Sequences (see 23.7.1).

12. Retry Sequence

When a Sequence has been abnormally terminated, the Sequence Initiator may retransmit the Sequence under guidance, authorization, or control of the FC-4 or upper level. Under guidance from the upper level, it may retransmit immediately under the Discard multiple Sequences with immediate Retransmission Exchange Error Policy in Class 1 (23.7.1.2). For other Exchange Error Policies, each individual FC-4 shall specify the procedure for Sequence retransmission, if any.

13. Update LESB

The Link Error Status Block is updated to track errors not directly related to an Exchange.

14. Perform Link Failure Protocol

Transmission or reception of the Not Operational Primitive Sequence initiates the Link Failure Protocol.

15. Error Policy processing

When an error is detected within a Sequence, the Sequence is either processed normally (process policy), or discarded (discard policy). See 23.6.1.1 for additional information.

16. Read Status Block

Read Status Block (either Sequence or Exchange) is accomplished by originating an Exchange with the appropriate Extended Link Service request.

24 Hunt Group

24.1 Introduction

A Hunt Group is a collection of one or more N_Ports controlled by a Common Controlling Entity. A Hunt Group is addressed using a Hunt Group Identifier (HG_ID) which is managed as an Alias. The Hunt Group shall be formed by the registration of a group of one or more N_Ports with the Alias Server. A N_Port may be a member of multiple Hunt Groups and may be addressed by either its base address identifier or any one of its Hunt Group Identifiers (Aliases). Other methods may be used to select one of N destination N_Ports using only base address identifiers, however, those methods do not constitute a Hunt Group.

N_Port_Login with a Hunt Group using the designated HG_ID shall be required prior to other communication. A N_Port using its base address identifier or HG_ID (Alias) and the normal N_Port_Login procedure may accomplish Login with a Hunt Group using the HG_ID as the destination address identifier. The formation and operation of a Hunt Group shall require specific support in both the N_Port and the Fabric.

The Alias Server handles the registration of N_Ports into a Hunt Group, and the deregistration of N_Ports from a Hunt Group. The Alias Server is not involved in the routing of Hunt Group frames.

Hunt Group routing is part of the adaptive addressing protocol specified in 30.7.

24.2 Function

The function of a Hunt Group is to provide a HG_ID (Alias) which shall be used by the Fabric to select a single available N_Port out of the group of possible N_Ports as the destination (D_ID) of a frame. This increases the likelihood of locating a destination N_Port that is not engaged in a Class 1 connection. In Class 2, Hunt Groups may also be used to increase aggregate bandwidth, reduce latency, or both.

24.3 Communication Model

The Hunt Group communication model is an extension of N_Port to N_Port communication. Since a Hunt Group is a collection of N_Ports, the following combinations are supported in this model:

- N_Port to N_Port
- N_Port to Hunt Group
- Hunt Group to Hunt Group

All combinations are supported as part of the adaptive addressing protocol, specified in 30.7.

24.4 Applicability

Hunt Group is an optional communication model. To support this model, Fabric and N_Port shall have the additional capability specified.

24.4.1 Classes

Class 1, 2, 3, or 4 may be used with Hunt Group. Common Controlling Entity (CCE) based or Port based allocation of resources required shall vary by Class of service (see 24.8).

24.4.2 Class 4 Addressing

Class 4 S_ID addressing is different from other Classes due to support of multiple Virtual Circuits per N_Port in Class 4. Since Class 4 S_ID is always associated with a specific Virtual

Circuit ID, any Class 4 S_ID used in Hunt Group shall be understood to have a specific Virtual Circuit ID associated with it, explicitly stated or not.

This address implication shall be valid for all mechanisms specified for Hunt Group for Class 4 N_Ports.

For example, a N_Port ID K1 in Class 4 implies specifically, say, K1(5) where 5 is the Virtual Circuit ID. A Hunt Group ID K in Class 4 implies specifically, say, K1(5), K2(9), and K3(25).

24.5 Formation

A Hunt Group is formed by a N_Port under control of the Common Controlling Entity performing Hunt Group Registration with the Alias Server. The N_Port performing the registration need not be a member of the Hunt Group being formed but shall be under control of the Common Controlling Entity constituting the Hunt Group.

The ultimate responsibility lies with Common Controlling Entity in allowing any of its N_Ports becoming a Hunt Group member and communicating common service parameters for such membership. The method by which the Common Controlling Entity informs its N_Ports is outside the scope of this document.

During registration, the Alias Server shall inform each member N_Port of its HG_ID.

Providing Hunt Group information to a Fibre Channel Directory Server or Management Server is the responsibility of the Alias Server. At the end of the Hunt Group registration, the Alias Server has the following information:

- Alias assigned
- Assigned Alias is associated with Hunt Group
- Node_Name associated with the list of N_Ports in the Hunt Group
- List of N_Ports in the Hunt Group.

The Fabric Controller is responsible for distribution of Aliases for Hunt Group Identifiers and Fabric-N_Port Service Parameters to the appropriate internal facilities and F_Ports within the Fabric. The method for distributing this information within the Fabric is outside the scope of this document.

24.5.1 Registration / deregistration

Registration of a list of N_Ports into or deregistration of a list of N_Ports from a Hunt Group by a Common Controlling Entity is performed by originating an Exchange with the Alias Server. Any N_Port under control of the Common Controlling Entity may be the Originator of the Exchange performing Registration / deregistration in a Class supported by the Fabric Controller and the N_Port. The third party authorization for registration or deregistration is the responsibility of the Common Controlling Entity which is outside the scope of the document.

The Exchange may be used to:

- create a Hunt Group with a list of member N_Ports,
- add a list of N_Ports to an existing Hunt Group, or
- delete a list of N_Ports of an existing Hunt Group.

Hunt Group Identifier is managed as an Alias. The detailed format and protocol for registration / deregistration are specified in clause 32.

The information transmitted by the Originator contains the following items:

Node_Name

The Node_Name of the Common Controlling Entity.

HG_ID

The Hunt Group Identifier that is associated with an existing Hunt Group for add and delete functions. In the case of creating a Hunt Group, the HG_ID may be unidentified (binary zeros) or a value requested by the Common Controlling Entity.

N_Ports

During creation of the Hunt Group, all member N_Ports are identified. During add or delete functions, member N_Ports being added or deleted are identified.

Common and Class Service Parameters

During the creation of a Hunt Group, the Fabric N_Port Service Parameters shared by each member N_Port for all three Classes of Service shall be included in the same format and meaning as during the N_Port to Fabric Login procedure (see clause 17).

24.5.2 Inquiry

The Hunt Group information may be available from the Directory Server, if Alias Server has communicated the information to the Directory Server.

If the membership of a Hunt Group is uncertain, the Common Controlling Entity may replace its members by using appropriate Alias Server functions for an existing HG_ID.

24.5.3 Hunt Group ID Removal

Removal of a Hunt Group by a Common Controlling Entity is performed by originating an Exchange with the Alias Server. Any N_Port under control of the Common Controlling Entity may be the Originator of the Exchange performing the removal of a Hunt Group in a class supported by the Alias Server and the N_Port. The details of the Exchange are specified in the in clause 26.

24.6 Destination N_Port_Login

Destination N_Port_Login shall be required on an address identifier basis. Therefore, Login shall be performed with the Hunt Group Identifier (HG_ID) prior to communication other than Basic and Extended Link Applications. Login with any instance of the alias address identifier of HG_ID completes Login with all members of the Hunt Group (HG_ID).

For example, HG_ID=K (K1, K2, K3) and HG_ID=M (M1, M2, M3) exist. M shall be required to Login with K before M to K or K to M communication even if K1 has performed Login with M2. Likewise, K3 shall be required to Login with M2 before K3 to M2 communication even if K and M Login have been performed. Destination N_Port_Login shall have been completed between the S_ID and D_ID N_Port address identifiers prior to communication of Upper Level Protocols for the first Data frame of the Exchange.

Hunt Group Service parameters

Each member N_Port of a specific Hunt Group shall operate with a single, identical set of Common and Class Service Parameters with the Fabric and destination N_Ports with which it communicates.

24.7 Addressing Protocol

An adaptive address identifier protocol is utilized when Hunt Group Identifiers (HG_ID) are being employed to allow for multiple strategies for managing Exchange and Sequence resources. This adaptive address identifier protocol supports any combination of communication models (see 24.3).

24.7.1 Originator

The Originator of an Exchange may specify its N_Port ID or an HG_ID as S_ID in the first Data frame of the Exchange. The Originator may specify as the Responder's N_Port ID or an HG_ID as D_ID in that frame.

24.7.2 Responder

In the ACK to that first Data frame, the Responder shall address the Originator with the same ID, which the Originator identified itself in the first Data frame. However, the Responder may choose to use either its N_Port ID or the Hunt Group ID with which it is addressed by the Originator.

NOTE - If the Responder specifies another HG_ID, the Originator may reject the Exchange with a reason code = protocol error.

24.7.3 Address Resolution

After the first ACK has been specified, the identifiers for both Originator and Responder shall remain the same. With X_ID invalidation, the Exchange shall resume from step specified in 24.7.1.

The address resolution for different cases is summarized in the Table 220.

Table 220 - Address Resolution		
Originator First Data frame	Responder ACK to First Data frame	Resolution
S_ID, D_ID	S_ID, D_ID	Originator ID, Responder ID
Originator HG_ID K, Originator N_Port ID x	Originator N_Port ID x, Originator HG_ID K	Originator HG_ID K, Originator N_Port ID x
Responder N_Port ID y, Responder HG_ID L	Responder HG_ID L, Responder N_Port ID y	Responder N_Port ID y, Responder HG_ID L
	Responder N_Port ID L _m , Responder N_Port ID y	Responder N_Port ID y, Responder N_Port ID L _m
Originator HG_ID K, Responder HG_ID L	Responder HG_ID L, Originator HG_ID K	Originator HG_ID K, Responder HG_ID L
Originator N_Port ID K _n , Responder HG_ID L	Responder N_Port ID L _m , Originator N_Port ID K _n	Originator N_Port ID K _n , Responder N_Port ID L _m
	Responder HG_ID L, Originator N_Port ID K _n	Originator N_Port ID K _n , Responder HG_ID L

Legend:

x = Originator N_Port ID

y = Responder N_Port ID

K_n = Originator N_Port IDL_m = Responder N_Port ID

K = Originator HG_ID

L = Responder HG_ID

24.7.4 Class

The address resolution is applicable to Class 1, 2 and 4, since ACK is used in the protocol. As there are no ACKs in Class 3, address resolution is not possible and HG_ID shall be used throughout the Exchange.

24.7.5 Class 4

The address resolution summarized in Table 220 shall be valid for Class 4 also, with the Class 4 address implication specified in 24.4.2.

24.8 Resource management

Since a Hunt Group may span multiple N_Ports, resources may be managed by the Common Controlling Entity and by N_Port. The sharing of resources may be implemented in various combinations. The arrangement of control and status block elements is only logical and does not necessarily imply any physical relationship.

A N_Port uses S_ID, D_ID, X_ID (OX_ID for Originator, RX_ID for Responder), SEQ_ID and Context bits to uniquely identify frames within a Sequence received by the N_Port. A N_Port is allowed to use any or all of the control entities for a given implementation.

When a N_Port supports multiple address identifiers or common address identifiers for S_ID, D_ID, or both, Sequence and frame identification alters N_Port behavior. The manner in which Hunt Group Identifiers are utilized within a N_Port leads to various logical models of N_Port resources.

Based on the organization of resources within a N_Port and CCE and the Class of Service, the Originator or Responder shall choose HG_ID or N_Port ID for an Exchange.

24.8.1 HG_ID use

For an Originator or a Responder to use a HG_ID, the following resource requirements shall be met:

- Class 1
 - Exchange resources (ESB, OX_ID) shall be Common Control Entity (CCE) managed
 - Sequence resources (SSB) may be N_Port managed or CCE managed
- Class 2
 - Exchange resources (ESB, OX_ID) shall be CCE managed
 - Sequence resources (SSB) shall be CCE managed
- Class 3
 - Exchange resources (ESB, OX_ID) shall be CCE managed
 - Sequence resources (SSB) shall be CCE managed
- Class 4

- Exchange resources (ESB, OX_ID) shall be CCE managed
- Sequence resources (SSB) shall be CCE managed

24.8.2 N_Port ID use

For an Originator or a Responder to use its N_Port ID, there are no additional requirements for resource organization. All resources may be CCE managed or N_Port managed or shared between them. If shared, the SSB, ESB, X_ID control hierarchy shall be retained.

24.9 Rotary Group

Rotary Group is another approach to increase the bandwidth and reduce latency but does not fall under the category of Hunt Group.

Existence of Rotary Group shall be transparent to compliant N_Ports or F_Ports.

24.9.1 Introduction

A Rotary Group is a collection of one or more N_Ports controlled by a Common Controlling Entity and a cooperative Fabric (See FC-SW - Rotary Groups) in which any member of the Rotary Group may respond in a transparent way to non-participating N_Ports. In this way, systems, which can handle more bandwidth than the attached N_Ports provide, can utilize multiple N_Port to Fabric connections to extend their achievable bandwidth.

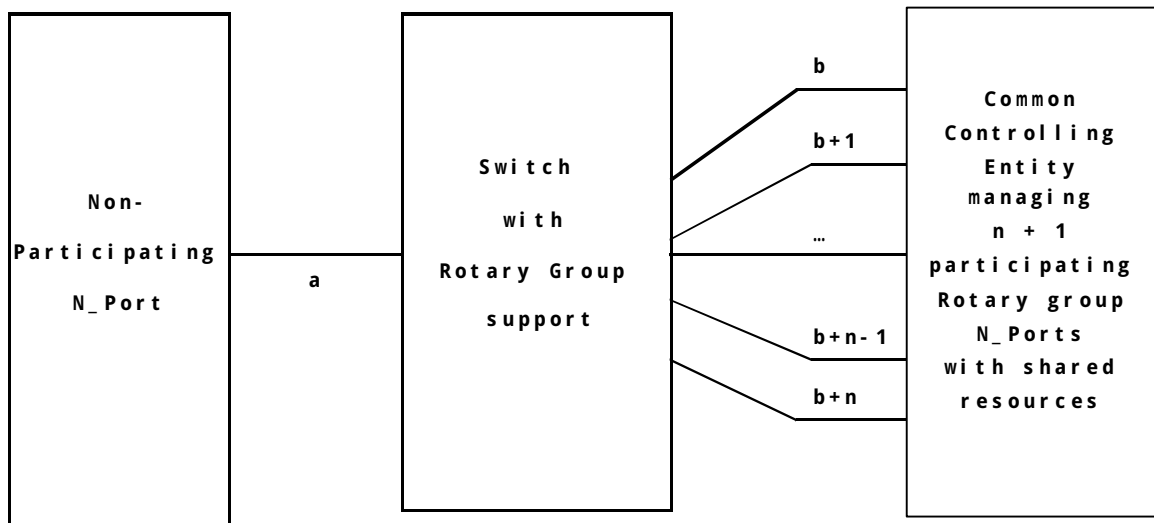
Rotary Group are formed and managed on a local basis, both in the participating N_Ports host (Common Controlling Entity) and in the Fabric, by the System administration.

The overall objective of a Rotary Group is that other N_Ports or Directory Services interact with the Group in a completely transparent manner (i.e., N_Ports which are not members of the Group do not know or care about the existence or non existence of the Rotary Group). Non-participating N_Ports address any N_Port of the Rotary group by its native address. No matter which N_Port of the Rotary Group responds, the Non-Participating N_Port receives acknowledgments and other communications associated with an established Exchanges with the Group as if it were communicating only with the native address of the Group with which it had originally established the exchange. (see Figure 66)

24.9.2 Function

In Class 1, a switch with Rotary Group support may be configured administratively to connect any connection request for a specific member of a Rotary group to any available member of that group. In Class 2 and 3, a Switch with Rotary Group support may be configured administratively to route any frame addressed for a specific member of a Rotary group to any available member of that group. (See FC-SW - Rotary Groups)

A Rotary Group member N_Port, under direction of its Common Controlling Entity, returns the address with which it was addressed as its source address (no matter which Rotary N_Port is actually comes from) in instances where it is the exchange responder. The Common Controlling Entity may use any available N_Port of its Rotary Group that is logged in to a destination N_Port for those exchanges where it is the exchange Originator. The Common Controlling Entity manages the resources (i.e., buffers, state tables, etc.) as a pool for all of the participating Rotary Group N_Ports.



Legend: a = address of the Non Participating N_Port,
b through b+n = addresses of n Participating N_Ports with one
Common Controlling Entity)

Figure 66 - Relationship of a Non-Participating to a Participating N_Port and switch

25 Multicast

25.1 Introduction

Multicast provides a service based on Fabric routing of Class 3 or Class 6 frames. For a multicast connection the Fabric routes frames from a source N_Port to many destination N_Ports.

25.2 Class 6 Multicast

Class 6 is a connection oriented multicast service for use with a Fabric. This is an optional service allowing the establishment of a Class 6 connection between a single source N_Port and multiple destination N_Ports. The Class 6 connection provides a multiplication of the bandwidth of the link between the source N_Port and the Fabric by the number of destination N_Ports.

A multicast Source establishes a Class 6 connection by generating a connect request frame to the Fabric using an Alias_ID to a multicast group. The Alias_ID is a 24-bit value placed in the D_ID field as defined in Clause 26. A multicast Source is the source N_Port for a Class 6 connection; a multicast group is a set of N_Ports that have subscribed to receive the same device data from a single multicast Source. The definition of the subscription and removal process is outside the bounds of this document. The Fabric recognizes the multicast Alias_ID, establishes connections between the multicast Source and the multicast group, and delivers the Class 6 connect request to the multicast group.

The Fabric routes all Link_Response frames (ACK, RJT, etc.) to a Multicast Server. The Multicast Server is responsible for collecting all responses and

Class 6 frames use SOF/EOF delimiters identical to Class 1 frames. Normal connection setup and removal follows the rules for Class 1 connections. Similarly, Class 6 connect request frames follow the rules for buffer-to-buffer flow control.

NOTE - A system implementation may provide less than reliable multicast service by implementing a single Sequence with infinite credit.

25.2.1 Class 6 Multicast Routing

The Fabric, based on the recognition that the D_ID of the transmitted frame is a multicast address does all routing of multicast frames. The exact frame that entered the Fabric is replicated to every destination N_Port indicated. The Fabric shall not alter the frame header or the frame contents in any manner during this replication.

Figure 67 illustrates the multicast routing. N_Ports B, C, D and E are members of a Multicast group. N_Port A is the multicast Source.

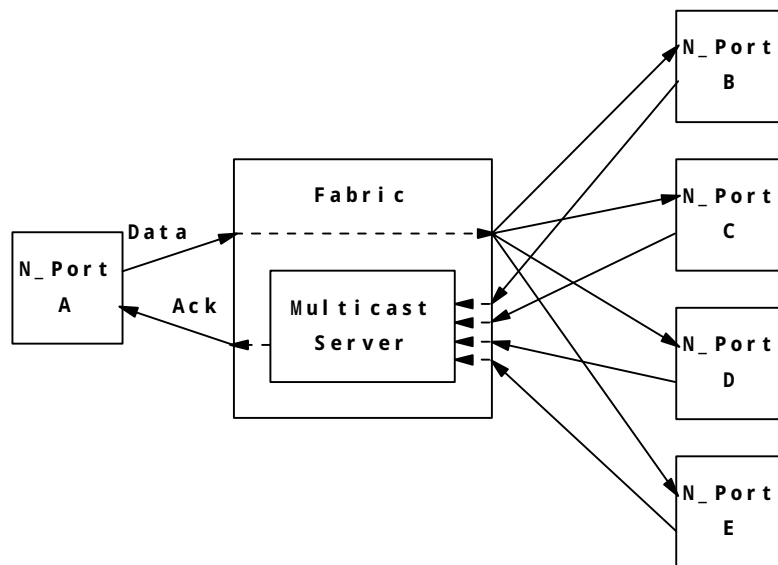


Figure 67 - Class 6 Multicast Routing

25.2.2 Class 6 Multicast Rules

- a) The multicast Source shall follow the uniqueness rules for OX_ID, SEQ_ID, and Sequence Count. X_ID interlock is not supported for Class 6 Exchanges. The multicast Source shall be the Connection Initiator for all multicast sequences. The multicast server shall set RX_ID for the Sequence to an appropriate value (established during Login).
- b) The D_ID in all device data frames shall be a multicast Alias_ID. The S_ID in all device data frames shall be the N_Port Identifier of the multicast Source and may also be an Alias_ID.
- c) The Fabric shall not alter the frame in any way. The Fabric shall simply replicate each frame to every group member.
- d) Sequence Initiative shall not be transferred.
- e) All Class 6 multicast frames shall be routed as uni-directional Dedicated Connections, regardless of the state of the Simplex bit in the CS_CTL Field (word 1 bit 31).
- f) If one or more F_Ports or destination N_Ports fail to return an R_RDY in response to the Class 6 connect request frame for a period greater than E_D_TOV, the Fabric shall initiate the Link Reset protocol for all participating N_Ports.
- g) If the Multicast Server receives identical, valid Link_Control frames from all multicast group members, it shall forward a single copy of that Link_Control frame to the multicast Source.
- h) If the Multicast Server receives at least one invalid Link_Control frame it shall cause the Fabric to initiate the Link Reset protocol for all participating N_Ports.
- i) If the Multicast Server receives at least two non-matching Link_Control frames it shall cause the Fabric to initiate the Link Reset protocol for all participating N_Ports.
- j) Class 1 service parameters, as specified during Login, shall be used to manage Class 6 multicast connections.
- k) The effect of preemption of the multicast Source or destination N_Port(s) shall result in all participating N_Ports being preempted.

25.2.3 Class 6 Multicast Server

To provide a single consistent response to the requesting N_Port, all destination N_Port responses shall be managed by a multicast server (at well-known address hex 'FF FF F5'). Based upon the type of responses returned from the destination N_Ports (i.e., ACK, P_RJT, or P_BUSY), the multicast server shall return a single ACK, busy, or reject (with a valid reason code) frame to the requesting N_Port as follows:

- a) Respond with an ACK only if all destination N_Ports respond with identical ACK frames before E_D_TOV has expired since the sending of the last multicast connect request or device data frame.
- b) Respond with a busy, with a "multicast busy" reason code (see Table 38), if all destination N_Ports respond with identical valid busy Link_Control Frames.
- c) Respond with a reject, with a "multicast error" reason code (see Table 40), if all destination N_Ports respond with identical valid reject Link_Control Frames.
- d) Initiate the Link Reset protocol for all participating N_Ports if all destination N_Ports do not respond with identical valid Link_Control Frames.
- e) Respond with a reject, with an EOFdt delimiter and a "multicast error terminate" reason code (see Table 40), if one or more destinations respond with an ACK, BSY, or RJT frame with an EOFdt delimiter.
- f) Initiate the Link Reset protocol for all participating N_Ports if one or more destination N_Ports respond with a Link Reset (LR), or if one or more destinations respond with a Link Reset Response (LRR).

Once a Class 6 connection is established, end-to-end credit is managed between the Connection Initiator and the multicast server.

25.2.4 Class 6 Multicast recovery

If the multicast Source receives from the Multicast Server a busy or a reject with an EOFdt delimiter, the multicast Source shall consider the multicast connection removed. Recovery is then performed at the Sequence or Exchange level. The mechanisms and conditions under which recovery may be performed are implementation-dependent.

25.3 Class 3 Multicast

Class 3 multicast provides a service based on Fabric routing of Class 3 frames. When the Fabric receives a Class 3 frame that is to be multicast, it causes the frame to be delivered to every destination in the Multicast Group. A Multicast Group is a list of N_Ports to which a frame being multicast is delivered. A Multicast Group is managed by the Alias Server, which handles the registration of N_Ports into a Multicast Group, and the de-registration of N_Ports from a Multicast Group. The Alias Server is not involved in the routing of multicast frames.

25.3.1 Registration and De-registration

The registration/de-registration by a N_Port as a member of a Multicast Group with the Alias Server is specified in clause 32. A N_Port may register itself as a member of one or more Multicast Groups. A N_Port may register a list of N_Ports as members of one or more Multicast Groups. The third party authorization for registration or de-registration is outside the scope of the document.

25.3.2 Multicast Routing

The Fabric, based on a recognition that the D_ID of the transmitted frame is an MG_ID Alias does all routing of multicast frames. The exact frame that entered the Fabric is replicated to every destination N_Port in the Multicast Group associated with the MG_ID of the frame. The

Fabric shall not alter the frame header or the frame contents in any manner during this replication.

The Sequence Initiator performs no special function to transmit a frame to be multicast other than use a D_ID indicating the MG_ID and the Multicast Group Service Parameters, rather than the Login Service Parameters. For example, the Receive Data Field Size for a multicast frame may be different than the Receive Data Field Size for a unicast frame.

If the Sequence Recipient is a member of a Multicast Group, it shall recognize the MG_ID as an Alias and accept the frame.

The Fabric shall reject Class1, Class 2, and Class 4 frames with a D_ID equal to a MG_ID.

Figure 68 illustrates the multicast routing. N_Port B, C, D, and E are members of a Multicast Group. N_Port A is the message sender, which may not be a member of the group. If it is a member, it may optionally receive the same message (see FC-GS.).

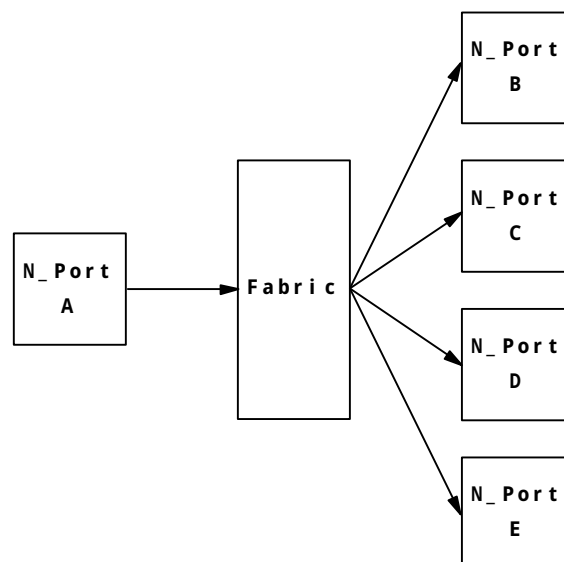


Figure 68 - Class 3 Multicast Routing

25.3.3 Class 3 Multicast rules

- The Sequence Initiator of the Multicast message shall follow the uniqueness rules for OX_ID, SEQ_ID, and Sequence Count. RX_ID for the Sequence shall be set to hex 'FF FF'.
- D_ID in all frames shall be the Alias for the Multicast Group. S_ID in all frames shall be the N_Port Identifier of the Sequence Initiator.
- The Fabric shall not alter the frame in any way. The Fabric shall simply replicate each frame to every group member.
- Sequence Initiative shall not be transferred.
- If any group member is not powered on or able to receive the frame for any other reason, the Fabric Controller shall not retransmit the frames.
- All multicast frames shall be routed in Class 3.

25.4 Broadcast

Broadcast is considered a simplification of multicast. No explicit registration or de-registration is required. Hex 'FF FF FF' is the Well-known address for Broadcast, which may be recognized by a N_Port.

NOTE - The Fabric may try to send to all possible N_Ports. However, the Fabric may not be able to deliver to all N_Ports for any number of reasons such as Class mismatch and N_Port not powered on.

The Broadcast Service Parameters shall be defined. All N_Ports that support the Broadcast shall support the Broadcast Service Parameters.

NOTE - The Broadcast Service Parameters shall be the least common denominator for all N_Ports. Normally the Broadcast Service Parameters may correspond to the default parameters.

25.5 Moviecast

Moviecast is an inherent feature of Multicast. An example for Class 3 Multicast: if a source N_Port is multicasting a movie to an Alias Group, another N_Port may join this Alias Group "in progress". That is, after the registration procedure is complete, the Fabric will start routing frames to the new N_Port. This is possible with Class 3 Multicast since a multicast group can be destination initiated. For a Class 6 Multicast the multicast group is source initiated and does not provide the direct capability for a N_Port to join a Multicast group "in progress". If intermix is supported a N_Port could request inclusion in a multicast group using a Class 2 or Class 3 frame, supporting an indirect destination initiated multicast group.

25.6 Other

Other forms of multicast are available in topology specific configurations. For examples, see FC-AL for a description of Selective Replicate to perform dynamic multicasting.

26 Aliases

26.1 Introduction

Alias is a group address recognized by a N_Port if the N_Port has registered with the Alias Server as a member of the group. Thus a N_Port may be able to recognize one or more Aliases in addition to its N_Port Identifier.

Presently defined Aliases are:

- Hunt Group Identifier
- Multicast Group Identifier.

A given N_Port may register with one or more Hunt Groups and/or with one or more Multicast Groups.

NOTE - Well-known addresses specified in Table 33 are allocated for special functions such as Directory Server. These special addresses are not to be confused with Aliases.

26.2 Alias Server

The Alias Server manages the registration and deregistration of Alias IDs for both Hunt Groups and Multicast Groups. The Alias Server is not involved in the routing of frames for any Group.

The Alias Server may be internal or external to the Fabric, but, in either case, it is addressed by means of the well-known address identifier, hex 'FF FF F8'. The following sections describe the registration/ deregistration process in more detail.

Authorization for Alias Server operations is provided.

26.3 Alias Service protocol

Alias registration and deregistration are managed through protocols containing a set of request/reply IUs supported by the Alias Server. These requests and replies use FC-FS constructs as defined in FC-GS.

26.4 Alias Routing

The Fabric, based on a recognition that the D_ID of the transmitted frame is an Alias_ID, does all routing of frames. For Multicast groups, the exact frame that entered the Fabric is replicated to every destination N_Port in the Multicast Group associated with the Alias_ID of the frame. The Fabric shall not alter the frame header or the frame contents in any manner during this replication. For Hunt Groups, the exact frame that entered the Fabric is routed to a single destination N_Port in the Hunt Group.

NOTE - The Fabric may assign Alias_IDs to easily partition Multicast Group Alias_IDs from Hunt Group Alias_IDs.

The Sequence Initiator performs no special function to transmit a frame other than to use a D_ID indicating the Alias_ID and to use the Alias Group Service Parameters, rather than the Login Service Parameters. For example, the Receive Data Field Size for a multicast frame may be different than the Receive Data Field Size for a unicast frame.

If the Sequence Recipient is a member of an Alias Group, it shall recognize the Alias_ID as an alias address identifier and accept the frame.

For Multicast Groups, the Fabric shall reject Class 1 and Class 2 frames with a D_ID equal to an Alias_ID.

26.5 Function Flow

Figure 69 illustrates the flow among the Originator N_Port, participating N_Ports, Alias Server, Directory Server, and Fabric Controller to create an Alias Group.

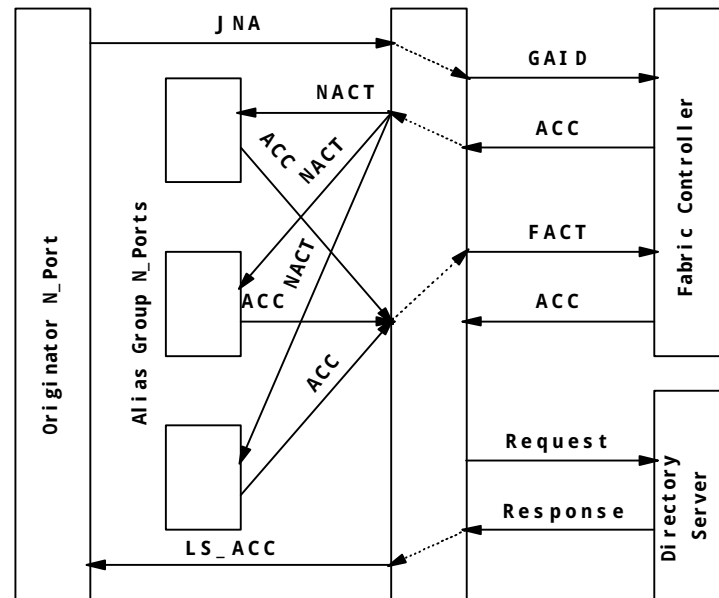


Figure 69 - Function Flow

26.6 IPA Considerations

26.6.1 Hunt Groups

For Hunt Groups, there are no IPA considerations, since there is a requirement that all members of a Hunt Group be within a single Common Controlling Entity, which cannot be spread across images. Therefore, the same IPA may be used no matter which N_Port receives the frame.

26.6.2 Multicast Groups

For Multicast Groups, the considerations for multicasting to N_Ports that require an Initial Process Associator are minimal as any multicasting behind the N_Port is handled internally. It is not necessary to know the IPA of each image behind a N_Port that belongs to a Multicast Group. Instead, if the Multicast Group requires an IPA, then the Sequence Initiator uses a Multicast Group IPA (MG_IPA). This MG_IPA is common for all images that belong to the same Multicast Group. An MG_IPA is set in the Association_Header as follows:

- The Responder Process Associator is set equal to the Alias_Qualifier for the Multicast Group.
- Bit 24 of the Association_Header Validity bits is set to binary '1'. This indicates an MG_IPA.

Internally, images shall register with their N_Ports to join Multicast Groups, they do not register with the Alias Server. The MG_IPA becomes an alias for the actual IPA of the image and is recognized as such by the N_Port as a result of the internal registration. When a frame is re-

ceived, the N_Port “multicasts” the payload to all images in the internal Multicast Group, based on the MG_IPA.

26.6.3 Broadcast

For Broadcast, there are no IPA considerations. Since the intent of Broadcast is to deliver to all possible recipients, it is the responsibility of the N_Port receiving a broadcast frame to broadcast the payload internally to all its images. Therefore, an IPA shall not be included in a frame being Broadcast.

27 Dedicated Simplex

27.1 Introduction

The Class 1 Dedicated Connection duplex, connection-oriented communication service or a Dedicated Simplex communication service. Thus, Class 1 service has two variants, duplex and simplex. This clause specifies Class 1 Dedicated Simplex.

Dedicated Simplex provides a means to combine the advantages of both the existing Class 1 and Class 2 in a single communications model as shown in Figure 70. It provides the high bandwidth flow-through of Class 1 Data frames for large block transfers with the ability of uncoupling the inbound and outbound data path (fibers) of Class 2. Dedicated Simplex allows a connection to exist on the outbound fibre to one N_Port while simultaneously having a connection on the inbound fibre with another N_Port. It increases the likelihood of simultaneous flow of data over both the outbound and the inbound fibre of a N_Port when compared with Class 1 dedicated connections.

27.2 Terms

The following terms are used with regard to Dedicated Simplex:

- Dedicated Simplex
- Class Specific Control (CS_CTL) field of Frame Header
- Simplex bit of CS_CTL field

27.3 Communications Model

The Dedicated Simplex communications model is essentially a unidirectional connection terms with ACKs transmitted in Class 2. More specifically, a unidirectional dedicated connection is established between the outbound fibre of the Connection_Initiator (CI) and the inbound fibre of the Connection_Recipient (CR). This is shown as paths 'M' and 'N' in Figure 70 and as paths '1' and '3' in Figure 71. Path '1' ('M') is a Dedicated Simplex connection from A (CI) to B (CR) and path '3' ('N') is a second (independent) Dedicated Simplex connection from C (CI) to A (CR). The in-bound fibers of the CIs and the out-bound fibers of the CRs are independent of the Dedicated Simplex connection. The CRs (destination N_Ports) will transmit all acknowledgments in Class 2 over paths 'm' and 'n' in Figure 70 and paths '2' and '4' in Figure 71.

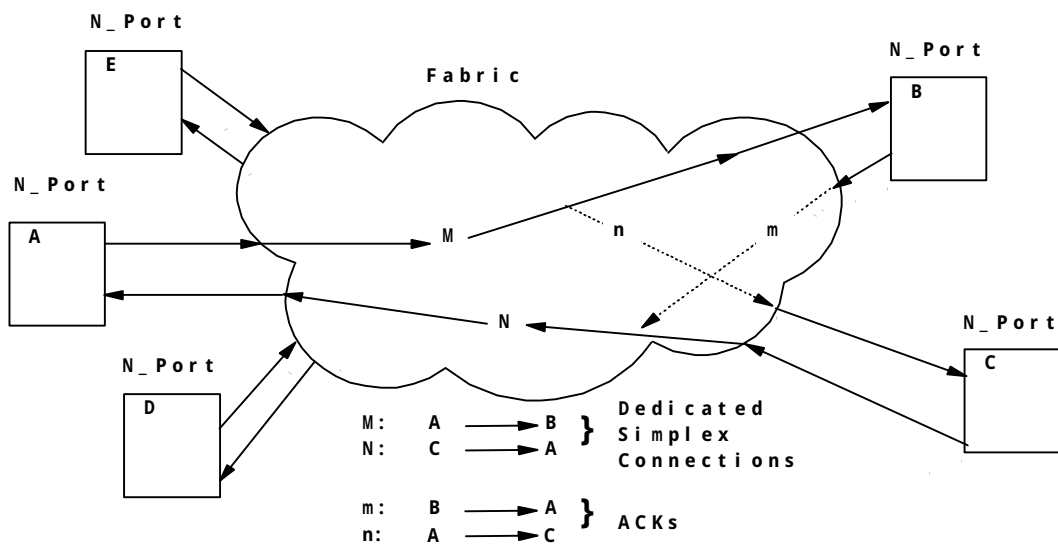


Figure 70 - Dedicated Simplex

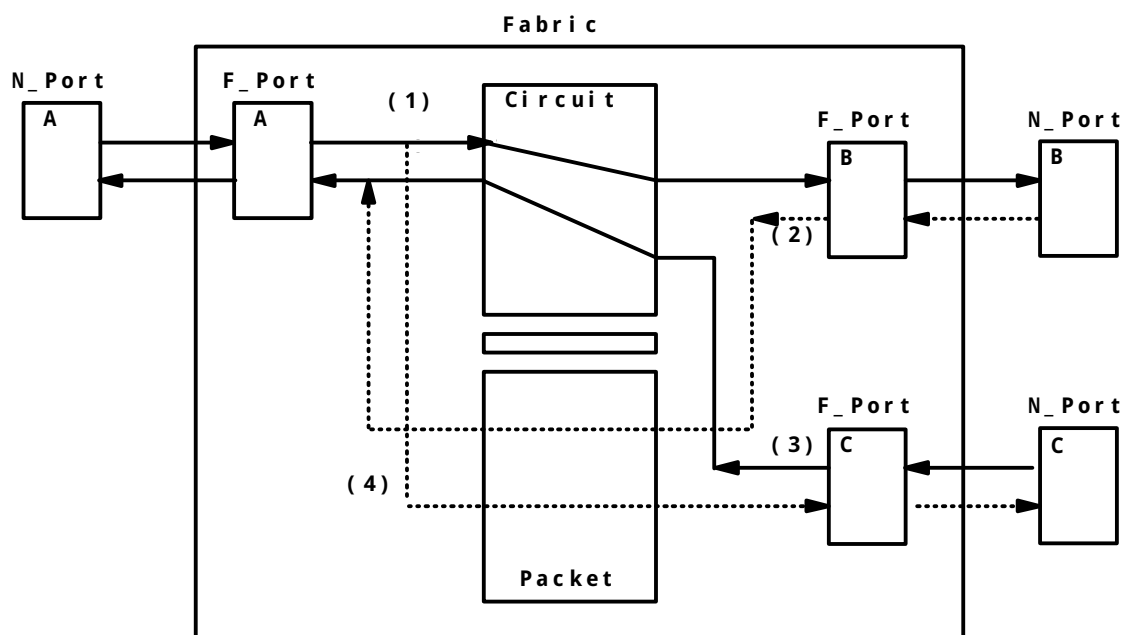


Figure 71 - Dedicated Simplex Fabric Circuit and Paths

Within a Dedicated Simplex connection, the CI shall transmit data frames in sequential order to the Fabric. The Fabric, if present, shall guarantee delivery of data frames within a Dedicated Simplex connection to the CR in the same order as they were received from the CI. The ACKs may be transmitted by the CR in any order and the Fabric, if present, may re-order them prior to delivery to the CI. Thus, Dedicated Simplex maintains in order delivery of data frames, while permitting the potential for out of order delivery of ACKs.

Dedicated Simplex is indicated by the CI through the use of a control bit in the frame header CS_CTL field (see 12.4.1) of the connect-request frame containing the SOFc1 delimiter.

A Dedicated Simplex connection is removed by the CI by transmitting an EOFdt delimiter on the last data frame (E_S=1) of the last sequence for this connection. The CR will respond by transmitting a Class 2 ACK with a EOFt delimiter containing the E_S=1.

When the Fabric receives a frame with the SOFc1 delimiter, it will check the Simplex bit of the CS_ CTL field of the frame header and take the following action.

- If the Simplex=0, the Fabric shall attempt to establish a Dedicated Duplex connection.
- If the Simplex=1, the Fabric shall attempt to establish a Dedicated Simplex connection.

On receipt of a frame containing a EOFdt delimiter and Simplex =1, the Fabric shall remove the Dedicated Simplex connection associated with the link on which the EOFdt was received (CI outbound fibre and the CR inbound fibre).

For Dedicated Simplex connections, the ACK, P_BSY, F_BSY, P_RJT and F_RJT frames are transmitted in Class 2 with EOFn or EOFt delimiters.

27.4 Applicability

Dedicated Simplex is an optional service that may be provided by the Fabric and N_Ports.

NOTE - Best efforts have been made to show applicability differences to FC-PH.

27.4.1 Permitted Functions

Dedicated Simplex may be used in conjunction with many of the functions defined herein. When other FC-FS functions are used, Dedicated Simplex will adhere to the rules defined in FC-FS except where specifically noted. The following list contains some of the FC-FS functions that may be used in conjunction with Dedicated Simplex:

- Sequence and Exchange management
- X_ID Interlock
- Stacked Connection request
- Camp-On (proposal)
- Intermix mode
- Continue Sequence Conditions
- Abort Sequence Conditions
- Login
- Class 1 End-to-End Credit rules
- Class 1 Receive Data Field Size
- Class 1 connect-request data field limits
- SOFc1, SOFi1, SOFn1, SOFn2
- EOFn, EOFt, EOFdt, EOFa, EOFni, EOFdti

27.4.2 Restricted Functions

Dedicated Simplex shall not use the following:

- SOFi3, SOFn3
- End_Connection bit protocol

27.4.3 Topology

Dedicated Simplex is applicable to the following topologies:

- Point-to-Point
- Fabric
- Arbitrated Loop

27.5 Requirements

27.5.1 Connection_Initiator N_Port

Dedicated Simplex requires that the CI N_Port has the capability to:

- support Class 1 and Class 2 delimiters
- support Intermix mode
- transmit the Class 1 connection request frame with Simplex=1.
- recognize Class 2 ACK (and other Link_ Control frames) for a sequence established in Dedicated Simplex

27.5.2 Connection_Recipient N_Port

Dedicated Simplex requires that the CR N_Port has the capability to:

- support Class 1 and Class 2 delimiters
- support Intermix mode
- support Dedicated Simplex connections

27.5.3 Fabric

Dedicated Simplex requires that the Fabric has the capability to:

- support Dedicated Simplex connections
- support Intermix mode
- support Class 1 and Class 2 delimiters

27.6 Connection Management

Class 1 shall use the Simplex bit (bit 31) of the CS_CTL field (word 1 bit 31-24) of the frame header to differentiate between Duplex and Dedicated Simplex (see 12.4.1.1).

27.7 Login

N_Ports and Fabrics shall indicate their ability to support Dedicated Simplex connections through Login. During Login, there is no additional requirement on any control bit values to support Dedicated Simplex function.

27.7.1 N_Port Class Service Parameters

The N_Port Class1 Service Parameters (see 17.7.4) Service Options field shall be used to indicate support for Dedicated Simplex connections (see 17.6.7.2).

27.7.2 F_Port Class Service Parameters

The F_Port Class1 Service Parameters (see 17.6.7) Service Options field shall be used to indicate support for Dedicated Simplex connections (see 17.6.7.2).

27.8 Delimiters

No new delimiters are required for Dedicated Simplex. Use of EOFdt with Dedicated Simplex differs (see 27.3 and 27.9.2) from Dedicated Connection.

27.8.1 Start of Frame

The CI shall use the SOFc1 delimiter together with the CS_CTL Simplex bit to establish a Dedicated Simplex connection. Once the Dedicated Simplex is established, the CI shall use the Class 1 delimiters SOFi1 and SOFn1 as defined for Class 1.

The CR and Fabric shall use the SOFn2 delimiter on Link_Control frames (ACK, P_BSY, F_BSY, P_RJT, and F_RJT) transmitted.

If a N_Port (F_Port) which does not support Dedicated Simplex connections receives a frame with a SOFc1 with the Simplex=1 (bit 31) of the CS_CTL field, that N_Port (F_Port) shall issue a Class 2 P_RJT (F_RJT) frame with a reason code of 'Dedicated Simplex not supported' (see Table 40).

27.8.2 End of Frame

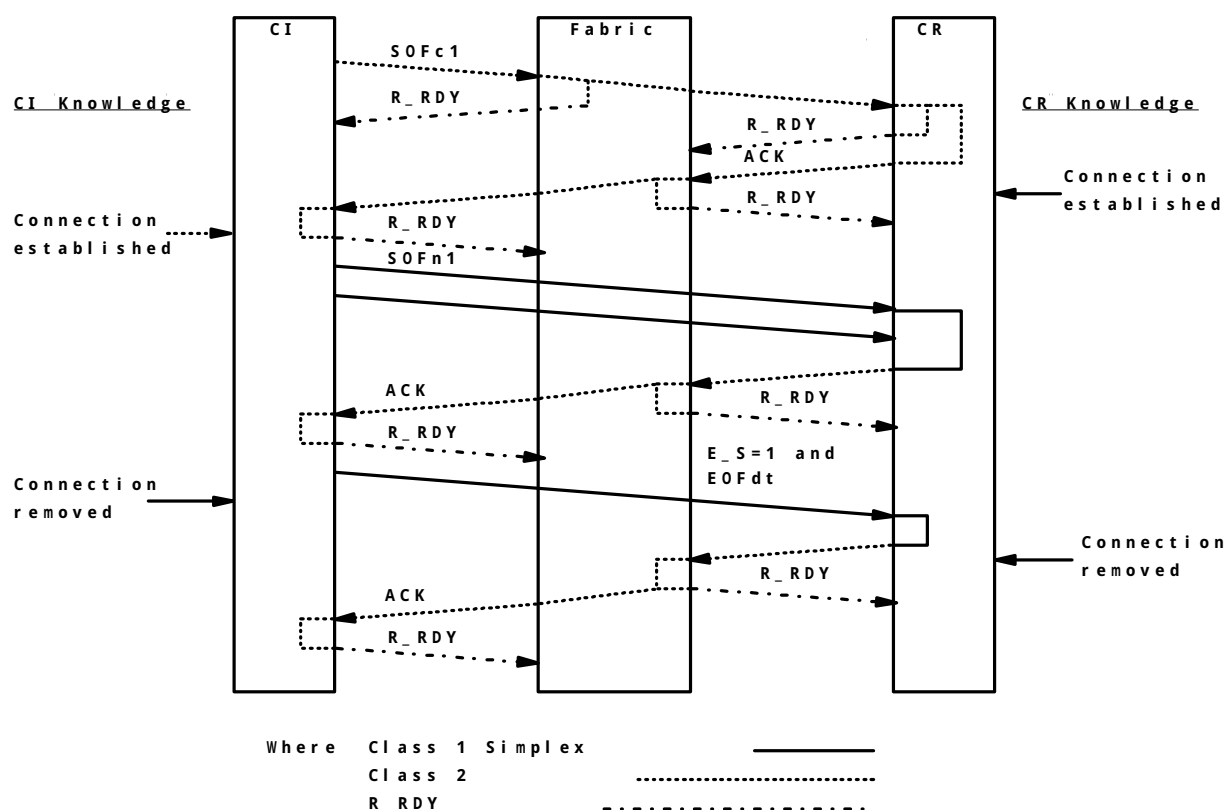
The CI shall use EOFn, EOFt, and EOFa delimiters. The CI shall also use the EOFdt delimiter to remove a Dedicated Simplex connection.

The CR shall use the EOFn or EOFt delimiter for all Link_Control reply frames transmitted within a Dedicated Simplex connection.

The EOFdti and EOFni rules defined in 11.6.2 shall apply to Dedicated Simplex.

27.9 Connections

Dedicated Simplex communication model is a connection-oriented service in one direction combined with connectionless ACK in the opposite direction as shown in Figure 72.



27.10 Rules

The following rules apply to Dedicated Simplex.

27.10.1 Restriction

The following restriction apply to the Dedicated Simplex class of service:

- A given N_Port shall not be simultaneously involved in a Class 1 Duplex connection and a Class 1 Dedicated Simplex connection. Class 1 Dedicated Duplex and Simplex connections are mutually exclusive. The exclusivity shall be enforced by the F _ Port.

27.10.2 Connection_Initiator N_Port

The following rules apply to the CI:

- The CI shall support transmission of Data_Frames with Class 1 delimiters and Link_Control frames with Class 2 delimiters.
- The CI shall support reception of Link_Control and Link_Data frame with Class 2 delimiters.
- The N_Port shall support Intermix mode.
- A N_Port shall initiate a Dedicated Simplex connection by transmitting a frame containing the SOFc1 delimiter with the CS_CTL Simplex bit set to a one.
- The CI shall adhere to all the rules of Class 1 connect-request frames with the exception that the Simplex=1 shall be used to establish the Dedicated Simplex connection.
- The CI shall consider the Dedicated Simplex connection established upon receipt of the ACK to the connection request frame.
- The CI shall not transmit any frames within the Dedicated Simplex connection following the connection request frame until the ACK frame is received or the appropriate time out period has expired as follows:
 - If Stacked Connect-request Transparent mode has been invoked, the time out period shall be $(CR_TOV) + (E_D_TOV)$
 - If Stacked Connect-request lock-down mode has been invoked, the time out period shall be $2(CR_TOV) + (E_D_TOV)$.
- All frames transmitted by the CI within the Dedicated Simplex connection following the connection request frame shall be transmitted with Class 1 SOFi1 or SOFn1 delimiters.
- Frames transmitted by the CI within a Dedicated Simplex connection shall adhere to the Class 1 credit management rules.
- The Sequence Count rules of 18.3.6 shall apply to Dedicated Simplex connections.
- Following transmission of a Class 1 Dedicated Simplex connection request frame, if the CI receives a connect-request frame for a Duplex Class 1 prior to the ACK to the connect-request and if Stacked Connects has not been invoked, the CI shall re-queue the connect-request frame for retransmission at a latter time.
- Following transmission of a Class 1 Dedicated Simplex connect-request, if the CI receives a connect-request frame for a Class 1 Dedicated Simplex prior to the ACK to the connect-request, the CI shall reply with an ACK. From the perspective of this CI, it will be engaged in two Dedicated Simplex connections.
- A N_Port may transmit a Dedicated Simplex connect-request with Stacked Connect-request invoked while engaged in a Dedicated Simplex connection which it initiated.
- A N_Port may transmit a Dedicated Simplex connect-request with Stacked Connect-request invoked while engaged in a Dedicated Duplex connection.

- The CI shall initiate the removal of a Dedicated Simplex connection by transmitting a frame containing the E_S=1 and an EOFdt delimiter.

27.10.3 Connection_Recipient N_Port

The following rules apply to the CR:

- The CR shall support reception of Data frames with Class 1 delimiters and Link_Control with Class 2 delimiters.
- The CR shall support transmission of Link_Control and Link_Data frame with Class 2 delimiters.
- The CR may reply to a connection request with either P_BSY or P_RJT transmitted in Class 2.
- The CR completes the establishment of a Dedicated Simplex connection by transmitting an ACK in Class 2 in reply to the connection request frame. Following transmission of the ACK frame in reply to the connection request frame, the CR shall consider the Dedicated Simplex connection to have been established.
- The CR shall respond to all frames received within the Dedicated Simplex connection Class 2 with SOFn2 and EOFn delimiter except a frame with E_S=1. Class 2 shall be used to transmit any Link_Control frames (ACK, P_BSY, or P_RJT) in response to the Connect frame.
- In response to the Data_Frame containing E_S=1 and with an EOFdt delimiter, the CR shall transmit an ACK frame with E_S=1 and an EOFt delimiter.
- The CR shall consider the Dedicated Simplex connection to have been removed upon receipt of a frame from the CI containing the EOFdt delimiter.
- If a CR which does not support Dedicated Simplex receives a Dedicated Simplex connect-request frame, that N_Port shall issue a Class 2 P_RJT with reason code of 'Class not supported'.

27.10.4 Fabric

Fabric shall obey the following rules:

- When the Fabric receives a Dedicated Simplex connect-request frame with a destination N_Port which is not engaged in a Class 1 Duplex connection or the inbound fibre of that destination N_Port is not engaged in a Class 1 Dedicated Simplex connection, the Fabric shall attempt to establish a Dedicated Simplex connection with the destination N_Port.
- When the Fabric receives a Dedicated Simplex connect-request frame with Stacked Connect-request = 1, the Fabric shall follow the Stacked Connection rules as defined in 22.5.2 and clause 30.
- When the Fabric receives a Dedicated Simplex connect-request frame with COR=1, the Fabric shall follow the Camp-On rules as defined in clause 29.
- The Fabric shall respond to all frames received within the Dedicated Simplex connection in Class 2 with SOFn2 and EOFt delimiters. Class 2 will be used to transmit any Link_Control frames (F_BSY, or F_RJT) in response to the Connect frame.
- Within a Dedicated Simplex connection, if the Fabric receives an END Link_Control command, it shall remove that connection.
- If a Fabric which does not support Dedicated Simplex receives a Dedicated Simplex connect-request frame, that F_Port shall issue a Class 2 F_RJT with reason code of 'Class not supported'.

27.10.5 F_Port

F_Port shall obey the following rules:

As specified in 22.3.1, the F_Port acts as the control point in resolving collisions or accepting the connect-requests.

- The F_Port shall resolve the collision, if any, between simultaneous connect-requests from the Link side and the internal side. These requests may be for Dedicated Simplex or Dedicated Duplex connections.
 - If these simultaneous requests relate to the same fibre pair or an individual fibre, then a collision results and the request from internal side shall always be accepted by the F_Port and the Link side request shall be discarded.
 - If these simultaneous requests relate to different fibers, then no collision results and the F_Port may be able to accept both the requests.
- After the F_Port has accepted a Dedicated Simplex connect-request from the Link side, only its inbound fibre is considered busy and the outbound fibre is available for Dedicated Simplex connect-request from internal side.

27.10.6 ACK Processing and Routing

Following rules apply to ACK processing and routing:

- With the Dedicated Simplex service, the CR shall transmit ACKs in Class 2.
- Any of the ACK forms (ACK_0 or ACK_1) may be used. The History protocol for ACKs as defined in 14.3.2.2 may be used.
- The CI may invoke the credit window rules defined in 18.3.9, item 4 to detect missing ACKs.

NOTE - The Fabric intermixes ACKs in place of Idles. Intermixing ACKs has an additional Fabric requirement, which is different from Intermix.

27.11 Error Recovery

27.11.1 Connection_Initiator

The CI performs the following for error recovery:

- The CI may invoke either the ABTS or ABTX protocol to abnormally terminate a Sequence or Exchange on detection of an abnormal condition.
- If the state of the connection is unknown due to some abnormal condition, the CI may transmit a END Link_Control command, NOP, ABTS, or ABTX frame containing a SOFx1 delimiter (as appropriate) and an EOFdt delimiter.
- E_D_TOV and R_A_TOV timer rules apply to frame transmitted within a Dedicated Simplex connection.

27.11.2 Connection_Recipient

The CR performs the following for error recovery:

- If the CR detects an error in the Sequence_Count of the frames received within a Dedicated Simplex connection, it shall invoke the Abort Sequence protocol through use of the Abort Sequence Conditions in the F_CTL of the ACK frame.
- E_D_TOV and R_A_TOV timer rules apply to frame transmitted within a Dedicated Simplex connection.

- If the state of the connection is unknown to the CR, the CR may transmit an END Link_Control command with an EOFdt delimiter.

28 Class 4 – Fractional

28.1 Introduction

Class 4 Service is a connection oriented service for use with a Fabric. This is an optional service allowing the establishment of a Class 4 circuit between a pair of communicating N_Ports. The Class 4 circuit provides a fractional allocation of the resources of the path between the N_Ports.

A Class 4 circuit is bi-directional with one Virtual Circuit (VC) operational in each direction. Each VC may have different Quality of Service (QoS) parameters. These QoS parameters include guaranteed bandwidth and latency. A N_Port may have up to 254 coexistent Class 4 circuits with the same or different N_Ports.

Class 4 operation is separated into two parts, circuit setup and circuit activation (see Figure 73). During the setup process, QoS parameters for both VCs are provided by the circuit initiator N_Port (CTI) and granted by the Fabric and the circuit recipient N_Port (CTR). The Fabric establishes a Class 4 circuit, in the Dormant state when the Fabric grants the setup. The Class 4 circuit setup process incurs a one-time round trip delay.

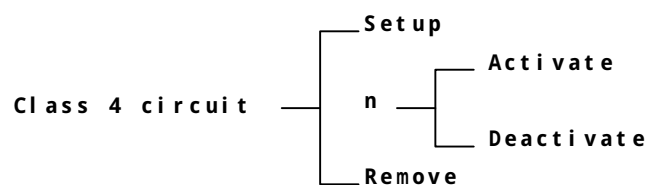


Figure 73 - Class 4 Circuit hierarchy

During the circuit activation process, both the Dormant VCs of the Class 4 circuit are put in the Live state to allow communication between the associated CTI and CTR.

The resources of an established Class 4 circuit are available on call so participating N_Ports can activate the circuit and deactivate it repeatedly (n) without paying any round trip delay penalty. Activation of a Class 4 circuit places it in the Live state and deactivation places the Class 4 circuit in the Dormant state.

A QoS Facilitator (QoSF) function is provided within the Fabric to establish and maintain QoS parameters for the Dormant VCs and manage them when the VCs are alive.

The Fabric regulates available bandwidth per Live VC by controlling the outbound frame flow out of the associated N_Ports. This is achieved by controlling credit, between the N_Port transmitter and the F_Port receiver, on a per VC basis, with the F_Port regulating credit to ensure the QoS parameters. This level of control permits congestion management for a Fabric.

Class 4 frames are never busied, since F_BSY and P_BSY are illegal responses to Data and Link_Control frames in Class 4.

In-order delivery of frames within a Class 4 circuit is guaranteed in Class 4 Service.

In Class 4, the Fabric is responsible for multiplexing frame belonging to different Live VCs between the same or different N_Port pairs.

28.1.1 Topology

Class 4 is applicable only to topologies where single N_Ports are attached to F_Ports.

28.2 Communication Model

Class 4 service provides a guaranteed bandwidth between communicating N_Ports. The Extended Link Services associated with Class 4 sets up a pair of unidirectional VCs between communicating N_Ports (see Figure 74) to guarantee bandwidth and frame transmission delay.

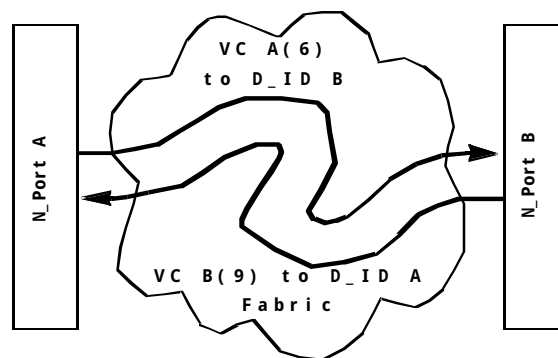


Figure 74 - Basic Class 4 circuit

Although there is a fixed route (path) through the Fabric for each VC between the communicating N_Ports and the Fabric regulates the VC_Credit, the path bandwidth may not be fully dedicated.

The Fabric regulates the N_Ports for frame transmission per VC. However, the communicating N_Ports may be unable to use the full requested bandwidth. This happens if there are no frames to transmit for the VC, if VC_Credit is unavailable or if EE_C4_Credit is unavailable. As a result that amount of bandwidth may be lost to the N_Port. It is possible that this lost bandwidth can be compensated for later, since excess bandwidth is allotted to Live Class 4 circuits and Class 2 or 3 traffic.

Multiple pairs of VCs may exist between one N_Port and one or more other N_Ports (see Figure 75). The amount of bandwidth needed for each VC is requested when the Class 4 circuit is setup.

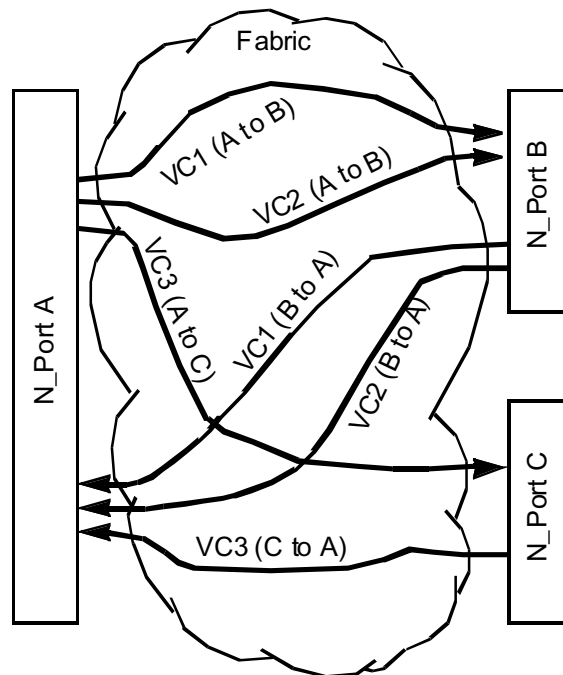


Figure 75 - Class 4 circuits – Example

28.2.1 In-order delivery

In-order delivery of Class 4 frames sent, on a Class 4 circuit, is guaranteed. However, no guarantees exist for frames sent on separate Class 4 circuits.

28.2.2 Guaranteed delivery

Class 4 provides acknowledged delivery of frames. This acknowledgment comes in the form of ACK, F_RJT or P_RJT Link_Control frames.

28.2.3 Fractional bandwidth management

For each Live VC the Fabric ensures the guaranteed bandwidth from a source N_Port to a destination N_Port. A Quality of Service Facilitator (QoSF) manages the guaranteed bandwidth. The Fabric enables a source N_Port to transmit frames in a timely manner.

In Class 4 the source N_Ports transmit frames into the Fabric when credit is available and the Fabric regulates available credit per VC to the source N_Port.

The Fabric routes frames based on the Virtual Circuit Identifier (VC_ID) field embedded in each Class 4 frame header (see Figure 75).

The Fabric shall make unused bandwidth available for Live Class 4 circuits, Class 2 and Class 3 frames. Unused bandwidth available for Live Class 4 circuits, shall be allocated approximately in proportion to their requested maximum bandwidth requirements.

The Fabric may ensure that a N_Port does not transmit frames on a VC in excess of the requested maximum bandwidth limit.

28.2.4 Class 4 circuit

A Class 4 circuit requires that two unidirectional VCs be setup through the Fabric before Class 4 frames may be transmitted between the N_Ports. Figure 74 shows a Class 4 circuit between two N_Ports 'A' and 'B', one VC (VC A(6)) from port 'A' to 'B' and one VC (VC B(9)) from port 'B' to 'A'. Each VC has associated with it a set of QoS parameters and a path (see 28.2.4.1).

Table 221 illustrates how a Class 4 circuit can be setup between two N_Ports 'A' and 'B'. N_Port 'A' supports a data rate of 100 MB/s, and 'B' a data rate of 25 MB/s. The VC from 'A' to 'B' (VC(1)) has a guaranteed minimum bandwidth of 15 MB/s and a maximum latency of 500 μ s. The VC from 'B' to 'A' (VC(6)) has a guaranteed minimum bandwidth of 3 MB/s and a maximum latency of 2 000 μ s. N_Port 'A' is left with an excess outbound bandwidth of 85 MB/s and an excess inbound bandwidth of 97 MB/s. The excess bandwidth for N_Port 'B' is 22 MB/s in the outbound direction and 10 MB/s in the inbound direction. Excess bandwidth may be used to enable transfers of more than the guaranteed minimum on the Class 4 circuit. The excess may be used for other Class 4 circuits or may be used for Class 2 or 3 Service.

Table 221 - Bandwidth Allocation Example					
Class 4 Connection		N_Port		Rate (MB/s)	Max. Delay (μ s)
		ID	VC_ID		
A $\Rightarrow$ B		A	(1)	15	500
A $\Leftarrow$ B		B	(6)	3	2 000
Free	A $\Rightarrow$	A	n/a	85	n/a
	A $\Leftarrow$		n/a	97	
	B $\Rightarrow$	B	n/a	22	n/a
	B $\Leftarrow$		n/a	10	

Class 4 circuits and their VCs exist in one of four states:

- Nonexisting,
- Pending,
- Dormant, or
- Live.

A Ports state diagram for a Class 4 circuit is shown in Figure 76. The states and state transitions are described in the sub-clauses that follow.

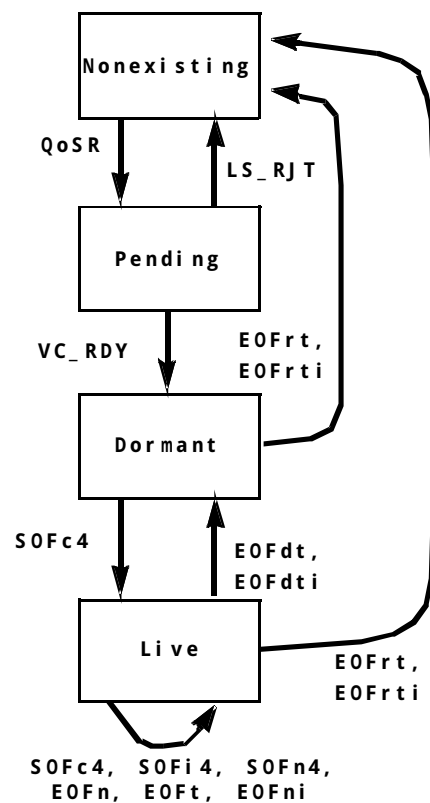


Figure 76 - Class 4 circuit - Port level state diagram

Ports that support Class 4 shall, regardless of the state of Class 4 circuits, support Intermix of Class 2 and Class 3 frames based on the rules of those two Classes of service.

28.2.4.1 Class 4 circuit – Setup

A N_Port requests setup of a Class 4 circuit to another N_Port by sending a Quality of Service request (QoS R) Extended Link Service command to the QoSF. The requesting N_Port becomes the circuit initiator (CTI) and the target N_Port the circuit recipient (CTR). The QoSR identifies the communicating N_Ports and the required QoS parameters.

The QoSR is granted if the QoSF can establish a path between the communicating N_Ports with the requested QoS parameters and if the CTR also can accept the QoS parameters (see 15.5.36.1). If the QoSR is accepted then a Class 4 circuit is established in the Pending state when the ACC Sequence is returned to the CTI (see Figure 77 (a)).

The QoSR is rejected if the QoSF is unable to establish a path between the communicating N_Ports with the requested QoS parameters. The CTR is never queried if the QoSF is unable to establish a path between the communicating N_Ports (see Figure 77 (b)).

The QoSR is rejected if the QoSF can establish a path between the communicating N_Ports with the requested QoS parameters but the CTR is unable or unwilling to accept the QoSR (see Figure 77 (c)).

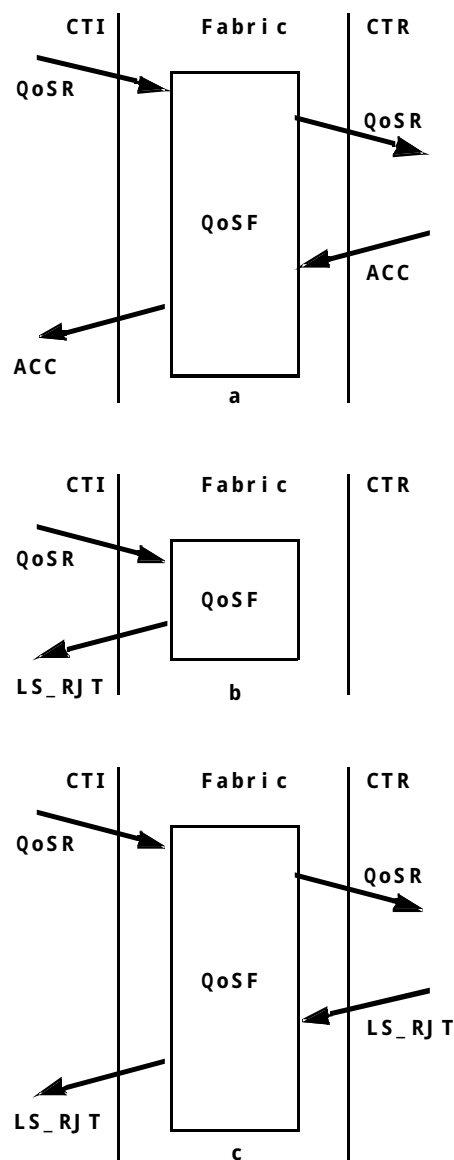


Figure 77 - Class 4 circuit – Setup

The CTI selects the QoS parameters for both VCs in the Class 4 circuit.

The QoSF receives the request from the CTI (S_ID = A and D_ID = hex 'FF FF F9'), validates it, sets up the path from the CTI to the CTR placing the Class 4 circuit in the Dormant state. It then sends a new request with the QoSF as the S_ID to the CTR (S_ID = hex 'FF FF F9' and D_ID = B).

If there is a conflict in parameters or insufficient resources, the request shall be rejected. If there is any error in the original transaction, the QoSF shall reject the request (see Figure 77 (b)).

General conditions associated with a Class 4 circuit are:

- A N_Port may establish up to 254 coexistent Class 4 circuits, to the same or different N_Ports. The total bandwidth allocated shall not exceed the total bandwidth available for the N_Port media rate of the port managing the Class 4 circuits.
- The bandwidth of any Class 4 circuit is restricted to the maximum bandwidth of the slowest link in the Class 4 circuit path.
- Once setup by the Fabric, a Class 4 circuit is retained and guaranteed by the Fabric. This Fabric service guarantees the requested minimum bandwidth between the communicating N_Ports once the Class 4 circuit is activated.
- The Quality of Service Request (QoSR) Extended Link Service shall be used by a CTI to provide its VC_ID (CIVC_ID) and identify the CTR along with QoS parameters.

The CTI actions are:

- The CTI shall set the VC_Credit_Limit, VC_Credit, EE_C4_Credit_Limit, and EE_C4_Credit to 0 before it sends the QoSR to the QoS. The CTI shall enter the Pending state for the VC as it sends the QoSR to the QoS.
 - The CTI shall release all resources assigned to the Class 4 circuit if an LS_RJT is received from the QoS.
 - The CTI shall, when it recognizes the ACC Sequence to the QoSR, set the Live_VC_Credit_Limit to the value obtained from the ACC Sequence. The Originating N_Port shall also set the EE_C4_Credit_Limit and the EE_C4_Credit to the value obtained from the ACC Sequence. CTI VC_Credit remains at 0 until a VC_RDY is received on the new Class 4 circuit.
 - The CTI shall use the address identifier in the CTR address identifier field in the ACC Sequence, as the D_ID for all Class 4 traffic on the new Class 4 circuit.
- NOTE - The CTR may elect to use any address identifier, alias or native, assigned to the CTR on the Class 4 circuit.
- The CTI shall enter the Dormant state when it receives the first VC_RDY for the VC. It shall also set the VC_Credit_Limit to 1 and VC_Credit to 1 when the VC_RDY is received.

The CTR actions are:

- The CTR shall enter the Pending state before it transmits an ACC Sequence to the QoS. Before the CTR transmits an ACC Sequence the CTR shall:
 - set the VC_Credit_Limit to 0;
 - set the Live_VC_Credit_Limit to the value obtained from the QoS Sequence;
 - set the EE_C4_Credit_Limit to the value obtained from the QoS Sequence;
 - set the EE_C4_Credit to the value obtained from the QoS Sequence; and
 - set VC_Credit to 0.
- If the CTR is unable to accept the QoS from the QoS it shall return an LS_RJT to the QoS and enter the Nonexisting state.
- The CTR shall use the address identifier in the CTI address identifier field in the QoS Sequence, as the D_ID for all Class 4 traffic on the new Class 4 circuit.

NOTE - The CTI may elect to use any address identifier, alias or native, assigned to the CTI on the Class 4 circuit.

- The CTR shall enter the Dormant state when it receives the first VC_RDY for the VC. It shall also set the VC_Credit_Limit to 1 and VC_Credit to 1 when the VC_RDY is received.

The QoS actions are:

- The QoSF shall set the F_Port, linked to the CTI, to the Pending state, once it recognizes the QoSR from the CTI.
- The QoSF shall set the F_Port, linked to the CTI, to the Dormant state when it completes the QoSR transaction with the CTI. The QoSR transaction is complete in Class 1,2 or 4 when the QoSF receives the ACK to the ACC Sequence, sent to the CTI. In Class 3 the QoSR transaction is complete when the QoSF transmits the ACC Sequence. The F_Port shall transmit one or more VC_RDY when the F_Port enters the Dormant state.
- The QoSF shall set the F_Port, linked to the CTR, to the Pending state, as it sends the QoSR to the CTR.
- The QoSF shall set the F_Port, linked to the CTR, to the Dormant state when it completes the QoSR transaction with the CTI. The QoSR transaction is complete in Class 1,2 or 4 when the QoSF transmits the ACK to the ACC Sequence, received from the CTR. In Class 3 the QoSR transaction is complete when the ACC Sequence is transmitted by the CTR. The F_Port, linked to the CTR, shall transmit one or more VC_RDY when the F_Port enters the Dormant state.
- The QoSF shall send the CTI an LS_RJT if an LS_RJT is received from the CTR.
- The QoSF shall set the F_Port, linked to the CTR, to the Nonexisting state, once it recognizes the LS_RJT from the CTR.
- The QoSF shall send the CTI an LS_RJT if it is unable to accept the CIs QoSR request.
- The QoSF shall set the F_Port, linked to the CTI, to the Nonexisting state, before it sends an LS_RJT to the CTI.
- The QoSF shall send ACC to the CTI only if an ACC has been received from the CTR.
- The QoSF shall remove the Class 4 circuit (see 28.2.4.4) if the QoSR transaction fails to complete.

The Class 4 circuit may be activated once it is setup (see 28.2.4.1).

28.2.4.2 Class 4 circuit – Activation

A Class 4 circuit activation request is used to activate the two VCs for the Class 4 circuit after which Class 4 frames may be transmitted between the communicating N_Ports. Either N_port may activate the Class 4 circuit. Once activated, the Class 4 circuit may be deactivated with a deactivation request or removed entirely with a removal request by either of the communicating N_Port.

A Class 4 circuit, in the Dormant state, is activated, enters the Live state, when a circuit activation request, an SOFc4 delimited frame, is sent or received (see Figure 78).

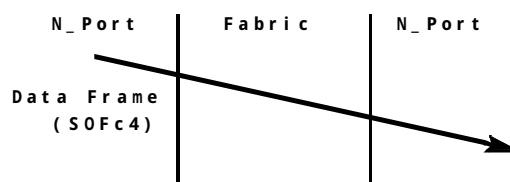


Figure 78 - Class 4 circuit – Activation

Bandwidth utilization is being monitored by the Fabric and is used to regulate VC_RDY transmission. Once the Class 4 circuit activation frame has been transmitted Class 4 frame flow is possible, only limited by available EE_C4_Credit and VC_Credit.

The Fabric manages buffer-to-buffer flow control and the Fabric regulates frame transmission from each source N_Port. The Fabric regulates each VC independently with the Virtual Circuit Ready (VC_RDY) Primitive Signal. Fabric flow control is accomplished using the VC_RDY Primitive Signal for each Live Class 4 circuit (see Figure 79).

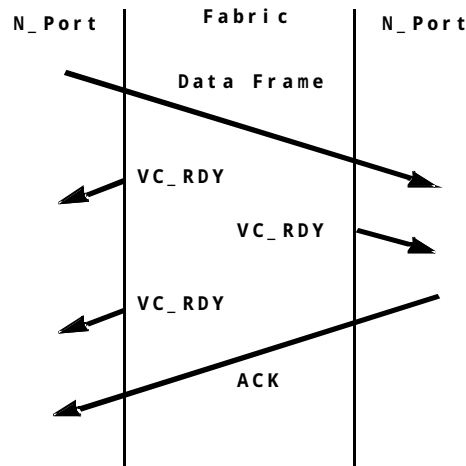


Figure 79 - Class 4 - Example of frame flow

Class 4 end-to-end flow control is managed by the communicating N_Ports using ACK frames.

28.2.4.3 Class 4 circuit – Deactivation

Once a Class 4 circuit is setup, it may be activated and deactivated one or more times.

The Sequence Initiator indicates in the frame header, on the last frame of a Sequence, that the Live Class 4 circuit is to be deactivated ($E_C = 1$, $R_C = 0$) (see 28.3). The ACK response then carries the EOFdt delimiter (see Figure 80).

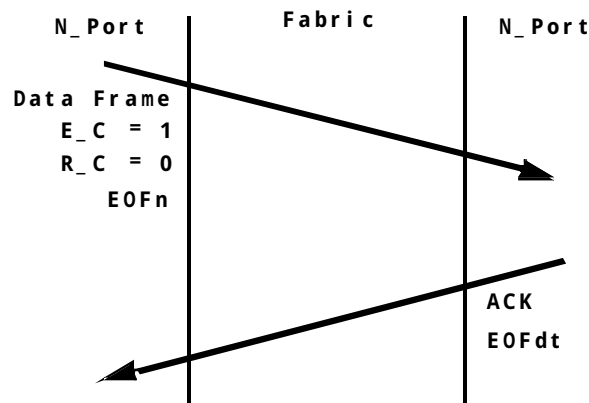


Figure 80 - Class 4 circuit - CTI or CTR initiated

The request to deactivate a Class 4 circuit may be sent by either of the communicating N_Ports.

A Class 4 circuit, in the Live state, is deactivated, and enters the Dormant state, when a circuit deactivation request, an EOFdt or EOFdti delimited frame, is sent or received (see Figure 80).

A deactivated Class 4 circuit may later be reactivated to the Live state (see 28.2.4.2).

The Fabric shall issue two END Link_Control commands, one on either VC, when the QoSF determines for any reason that a Class 4 circuit should be deactivated. The Fabric shall use the addresses of the communicating N_Ports, as the S_ID and D_ID, in the generated END Link_Control commands, and the VC_ID shall match with the S_ID field. The END Link_Control command frame delimiters shall be EOFdt and SOFn4 (see Figure 81). The Fabric shall use this mechanism to deactivate Class 4 circuits when the N_Port or F_Port enters the Link Reset state.

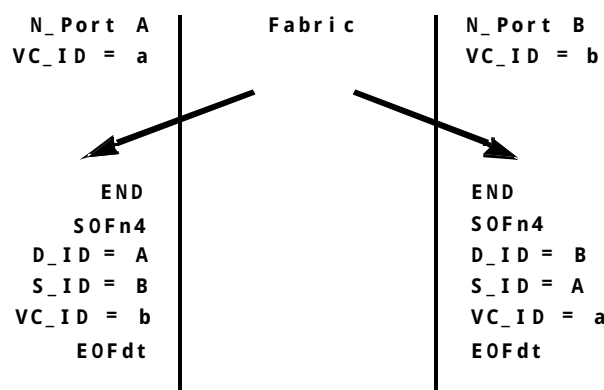


Figure 81 - Class 4 circuit - Fabric initiated deactivation

When a Class 4 circuit is deactivated, an N_Port shall reclaim all EE_C4_Credit when the N_Port enters the Dormant state (i.e., set EE_C4_Credit equal to EE_C4_Credit_Limit when using the count down model for credit management).

28.2.4.4 Class 4 circuit – Removal

From the Live state, a Class 4 circuit may be removed, i.e., it may enter the Nonexisting state. The Sequence initiator indicates, in the frame header, on the last frame of a Sequence, that the Live Class 4 circuit is to be removed ($E_C = 1$, $R_C = 1$) (see 28.3). The ACK response then carries EOFrt delimiter (see Figure 82).

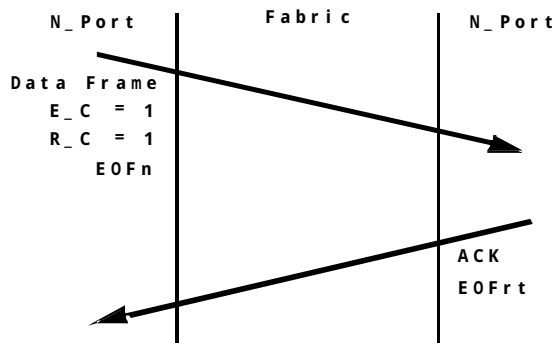


Figure 82 - Class 4 circuit - CTI or CTR initiated removal

The request to remove a Live Class 4 circuit may be sent by either of the communicating N_Ports.

A Class 4 circuit, in the Live or Dormant state, is removed, and enters the Nonexisting state, when a circuit removal request, an EOFrt or EOFrti delimited frame, is received (see Figure 81). A Class 4 circuit, in the Live state, is removed, and enters the Nonexisting state, when a circuit removal request, an EOFrt or EOFrti delimited frame, is sent (see Figure 81).

The two N_Ports shall not reactivate the Class 4 circuit when the Class 4 circuit has been removed. However, the VC_IDs associated with the removed Class 4 circuit (CIVC_ID and CRVC_ID) may be reused. The QoSF shall return any resources it has allocated to a common pool for use by other Live Class 4 circuits or other Classes of Service when the Class 4 circuit has been removed.

Another Class 4 circuit may be used for future Class 4 communication between the N_Ports or the two N_Ports may setup a new Class 4 circuit.

If required for any reason, the Fabric may remove a Class 4 circuit from either the Live state or the Dormant state. To achieve this the Fabric shall issue two END Link_Control commands, one on either VC. The Fabric shall use the addresses of the communicating N_Ports, as the S_ID and D_ID, in the generated END Link_Control commands, and the VC_ID shall match with the S_ID field. The END Link_Control command frame delimiters shall be EOFrt and SOFn4 (see Figure 83). If the Class 4 circuit is alive, then the Fabric shall first deactivate the Class 4 circuit before removing the circuit (see 28.2.4.3).

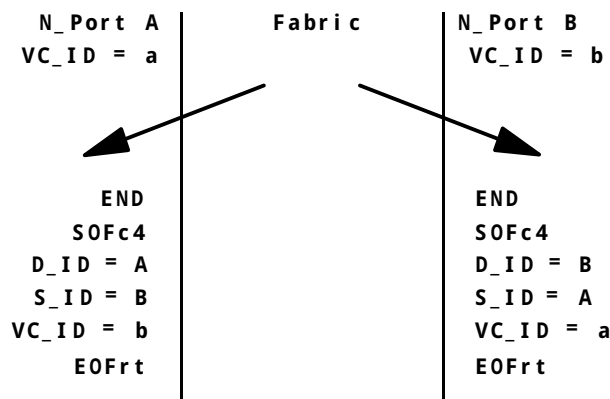


Figure 83 - Class 4 circuit - Fabric initiated removal

The Fabric shall use this mechanism to remove Class 4 circuits when:

- Fabric re-Login is attempted,
- N_Port or the F_Port enters the Offline state,
- Link failure is detected at the N_Port,
- error recovery for the setup process (see 28.2.4.1), or
- internal Fabric failures of the Class 4 circuit path are detected (i.e., the Fabric is unable to provide the guaranteed quality of service).

A N_Port shall reclaim all EE_C4_Credit when the N_Port enters the Nonexisting state (i.e., set the EE_C4_Credit to the EE_C4_Credit_Limit limit when using the count down credit management model).

28.3 Class 4 circuit – Deactivation, removal contention and error recovery

If a CTI has requested removal of a Class 4 circuit (i.e., set E_C = 1, R_C = 1 on the last Data frame transmitted) and this Data frame has not yet been acknowledged by the CTR, the CTI shall react as follows:

- a) The CTI shall acknowledge Data frames received from the CTR on the Class 4 circuit. The EOF delimiter used on the ACK frames shall be either EOFn or EOFt, preventing the CTI from responding to deactivation or removal requests from the CTR.
- b) Requests to remove or deactivate the Class 4 circuit from the CTR shall be denied (see 28.3(1)).
- c) Data frame rejection (P_RJT) criteria for the CTI is unaffected by the Class 4 circuit removal request.
- d) The CTI shall not transmit any Data frames on the Class 4 circuit until either an acknowledgment to the removal request frame has been received or until Sequence timeout is detected.
- e) If Sequence timeout is detected, then the CTI may attempt Sequence recovery or it shall transmit an END Link_Control command to deactivate the Class 4 circuit, followed by an END Link_Control command to remove the circuit.

If a CTR has requested removal of a Class 4 circuit (i.e., set E_C = 1, R_C = 1 on the last Data frame transmitted) and this Data frame has not yet been acknowledged by the CTI, the CTR shall react as follows:

- f) The CTR shall acknowledge Data frames received from the CTI on the Class 4 circuit. The EOF delimiter used on the ACK frames shall be EOFn, EOFt or EOFrt.
- g) Requests to deactivate the Class 4 circuit from the CTI shall be denied (see 28.3(6)).
- h) Requests to remove the Class 4 circuit from the CTI shall be accepted (see 28.3 (6)).
- i) Data frame rejection (P_RJT) criteria for the CTR is unaffected by the Class 4 circuit removal request.
- j) The CTR shall not transmit any Data frames on the Class 4 circuit until either an acknowledgment to the removal request frame has been received or until Sequence timeout is detected.
- k) If Sequence timeout is detected, then the CTR may attempt Sequence recovery or it shall transmit an END Link_Control command to deactivate the Class 4 circuit, followed by an END Link_Control command to remove the circuit.

If a CTI has requested deactivation of a Class 4 circuit (i.e., set E_C = 1, R_C = 0 on the last Data frame transmitted) and this Data frame has not yet been acknowledged by the CTR, the CTI shall react as follows:

- l) The CTI shall acknowledge Data frames received from the CTR on the Class 4 circuit. The EOF delimiter used on the ACK frames shall be either EOFn, EOFt or EOFrt, preventing the CTI to respond to deactivation request from the CTR but allowing the CTI to respond to removal request from the CTR.
- m) Requests to deactivate the Class 4 circuit from the CTR shall be denied (see 28.3 (9)).
- n) Requests to remove the Class 4 circuit from the CTR shall be accepted (see 28.3 (9)).
- o) Data frame rejection (P_RJT) criteria for the CTI is unaffected by the Class 4 circuit deactivation request.
- p) The CTI shall not transmit any Data frames on the Class 4 circuit until either an acknowledgment to the deactivation request frame has been received or until Sequence timeout is detected.
- q) If Sequence timeout is detected, then the CTI may attempt Sequence recovery or it may transmit an END Link_Control command to deactivate the Class 4 circuit.

If a CTR has requested deactivation of a Class 4 circuit (i.e., set E_C = 1, R_C = 0 on the last Data frame transmitted) and this Data frame has not yet been acknowledged by the CTI, the CTR shall react as follows:

- r) The CTR shall acknowledge Data frames received from the CTI on the Class 4 circuit. The EOF delimiter used on the ACK frames shall be EOFn, EOFt, EOFdt or EOFrt, enabling the CTR to respond to both deactivation and removal requests from the CTI.
- s) Requests to deactivate or remove the Class 4 circuit from the CTI shall be accepted (see 28.3 (18)).
- t) Data frame rejection (P_RJT) criteria for the CTR is unaffected by the Class 4 circuit removal request.
- u) The CTR shall not transmit any Data frames on the Class 4 circuit until either an ACK frame to the deactivation request frame has been received or until Sequence timeout is detected.
- v) If Sequence timeout is detected, then the CTR may attempt Sequence recovery or it shall transmit an END Link_Control command to deactivate the Class 4 circuit, followed by an END Link_Control command to remove the circuit.

If a CTI has no outstanding deactivation or removal request, the CTI shall react as follows:

- w) The CTI shall acknowledge Data frames received from the CTR on the Class 4 circuit. The EOF delimiter used on the ACK frames shall be EOFn, EOFt, EOFdt or EOFrt, enabling the CTI to respond to both deactivation and removal requests from the CTR.
- x) Requests to deactivate or remove the Class 4 circuit from the CTR may be accepted (see 28.3(21)).

If a CTR has no outstanding deactivation or removal request, the CTR shall react as follows:

- y) The CTR shall acknowledge Data frames received from the CTI on the Class 4 circuit. The EOF delimiter used on the ACK frames shall be EOFn, EOFt, EOFdt or EOFrt, enabling the CTR to respond to both deactivation and removal requests from the CTI.
- z) Requests to deactivate or remove the Class 4 circuit from the CTI may be accepted (see 28.3 (25)).

28.4 Ordered Sets for Class 4

Class 4 Frame delimiter usage for Data and Link_ Control frames is shown in Table 222.

Table 222 - Frame delimiter usage	
Frame Types	Delimiters
Data	SOFc4, SOFi4, SOFn4, EOFn
Link_Control	SOFc4, SOFn4, EOFn, EOFt, EOFdt, EOFrt

28.4.1 Start Of Frame (SOF)

A Class 4 circuit in the Dormant state is activated to Live state by using an SOFc4 delimiter for the first frame of a Sequence.

The frames that initiate a Sequence while the Class 4 circuit is in the Live state shall begin with an SOFi4 delimiter.

All other frames for a Class 4 circuit in the Live state shall begin with an SOFn4 delimiter.

28.4.1.1 SOF – Circuit Activate Class 4 (SOFc4)

The SOFc4 shall be used to activate a Class 4 circuit. This delimiter shall also identify the first frame of a Sequence. Interaction with the Fabric and the destination N_Port are shown in Figure 84 (a).

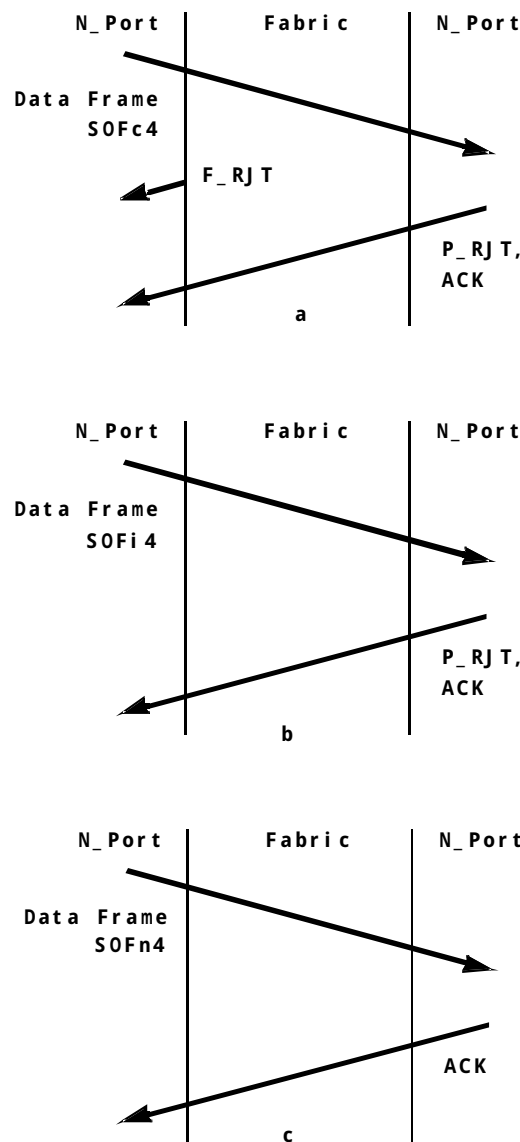


Figure 84 - Class 4 circuit - Data frame responses

A N_Port shall only transmit an SOFc4 delimited frame if the Class 4 circuit is in the Dormant state. N_Ports shall discard SOFc4 delimited frames received for Class 4 circuits in the Non-existing or Pending state. Any SOFc4 delimited frames received by a N_Port for a Class 4 circuit in the Dormant or Live state shall qualify for further processing by the N_Port.

The END Link_Control command is the only Link_Control frame, which shall make use of the SOFc4 delimiter.

Figure 84 (a) shows all possible responses to a SOFc4 delimited data frame, in the absence of errors.

28.4.1.2 SOF – Initiate Class 4 (SOFi4)

The SOFi4 shall be used on the first frame of a Sequence for Class 4 Service, except when SOFc4 is used.

If the Class 4 circuit is not in the Live state the Port shall discard any SOFi4 delimited frame received.

Figure 84 (b) shows all possible responses to a SOFi4 delimited data frame, in the absence of errors.

28.4.1.3 SOF – Normal (SOFn4)

The SOFn4 shall be used for all frames except the first frame of a Sequence for Class 4.

If the Class 4 circuit is not in the Live state the Port shall discard any SOFn4 delimited frame received.

Figure 84 (c) shows all possible responses to a SOFn4 delimited data frame, in the absence of errors.

28.4.2 End_of_Frame (EOF)

All frames other than the last frame of a Sequence shall be terminated with an EOFn delimiter.

Each Sequence other than a Sequence that deactivates or removes a Class 4 circuit shall terminate with an EOFt delimiter.

The EOFni delimiter shall be used by the Fabric to replace either an EOFn or an EOFt on frames for which the Fabric detects a frame error.

The EOFa delimiter shall be used to terminate a partial frame due to a malfunction during transmission. It is also used by the Fabric to replace missing EOF delimiters or to truncate overlength frames.

28.4.2.1 EOF – Deactivate Terminate (EOFdt)

The EOFdt delimiter shall deactivate the Class 4 circuit.

The EOFdt shall identify the last ACK frame of a Sequence or may be sent on an END Link_Control frame. It shall indicate that all Class 4 Sequences associated with this Class 4 circuit are terminated. Open Class 4 Sequences for this Class 4 circuit, other than the SEQ_ID specified in the ACK frame containing EOFrt, shall be abnormally terminated and may require Sequence recovery on a ULP protocol-dependent basis.

28.4.2.2 EOF – Deactivate Terminate Invalid (EOFdti)

The EOFdti delimiter shall replace a recognized EOFdt delimiter on a frame with invalid frame content.

All other actions for an EOFdti shall be the same as for an EOFdt delimiter (see 28.4.2.1).

28.4.2.3 EOF – Remove Terminate (EOFrt)

The EOFrt delimiter shall remove the Class 4 circuit.

The EOFrt shall identify the last ACK frame of a Sequence or may be sent on an END Link_Control frame. It shall indicate that all Class 4 Sequences associated with this Class 4 circuit are terminated. Open Class 4 Sequences for this Class 4 circuit, other than the SEQ_ID specified in the ACK frame containing EOFrt, shall be abnormally terminated and may require Sequence recovery on a ULP protocol-dependent basis.

28.4.2.4 EOF – Remove Terminate Invalid (EOFrti)

The EOFrti delimiter shall replace a recognized EOFrt delimiter on a frame with invalid frame content.

All other actions for an EOFrti shall be the same as for an EOFrt delimiter (see 28.4.2.3).

28.4.3 Virtual Circuit Ready (VC_RDY)

The VC_RDY Primitive Signal shall only be transmitted from a F_Port. The VC_RDY shall indicate that a single Class 4 frame is needed from the N_Port if it wish to maintain the requested bandwidth.

VC_RDY transmission shall obey the following:

- A minimum of two Idle Primitive Signals shall separate a VC_RDY from an EOF or SOF delimiter.
- A minimum of two Idle Primitive Signals shall separate a VC_RDY from a R_RDY Primitive Signal.
- At least one Idle Primitive Signal shall separate VC_RDYs.
- A N_Port shall not transmit the VC_RDY Primitive Signal.

A F_Port shall discard any VC_RDY Primitive Signals received.

28.5 Primitive Sequences

The effect of Primitive Sequences on Class 4 circuits depends on the Port state and is described in the following sub-clauses.

28.5.1 LR Transmit state (LR1)

A N_Port in the LR Transmit state:

- shall retain Class 4 circuit states,
- shall set VC_Credit = 0 for all Class 4 circuits,
- may discard Class 4 frames received on Live Class 4 circuits.

The locally attached F_Port in the LR Transmit state:

- shall retain Class 4 circuit states,
- shall discard Class 4 frames destined for the N_Port,
- may discard Class 4 frames received from the N_Port.

The Fabric with the locally attached F_Port in the LR Transmit state:

- shall, for Live Class 4 circuits, send END Link_Control commands to deactivate the Class 4 circuits when the LR Transmit state is entered (see 28.2.4.3),
- may suspend, to remote N_Ports, VC_RDY pacing for all Live Class 4 circuits.

28.5.2 LR Receive state (LR2)

A N_Port in the LR Receive state:

- shall retain Class 4 circuit states,
- shall set VC_Credit = 0 for all Class 4 circuits.

The locally attached F_Port in the LR Receive state:

- shall retain Class 4 circuit states,
- shall discard Class 4 frames destined for the N_Port.

The Fabric with the locally attached F_Port in the LR Receive state:

- shall suspend VC_RDY transmission to the N_Port,
- shall, for Live Class 4 circuits, send END Link_Control commands to deactivate the Class 4 circuits when the LR Receive state is entered (see 28.2.4.3),
- may suspend, to remote N_Ports, VC_RDY pacing for all Live Class 4 circuits.

28.5.3 LRR Receive state (LR3)

A N_Port in the LRR Receive state:

- shall retain Class 4 circuit states.

The locally attached F_Port in the LRR Receive state:

- shall retain Class 4 circuit states,
- shall discard Class 4 frames destined for the N_Port.

The Fabric with the locally attached F_Port in the LRR Receive state:

- shall retain the suspension of VC_RDY transmission to the N_Port,
- shall if suspended retain suspension, to remote N_Ports, of VC_RDY pacing for all Live Class 4 circuits.

28.5.4 NOS Receive state (LF1)

A N_Port in the NOS Receive state:

- shall set all Class 4 circuits to the Nonexisting state.

The locally attached F_Port in the NOS Receive state:

- shall set all Class 4 circuits to the Nonexisting state.

The Fabric with the locally attached F_Port in the NOS Receive state:

- shall remove Dormant or Live Class 4 circuits when the NOS Receive state is entered (see 28.2.4.4).

28.5.5 NOS Transmit state (LF2)

A N_Port in the NOS Transmit state:

- shall set all Class 4 circuits to the Nonexisting state.

The locally attached F_Port in the NOS Transmit state:

- shall set all Class 4 circuits to the Nonexisting state.

The Fabric with the locally attached F_Port in the NOS Transmit state:

- shall remove Dormant or Live Class 4 circuits when the NOS Transmit state is entered (see 28.2.4.4).

28.5.6 OLS Transmit state (OL1)

A N_Port in the OLS Transmit state:

- shall set all Class 4 circuits to the Nonexisting state.

The locally attached F_Port in the OLS Transmit state:

- shall set all Class 4 circuits to the Nonexisting state.

The Fabric with the locally attached F_Port in the OLS Transmit state:

- shall remove Dormant or Live Class 4 circuits when the OLS Transmit state is entered (see 28.2.4.4).

28.5.7 OLS Receive state (OL2)

A N_Port in the OLS Receive state:

- shall set all Class 4 circuits to the Nonexisting state.

The locally attached F_Port in the OLS Receive state:

- shall set all Class 4 circuits to the Nonexisting state.

The Fabric with the locally attached F_Port in the OLS Receive state:

- shall remove Dormant or Live Class 4 circuits when the OLS Receive state is entered (see 28.2.4.4).

28.5.8 Wait for OLS state (OL3)

A N_Port in the 'Wait for OLS' state:

- shall retain Class 4 circuits in the Nonexisting state.

The locally attached F_Port in the 'Wait for OLS' state:

- shall retain Class 4 circuits in the Nonexisting state.

28.6 Data frames and responses

28.6.1 Class 4 frame size

The maximum frame size used for all frames for a Class 4 circuit shall be the VC Data Field size decided during the QoS Extended Link Service request.

The Fabric may discard or truncate any Class 4 frames exceeding the Data Field Size limit. If the Fabric truncates a Class 4 frame it shall be terminated with an EOFa delimiter.

28.6.2 QoSF Well-known Address identifier

The well-known address identifier of the Quality of Service Facilitator (QoSF) shall be hex 'FF FF F9'.

28.6.3 Class 4 – CS_CTL field usage

The Class 4 VC_ID field is a Class of Service specific use of the CS_CTL field frame header. The VC_ID identifies which VC the frame is associated with from the point of view of the S_ID in the frame header.

An unique VC is identified from S_ID || VC_ID in the frame header. During the Class 4 circuit setup process, the QoSF associates S_ID || VC_ID with a unique destination N_Port D_ID value which is also identified in the QoSR. For each D_ID, the S_ID||VC_ID identifies the route for the VC through the Fabric. The association between VC_ID and S_ID shall be done for the CTI when the QoSR is recognized by the QoSF and for the CTR when it's ACC is recognized by the QoSF.

VC_ID values are assigned as follows:

- The value hex '00' is reserved.
- The value in the VC_ID associated with a reserved VC shall be value in the range of hex '01' through hex 'FE'.
- The value hex 'FF' is reserved.

This assignment of VC_ID values permits a maximum of 254 Dormant or Live Class 4 circuits for each N_Port, shared between the N_Ports address identifier and any alias address identifier.

The Fabric uses the VC_ID field in the frame header to route frames to a destination N_Port. The VC_ID field value is picked by the N_Port during the Class 4 circuit setup process. Each N_Port pick its own VC_ID value.

The value in the field is either a CTI VC_ID (CIVC_ID) or a CTR VC_ID (CRVC_ID), depending on which of the two communicating N_Ports is transmitting the frame.

28.6.4 Frame control (F_CTL)

The E_C bit shall be set to one in the last Data frame of a Sequence to indicate that the N_Port transmitting E_C is beginning the deactivation or removal procedure. The N_Port transmitting E_C = 1 on the last Data frame of a Sequence is requesting the receiving N_Port to transmit an ACK frame terminated by EOFdt or EOFrt if the receiving N_Port has completed all active Sequences. If the R_C = 1 then a Class 4 circuit removal is requested otherwise a deactivation is requested.

If the receiving N_Port is not able to deactivate or remove the Class 4 circuit, E_C = 1 requests that the receiving N_Port complete all active Sequences and not initiate any new Sequences during the current activation cycle.

28.6.5 Data Frames

Permission to transmit frames is granted by a VC_RDY. Each N_Port has a VC_ID that identifies the VC for its half of the Class 4 circuit.

If a data frame is transmitted in a Class 4 circuit, the VC_ID field shall contain the appropriate CIVC_ID or CRVC_ID value corresponding to the S_ID field in the same frame (see Figure 85).

As shown in Figure 85, in an ACK frame, the VC_ID field in the frame header is substituted with the VC_ID associated with the switched S_ID in the frame header (i.e., changed to the other VC_ID associated with the Class 4 circuit).

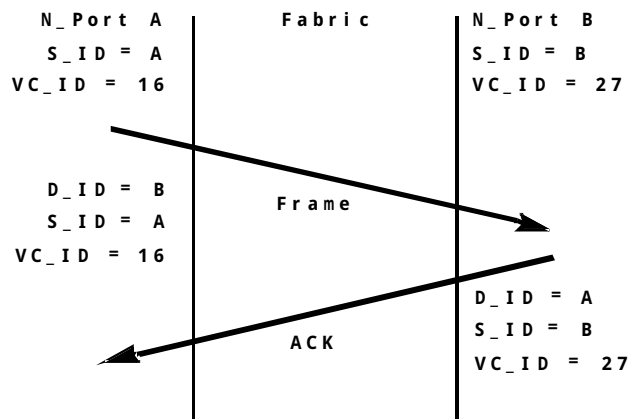


Figure 85 - CVC_ID and CRVC_ID management

28.6.6 Link_Continue function

The Link_Continue functions used for Class 4 are

- VC_RDY Primitive Signal (see 28.6.6.1)
- ACK frames (see 28.6.6.2).

28.6.6.1 VC_RDY Primitive Signal

In Class 4, once a Class 4 circuit is activated to provide fractional bandwidth service between communicating N_Ports, the Fabric shall transmit the VC_RDY Primitive Signal to indicate that the receiving N_Port may send a frame for the VC_ID identified in the VC_RDY Primitive Signal.

The VC_RDY Primitive Signal shall only be used for flow control and shall indicate that the VC identified in the VC_RDY still exists in the Fabric. The VC_RDY primitive signal from a F_Port also indicates that a frame buffer is free for the identified VC only.

The VC_RDY Primitive Signal is used for F_Port-to-N_Port buffer-to-buffer flow control for an individual VC. The VC_RDY Primitive Signal shall be transmitted when the Fabric determines that the N_Port may send a single frame to maintain its QoS parameters for the associated VC. This Primitive Signal shall only be transmitted when a Dormant or Live Class 4 circuit exists with a N_Port.

The VC_RDY Primitive Signal shall indicate that the Source N_Port needs to transmit a single frame, but not to exceed its available EE_C4_Credit or the VC_Credit, for the indicated VC Identifier (VC_ID).

The event associated with the receipt of a VC_RDY may be indicated to level above FC-2 within a N_Port to cause a frame for the indicated VC to be transmitted.

28.6.6.2 ACK frames

Class 4 shall use either ACK_1 or ACK_0 frames for frame acknowledgment.

The frame's VC_ID field shall contain the appropriate CVC_ID or CRVC_ID value. This value may be different from the value in the VC_ID field in the data frame being acknowledged.

The ACK frame used to terminate a Sequence shall have an EOFt delimiter. The ACK frame used to deactivate a Class 4 circuit and terminate a Sequence shall use the EOFdt delimiter.

The ACK frame used to remove a Class 4 circuit and terminate a Sequence shall use the EOFrt delimiter. All other ACK frames that do not terminate a Sequence shall use an EOFn delimiter.

28.6.7 Link_Response frames

Class 4 shall use the following Link_Response frames in the same manner as used by Class 1.

- F_RJT (Fabric Reject), or
- P_RJT (N_Port Reject).

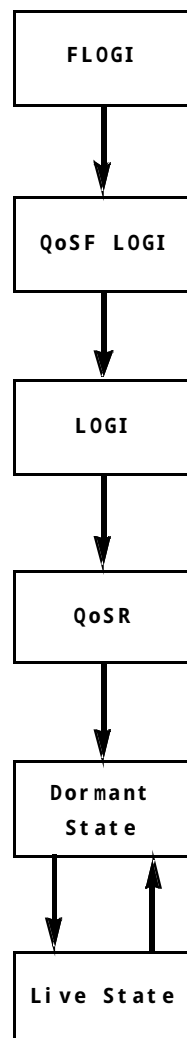


Figure 86 - Class 4 circuit – Management

28.6.7.1 Reject (P_RJT, F_RJT)

Class 4 shall use the F_RJT and P_RJT. Reject Reason Codes specifically for Class 4 are specified in Table 196.

If the EE_C4_Credit rules are violated by a source N_Port, the destination N_Port shall reject the frame (P_RJT) with a Reject Reason Code of Protocol Error (see 14.3.3.3).

The frame's VC_ID field shall contain the appropriate CIVC_ID or CRVC_ID value. This value may be different from the value in the VC_ID field in the data frame being rejected.

The Fabric shall not reject any SOFn4 or SOFi4 delimited frames.

The VC_ID field shall be = 0 in an F_RJT frame with the reason code 'Invalid VC_ID'.

Neither the N_Port nor the Fabric shall reject an END Link_Control command. Both the Fabric and the N_Ports shall discard END Link_Control commands with invalid D_ID || S_ID || VC_ID frame header content.

28.6.7.2 Fabric Busy (F_BSY)

The Fabric shall not F_BSY any Class 4 frames.

28.6.7.3 N_Port Busy (P_BSY)

The N_Port shall not P_BSY any Class 4 frames.

28.6.8 Link Credit Reset (LCR) Link_Control command

Class 4 behavior shall not be affected by LCR.

28.6.9 END Link_Control command

The END frame (see 14.3.4.3) shall, in Class 4, indicate that the Class 4 circuit is to be deactivated or removed. The Class 4 circuit is removed if the End Of Frame delimiter is EOFrt or EOFrti. The Class 4 circuit is deactivated if the End Of Frame delimiter is EOFdt or EOFdti. The END Link_Control frame shall be discarded if the End Of Frame delimiter is EOFn, EOFt, EOFni, EOFa or if the end of frame delimiter is missing. Upon the reception of the END Link_Control frame all active Sequences between the communicating N_Ports, using the Class 4 circuit are abnormally terminated.

The communicating N_Ports, at the discretion of the appropriate Upper Level protocol shall perform Exchange and Sequence recovery.

N_Ports shall use the END Link_Control frame to remove or deactivate a Class 4 circuit if the N_Port is unable to determine the state of the Class 4 circuit.

The Fabric shall, when required to remove a Class 4 circuit, send END Link_Control commands to both N_Ports in the Class 4 circuit (see 28.2.4.4).

28.7 Login and Service Parameters

The applicability of Class 4 service parameters for Fabric and N_Port_Login are shown in Table 202 and Table 204.

The Fabric shall remove all Class 4 circuits to a N_Port when the N_Port makes a Fabric Login. Class 4 shall not be used for Fabric Login.

The Class 4 default Fabric Login service parameters shall be:

- Class validity = 0 (invalid).

End-to-end N_Port_Login shall not affect the state of Class 4 circuits.

The Class 4 default Login service parameters prior to end-to-end N_Port_Login shall be as shown in Table 223.

Table 223 - N_Port – Default Class 4 service parameters	
Service Parameter	Default value
Class validity	1 (Class supported)
Initiator control	
X_ID reassignment - obsolete	Shall be zero
Initial CTR Process Associator	00 (not supported)
ACK_0 capable	0 (ACK_0 incapable)
ACK generation assistance	0 (no ACK generation assistance)
Recipient control	
ACK_0 capable	0 (ACK_0 incapable)
X_ID interlock	1 (interlock required)
Error Policy supported	00 (discard policy only)
Categories per Sequence	1
Receive Data Field size	256
Concurrent Sequences	1
N_Port EE_C4_Credit	n/a
Open Sequences per Exchange	1

28.8 Link Services

A Class 4 circuit is setup using Extended Link Service commands associated with Class 4 service.

- QoS SR - Quality of Service Request (see 15.5.36.1)

28.8.1 Basic Link Services

For any Basic Link Service response, if the request is carried on a Class 4 circuit, the VC_ID field shall contain the appropriate CIVC_ID or CRVC_ID value.

28.8.2 Extended Link Services

For any extended link service response, if the request is carried on a Class 4 circuit, the VC_ID field shall contain the appropriate CIVC_ID or CRVC_ID value.

The following Extended Link Service requests shall be supported for Class 4 management (see 15.5.36):

- Quality of Service Request (QoS SR)
- Read Virtual circuit status (RVCS).

28.9 Class 4 procedures

The following procedures specify the behavior for the various phases of a Class 4 circuit.

The flow chart in Figure 86 outlines the procedures involved in Class 4 circuit management.

- FLOGI - Each N_Port shall perform Fabric login indicating support for Class 4. (see 28.9.1).
- Each N_Port shall perform N_Port_Login with the QoSF.
- LOGI - Each N_Port shall perform N_Port_Login with each N_Port for which it intends to establish a Class 4 circuit indicating support for Class 4 (see 28.9.1).
- One of the terminal N_Ports shall start the QoS link service request with the QoSF. When successful, the Class 4 circuit is placed in the Dormant state (see 28.2.4).
- From the Dormant state, the Class 4 circuit may be activated or reactivated to the Live state by a frame with a circuit activation request between the terminal N_Ports (see 28.2.4.2). The Class 4 circuit may also be removed by the Fabric (see 28.2.4.4).
- While in the Live state, normal interaction between the communicating N_Ports takes place across the Class 4 circuit, moderated by the Fabric.
- When data transfers are complete or are to be suspended, the Class 4 circuit may be moved back to the Dormant state (see 28.2.4.3).

28.9.1 Class 4 login

Before any operations can begin in Class 4, the potential communicating N_Ports shall support Class 4. Their attempt to login with the Fabric provides an indication to the Fabric of their capability to support Class 4 operation and of the Fabric to support Class 4 operation.

If Fabric login ends successfully with Class 4 available to both N_Ports and the Fabric, the N_Ports must then perform N_Port_Login with each other. Again, the N_Port_Login procedure indicates the capability of the two N_Ports to support Class 4. If the N_Port_Login is successful for Class 4, a Class 4 circuit may be setup (see 28.2.4.1).

N_Ports shall Login with the QoSF prior to any QoSR by the N_Port.

28.9.2 Class 4 rules - summary

The following rules apply exclusively to Class 4 circuits. See 28.9.3 for additional rules applicable to the mandatory supported Intermix with Class 4. To provide a Class 4 circuit, the source and destination N_Ports and the Fabric shall obey the following rules:

- a) Except for some Link Service protocols a N_Port shall have performed Fabric Login and N_Port_Login with the ports it intends to communicate with, either explicitly or implicitly (see 28.9.1). To login explicitly, the requesting N_Port shall use the Fabric and N_Port Login protocols.
- b) The Fabric shall be responsible for Class 4 circuit setup. The Fabric shall retain the setup Class 4 circuit until one of the communicating N_Ports in the Class 4 circuit requests removal of the Class 4 circuit unless one of the legal Fabric exceptions occur. Legal Fabric exceptions include: path failure and Fabric Login.
- c) To activate a Class 4 circuit, the requesting N_Port shall use the Class 4 delimiters as specified in 28.4. A N_Port shall only send an SOFc4 delimited frame on a Class 4 circuit in the Dormant state.
- d) To deactivate a Class 4 circuit, the requesting N_Port shall use the Class 4 delimiters as specified in 28.4. The Class 4 circuit may be re-activated at a later time by using the procedure in 28.2.4.2.
- e) To remove a Class 4 circuit, the requesting N_Port shall use the Class 4 delimiters as specified in 28.4.
- f) A destination N_Port shall acknowledge delivery of each valid Data Frame (see 28.9.4.1).

- g) The Sequence Initiator shall increment the SEQ_CNT field on each successive frame transmitted within a Sequence. The Fabric shall guarantee delivery of the frames at the destination N_Port in the order of transmission by the source N_Port.
- h) Each N_Port of a Live Class 4 circuit may originate multiple Exchanges and initiate multiple Sequences with one or more N_Ports. Each N_Port shall assign a unique X_ID as appropriate for its role in each Exchange. Thus, the value of the OX_ID and RX_ID is unique for each Exchange with a N_Port pair. A Sequence Initiator shall assign a unique Sequence_Qualifier for each Sequence it initiates which is unique for the communicating N_Port pair.
- i) Communicating N_Ports shall be responsible to manage end-to-end flow control without any Fabric involvement. ACK frames shall be used to perform end-to-end flow control (see 28.9.4.1). All Class4 frames, except for the END Link_Control command, shall participate in end-to-end flow control.
- j) The Fabric shall be responsible to regulate the rate at which a N_Port transmits frames. It does so by transmitting the VC_RDY Primitive Signal to a N_Port to request that frames be transmitted. Each VC_RDY shall identify the VC for which a frame is being requested. The Fabric shall measure the QoS for bandwidth by measuring all transmission words associated with the VC (SOF delimiters, EOF delimiters, and frame content).
- k) The Fabric may reject a request to activate a Class 4 circuit with a valid reason code. Once the Class 4 circuit is alive the QoSF shall only interfere with a VC to the extent that it supplies the VC_RDY Primitive Signals in a timely manner.
- l) The Fabric shall respond with an F_BSY Link_Response frame to any received Class 1 connect-request frame (i.e., a SOFc1 delimited frame) if one or more Class 4 circuits are setup with the destination N_Port frame).
- m) The destination N_Port for a request to establish a VC may respond with a P_RJT with an appropriate reason code.
- n) A N_Port shall not respond with a P_BSY for any Class 4 frame received.
- o) The EE_C4_Credit, determined during the circuit setup process shall be honored.
- p) The Fabric shall guarantee that it provides VC_RDY Primitive Signals to the communicating N_Ports at the correct rate and time intervals to meet or exceed the QoS parameters guaranteed for each VC. The latency guarantee relates only to the frame transit within the Fabric.
- q) A N_Port shall only transmit Class 4 frames if available VC_Credit permits (see 28.9.4.2).
- r) N_Ports shall discard all Class 4 frames received on a Nonexisting Class 4 circuit, or if the Class 4 circuit is in the Pending state.
- s) The N_Port shall discard all SOFi4 and SOFn4 delimited frames received on a Class 4 circuit in the Dormant state.
- t) A F_Port shall discard all SOFi4 and SOFn4 delimited frames received unless the Class 4 circuit is in the Live state.
- u) EE_C4_Credit shall remain unaffected by any discarded Class 4 frames.
- v) If VC_Credit is = 0 for a time of more than E_D_TOV the N_Port may use the RVCS protocol to read the state of the Class 4 circuit or the N_Port shall perform the Online to Offline protocol. If the N_Port obtains the Class 4 circuit state from the ACC to the RVCS request and it shows that the Class 4 circuit is Nonexisting, the N_Port shall set the circuit state to Nonexisting.
- w) Frames within a Sequence are tracked on a Sequence_Qualifier and SEQ_CNT basis.
- x) If a N_Port, not supporting Class 4, receives a Class 4 circuit activation frame, the N_Port may return a P_RJT frame with a reason code of "Class not supported", or the N_Port may discard the Class 4 frame. If an F_Port, not supporting Class 4, receives a Class 4 circuit

activation frame, the F_Port may return a F_RJT response frame with a reason code of "Class not supported".

- y) The Fabric shall use the VC_ID from the frame header for routing frames to a destination N_Port (D_ID). The QoSF maintains a translation from S_ID||VC_ID to D_ID from the original QoS transaction. Prior to routing a frame, if there is a mismatch on S_ID, VC_ID and D_ID, the frame shall be discarded by the Fabric.
- z) A N_Port supporting Class 4 shall not send Class 1 frames nor attempt to establish a Class 1 Dedicated Connection while a Class 4 circuit exists with the N_Port or while a request to setup a Class 4 circuit has been made by the N_Port.

28.9.3 Class 4 intermix

Intermix is always enabled for service Class 4.

- N_Ports may interleave Class 2 or Class 3 frames on the outbound fibre while a Class 4 circuit exists. The same N_Port may receive Class 2 or Class 3 frames on the inbound fibre while a Class 4 circuit exists.
- Transmission of Class 4 frames shall have priority over Class 2 and 3, but a Class 2 or Class 3 frame shall not be aborted to send a Class 4 frame.

28.9.4 Class 4 flow control

28.9.4.1 End-to-end flow control

ACK frames shall be used to perform Class 4 end-to-end flow control. All Class 4 frames shall participate in end-to-end flow control.

NOTE - End-to-end flow control is the only mechanism available for flow control by a N_Ports receiver. A small EE_C4_Credit value helps to decrease peak demands.

28.9.4.2 Buffer-to-buffer flow control

Summary rules for VC credit management:

- N_Ports shall track available credit independently for each VC.
- Maximum available credit (VC_Credit) on a VC shall be limited to a value of less than 256. $VC_Credit_Limit < 256$
- A N_Port shall not transmit any Class 4 frames on a given VC unless VC_Credit is available for the VC.
- One credit shall be consumed for every Class 4 frame transmitted on a VC ($VC_Credit \leftarrow VC_Credit - 1$).
- Credit shall be regained when a VC_RDY is received for a given VC ($VC_Credit \leftarrow VC_Credit + 1$ if $VC_Credit < VC_Credit_Limit$)

NOTE - Limiting available credit allows the Fabric to generate more VC_RDYs than the number of frames transmitted. Allowing unused bandwidth to be used by other VCs or for intermix frames.

The state of a VC shall be the same as the Class 4 circuit state for that VC. The maximum available credit (VC_Credit) shall be limited as follow:

- For the CTR to the value of the Live VC credit limit field in the QoS, if the VC is alive.
- For the CTI to the value of the Live VC credit limit field in the accept frame to the QoS, if the VC is alive.
- Maximum available credit shall be 0 for Nonexisting VC or VCs in the Pending state. ($VC_Credit_Limit = 0$)

NOTE - The CTI and the CTR start with an available VC_Credit of 0 when a new VC is setup.

- Maximum available credit shall be 1 for VCs in the Dormant state. ($VC_Credit_Limit = 1$)
- Maximum available credit shall be set to the `Live_VC_Credit_Limit` for VCs in the Live state.
- Available `VC_Credit` shall be 0 after Link Reset. ($VC_Credit \leftarrow 0$)

If the `N_Port` has a frame ready to transmit and `EE_C4_Credit` available when a `VC_RDY` is received for a VC, the `N_Port` transmits the frame at the next opportunity to transmit an out-bound frame. The `N_Port` may be in the middle of transmitting a frame when a `VC_RDY` is received.

The `N_Port` shall not accumulate a `VC_RDY` buffer-to-buffer credit that exceeds the allotted limit. The `N_Port` shall not use a count in excess of the preset limit to send several frames.

Receipt of a `VC_RDY` is a guarantee by the Fabric that bandwidth is available to transmit a frame for the identified VC.

The rate at which the Fabric provides `VC_RDYs` to a `N_Port` can be used to provide congestion control for the Fabric by keeping unwanted frames in the `N_Ports` until the QoS parameters can be met.

28.10 Quality of Service

Class 4 through the QoSF, guarantees that the minimum bandwidth is available to the communicating `N_Ports` in a Class 4 circuit. Some applications may cause the communicating `N_Ports` not to use all of that guaranteed bandwidth, but the Fabric shall offer the guaranteed bandwidth to each `N_Port` for each Live Class 4 circuit.

Therefore, the measure of effectiveness of Class 4 in meeting its goal of guaranteed bandwidth cannot be measured for a Fabric without consideration for the participating `N_Ports`. The measure of effectiveness can only be obtained indirectly by the `VC_RDY` Primitive Signals transmitted to the participating `N_Ports`. The bandwidth utilization for a VC is also a function of the available `EE_C4_Credit` and the ability of the participating ports to sustain the negotiated bandwidth.

The Fabric shall guarantee a worst case end-to-end delay for each frame once it enters the Fabric.

To help establish a frame rate associated with a calculation of the negotiated bandwidth, the Quality of Service Extended Link Service requests permit the `N_Ports` to negotiate a maximum frame Data Field size that is equal to or smaller than the current allowable maximum between the communicating `N_Ports` specifically for Class 4. The maximum value is determined during Fabric and `N_Port_Login`.

All QoS parameters are negotiated through the QoSF for each VC. The QoS parameters are independently negotiated for each VC. This permits an asymmetric allocation of resources.

- Bandwidth range
- Worst case end-to-end delay for a frame for fractional bandwidth mode
- VC Data Field size.

For the first parameter there are two levels of negotiation. The desirable quality of service (Desirable QoS) and the acceptable quality of service (Acceptable QoS).

The Acceptable QoS is guaranteed if the Class 4 circuit is setup. The desirable QoS may be attempted, but is not guaranteed. When applied to the list above, the negotiable parameters become:

- Desirable Bandwidth
- Acceptable Bandwidth
- Acceptable Worst case end-to-end delay for a frame
- VC Data Field size.

The VC Data Field size is not a variable quantity since it represents the maximum Data field size that an FC-2 can generate for each frame for Class 4 and is not controllable by the Fabric.

The negotiation for QoS is accomplished using an Extended Link Service request, QoS Request (QoSR) (see 15.5.36.1).

Each request is handled in two stages: CTI to QoSF and QoSF to CTR. This process is illustrated in the specification of the QoSR link service request.

- If the QoSF cannot perform the function for the Class 4 circuit (e.g., too much bandwidth requested versus that available at the time of the request) the QoSF shall send a LS_RJT response to the CTI with an appropriate reason code.
- If the CTR cannot perform the function for the Class 4 circuit (e.g., too much bandwidth requested versus that available at the time of the request) the CTR shall send an LS_RJT to the CTI to inform it of the rejection and the reason for it.

29 Camp-On

29.1 Introduction

Camp-On introduces a level of fairness into the connection establish mechanism that is enforced by the Fabric. It ensures the Connection Initiator (CI) eventual access to the destination N_Port. The Camp-On protocol requires active participation by the CI and the Fabric while the Connection_Recipient (CR) has a passive role in the Camp-On connect-request protocol.

Camp-On provides a N_Port the means to alter the behavior of the Fabric during the connection request process. Camp-On allows the source N_Port to get on a queue at the destination F_Port when the destination N_Port is engaged in a connection and thus be assured of eventual access to the destination N_Port. Camp-On is invoked on a connection by connection basis.

The Connection Initiator (CI) can direct the Fabric to immediately (within E_D_TOV) 'busy' the connect-request if the destination N_Port is engaged in a connection or to request that the Fabric 'hold' the connection request for an extended time period.

NOTE - No new delimiters are required.

29.2 FC-2 Mechanisms

Enhanced FC-2 mechanisms used for Camp-On are:

- Camp-On Request (COR): CS_CTL bit (see 12.2)
- Notify sequence protocol (see 14.3.4.2)
- End Link_Control Command

29.3 Applicability

Camp-On is limited to Class 1 service. Camp-On is an optional function that may be supported by the CI and the Fabric. The CR is not aware of the function. It is related to Stacked Connection Request, which is also limited to Class 1 service.

29.4 Communications Model

The Connection Initiator (CI) N_Port requesting a connection will set or reset the COR bit in the CS_CTL field of the connection request to invoke or not to invoke the Camp-On function in the Fabric. If the COR bit is reset (=0) and if the destination N_Port is engaged in a connection, the Fabric will immediately (within E_D_TOV) busy off the connection request. When the COR bit is set (=1) and if the destination N-Port is currently engaged in a connection, the Fabric will place the connection request on a queue for that destination F_Port. If that destination N_Port supports Intermix, the Fabric will initiate a Class_2 Notify (see 14.3.4.2) sequence to that N_Port. If the destination N_Port does not support Intermix, the Fabric shall not transmit the Notify sequence. If the destination N_Port is not engaged in a connection when the Fabric receives the connect-request, the Fabric will immediately initiate the connection process (normal operation).

If the Fabric is unable to service the connect-request due to any Fabric conditions such as queue being full, the Fabric shall transmit an F_BSY to the CI and discard the connect-request.

If the CI does not receive F_BSY nor get the connection established and if the CI chooses not to wait any longer, the CI may send an End Link_Control command with EOFdt and the Fabric shall remove the connect-request off the queue and discard it. If the Fabric has already routed the connect-request to the CR, then the Fabric shall transmit an F_RJT.

29.5 Requirements

29.5.1 Connection Initiator

The CI shall support:

- Class 1
- COR CS_CTL bit
- CI rules stated below.

29.5.2 Connection Recipient

The CR may optionally support:

- Notify protocol.

29.5.3 Fabric

The Fabric shall support

- Class 1
- Intermix
- Notify protocol.
- COR CS_CTL bit

29.6 Login

N_Ports and Fabric shall indicate their ability to support Camp-On during Fabric Login.

29.6.1 F_Port Class Service Parameters

The F_Port Class Service Parameters (see 17.7.4) shall be used to indicate support for the Camp-On facility (see 17.7.4.2).

29.7 Camp-On Invoked

The CI shall invoke the Camp-On function on a connection by connection basis. To invoke Camp-On, the CI shall transmit a connect-request frame containing the SOFc1 delimiter and setting the CS_CTL COR=1. If COR=0 on a connect-request frame, the Camp-On function is not invoked.

Table 224 - Camp-On vs. Stacked Connect-request		
Support Indicated:	Stacked Connect-request by Fabric during Login	Camp-On by N_Port & Fabric during Login
Invoked by	CI through Stacked Connect-request CS_CTL on a connection by connection basis	CI through COR CS_CTL on a connection by connection basis
Response Received to Connect-request		
with S_ID=A and	D_ID=B, C, D,...	D_ID=B
ACK with S_ID=B	Connection established with one of the de- sired destinations	- same as Stacked Connect-request
SOF(Cx) with S_ID=X	- Connect est. with third party	- Connection est. with third party

	- Connect-request maintained within Fabric	- Connect-request failed - Retransmit Connect-request
F_BSY	Retransmit Connect-request	same as Stacked Connect-request
F_RJT	Retransmit Connect-request	same as Stacked Connect-request
P_BSY	Retransmit Connect-request	same as Stacked Connect-request
P_RJT	Retransmit Connect-request	same as Stacked Connect-request
CR_TOV use	- CR_TOV + E_D_TOV or 2 (CR_TOV) + E_D_TOV - Connection could not be established within time out period. - Retransmit Connect-request	Not applicable

29.8 Rules

29.8.1 Connection Initiator

- To invoke Camp-On, the CI shall transmit a connect-request frame containing the SOFc1 delimiter and setting the CS_CTL COR=1.
- When the CI receives an F_BSY in response to a connect-request frame with COR set to a one (=1), it shall know that its connect-request has been discarded and the requested connection will not be established. It must retransmit the connect-request at a later time.
- If neither the connection has been established, nor an F_BSY has been received and if the CI chooses not to wait any longer, the CI may transmit an End Link_Control command with EOFdt. Through the End Link_Control command, the CI is indicating that the Fabric shall discard the connect-request.

29.8.2 Connection Recipient

- The CR shall transmit all ACK, P_BSY, and P_RJT frames in the appropriate class of service according to the state of the Simplex CS_CTL bit in the connect-request frame:
- if Dedicated Duplex (Simplex=0), the CR shall use Class 1 SOF delimiters.
- if Dedicated Simplex (Simplex=1), the CR shall use Class 2 SOF delimiters.
- If the CR receives the NTY Link_Control command (see 14.3.4.2), the CR shall end the existing connection by initiating the End-Connection protocol as defined in sections 12.5 and 22.4.3.

29.8.3 Fabric

- The Fabric shall transmit all ACK, F_BSY, and F_RJT frames in the appropriate class of service according to the state of the Simplex CS_CTL bit in the connect-request frame:
- If Dedicated Duplex (Simplex=0), the Fabric shall use Class 1 SOF delimiters.
- If Dedicated Simplex (Simplex=1), the Fabric shall use Class 2 SOF delimiters.
- The Fabric shall transmit the NTY Link_Command frame when Camp-On connect-request is queued waiting for the destination N_Port to become available if the destination N_Port supports intermix.
- If the Fabric is unable to service the connect-request due to Fabric conditions such as queue full, the Fabric shall discard the connect-request frame, transmit the F_BSY as the response to the connect-request, and take no further action to make the connection. The F_BSY shall use the appropriate SOF delimiter and an EOFt delimiter.

- If the Fabric receives the End Link_Control command and has not routed the connect-request to the CR, the Fabric shall remove the connect-request off the queue and discard it.
- If the Fabric receives the End Link_Control command but has already routed the connect-request to the CR, the Fabric shall issue a F_RJT and establish the connection when it becomes possible.

29.8.4 Multiple requests

- If both Stacked Connect-request and Camp-On requests are indicated for the same connection in SOFc1, then the Fabric shall reject the request by issuing a F_RJT with a reason code = protocol error.

30 Stacked Connect-request

30.1 Introduction

The ability to support Stacked Connect-request is indicated by a Fabric during the Fabric Login procedure and invoked by a N_Port through the CS_CTL Stacked Connect-request bit of the connect-request frame. It is invoked on a connection by connection basis. The CR_TOV timer defined in Clause 29 is used by the Connection Initiator (CI), the Connection Recipient (CR), and the Fabric to time the connection establishment procedure.

NOTE - No new delimiters are required.

30.2 FC-2 Mechanisms

The FC-2 mechanisms used for this function are:

- Stacked Connect-request
- Class Specific Control field (CS_CTL)
- CR_TOV: Connection request time out value.

30.3 Applicability

The Stacked Connect-request is an optional function. If this function is invoked, the use of the CR_TOV is mandatory.

Stacked Connect-request is applicable to Class 1 Dedicated Duplex and Dedicated Simplex connections.

30.4 Communications Model

The ability of the Fabric to support Stacked Connect-request is indicated during the Fabric Login procedure through the service parameters supplied by the Fabric. The Stacked Connect-request function is invoked by the CI through the Stacked Connect-request CS_CTL bit in the Frame Header of the connect-request frame. Thus the Stacked Connect-request function is invoked on a connection by connection basis under control of the CI.

30.5 Requirements

30.5.1 Connection Initiator

The CI shall support:

- Class 1
- Stacked Connect-request CS_CTL bit
- CR_TOV timer.
- Stacked Connect-request in Login

30.5.2 Connection Recipient

The CR shall support:

- Class 1
- CR_TOV timer.

30.5.3 Fabric

The Fabric shall support

- Class 1
- Stacked Connect-request CS_CTL bit
- CR_TOV timer.
- Stacked Connect-request in Login

30.6 Login

The ability to support the Stacked Connect-request function by the Fabric is indicated during Fabric Login in the F_Port Class Service parameters (see 17.7.4.2), as stated in 17.7. The N_Port service parameters interchanged during Fabric Login (see 17.6.7.2) for the Stacked Connect-request function are not used.

30.7 Stacked Connect-request Invoked

The CI shall invoke the Stacked Connect-request function on a connection by connection basis. To invoke Stacked Connect-request, the CI shall transmit a connect-request frame containing the SOFc1 delimiter and setting the CS_CTL Stacked Connect-request = 1. If Stacked Connect-request = 0 on a connect-request frame, the Stacked Connect-request function is not invoked.

30.8 Timer

Stacked Connect-request utilizes both the CR_TOV timer and the E_D_TOV timer.

30.8.1 CR_TOV

The CR_TOV timer shall be used by the Fabric to define the maximum time period that the Fabric shall hold a connect-request frame when Stacked Connect-request is invoked and the destination N_Port is engaged in a prior connect. The Fabric shall attempt to establish the requested connection within the CR_TOV time out period. If the CR_TOV timer expires before the Fabric transmits the connect-request to the destination N_Port, the Fabric shall discard the connect-request and transmit a F_BSY to the source N_Port.

30.8.2 E_D_TOV

The E_D_TOV timer shall be used by the Fabric to define the maximum time period that the Fabric may hold a connect-request when Stacked Connect-request is not invoked.

30.9 Rules

30.9.1 Connection Initiator

- To invoke Stacked Connect-request, the CI shall transmit a connect-request frame containing the SOFc1 delimiter and setting the CS_CTL Stacked Connect-request = 1.
- If the Fabric indicated support for Stacked Connect-request in Fabric Login (see 17.7.4.2 and Table 208), the CI shall wait a time out period as specified in 23.2.6.6.

30.9.2 Connection Recipient

- The CR shall wait a time out period as specified in. 23.2.6.6.

30.9.3 Fabric

- If a Fabric received a connect-request frame with the Stacked Connect-request CS_CTL bit 30 = 1, it shall invoke the rules for the Stacked Connect-request mode indicated in its service parameters during Fabric Login (see 17.7.4.2 and Table 208).
- The Fabric shall manage CR_TOV timer as specified in. 23.2.6.6.

30.9.4 Multiple requests

- If both Stacked Connect-request and Camp-On requests are indicated for the same connection in SOFc1, then the Fabric shall reject the request by issuing a F_RJT with a reason code = protocol error.

31 Buffered Class 1 Service

Buffered Class 1 extends Class 1 Service to allow N_Ports of differing data rates to establish and make use of Class 1 Service. However, only maximal bandwidth rather than full bandwidth is guaranteed between the N_Ports. The data rate translator facility, if present, shall be located in the Fabric. The Fabric shall not preclude the use of Buffered Class 1 Service between equal data rate N_Ports, nor shall the availability of Buffered Class 1 Service preclude normal unbuffered Class 1 usage. Buffered Class 1 may be used in combination with any of the other Class 1 features, i.e., both types of Stacked Connect-request, Camp-On, Dedicated Simplex and Unidirectional or Bi-directional Duplex type connections.

When Buffered Class 1 Service is enabled, a N_Port can freely elect to use or not use Buffered Class 1 each time a Class 1 Connection is established. Buffered and unbuffered Class 1 connect-requests can be mixed without any restrictions even if Stacked Connect-requests or Camp-On is enabled.

31.1 Fabric Login

The support for Buffered Class 1 Service is negotiated during Fabric Login (see 17.3). If a N_Port requests Buffered Class 1 Service and the attached F_Port supports it, only then is Buffered Class 1 Service enabled and available for use by the N_Port and Fabric.

Buffered Class 1 uses BB_Credit rules between the N_Port and the F_Port. The BB_Credit rules applies to all Class 1 frames transmitted by a N_Port, on an established Buffered Class 1 connection, but do not apply to any SOFi1 and SOFn1 delimited frames transmitted by the F_Port. Neither the Fabric nor the F_Port shall restrict the frame size of Class 1 SOFi1 and SOFn1 delimited frames below the 2 148-byte maximum frame size limit.

31.2 Applicability

Buffered Class 1 is an optional service that may be provided by the Fabric and N_Ports.

31.2.1 Topology

Buffered Class 1 is applicable to the following topologies:

- Fabric;
- Fabric attached Arbitrated Loop.

Both the point-to-point and the stand-alone Arbitrated Loop topology are restricted to a common data rate and therefore do not require Buffered Class 1 service.

31.3 Fabric guaranteed bandwidth

Buffered Class 1 Service provides a maximal bandwidth guarantee across the Dedicated Simplex or Duplex connection. The following ensures this guarantee:

- The internal Fabric path, used for an established Buffered Class 1 Dedicated connection shall not be used to carry any Class 2, 3 or 4 frames while the Buffered Class 1 Dedication connection exists.
- If intermix is enabled, shall Class 1 frames have priority over Class 2 and 3 Intermix frames.

31.4 Delimiters

Buffered Class 1 does not affect delimiter usage of the Class 1 service.

31.5 CS_CTL bit

CS_CTL bit 28 is used for Buffered Class 1 Request (BCR) (see 31.6).

If bit 28 in the CS_CTL field is = 1 then Buffered Class 1 is requested. If bit 28 in the CS_CTL field is = 0 then unbuffered, full bandwidth, Class 1 is requested. This bit is meaningful on all Class 1 frames.

31.6 Rules

The following rules apply specifically to Buffered Class 1. All other Class 1 rules still apply. In case of conflict, the rules defined in this clause shall take precedence over the rules defined in previous clauses of this document. Examples of the Class 1 frame and R_RDY flow for both Buffered and unbuffered Class 1 are shown in Figure 87.

31.6.1 Connection_Initiator N_Port

The N_Port shall set the Buffered Class 1 Request (BCR) bit in the CS_CTL field = 1 on the Class 1 connect frame transmitted to establish a Buffered Class 1 connection.

- The N_Port shall set the BCR bit in the CS_CTL field = 1 on all Class 1 frames transmitted to an established Buffered Class 1 connection.
- The N_Port shall set the BCR bit in the CS_CTL field = 1 on all Class 1 frames transmitted on an established Buffered Class 1 connection.
- The N_Port shall only transmit a Class 1 frame on an established Buffered Class 1 connection if permitted by BB_Credit rules (see 31.6.4).
- The N_Port shall obey all other rules defined for the established Class 1 connection type, i.e., the Unidirectional Duplex, Bi-directional Duplex or Dedicated Simplex rules.

31.6.2 Connection_Recipient N_Port

- The N_Port shall set the BCR bit in the CS_CTL field = 1 on all Class 1 frames transmitted on an established Buffered Class 1 connection.
- The N_Port shall only transmit Class 1 frames on an established Buffered Class 1 connection if permitted by BB_Credit rules.
- The N_Port shall obey all other rules defined for the established Class 1 connection type, i.e., the Unidirectional Duplex, Bi-directional Duplex or Dedicated Simplex rules.

31.6.3 F_Port

- The Fabric shall F_RJT any Buffered Class 1 Connect-requests, SOFc1 delimited frames, destined for N_Ports for which Buffered Class 1 is not enabled. The reason code shall be 'Fabric path not available'. This ensures compatibility with FC-FS compliant N_Ports.
- The Fabric shall discard any SOFi1 or SOFn1 delimited frames on an established Class 1 connection, if the BCR bit in the CS_CTL field is set = 1 and Buffered Class 1 is enabled but unbuffered Class 1 was indicated by the Connect frame.
- The Fabric may discard any SOFi1 or SOFn1 delimited frames on an established Buffered Class 1 connection, if the BCR bit in the CS_CTL field is set = 0.
- The Fabric may discard any SOFi1 or SOFn1 delimited frames on an established Buffered Class 1 connection, if BB_Credit rules are violated (see 31.6.4).
- The Fabric may F_RJT any Class 1 connect frames received from a N_Port for which Buffered Class 1 was not enabled. If the Fabric rejects the frame the reason code shall be 'Invalid CS_CTL field'.

- The Fabric response is unpredictable if an SOFi1 or SOFn1 delimited frames is received in which the BCR bit in the CS_CTL field is set 1 if Buffered Class 1 is not enabled.
- The F_Port shall return an R_RDY for every SOFi1 or SOFn1 delimited frame received on an established Buffered Class 1 connection,. The R_RDY shall be transmitted as receive buffers becomes available.
- SOFi1 and SOFn1 delimited frames received on an established Buffered Class 1 connection shall be forwarded by the Fabric to the destination N_Port unless discarded for one of the above reasons. The Fabric shall perform no address validation or routing on these frames. However, frame validation as specified by clause 11.6.2 shall be performed.

31.6.4 BB_Credit rules for Buffered Class 1

BB_Credit rules are largely unaffected by Buffered Class 1, except that a N_Port follows BB_Credit rules when it transmits SOFi1 and SOFn1 delimited frames on an established Buffered Class 1 connection. The BB_Credit rules for Buffered Class 1 are:

- N_Ports shall only transmit SOFi1 or SOFn1 delimited frames on established Buffered Class 1 connections if BB_Credit allows.
- Transmission of an SOFi1 or SOFn1 delimited frames on established Buffered Class 1 connections shall consume 1 BB_Credit.
- Reception of an R_RDY shall reclaim 1 BB_Credit.
- Unless the alternative credit model is in effect, as specified in FC-AL, the BB_Credit shall be limited to value obtained during Fabric Login.

31.6.5 Buffered Class 1 frame size

The rules determining Class 1 frame size shall remain unaffected by Buffered Class 1. The frame size rules for Buffered Class 1 are summarized below:

- SOFc1 delimited frames are limited in size to the lesser of the Fabric or the recipient N_Ports Receive Data_Field size.
- SOFi1 or SOFn1 delimited frames are limited in size only by the recipient N_Ports Receive Data_Field size.
- The Fabric shall truncate SOFi1 or SOFn1 delimited frames longer than the maximum allowed, 2112 Byte Receive Data_ Field size. Truncated frames shall be terminated by an EOFa end of frame delimiter.

31.6.6 Intermix

Intermix rules and behavior shall remain unaffected by Buffered Class 1. Class 2 and 3 frames may freely be intermixed if Intermix is enabled.

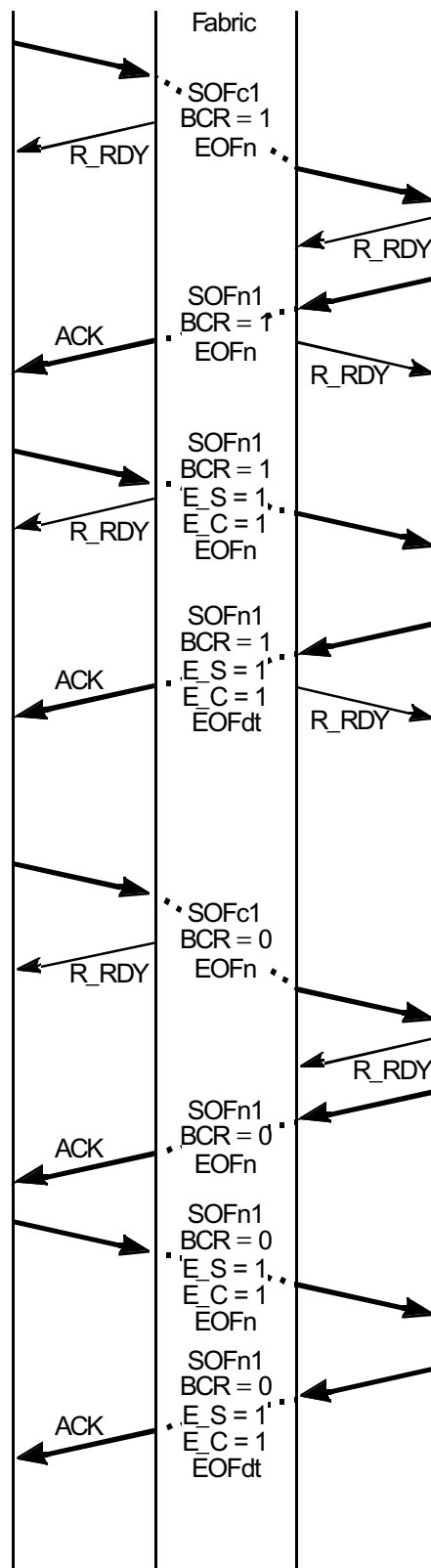


Figure 87 - Buffered and unbuffered Class 1 frame and R_RDY flow

32 Clock synchronization service

32.1 Introduction

32.1.1 References

See Informative Annex O for implementation details related to Clock Synchronization.

32.1.2 Applicability

Conventional network technologies utilize clock distribution protocols (e.g. Network Time Protocol) which synchronize the computer's time-of-day clock. Such protocols typically provide clock synchronization accuracies on the order of a few milliseconds with highly tuned versions producing accuracies on the order of 50 microseconds.

The synchronization that naturally occurs between transmitters and receivers in a Fibre Channel fabric offers a highly accurate reference for maintaining clock synchronization. If all delays are accounted for, accuracies on the order of a few nanoseconds become practical. The protocols defined in this clause allow clocks located within N_Ports to be synchronized to nanosecond accuracies.

32.1.3 Function

Clock Synchronization over Fibre Channel is attained through a Clock Synchronization Server, which contains a reference clock. The Server synchronizes Client's clocks to the reference clock on a periodic basis using either Primitive Sequences or ELS frames. The Server can be built into a fabric or it can be implemented as an independent node that services one or more Nx_Ports in either a Fabric or an Arbitrated Loop topology.

When all the Clients are synchronized with the Server, they will be synchronized with each other. By tagging data with the current value of their synchronized clock, one client can accurately exchange time sensitive data with another client.

32.2 ELS Command Service

See 15.5.39 for more information on Extended Link Services.

32.2.1 Assumptions

A simplifying assumption in the ELS method is that propagation delays over the medium are insignificant. This eliminates the need for the Server to calculate and maintain the media delay to each Client.

Very accurate clock synchronization will be accomplished without the use of media propagation delays through the techniques describe below. If the system requires even greater accuracy, "canned" propagation delays could be added in the Client's software or hardware.

32.2.2 Fabric Topology

The Server shall provide Clock Synchronization Service to one or more Client Ports. Each Client/Server pair must operate with common speeds and classes of service. (The Clients themselves can operate at different speeds and classes of service.)

The basic Model of the ELS method in a Fabric is shown in Figure 88.

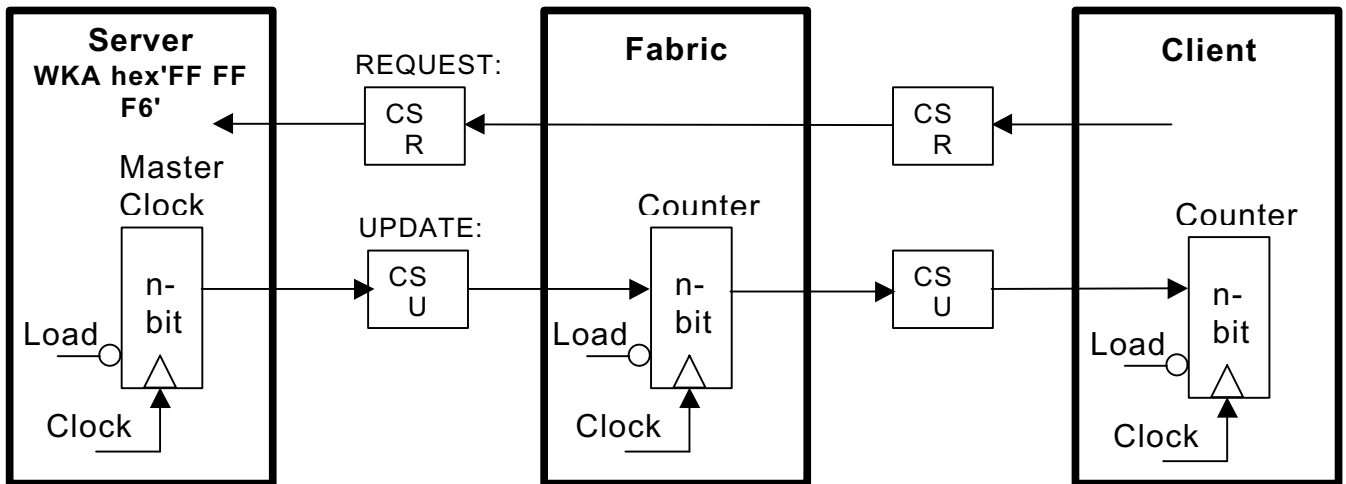


Figure 88 ELS Clock Sync Model – Fabric

There are three components in this Model: Server, Fabric, and Client.

32.2.2.1 Server Rules

The Clock Synchronization Server shall have an n-bit binary counter. This counter shall act as the Master Clock to the Clients.

The Server shall periodically issue the Clock Synchronization Update (CSU) ELS command to the Clients. When a CSU command is sent, the Server shall place the current value of the Master Clock in the payload.

The format of the CSU command is defined in Clause 32.2.4.4. The Clock value in the payload shall contain 8 bytes. The System Designer determines the number of bits that are actually meaningful in a given implementation. Any bits that are unused shall be set to 0.

The Server shall control the Clock Sync update rate. All Clients shall be updated at the same rate.

The Server is at Well-Known Address hex 'FF FF F6'. The Server may be physically located in a Fabric or a N_Port.

32.2.2.2 Server Options

Depending on the implementation, the Server may receive the Clock Synchronization Request (CSR) ELS command from Clients that wish to initiate Clock Sync Service. The format of the CSR command is defined in Clause 32.2.4.3. When the Server accepts the CSR command, it shall notify the Client that Clock Sync Service is enabled.

Although N_Port or Fabric Login is not required to use the CSR or CSU commands, if Login is used the Clock Synchronization ELS Capable bit in the Initiator Control section of the N_Port Class Service Parameters shall be used to indicate whether the server is capable of supporting Clock Synchronization.

32.2.2.3 Fabric Rules

A Fabric shall have its own n-bit binary counter.

When a CSU is received, the Fabric shall load its counter with the incoming value in the payload of the CSU command, regardless of the content of the D_ID field of the header. The Fabric shall then transmit a modified CSU command to the D_ID specified in the received CSU command. The Fabric shall place the current value of its counter in the payload of the outgoing CSU command and shall update the CRC value. All other elements of the outgoing CSU frame shall be the same as in the incoming CSU frame.

32.2.2.4 Client Rules

A Client shall have an n-bit binary counter.

When a CSU is received, the Client shall load its counter with the incoming value in the payload of the CSU command. The time value used by the Client for purposes of time stamping may be the raw counter value itself; or it may be a filtered version of the counter value. The filtered version may provide such benefits as monotonicity, lack of abrupt changes in value, reduced RMS error, etc.

32.2.2.5 Client Options

Depending on the implementation, the Client may issue the Clock Synchronization Request (CSR) ELS command to the Server to initiate Clock Sync Service. The format of the CSR command is defined in Clause 32.2.4.3. When the Server accepts the CSR command, it shall notify the Client that Clock Sync Service is enabled.

Although N_Port or Fabric Login is not required to use the CSR or CSU commands, if Login is used the Clock Synchronization ELS Capable bit in the Recipient Control section of the N_Port Class Service Parameters shall be used to indicate whether the client is capable of supporting Clock Synchronization.

32.2.3 Loop Topology

The Server shall provide Clock Synchronization Service to one or more Client Ports. Each Client/Server pair must operate with common classes of service. (The Clients themselves can operate with different classes of service.)

The basic Model of the ELS method in a Loop is shown in Figure 89.

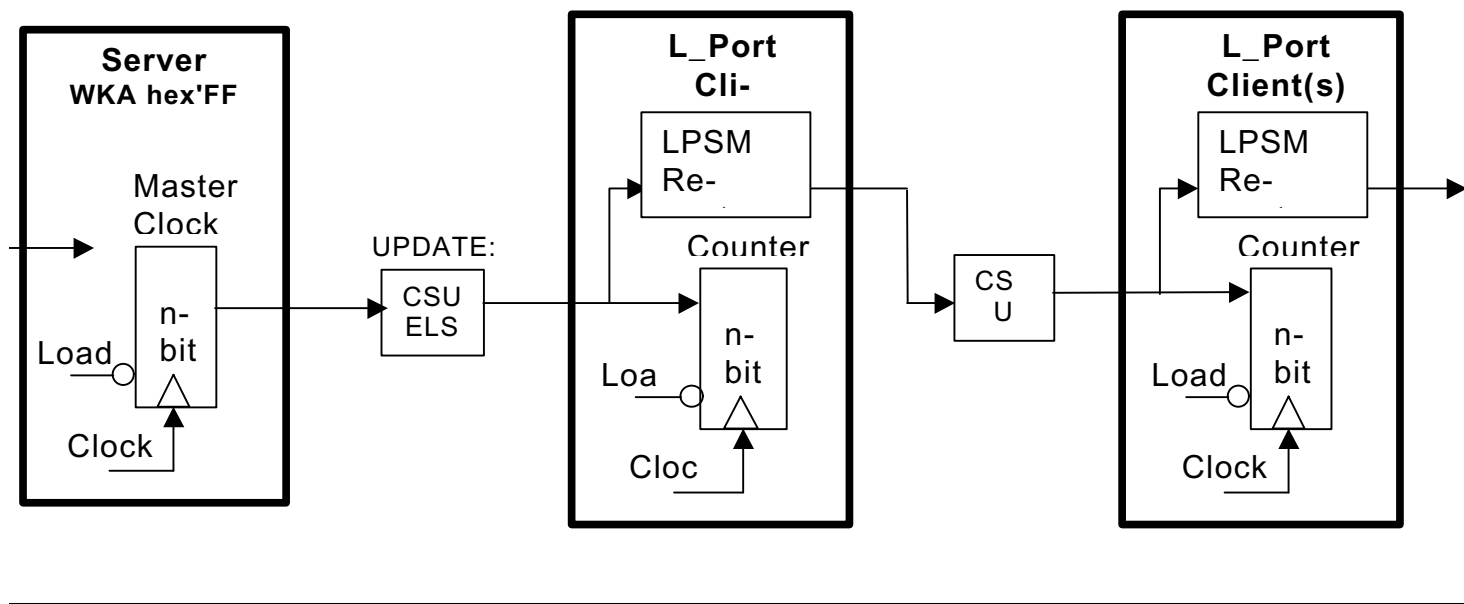


Figure 89 ELS Clock Sync Model – Loop

There are two components in this Model: L_Port Server and L_Port Client(s).

32.2.3.1 L_Port Server Rules

The Clock Synchronization Server shall have an n-bit binary counter . This counter shall act as the Master Clock to the Clients.

The Server shall periodically issue the Clock Synchronization Update (CSU) ELS command to the Clients. When a CSU command is sent, the Server shall place the current value of the Master Clock in the payload.

The format of the CSU command is defined in Clause 32.2.4.4. The Clock value in the payload shall contain 8 bytes. The System Designer determines the number of bits that are actually meaningful in a given implementation. Any bits that are unused shall be set to 0.

The Server shall control the Clock Sync update rate. All Clients shall be updated at the same rate.

The Server is at Well-Known Address hex 'FF FF F6'.

32.2.3.2 L_Port Server Options

Depending on the implementation, the Server may receive the Clock Synchronization Request (CSR) ELS command from Clients that wish to initiate Clock Sync Service. The format of the CSR command is defined in Clause 32.2.4.3. When the Server accepts the CSR command, it shall notify the Client that Clock Sync Service is enabled.

Although N_Port or Fabric Login is not required to use the CSR or CSU commands, if Login is used the Clock Sync Capable bit in the Class Specific Login Service Parameters shall be used to indicate whether the server is capable of supporting Clock Synchronization.

32.2.3.3 L_Port Client Rules

A Client L_Port shall have an n-bit binary counter.

When a CSU is received, the Client shall load its counter with the incoming value in the payload of the CSU command.

32.2.3.4 L_Port Client Options

Depending on the implementation, the Client may issue the Clock Synchronization Request (CSR) ELS command to the Server to initiate Clock Sync Service. The format of the CSR command is defined in Clause 32.2.4.3. When the Server accepts the CSR command, it shall notify the Client that Clock Sync Service is enabled.

Although N_Port or Fabric Login is not required to use the CSR or CSU commands, if Login is used the Clock Sync Capable bit in the Class Specific Login Service Parameters shall be used to indicate whether the client is capable of supporting Clock Synchronization.

32.2.4 Use of FC-FS Constructs

32.2.4.1 Login/Logout

A Client does not need to log in with the Clock Sync Server in order to receive Clock Sync Updates using the ELS method. Neither the CSR or CSU ELS commands require Login.

32.2.4.2 Common Required FC Parameters

Class of Service

The Server shall support all Classes of Service supported by the Fabric Region to which it is attached. A Client port may communicate with the Server using any desired Class.

R_CTL Routing Bits

Routing Bits of the R_CTL field shall be 0010 indicating Extended Link_Data.

R_CTL Information Category

Information Category of the R_CTL field shall be 0010 for the CSR ELS command and 0001 for the CSU ELS command. It shall be 0011 for Accept or Reject responses to the CSR ELS command.

Type

The Type field shall be hex '01' indicating Extended Link Service.

32.2.4.3 Clock Synchronization Request (CSR) ELS command

The format of the CSR ELS command payload shall be as shown in Table 225.

Table 225 - CSR ELS Payload				
WORD	Bits 31-24	Bits 23-16	Bits 15-8	Bits 7-0
0	Command Code	hex '00'	hex '00'	hex '00'
1	Clock Sync Mode	Reserved	Reserved	Reserved

Command Code

hex '68'

Clock Sync Mode

hex '00' Enable clock synchronization service to this client

hex '01'- hex 'FE' Reserved

hex 'FF' Disable clock synchronization service to this client

The format of the CSR ELS accept payload shall be as shown in Table 226.

Table 226 - CSR ELS Accept Payload				
WORD	Bits 31-24	Bits 23-16	Bits 15-8	Bits 7-0
0	Command Code	hex '00'	hex '00'	hex '00'
1	Clock Sync Mode	Reserved	Reserved	Reserved

Reply Code

hex '02' Accept

Clock Sync Mode

hex '00' Clock synchronization service enabled to this client

hex '01' - hex 'FE' Reserved

hex 'FF' Clock synchronization service disabled to this client

The format of the CSR ELS reject payload shall be as shown in Table 227.

Table 227 - CSR ELS Reject Payload				
WORD	Bits 31-24	Bits 23-16	Bits 15-8	Bits 7-0
0	Reply Code	hex '00'	hex '00'	hex '00'
1	Reserved	Reason Code	Reason Explanation	Vendor Unique

Reply Code

hex '01' Reject

Reason Code and Reason Explanation

See 15.6.2

32.2.4.4 Clock Synchronization Update (CSU) ELS command

The format of the CSU ELS command payload shall be as shown in Table 228. Note that there is no reply to the CSU ELS command.

Table 228 - CSU ELS Payload				
WORD	Bits 31-24	Bits 23-16	Bits 15-8	Bits 7-0
0	Command Code	hex '00'	hex '00'	hex '00'
1	Clock Count Byte 0 (MSB)	Clock Count Byte 1	Clock Count Byte 2	Clock Count Byte 3
2	Clock Count Byte 4	Clock Count Byte 5	Clock Count Byte 6	Clock Count Byte 7 (LSB)

Command Code

hex '69'

Clock Count

Byte 0 Counter value, byte 0 (MSB)

Byte 1 Counter value, byte 1

Byte 2 Counter value, byte 2

Byte 3 Counter value, byte 3

Byte 4 Counter value, byte 4

Byte 5 Counter value, byte 5

Byte 6 Counter value, byte 6

Byte 7 Counter value, byte 7 (LSB)

The bit values are derived from clock frequencies that are used in 1Gbaud Fibre Channel and shall be defined as follows:

The value of the Bit 7 in Word 2 shall be equal to 1/106.25MHz, roughly 9.4ns. Every other bit value is a binary multiple of this value. The next most significant bit is 2x that value, or 18.8ns. The next least significant value is ½ that value, or 4.7ns. The overall least significant bit is 73.5ps. The overall range that can be represented is 1.36×10^9 sec, approximately equal to 43 years.

The Clock Count value shall represent the time at which the most significant bit was placed on the link by the CSU ELS originator.

32.3 Primitive Signal Service

Primitive Signal Service for Clock Synchronization provides a low cost, no overhead, very accurate solution for real-time clock synchronization. This service is compatible with all Fibre Channel topologies, point-to-point, arbitrated loop, and fabric based networks.

32.3.1 Communication Model

Figure 90 illustrates the protocol for synchronizing client's real-time clocks with a clock synchronization server real-time clock. To accomplish this the server Periodically generates a synchronization event. The synchronization event is the transfer of clock synchronization primitives from the server to the Clients with the period between synchronization events controlled by the Server. Embedded within the clock synchronization primitives is the necessary data to update the client's real-time clock. For the client to receive an accurate real-time clock update in a fabric based network the fabric must, to the degree required by the application(s) of interest, compensate for the delay of moving the real-time clock value from the server to the clients.

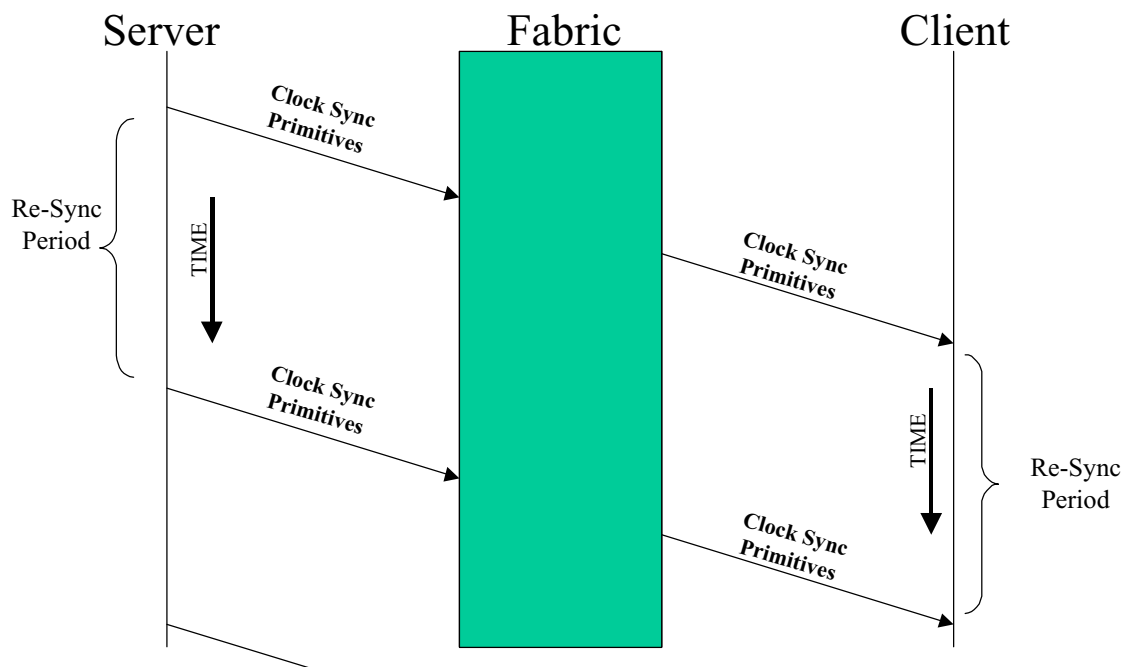


Figure 90 - Clock Synchronization Data Distribution

32.3.2 Requirements

The Clock Synchronization Server shall initiate clock synchronization events by substituting three synchronization primitives for a sequence of three consecutive IDLE primitives in the inter-Frame interval, as shown in Figure 91. This shall be done in such a way as to insure that the synchronization symbols are bracketed at both ends by at least two IDLE primitives.

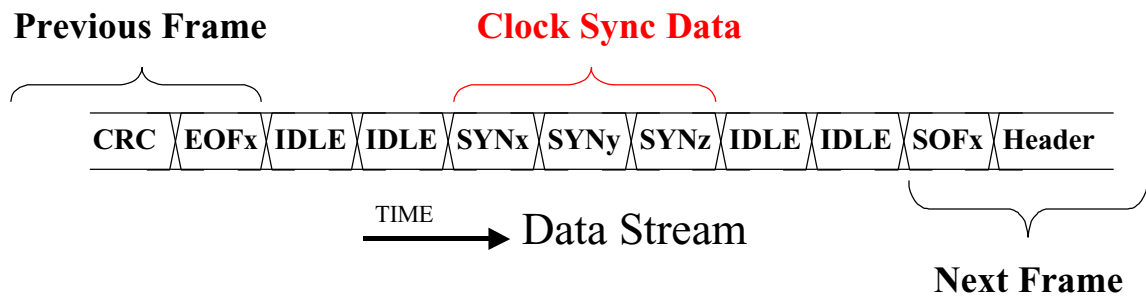


Figure 91 - Synchronization Primitive Substitution for IDLE Primitives in Inter-Frame Interval

Clock synchronization primitives shall consist of the ordered sets shown in Figure 92. The 14-bit values contained within each primitive (SYNx, SYNy, and SYNz) are the concatenation of two 7-bit values (X and X', Y and Y', Z and Z' respectively). Each 7-bit value shall have an equivalent neutral disparity Data Character (CS_X and CS_X', CS_Y and CS_Y', CS_Z and CS_Z') as shown in Table 229.

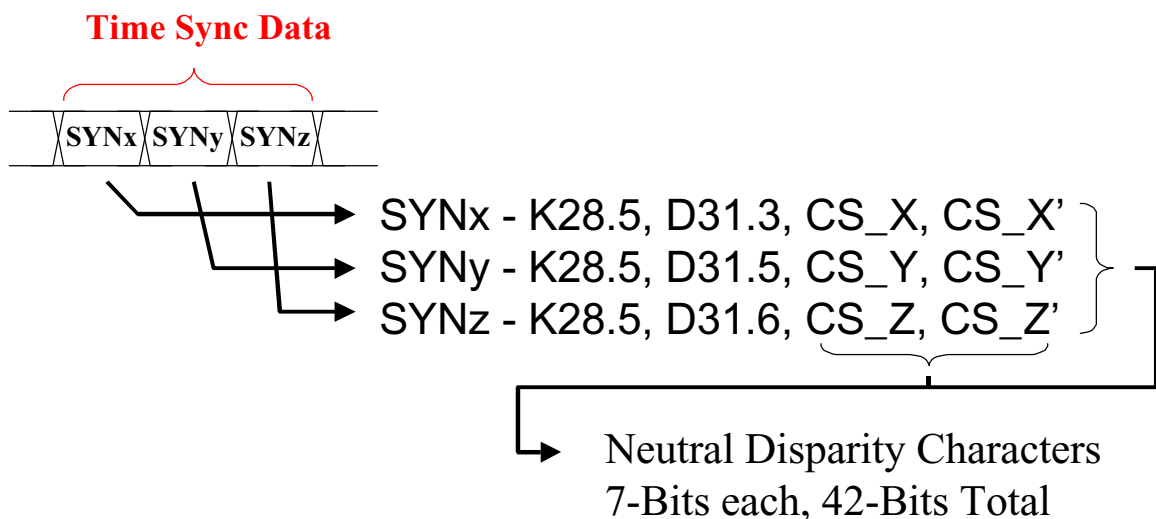


Figure 92 - Clock Synchronization Primitive Ordered Sets

Table 229 - Neutral Disparity Character Values									
Symbol: Dxx.y									Neutral Disparity Character (Hex)
xx	y								
	0	1	2	3	4	5	6	7	
00	(126)				(56)			(5)	00, 80, E0
01	(125)				(55)			(4)	01, 81, E1
02	(124)				(54)			(3)	02, 82, E2
03		(113)	(94)	(75)		(43)	(24)		23, 43, 63, A3, C3

Table 229 - Neutral Disparity Character Values									
Symbol: Dxx.y									Neutral Disparity Character (Hex)
xx	y								
	0	1	2	3	4	5	6	7	
04	(123)				(53)			(2)	04, 84, E4
05		(112)	(93)	(74)		(42)	(23)		25, 45, 65, A5, C5
06		(111)	(92)	(73)		(41)	(22)		26, 46, 66, A6, C6
07		(110)	(91)	(72)		(40)	(21)		27, 47, 67, A7, C7
08	(122)				(52)			(1)	08, 88, E8
09		(109)	(90)	(71)		(39)	(20)		29, 49, 69, A9, C9
10		(108)	(89)	(70)		(38)	(19)		2A, 4A, 6A, AA, CA
11		(107)	(88)	(69)		(37)	(18)		2B, 4B, 6B, AB, CB
12		(106)	(87)	(68)		(36)	(17)		2C, 4C, 6C, AC, CC
13		(105)	(86)	(67)		(35)	(16)		2D, 4D, 6D, AD, CD
14		(104)	(85)	(66)		(34)	(15)		2E, 4E, 6E, AE, CE
15	(121)				(51)			(0)	0F, 8F, EF
16	(120)				(50)			(133)	10, 90, F0
17		(103)	(84)	(65)		(33)	(14)		31, 51, 71, B1, D1
18		(102)	(83)	(64)		(32)	(13)		32, 52, 72, B2, D2
19		(101)	(82)	(63)		(31)	(12)		33, 53, 73, B3, D3
20		(100)	(81)	(62)		(30)	(11)		34, 54, 74, B4, D4
21		(99)	(80)	(61)		(29)	(10)		35, 55, 75, B5, D5
22		(98)	(79)	(60)		(28)	(9)		36, 56, 76, B6, D6
23	(119)				(49)			(132)	17, 97, F7
24	(118)				(48)			(131)	18, 98, F8
25		(97)	(78)	(59)		(27)	(8)		39, 59, 79, B9, D9
26		(96)	(77)	(58)		(26)	(7)		3A, 5A, 7A, BA, DA
27	(117)				(47)			(130)	1B, 9B, FB
28		(95)	(76)	(57)		(25)	(6)		3C, 5C, 7C, BC, DC
29	(116)				(46)			(129)	1D, 9D, FD
30	(115)				(45)			(128)	1E, 9E, FE
31	(114)				(44)			(127)	1F, 9F, FF
Total 134	13	19	19	19	13	19	19	13	
Legend: (x) = Decimal value of neutral disparity character									

32.3.2.1 Server Rules

The Server shall be capable of initiating clock synchronization events on a periodic basis or be disabled. The default synchronization event period shall be 1 second. The synchronization event period shall be settable from 1 microsecond to at least 60 seconds in 1-microsecond increments or set to 0.

The Server shall, at a minimum, maintain a 42-Bit real-time clock register. The real-time clock register's 42 transmit bits¹ shall be settable. The location of these 42-Bits in the real-time clock register shall be determined such that the least significant bit (LSB) aligns with the ELS clock count (33.2.4.4) bit location having the resolution required by the application(s) of interest. For example, if a 64-Bit real-time clock register is used and 1 microsecond resolution is required the LSB of the 42 transmit bits will be located at Byte-6 Bit-5, providing a resolution of 602.4ns. Bit-6 can not be used since this would provide a resolution of 1.2 microseconds.

The real-time clock register shall be maintained by incrementing its value at the rate that will toggle the LSB of the 42 transmit bits at the rate determined above.

The Server shall be addressed using Well-Known Address hex 'FF FF F6'. The Server may be physically located in a Fabric or with a N_Port.

32.3.2.2 Fabric Rules

Fabrics shall provide one port designated as the Fabric Clock Synchronization (FCS) Server port. All fabric ports shall be capable of periodically receiving clock synchronization primitives. Received clock synchronization primitives shall be interpreted the same as IDLE primitives by all ports except the FCS Server port. Following reception by the FCS Server port all fabric ports shall transmit clock synchronization primitives, except for the FCS Server port, using available inter-frame intervals. The real-time clock value transmitted by fabric ports shall be equal to the real-time clock value in the clock synchronization server, within the accuracy limits defined by the application(s) of interest.

32.3.2.3 Client Rules

Clients that support clock synchronization shall be capable of periodically receiving clock synchronization primitives. Clients that do not support clock synchronization shall interpret received clock synchronization primitives the same as IDLE primitives or ignore them.

Supporting clients shall, at a minimum, maintain a 42-Bit real-time clock register. Following reception of the three consecutive clock synchronization primitives the clients real-time clock register shall be loaded. The location of the received 42-Bits in the real-time clock register shall be determined such that the LSB aligns with the ELS clock count bit location having the resolution required by the application(s) of interest. The real-time clock register shall be maintained by incrementing its value at a rate such that the LSB of the 42-Bit value is toggled at a rate determined by the location of the LSB in the ELS clock count. For example: the received 42-Bits have a resolution of 1 microsecond and are to be placed into a 48-Bit real-time clock register with its LSB aligned with ELS clock count Byte-7 Bit-7. The 42 bits are located such that its LSB is at Bit 5 in the 48-Bit register with the register incremented at 106.25MHz.

¹ The 42-Bits out of the real-time clock register that will be used to generate the clock synchronization primitives.

33 Link Speed Negotiation

33.1 Speed Negotiation overview

The optional Speed Negotiation method may be used to enable Ports that are capable of multiple data transfer rates to establish in-band communications on an FC link (all Port types). The term “speed” as used in this clause refers to the bit transfer rate expressed in Baud. This method finds the highest speed common to the Ports and to the infrastructure connecting the ports. Each Port can support up to a maximum of 4 speeds in the negotiation process. The exact speeds are not specified. Different Ports may negotiate with different speed ranges up to a maximum of 4 speeds each and Speed Negotiation shall converge provided there is a least one common speed. The link quality for Speed Negotiation purposes is error free word synchronization for a minimum of 1000 consecutive words.

Because the link quality requirements for Speed Negotiation are not quite as stringent as for normal operation it is possible to complete Speed Negotiation yet have an excessive error rate in normal operation. Excessive error rates may also happen in non Speed Negotiating links for the same reasons. Means of determining bit error rate are out side the scope of Speed Negotiation. The corrective action to be taken in the event of excessive error rate in normal operation is also determined by means not specified by the Speed Negotiation requirements.

Note: One possible corrective action is to eliminate the speed that produced the excessive error rate from the speed capability list for the Port and to re-enter Speed Negotiation (return to state 10 in Figure 95 with a reduced list). This process may be repeated until a satisfactory speed is found or until all speeds are eliminated. Note that an eliminated speed may have been the only common speed useful for diagnostic communication over the link.

Speed Negotiation shall complete within 2 seconds. If the speed negotiation does not complete within 2 seconds no common speed exists.

Connection of Speed Negotiating Ports to an operating set of devices shall not disrupt the operation of those devices unless the Ports being connected can operate at the speed of those devices. For example, connection of Ports to a loop causes disruption of devices already in operation only at the operating speed.

Once a particular speed has been established, Speed Negotiation is not attempted unless a LINK FAILURE is detected or by means outside the scope of this clause.

33.2 Link physical architecture and requirements

The physical architecture of the link is assumed to be as shown in Figure 93.

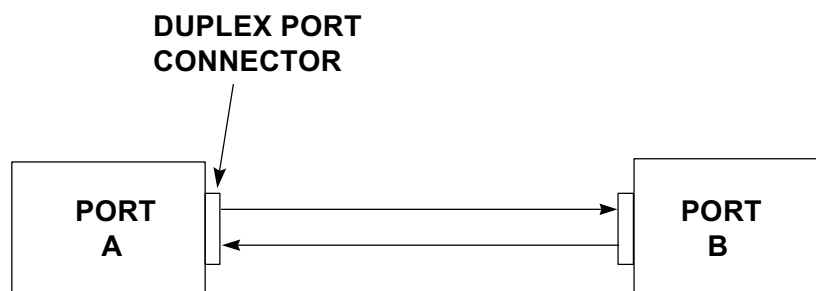


Figure 93 - Physical architecture of the Speed Negotiating Link

There are several points derived from this physical architecture that bear on the Speed Negotiation algorithm:

- The Speed Negotiation algorithm is specified for only one Port at a time. That is, when Port “A” is involved, the term transmitter applies only to the transmitter in Port “A” and the term receiver applies only to the receiver in Port “A”. Similar comments apply for all other features described. The algorithm may be executing on both Ports at the same time.
- Timing specifications apply at the duplex port connector and have a granularity of several milliseconds. Since the propagation time is short, typically nanoseconds, between the external connector and the circuitry in the port that is executing the algorithm the differences in propagation time behind the port connector with different hardware implementations are irrelevant to the execution of the Speed Negotiation algorithm.
- No requirements are explicitly placed by the algorithm on the means for controlling the transceiver speed capabilities; however, Ports implementing this algorithm shall NOT attain word sync unless the incoming signal is within $\pm 10\%$ of the receive rate set by the Port implementing the algorithm.
- A stable physical environment (no power cycles, no cable flexing, no transient noise sources, etc.) is expected during Speed Negotiation. Otherwise, Speed Negotiation may settle to a sub-optimum speed.
- The transmitter and receiver shall be capable of working at different speeds at the same time during Speed Negotiation.
- Each Port shall support up to a maximum of any 4 speeds.

33.3 Speed Negotiation requirements on L_Ports

Removal of an L_Port from a loop shall not cause Speed Negotiation to occur. This requirement maintains even if the removal of the L_Port allows negotiation of a higher common speed.

Hubs are set to a single speed during Speed Negotiation by some out-of-band means. This does not preclude intelligent hubs from being able to operate at multiple speeds.

33.4 Speed Negotiation algorithm

In this clause the following terminology is used:

Speeds

- “Speed list” refers to the set of speeds that are currently available for negotiation by the Port. The list may change during Negotiate_master. Speed changes and tests in the algorithm shall always be based on the speed list that is currently set.
- MAX refers to the maximum speed of the current speed list.
- TX refers to the present transmitter speed; RX refers to the receiver speed.
- Normally “Change TX” or “Change RX” instructions require the device to step down to the next lower rate in the list unless already at the lowest rate. In that case, the Port shall change to the highest rate. If the speed list has been reduced in state 2A, the speed going into state 29 may be outside the new list. In this case change to the new max speed.

Timing

- Pass sync_test? (states 11,21,27,34) requires that WORD_SYNC be maintained for at least 1000 words. Counting of 8b/10b invalid characters and disparity errors may be used for the 1000 word times to insure robustness, if available to the firmware. If counted, the number

of errors allowed shall be zero. If the number of errors is not zero, then WORD_SYNC (passed sync test?) is not considered to have occurred and a different speed is negotiated or the algorithm does not converge. In contrast, Sync? in state 31 is WORD_SYNC.

- The algorithm presumes that all Pass sync_test? decision states are evaluated at the beginning of the next major sample interval (t_{rxcycl}), but that all other instructions following or resulting from the decision are delayed until the next Pass sync_test? state is reached.
- Elapsed-time timers are compared against constants in several places.
 - t_{tx} , t_{neg} , and t_{sync} start where shown being (re)set to 0 in the algorithm.
 - t_{tx} is compared against t_{txcycl} .
 - t_{neg} is compared against $t_{fail..}$.
 - t_{sync} is compared against $t_{stbl..}$.
 - The R_T_TOV watchdog timer begins anytime WORD_SYNC is lost during Normal_ operation.

33.4.1 State machines

There are two state machines involved (1) an FC state machine that executes initialization and (2) a Speed Negotiation state machine that executes the speed negotiation algorithm. The Speed Negotiation algorithm shall complete successfully before the FC state machine is invoked.

33.4.2 Primitives

Either Lip (F7,F7) or OLS shall be the primitive transmitted for Speed Negotiation purposes.

33.4.3 Algorithm overview

Figure 94 shows an overview of the Speed Negotiation algorithm.

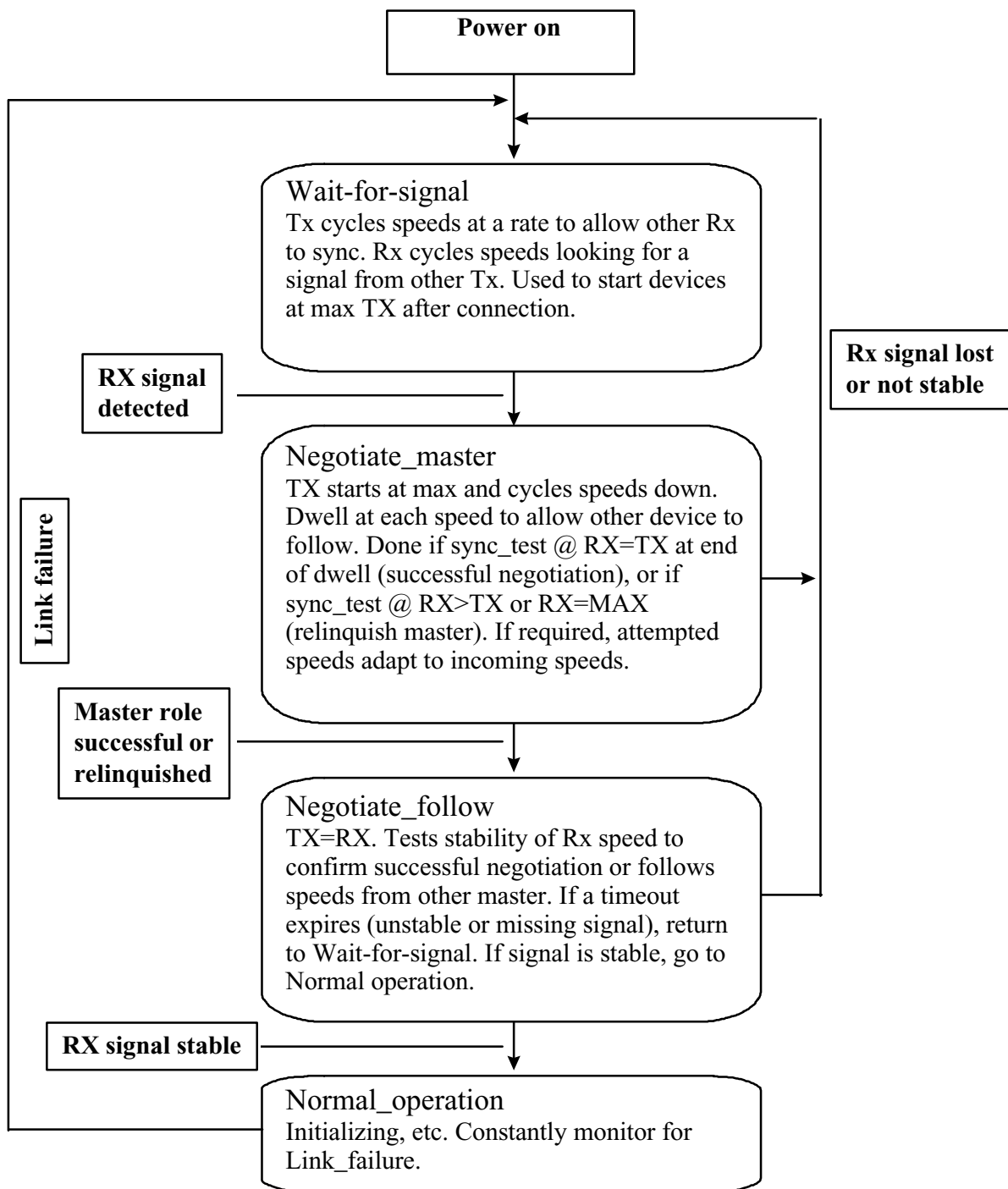


Figure 94 - Overview of the Speed Negotiation algorithm

33.4.4 Stage 1 - Wait_for_signal

Figure 95 shows the flowchart for the wait_for_signal stage.

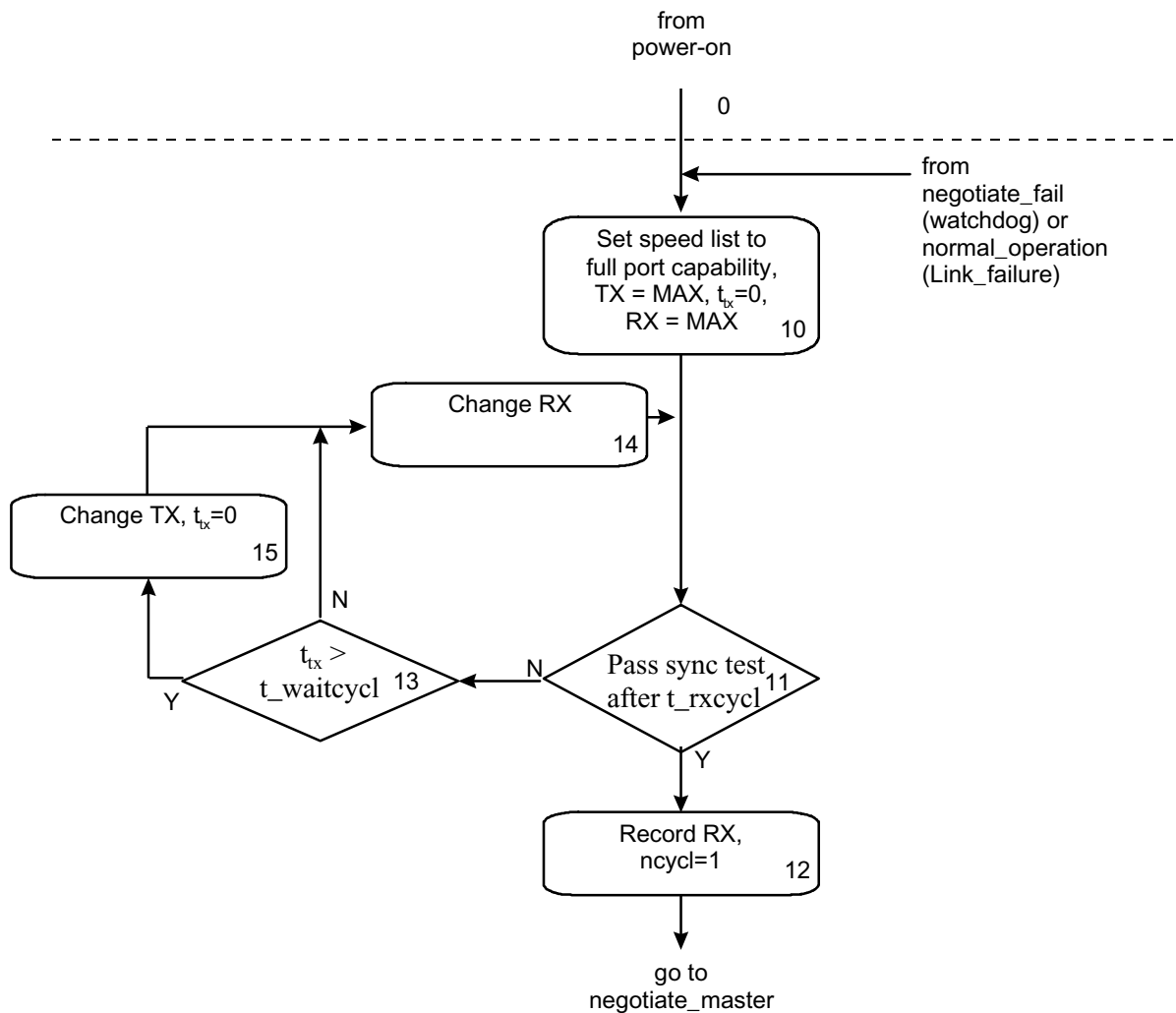


Figure 95 - Wait_for_signal flowchart

Description

- The device sets default parameters in state 10. States 11, 13, 14 cycle Rx speeds looking for the presence of an incoming signal from the other device that it can sync to. If found, RX is recorded, and the device moves onto Negotiate_master.

Tx speeds are cycled slowly compared to the time spent in 1 Rx speed which allows for the receiving side of the opposite Port to cycle through at least 4 Rx speeds. This gives the receiver time to sync at each speed. The Pass sync_test? is performed and the Rx speed is changed at the end of each Receiver Cycle Time (t_rxcycl)

RX_LOS, if implemented, may be used as an option to cycling Rx speeds (leave RX = MAX) and running the sync_test. However, if this is done, then ncycl = 0 in state 12 and no Rx speed is recorded. Tx speeds must still be cycled. If RX_LOS is used as a means to detect attached cables or Ports it may not accurately represent the state of the Tx connection on the Port.

33.4.5 Stage 2 - Negotiate_master (& Negotiate_fail)

Figure 96 shows the flowchart for the Negotiate_master (and Negotiate_fail) stages.

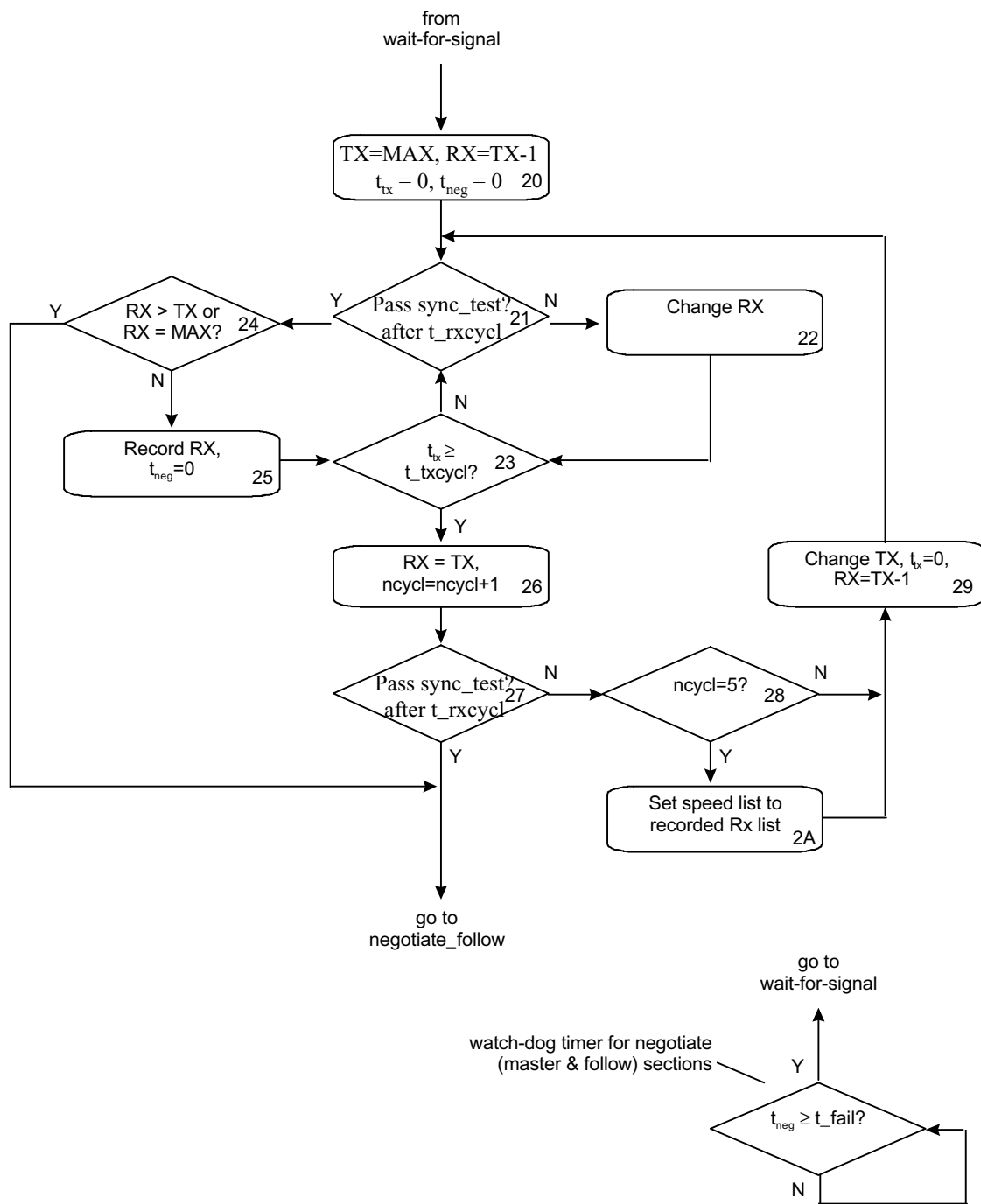


Figure 96 - Negotiate_master and Negotiate_fail flowchart

Description:

- The general operation of the algorithm is to start at the highest speed and step down until both devices agree on a speed. With this approach, lower speeds are tried only if higher speeds fail.
- RX is set to 1 speed lower than the transmission speed in States 20 and 29 (denoted as TX-1). This insures that at least the 3 possible speeds, other than the highest speed will have been observed during t_{txcycl} . The 4th or highest speed will be observed in State 26, insuring that that all 4 speeds from the other transmitter will have been observed.
- States 23 & 26-2A control TX. The idea is to force a transmit speed (starting at the highest speed) for sufficient time (t_{txcycl}) for the other device to sync and follow (Negotiate_follow is discussed below) and return TX back to the master. If NO from state 27, another (lower) Tx speed is attempted; if YES, the master role is assumed to be successful, and the algorithm moves to state 30 in Negotiate_follow. Note - there are 2 conditions that can cause YES in state 27: the other device is in follow mode as described above, or the other device is also in master mode transmitting at the same speed. If the latter, the master role is effectively relinquished to the other (later) master.
- If the other device has had time to transmit all of its possible speeds (maximum of 4 speeds), but master role has not completed, states 28 & 2A adapt the speed list to the incoming speed recorded by the Rx. This is usually not necessary unless the Port supports a very limited set of speeds.
- States 21-25 control RX. The idea here is to constantly monitor for an incoming rate that is higher than TX or equal to the maximum rate in the speed list. If found (sync_test passed), the device relinquishes its master role to the other device, transfers to follow role by changing its Tx rate to follow the faster or max Rx rate, and moves to state 30.
- A watchdog timer, t_{neg} , keeps track of time between passed sync_tests (state 27). If t_{neg} exceeds t_{fail} the Port returns to Wait-for-signal.

RX_LOS shall not be used during Negotiate_master stage.

33.4.6 Stage 3 - Negotiate_follow

Figure 97 shows the flowchart for the Negotiate_follow stage.

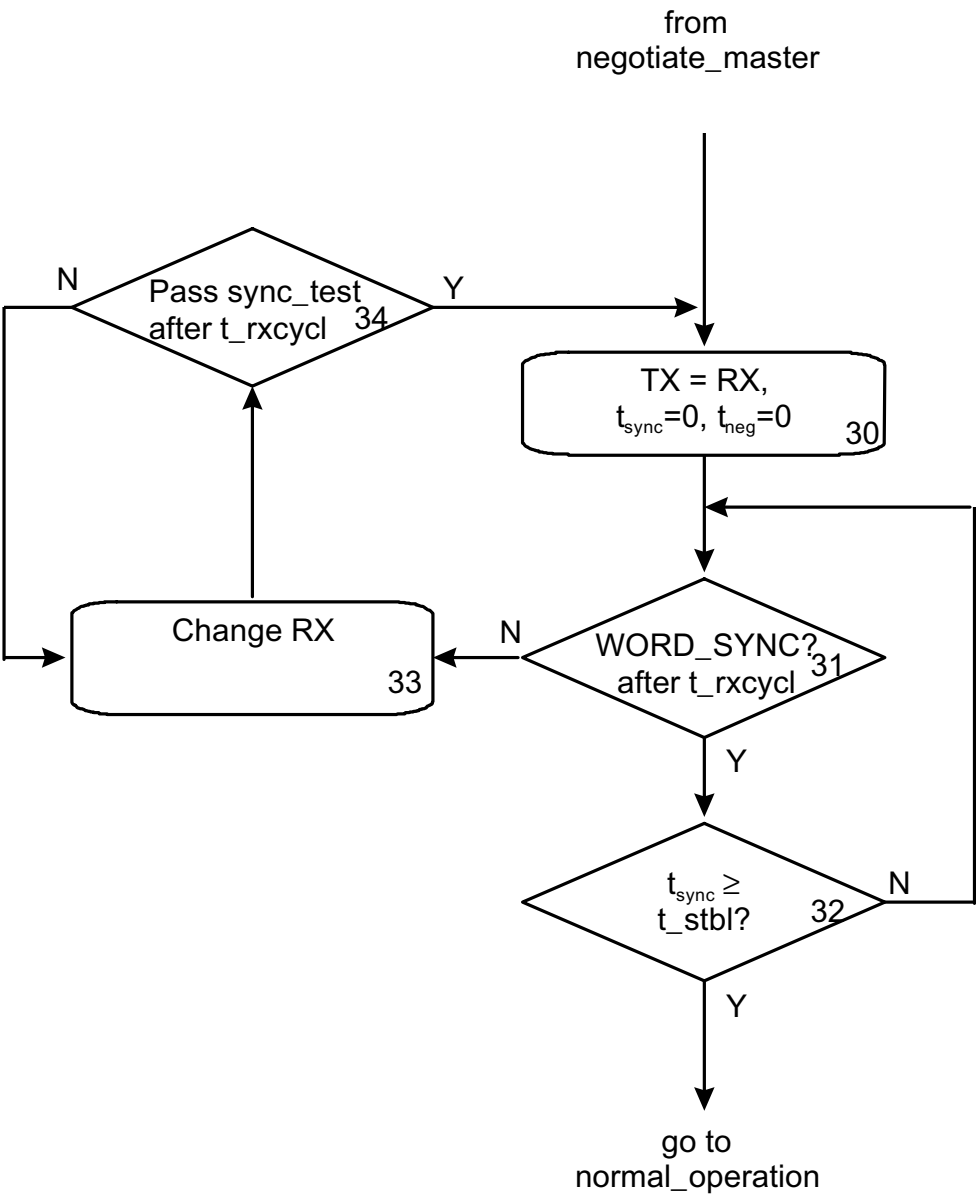


Figure 97 - Negotiate_follow flowchart

Description:

- A Port in the Negotiate_follow stage transmits its incoming speed. If the incoming signal speed changes and the pass sync test is true with the changed speed then the transmitted speed is changed to match.
- This continues until sync is held for at least t_stbl in state 31 (in the case where the other master is not driving other speeds). Note that during this time, TX and RX have been matched, allowing the other device to come to a YES decision in state 27. After t_stbl , the Port moves to state 40.
- If sync is lost (for example due to an incoming signal speed change), the Port seeks sync at another Rx speed. If obtained, TX is adjusted to follow the new RX, and the test for t_stbl starts over.
- The same watchdog time used in Negotiate_master is also used in Negotiate_follow.

RX_LOS shall not be used during Follow mode.

33.4.7 Stage 4 - Normal operation

Figure 98 shows the flowchart for the normal operation stage

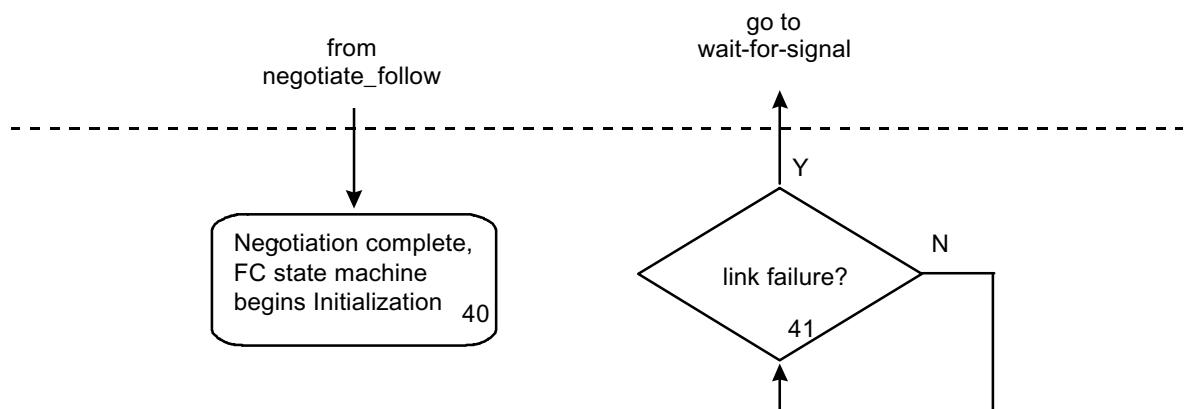


Figure 98 - Normal operation flowchart

Description:

- State 40 - negotiation is completed and initialization can begin.
- State 41 - whenever subsequent link failure occurs, negotiation re-enters at state 0.

33.4.8 Timing requirements

This section describes the timing requirements for the Speed Negotiation algorithm.

Receiver Cycle Time, t_rxcycl : the length of time the receiver is set to a particular speed while the receiver is actively changing speeds

Wait_Transmitter Cycle Time, $t_waitcycl$: the minimum length of time the transmitter is set to a particular speed in the Wait_for_signal stage.

Master_Transmitter Cycle Time, t_{txcycl} : The length of time the transmitter transmits at a given speed before setting its receive speed equal to its transmit speed for its final test in the Negotiate_master stage.

Speed stability time, t_{stbl} : Time required to ensure speed stability of the other Port

Maximum stage convergence time, t_{fail} : Time allowed for the algorithm to execute the stage

Table 230 - Timing requirements		
Timing parameter	Min (ms)	Max (ms)
t_{rxcycl}	16	30
$t_{waitcycl}$	122	152
t_{txcycl}	106	136
t_{stbl}	182	212
t_{fail}	1,500	

Annex A
(informative)
FC-1 service interface

FC-1 presents a decoded-Transmission-Word-oriented service interface to FC-2. This interface uses the same primitive types as those defined by the OSI model.

The FC-1 service interface provides a conceptual view of FC-1 function from the perspective of the FC-2 level and does not restrict FCS implementation flexibility. Figure A.1 graphically portrays the Fibre Channel conceptual model as applied to the FC-1 service interface.

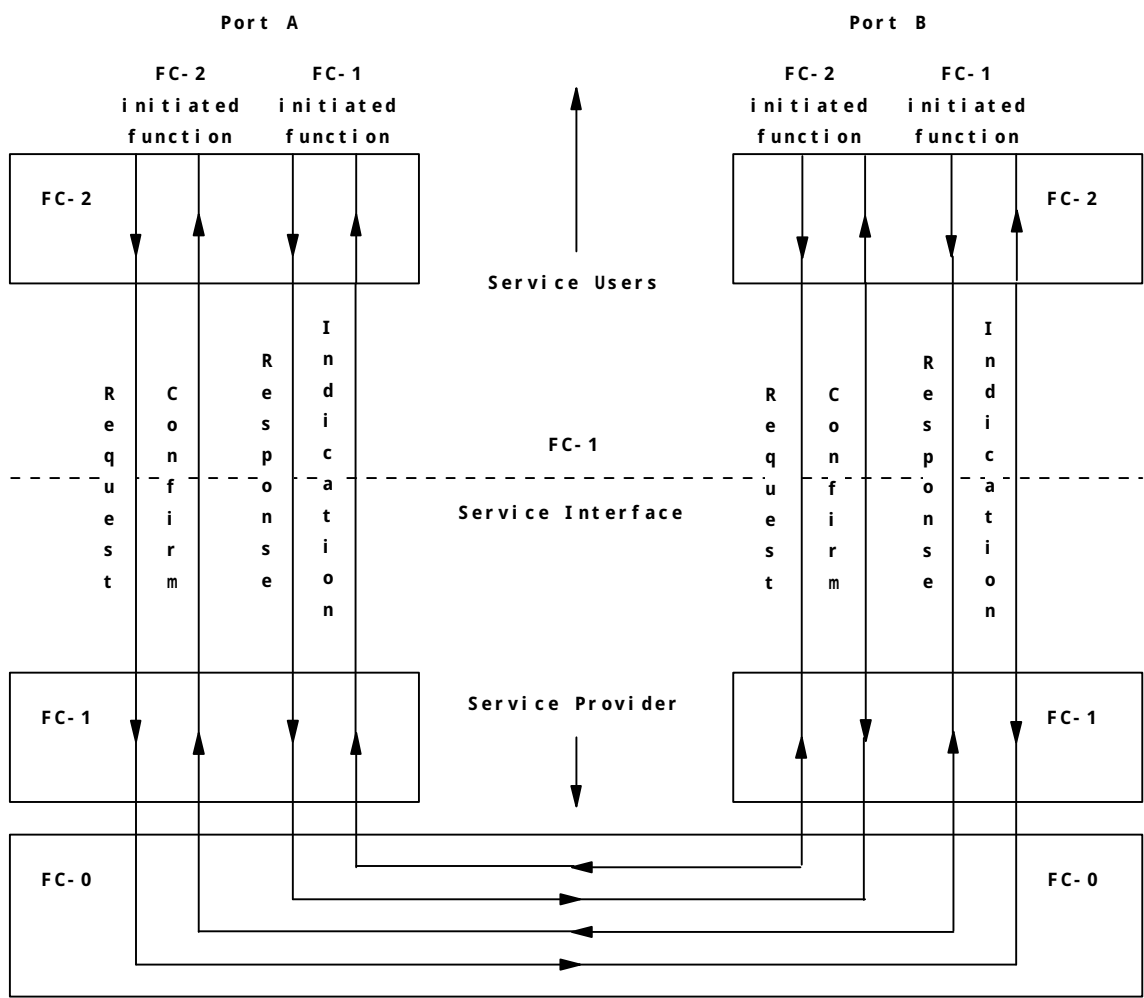


Figure A.1 - FC-1 service interface

Primitives are of four generic types:

- Request** The request primitive is passed from the FC-2 level to the FC-1 level to request that a service be initiated.
- Indication** The indication primitive is passed from the FC-1 level to the FC-2 level to indicate an

internal FC-1 event which is significant to the FC-2 level. This event may be logically related to a remote service request, or may be caused by an event internal to the FC-1 level.

Response The response primitive is passed from the FC-2 level to the FC-1 level to complete a procedure previously invoked by an indication primitive.

NOTE - The response primitive is not used by the FC-1 service interface.

Confirm The confirm primitive is passed from the FC-1 level to the FC-2 level to convey the results of one or more associated previous service request(s).

Figure A.2 illustrates the primitive combinations that are supported by the FC-1 service interface. Each primitive described in this annex is correlated to one of the subfigures (a), (b), (c), or (d) of Figure A.2. This figure also indicates the logical relationship of the primitive types. Primitive types that occur earlier in time and are connected by dotted lines in the diagrams are the logical antecedents of subsequent primitive types.

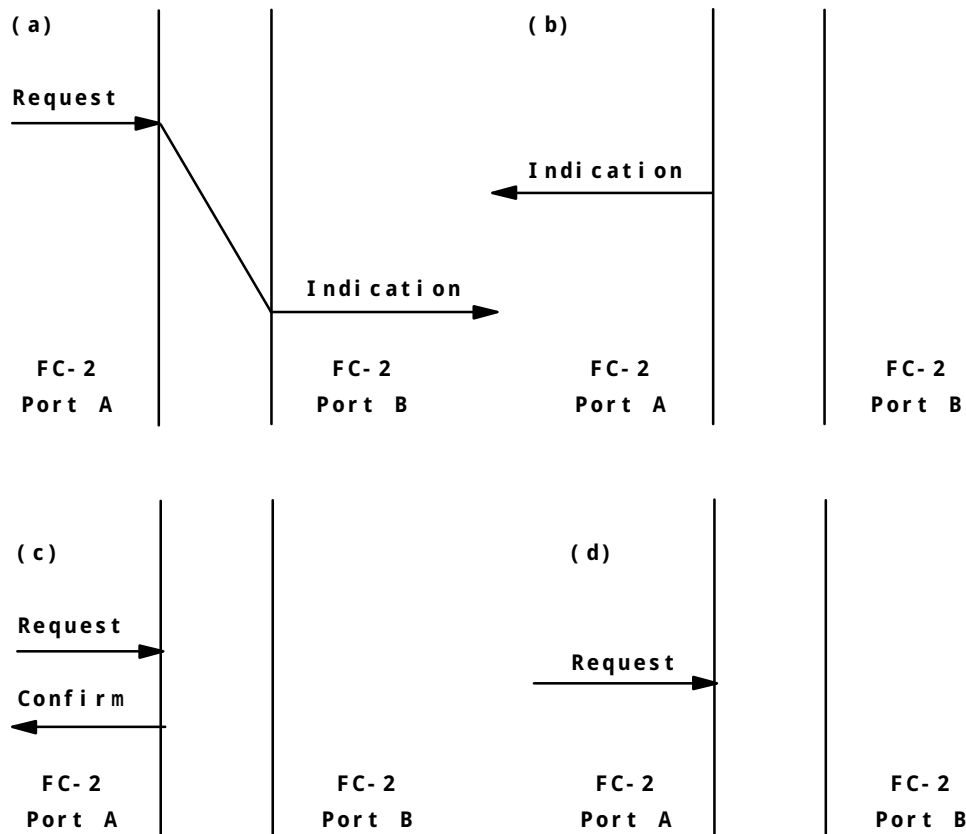


Figure A.2 - FC-1 service interface primitive combinations

The following service primitives are defined by the FC-1 service interface:

- Receiver_State
 - Indication

- FC-1_Transmitter_State
 - Indication
- FC-1_Data_Path_Width *
 - Request
 - Confirm
- FC-1_Transmitter_Clock
 - Indication
- FC-1_Data
 - Request
 - Indication
- FC-1_Receiver_Control
 - Request
- FC-1_Transmitter_Control
 - Request
- FC-1_Loopback_Control
 - Request
 - Confirm

* optional primitive

A.1 FC-1_Receiver_State.Indication

A.1.1 Function

This primitive is used by the FC-1 level to indicate the state of its receiver to an associated FC-2 level.

A.1.2 Semantics

FC-1_Receiver_State.Indication(State)

State	The functional state of the FC-1 level's receiver. The allowable values of this parameter are Loss_Of_Synchronization, Synchronization_Acquired, and Reset. The format of this parameter is not defined.
-------	--

A.1.3 When generated

A FC-1 level issues the FC-1_Receiver_State.Indication primitive with State = Loss_Of_Synchronization to indicate to the FC-2 level that its receiver has lost Synchronization on the signal received from its attached Fibre (see 6.1.2 for a description of Synchronization). This state may be entered from the Synchronization-Acquired or Reset states. While in the Loss-Of-Synchronization state, the FC-1 receiver is considered Operational but does not provide FC-1_Data.Indication primitives to the FC-2 level.

A FC-1 level issues the FC-1_Receiver_State.Indication primitive with State = Synchronization_Acquired to indicate to the FC-2 level that its receiver has achieved Synchronization on the signal received from its attached Fibre (see 6.1.2 for a description of Synchronization). This state may be entered from the Loss-Of-Synchronization state. While in the Synchronization-Acquired state, the FC-1 receiver is considered Operational and provides FC-1_Data.Indication primitives to the FC-2 level according to the rules described in K.6.

A FC-1 level issues the FC-1_Receiver_State.Indication primitive with State = Reset to indicate to the FC-2 level that it has initiated a reset condition at its receiver. This state may be entered from any other state. While in the Reset state, the FC-1 receiver is considered Not Operational and does not provide FC-1_Data.Indication primitives to the FC-2 level.

A.1.4 Effect on receipt

Receipt of this primitive with State = Synchronization_Acquired causes the FC-2 level to prepare to accept FC-1_Data.Indication primitives from the FC-1 level. Receipt of this primitive with State = Loss_Of_Synchronization notifies the FC-2 level that FC-1_Data.Indication primitives will not be forthcoming from the FC-1 level. Receipt of this primitive with State = Reset notifies the FC-2 level that the FC-1 level is incapable of reception because it has had a receiver reset condition imposed upon it, either internally or externally.

A.1.5 Additional comments

The Loss-Of-Synchronization state is the default state of an FC-1 level's receiver upon completion of Initialization.

This primitive is illustrated by Figure A.2 (b).

A.2 FC-1_Transmitter_State.Indication

A.2.1 Function

This primitive is used by the FC-1 level to indicate the state of its transmitter to an associated FC-2 level.

A.2.2 Semantics

FC-1_Transmitter_State.Indication (State)

State	The functional state of the FC-1 level's transmitter. The allowable values of this parameter are Failure, Not_Enabled, Open_Fibre, and Working. The format of this parameter is not defined.
-------	--

A.2.3 When generated

A FC-1 level issues the FC-1_Transmitter_State.Indication primitive with State = Failure to indicate to the FC-2 level that its transmitter has entered the Failure state as described by 12.3.5. This state may be entered from the Working state. While in the Failure state, the FC-1 transmitter is considered Not Operational, does not accept FC-1_Data.Request primitives from the FC-2 level, and does not provide FC-1_Transmitter_Clock.Indication primitives to the FC-2 level.

A FC-1 level issues the FC-1_Transmitter_State.Indication primitive with State = Not_Enabled to indicate to the FC-2 level that its transmitter is Operational but is prevented from transmitting a signal onto its associated Fibre. This state may be entered from the Working or Open-Fibre states. It may result from the processing of a FC-1_Transmitter_Control.Request primitive or from an external event (e.g., FC-2 link reset, removal of laser safety condition when a transmitter disable request remains outstanding). While in the Not-Enabled state, the FC-1 transmitter is considered Operational and accepts FC-1_Data.Request primitives from the FC-2 level in synchronization with the FC-1_Transmitter_Clock.Indication primitives provided by it to the FC-2 level. However, no signals resulting from the accepted FC-1_Data.Request primitives are transmitted onto its associated Fibre. If the FC-1 level is in Loopback mode, the encoded bit stream resulting from the accepted FC-1_Data.Request primitives is provided to the FC-1 receiver.

A FC-1 level issues the FC-1_Transmitter_State.Indication primitive with State = Open_Fibre to indicate to the FC-2 level that its transmitter is Operational but is prevented from transmitting a signal onto its associated Fibre. This state may be entered from the Working or Not-Enabled states. It results from detection by FC-0 laser safety procedures of a laser safety condition (i.e., a condition that requires that the transmitter cease transmission). While in the Open-Fibre state, the FC-1 transmitter is considered Operational and accepts FC-1_Data.Request primitives from the FC-2 level in synchronization with the FC-1_Transmitter_Clock.Indication primitives provided by it to the FC-2 level. However, no signals resulting from the accepted FC-1_Data.Request primitives are transmitted onto its associated Fibre. If the FC-1 level is in Loopback mode, the encoded bit stream resulting from the accepted FC-1_Data.Request primitives is provided to the FC-1 receiver.

A FC-1 level issues the FC-1_Transmitter_State.Indication primitive with State = Working to indicate to the FC-2 level that its transmitter is operating normally according to the error-monitoring procedure described by 12.3.5. This state may be entered from the Not-Enabled or Open-Fibre states. While in the Working state, the FC-1 transmitter is considered Operational and accepts FC-1_Data.Request primitives from the FC-2 level in synchronization with the FC-1_Transmitter_Clock.Indication primitives provided by it to the FC-2 level. Signals resulting from the accepted FC-1_Data.Request primitives are transmitted onto its associated Fibre. If the FC-1 level is in Loopback mode, the encoded bit stream resulting from the accepted FC-1_Data.Request primitives are also provided to the FC-1 receiver.

A.2.4 Effect on receipt

Receipt of this primitive with State = Working causes the FC-2 level to continue issuance of FC-1_Data.Request primitives to the FC-1 level (synchronous to FC-1_Transmitter_Clock.Indication primitives from the FC-1 level). Receipt of this primitive with State = Not_Enabled causes the FC-2 level to begin or continue issuance of FC-1_Data.Request primitives to the FC-1 level (synchronous to FC-1_Transmitter_Clock.Indication primitives from the FC-1 level). No transmitted signals result from the receipt of FC-1_Data.Request primitives. The transmitter remains Operational. Receipt of this primitive with State = Open_Fibre causes the FC-2 level to begin or continue issuance of FC-1_Data.Request primitives to the FC-1 level (synchronous to FC-1_Transmitter_Clock.Indication primitives from the FC-1 level). No transmitted signals result from the receipt of FC-1_Data.Request primitives. The transmitter remains Operational. Receipt of this primitive with State = Failure notifies the FC-2 level that FC-1_Transmitter_Clock.Indication primitives will not be forthcoming from the FC-1 level and that FC-1_Data.Request primitives will not be accepted by the FC-1 level. The FC-2 level may initiate maintenance activity.

A.2.5 Additional comments

The Not-Enabled state is the default state of an FC-1 level's transmitter upon completion of Initialization.

This primitive is illustrated by Figure A.2 (b).

A.3 FC-1_Data_Path_Width.Request

A.3.1 Function

This primitive is used to establish the path width of a variable-width data interface between an FC-1 and an FC-2 level.

A.3.2 Semantics

FC-1_Data_Path_Width.Request (Width)

Width The data path width of the interface between the FC-1 and FC-2 levels. The allowable values of this parameter are Byte, Half_Word, and Word. The format of this parameter is not defined.

A.3.3 When generated

A FC-2 level issues the FC-1_Data_Path_Width.Request primitive to specify to the FC-1 level the width of the data path to be used (Byte, Half_Word, or Word) between the FC-1 and FC-2 levels.

A.3.4 Effect on receipt

Receipt of the FC-1_Data_Path_Width.Request primitive causes a FC-1 level to attempt to establish the requested path width for its data interface to FC-2.

A.3.5 Additional comments

This primitive is not required for those implementations that use an FC-1 level with a fixed data path width. For those implementations that use an FC-1 level with a variable data path width, the attached FC-2 level issues this primitive to indicate the preferred path width.

A FC-1 level accepts an FC-1_Data_Path_Width.Request primitive or a fixed data path width is defined before FC-1_Data and FC-1_Transmitter_Clock primitives can be issued.

The Half_Word value of Width corresponds to a 2-byte interface. The Word value of Width corresponds to a 4-byte interface.

This primitive is illustrated by the request portion of Figure A.2 (c).

A.4 FC-1_Data_Path_Width.Confirm

A.4.1 Function

This primitive is used to accept or reject the establishment of the path width of a variable-width data interface between an FC-1 and an FC-2 level.

A.4.2 Semantics

FC-1_Data_Path_Width.Confirm (Status)

Status The results of the FC-1_Data_Path_Width.Request primitive. The allowable values of this parameter are Accept and Reject. The format of this parameter is not defined.

A.4.3 When generated

A FC-1 level issues the FC-1_Data_Path_Width.Confirm primitive to a FC-2 level to indicate the acceptance or rejection of a previously issued FC-1_Data_Path_Width.Request primitive.

A.4.4 Effect on receipt

Receipt of the FC-1_Data_Path_Width.Confirm primitive with Status = Accept informs the FC-2 level that path width establishment was successful. Receipt of the FC-1_Data_Path_Width.Confirm primitive with Status = Reject informs the FC-2 level that path width establishment was unsuccessful.

A.4.5 Additional comments

This primitive is not required for those implementations that use an FC-1 level with a fixed data path width. For those implementations that use an FC-1 level with a variable data path width, the attached FC-1 level must issue this primitive in response to an FC-1_Data_Path_Width.Request primitive.

This primitive is illustrated by the confirm portion of Figure A.2 (c).

A.5 FC-1_Transmitter_Clock.Indication

A.5.1 Function

This primitive is used by the FC-1 level to provide data transmission synchronization information, including an indication of Transmission Word alignment boundaries, to the FC-2 level.

A.5.2 Semantics

FC-1_Transmitter_Clock.Indication (Boundary)

Boundary An indication of the transmission boundary presented by the transmitter clock. The allowable values of this parameter are Word and Null. The format of this parameter is not defined.

A.5.3 When generated

A FC-1 level with a transmitter in the Working, Open-Fibre, or Not-Enabled states issues the FC-1_Transmitter_Clock.Indication primitive to provide a data transmission synchronization signal to the FC-2 level. Synchronous to the receipt of an FC-1_Transmitter_Clock.Indication primitive, the FC-2 level may choose to issue an FC-1_Data.Request primitive to the FC-1 level to present data to be transmitted (see B.6 for a description of the rules associated with data transmission).

The FC-1_Transmitter_Clock.Indication primitive is issued periodically to the FC-2 level with a frequency based upon the FC-0 transmission frequency and the data path width between the FC-1 and FC-2 levels. When a byte-wide data path is provided, the frequency of issuance = FC-0 transmission frequency / 10. When a half-word-wide data path is provided, the frequency of issuance = FC-0 transmission frequency / 20. When a word-wide data path is provided, the frequency of issuance = FC-0 transmission frequency / 40.

The FC-1 level uses the Boundary parameter to indicate whether the transmission data presented on an FC-1_Data.Request primitive that is associated with the FC-1_Transmitter_Clock.Indication primitive may be of the type that requires a Transmission-Word boundary. When Boundary = Word, any information type (Valid_Data_Byte, Delimiter, or Primitive_Signal) may be presented on the Type1 parameter of the associated FC-1_Data.Request primitive. When Boundary = Null, only the type Valid_Data_Byte may be presented on the Type1 parameter of the associated FC-1_Data.Request primitive.

Transmission-Word alignment is established by a FC-1 transmitter at the time it enters the working state and begins to transmit idle words as specified in B.6. The established Transmission Word alignment is indicated by the FC-1 level to the FC-2 level through the FC-1_Transmitter_Clock.Indication primitive. When a byte-wide data path width has been specified, the FC-1 level indicates a Transmission-Word boundary on every fourth FC-1_Transmitter_Clock.Indication primitive. When a half-word-wide data path width has been specified, the FC-1 level indicates a Transmission-Word boundary on every second FC-1_Transmitter_Clock.Indication primitive. When a word-wide data path width has been specified, the FC-1 level indicates a Transmission-Word boundary on every FC-1_Transmitter_Clock.Indication primitive.

A.5.4 Effect on receipt

Receipt of this primitive informs an FC-2 level that a request for information transmission can occur via issuance of an FC-1_Data.Request primitive.

A.5.5 Additional comments

This primitive is illustrated by Figure A.2 (b).

A.6 FC-1_Data.Request**A.6.1 Function**

This primitive is used by the FC-2 level to pass transmission requests to an associated FC-1 level.

A.6.2 Semantics

FC-1_Data.Request (Type1, Code1,
Type2, Code2,
Type3, Code3,
Type4, Code4)

Type The type of transmission information that is to be encoded and transmitted by the FC-1 level. The allowable values of this parameter are Valid_Data_Byte, Non_Repeating_Ordered_Set, and Repeating_Ordered_Set. The format of this parameter is not defined.

When a Non-Repeating Ordered Set or Repeating Ordered Set is indicated, a single Type field (Type1) is provided regardless of data path width. When a Valid Data Byte is indicated, a separate Type field is provided for each byte of the data path (i.e., Type1 is provided when a byte-wide data path width is specified, Type1 and Type2 are provided when a half-word-wide data path width is specified, and Type1, Type2, Type3, and Type4 are provided when a worldwide data path width is specified).

The values Non_Repeating_Ordered_Set and Repeating_Ordered_Set are restricted to the Type1 parameter. When present, the Type2, Type3, and Type4 parameters are set to the Valid_Data_Byte value.

When multiple Type parameters are presented in an FC-1_Data.Request primitive, the corresponding Transmission Characters are transmitted in the order given (i.e., Type1 corresponds to the first character received, followed by Type2, Type3 (if present), and Type4 (if present)).

Code The Valid Data Byte or Ordered Set that is to be encoded and transmitted by the FC-1 level. The contents of the Code parameter are defined as follows:

- When Type = Valid_Data_Byte, the Code parameter contains information in a format not defined. Standard which indicates the Valid Data Byte.
- When Type = Non_Repeating_Ordered_Set the Code parameter contains information in a format not defined. which indicates the Non-Repeating Ordered Set. All Ordered Sets specified in 11.4, "Ordered Sets" may be specified as a Non-Repeating Ordered Set.
- When Type = Repeating_Ordered_Set, the Code parameter contains information in a format not defined. Repeating Ordered Set. FC-2 must guarantee that only Primitive Sequences and the Idle Primitive Signal are specified as Repeating Ordered Sets.

When a Repeating Ordered Set or Non-Repeating Ordered Set is indicated, a single Code field (Code1) is provided regardless of data path width. When a Valid Data Byte is indicated by Type, a separate Code field is provided for each byte of the data path (i.e., Code1 is provided when a byte-wide data path width is specified, Code1 and Code2 are provided when a half-word-wide data path width is specified, and Code1, Code2, Code3, and Code4 are provided when a word-wide data path width is specified).

When multiple Code parameters are presented in an FC-1_Data.Request primitive, the corresponding Transmission Characters are transmitted in the order given (i.e., Code1 corresponds to the first character received, followed by Code2, Code3 (if present), and Code4 (if present)).

A.6.3 When generated

An FC-2 level issues the FC-1_Data.Request primitive to the FC-1 level synchronous to a received FC-1_Transmitter_Clock.Indication primitive when it wishes to specify information to be encoded and transmitted by the FC-1 level. Information to be encoded is passed by the FC-2 level to the FC-1 level according to the following rules:

- a) When the data path width between the FC-1 and FC-2 levels is defined to be Byte, the information passed by the FC-1_Data.Request primitive may be a Valid Data Byte, a Non-Repeating Ordered Set, or a Repeating Ordered Set. If a Repeating Ordered Set or Non-Repeating Ordered Set is passed by the FC-1_Data.Request primitive, the FC-2 level does not issue an FC-1_Data.Request primitive synchronous to the following three received FC-1_Transmitter_Clock.Indication primitives.
- b) When the data path width between the FC-1 and FC-2 levels is defined to be Half_Word, the information passed by the FC-1_Data.Request primitive may be two Valid Data Bytes, a Non-Repeating Ordered Set, or a Repeating Ordered Set. If a Repeating Ordered Set or Non-Repeating Ordered Set is passed by the FC-1_Data.Request primitive, the FC-2 level does not issue an FC-1_Data.Request primitive synchronous to the following received FC-1_Transmitter_Clock.Indication primitive.
- c) When the data path width between the FC-1 and FC-2 levels is defined to be Word, the information passed by the FC-1_Data.Request primitive may be four Valid Data Bytes, a Repeating Ordered Set, or a Non-Repeating Ordered Set.

The Boundary parameter of the FC-1_Transmitter_Clock.Indication primitive may restrict what a FC-2 level is allowed to present on a FC-1_Data.Request primitive. When Boundary = Word, any information type (Valid_Data_Byte, Non_Repeating_Ordered_Set, or Repeating_Ordered_Set) may be presented on the Type1 parameter of the associated FC-1_Data.Request primitive. When Boundary = Null, only Valid_Data_Byte may be presented on the Type1 parameter of the associated FC-1_Data.Request primitive. When information types other than Valid_Data_Byte are presented, unpredictable transmitter behavior results and the transmitter enters the Failure state.

A.6.4 Effect on receipt

Receipt of a FC-1_Data.Request primitive by a FC-1 transmitter in the Not-Enabled or Open-Fibre state causes the FC-1 level to attempt to encode the data indicated by the primitive. If the FC-1 level is in Loopback mode, data are provided to the FC-1 receiver for decoding and presentation to the FC-2 level. No signal is transmitted onto the attached Fibre as the result of a FC-1_Data.Request primitive received by a FC-1 transmitter in the Not-Enabled state.

Receipt of a FC-1_Data.Request primitive by a FC-1 transmitter in the Working state causes the FC-1 level to attempt to encode and transmit the data indicated by the primitive onto its attached Fibre. If the FC-1 level is in Loopback mode, this data is also provided to the FC-1 receiver for decoding and presentation to the FC-2 level.

A FC-1 transmitter in the Working, Not-Enabled, or Open-Fibre state according to the following rules processes encoded information:

- a) Upon entering the Working state, the FC-1 transmitter must be presented with a proper FC-1_Data.Request primitive by FC-2.
- b) Upon receipt of a FC-1_Data.Request primitive specifying Repeating_Ordered_Set, the FC-1 transmitter begins transmitting and/or providing the specified encoded Ordered Set and continues to transmit and/or provides this Ordered Set until receipt of a subsequent FC-1_Data.Request primitive.
- c) Upon receipt of a FC-1_Data.Request primitive specifying Valid_Data_Byte or Non_Repeating_Ordered_Set the FC-1 transmitter encodes and transmits and/or provides the specified information. Receipt of such a FC-1_Data.Request primitive ends the continuous transmission and/or provision of Repeating Ordered Sets if a prior FC-1_Data.Request primitive has previously established such transmission and/or provision.

Receipt of a FC-1_Data.Request primitive not conforming to the rules described previously results in unpredictable FC-1 level behavior.

Encoded information is not transmitted and/or provided and an FC-1 transmitter in the Failure state does not accept FC-1_Data.Request primitives.

A.6.5 Additional comments

This primitive is illustrated by the request portion of Figure A.2 (a).

A.7 FC-1_Data.Indication

A.7.1 Function

This primitive is used by the FC-1 level to pass received and decoded data to an associated FC-2 level.

A.7.2 Semantics

FC-1_Data.Indication (Type1, Code1,
Type2, Code2,
Type3, Code3,
Type4, Code4)

Type The type of transmission information that has been received and decoded by the FC-1 level. The allowable values of this parameter are Valid_Data_Byte, Ordered_Set, Special_Code, Code_Violation, Invalid_Special_Code_Alignment, and Invalid_Beginning_Running_Disparity. The latter four values are associated with the detection of invalid Transmission Words as specified by "Invalid Transmission Word rules." Special Codes and Valid Data Bytes may be reported in conjunction with these values as defined by the rules specified in K.7.3. The format of this parameter is not defined.

When an Ordered Set is indicated, a single Type field (Type1) is provided regardless of data path width. When information other than an Ordered Set is indicated, a separate Type field is provided for each byte of the data path (i.e., Type1 is provided when a byte-wide data path width is specified, Type1 and Type2 are provided when a half-word-wide data path width is specified, and Type1, Type2, Type3, and Type4 are provided when a word-wide data path width is specified).

The values Ordered Set and Invalid_Beginning_Running_Disparity are restricted to the Type1 parameter. When present, the Type2, Type3, and Type4 parameters are set to

the Valid_Data_Byte, Code_Violation, Special_Code, or Invalid_Special_Code_Alignment value.

When multiple Type parameters are presented in an FC-1_Data.Indication primitive, the corresponding Transmission Characters were received in the order given (i.e., Type1 corresponds to the first character received, followed by Type2, Type3 (if present), and Type4 (if present)).

Code The information (Valid Data Byte, Ordered Set, or invalid information) that has been received and decoded by the FC-1 level. The contents of the Code parameter are defined as follows:

- When Type = Valid_Data_Byte, the Code parameter contains information in a format not defined. which indicates the Valid Data Byte.
- When Type = Special_Code, the Code parameter contains information in a format not defined. which indicates the Special Code.
- When Type = Ordered_Set, the Code parameter contains information in a format not defined. which indicates the Ordered Set.
- When Type = Invalid_Special_Code_Alignment, the Code parameter contains information in a format not defined. which indicates the Special Code that was detected in an invalid position in the second, third, or fourth character of the Transmission Word.
- When Type = Code_Violation, the Code parameter is not meaningful and is ignored.
- When Type = Invalid_Beginning_Running_Disparity, the Code parameter contains information in a format not defined. which indicates the otherwise valid Ordered Set which was received with improper Beginning Running Disparity.

When an Ordered Set or an indication of invalid Beginning Running Disparity is indicated, a single Code field (Code1) is provided regardless of data path width. When information other than an Ordered Set or an indication of invalid Beginning Running Disparity is indicated by Type, a separate Code field is provided for each byte of the data path (i.e., Code1 is provided when a byte-wide data path width is specified, Code1 and Code2 are provided when a half-word-wide data path width is specified, and Code1, Code2, Code3, and Code4 are provided when a word-wide data path width is specified). Note that the Code field associated with a Type field indicating Code_Violation, while present, is not meaningful.

When multiple Code parameters are presented in an FC-1_Data.Indication primitive, the corresponding Transmission Characters were received in the order given (i.e., Code1 corresponds to the first character received, followed by Code2, Code3 (if present), and Code4 (if present)).

A.7.3 When generated

A FC-1 receiver in the Synchronization-Acquired state issues the FC-1_Data.Indication primitive to pass received and decoded Transmission Words to the FC-2 level. When the FC-1 level is in normal mode, Transmission Words are received from the Fibre attached to the FC-1 receiver. When the FC-1 level is in Loopback mode, the FC-1 transmitter provides received Transmission Words.

Decoded Transmission Words that are determined to be valid by the FC-1 level are passed to the FC-2 level according to the following rules:

- a) When the data path width between the FC-1 and FC-2 levels is defined to be Byte, the information passed by the FC-1_Data.Indication primitive may be a Valid Data Byte, a Spe-

cial Code, or an Ordered Set (a Special Code is passed only when an Unrecognized Ordered Set is received and decoded by the receiver).

- b) When the data path width between the FC-1 and FC-2 levels is defined to be Half_Word, the information passed by the FC-1_Data.Indication primitive may be two Valid Data Bytes, a Special Code followed by a Valid Data Byte, or an Ordered Set (the Special Code / Valid Data Byte pair is passed only when an Unrecognized Ordered Set is received and decoded by the receiver).
- c) When the data path width between the FC-1 and FC-2 levels is defined to be Word, the information passed by the FC-1_Data.Indication primitive may be four Valid Data Bytes, a Special Code followed by three Valid Data Bytes, or an Ordered Set (the Special Code / Valid Data Byte / Valid Data Byte / Valid Data Byte string is passed only when an Unrecognized Ordered Set is received and decoded by the receiver).

Decoded Transmission Words that are determined to be invalid by the FC-1 level are passed to the FC-2 level according to the following rules:

- d) When the data path width between the FC-1 and FC-2 levels is defined to be Byte, the information passed by the FC-1_Data.Indication primitive may be one of the following: a Code Violation, an invalid Special Code alignment, or an indication of invalid Beginning Running Disparity. Preceding or subsequent FC-1_Data.Indication primitives associated with the invalid Transmission Word may contain Valid Data Bytes and/or Special Codes (note that an invalid Transmission Word as specified by "Invalid Transmission Word rules" may contain one or more valid Transmission Characters; it is these characters that are passed as Valid Data Bytes or Special Codes in the four FC-1_Data.Indication primitives which represent the invalid Transmission Word).
- e) When the data path width between the FC-1 and FC-2 levels is defined to be Half_Word, the information passed by the FC-1_Data.Indication primitive may be one or more of the following: a Code Violation, an invalid Special Code alignment, or an indication of invalid Beginning Running Disparity. Indications of invalid Beginning Running Disparity may be specified only in the Type1 parameter. Code Violations and invalid Special Code alignments may be specified in either or both of the Type1 and Type2 parameters. These values may be combined with a Valid Data Byte or Special Code value in one of the Type1 and Type2 parameters according to the contents of the received and decoded Transmission Word and the usage rules described in the definition of the Type parameter. Preceding or subsequent FC-1_Data.Indication primitives associated with the invalid Transmission Word may also contain Valid Data Bytes and/or Special Codes (note that an invalid Transmission Word as specified by "Invalid Transmission Word rules" may contain one or more valid Transmission Characters; it is these characters that are passed as Valid Data Bytes or Special Codes in the two FC-1_Data.Indication primitives which represent the invalid Transmission Word).
- f) When the data path width between the FC-1 and FC-2 levels is defined to be Word, the information passed by the FC-1_Data.Indication primitive may be one or more of the following: a Code Violation, an invalid Special Code alignment, or an indication of invalid Beginning Running Disparity. Indications of invalid Beginning Running Disparity may be specified only in the Type1 parameter. Code Violations may be specified in one or more of the Type1, Type2, Type3, and Type4 parameters. Invalid Special Code alignments may be specified in one or more of the Type2, Type3, and Type4 parameters. These values may be combined with Valid Data Byte and Special Code values in one or more of the Type1, Type2, Type3, and Type4 parameters according to the contents of the received and decoded Transmission Word and the usage rules described in the definition of the Type parameter (note that an invalid Transmission Word as specified by "Invalid Transmission Word rules" may contain one or more valid Transmission Characters; it is these characters that are passed as Valid Data Bytes or Special Codes in the FC-1_Data.Indication primitive which represents the invalid Transmission Word).

When information other than Ordered Sets or an indication of invalid Beginning Running Disparity is received and decoded by the FC-1 level, the FC-1_Data.Indication primitive is issued periodically to the FC-2 level with a frequency based upon the FC-0 transmission frequency and the data path width between the FC-1 and FC-2 levels. When a byte-wide data path is provided, the frequency of issuance = FC-0 transmission frequency / 10. When a half-word-wide data path is provided, the frequency of issuance = FC-0 transmission frequency / 20. When a word-wide data path is provided, the frequency of issuance = FC-0 transmission frequency / 40. When an Ordered Set or an indication of invalid Beginning Running Disparity is received by the FC-1 level, the FC-1_Data.Indication primitive is issued according to the rules described below.

Received and decoded information is passed to the FC-2 level by an FC-1 receiver in the Synchronization-Acquired state according to the following rules:

- g) Upon entering the Synchronization-Acquired state, the FC-1 receiver indicates receipt of the Ordered Set which allowed it to enter the Synchronization-Acquired state. When the Ordered Set is recognized, this indication is completed by issuing an FC-1_Data.Indication with Type = Ordered_Set and Code indicating the appropriate value. When the Ordered Set is unrecognized, this indication is completed by issuing one or more FC-1_Data.Indication primitives sufficient to represent the Unrecognized Ordered Set according to the defined data path width between the FC-1 and FC-2 levels.
- h) Upon receipt of an incoming bit stream representing a recognized Ordered Set, the FC-1 receiver indicates receipt of this Ordered Set by issuing an FC-1_Data_Indication with Type = Ordered_Set and Code set to the appropriate value.
- i) Upon receipt of an incoming bit stream representing an Unrecognized Ordered Set, the FC-1 receiver indicates receipt of this Ordered Set by issuing one or more FC-1_Data.Indication primitives sufficient to represent the Unrecognized Ordered Set according to the defined data path width between the FC-1 and FC-2 levels.
- j) Upon receipt of an incoming bit stream representing a data character that is not part of an Ordered Set, the FC-1 receiver indicates receipt of this character by issuing an FC-1_Data.Indication with Type = Valid_Data_Byte and Code set to the appropriate value. Additional Valid_Data_Byte entries are provided when the data path width between the FC-1 and FC-2 levels is Half_Word or Word.
- k) Upon receipt of an incoming bit stream representing an invalid Transmission Word as specified by "Invalid Transmission Word rules," the FC-1 receiver indicates receipt of this word by issuing an FC-1_Data.Indication with Type indicating the appropriate condition or conditions according to the rules described previously in this annex.

Decoded information is not passed to the FC-2 level and a FC-1 receiver in the Loss-Of-Synchronization or Reset states does not issue FC-1_Data.Indication primitives.

A.7.4 Effect on receipt

Receipt of a FC-1_Data.Indication primitive by a FC-2 level causes the FC-2 level to accept the received and decoded data indicated by the primitive from the FC-1 level.

A.7.5 Additional comments

This primitive is illustrated by the indication portion of Figure A.2 (a).

A.8 FC-1_Receiver_Control.Request

A.8.1 Function

The FC-2 level to request that the FC-1 receiver be reset uses this primitive.

A.8.2 Semantics

FC-1_Receiver_Control.Request (Control)

Control The receiver control request made by the FC-2 level. The allowable value of this parameter is Reset. The format of this parameter is not defined.

A.8.3 When generated

A FC-2 level issues the FC-1_Receiver_Control.Request primitive with Control = Reset to request that the FC-1 level receiver be reset.

A.8.4 Effect on receipt

Receipt of this primitive with Control = Reset causes the FC-1 level to initiate a reset condition in its receiver. If the receiver is not already in the reset state, the FC-1 level concurrently issues an FC-1_Receiver_State.Indication primitive to the FC-2 level with State = Reset. The FC-1 level is incapable of providing FC-1_Data.Indication primitives to the FC-2 level while a reset condition exists in its receiver. When the receiver reset condition is exited, the FC-1 level issues an FC-1_Receiver_State.Indication primitive to the FC-2 level with State = Loss_Of_Synchronization.

A.8.5 Additional comments

A FC-1 reset condition may result in a request to the FC-0 receiver to attempt to reacquire bit synchronization.

This primitive is illustrated by Figure A.2 (d).

A.9 FC-1_Transmitter_Control.Request

A.9.1 Function

The FC-2 level to request that the FC-1 transmitter be enabled or disabled uses this primitive.

A.9.2 Semantics

FC-1_Transmitter_Control.Request (Control)

Control The enable control request made by the FC-2 level. The allowable values of this parameter are Enable and Disable. The format of this parameter is not defined.

A.9.3 When generated

A FC-2 level issues the FC-1_Transmitter_Control.Request primitive with Control = Enable to request that the FC-1 level transmitter be enabled.

A FC-2 level issues the FC-1_Transmitter_Control.Request primitive with Control = Disable to request that the FC-1 level transmitter be disabled.

A.9.4 Effect on receipt

Receipt of this primitive with Control = Enable causes the FC-1 level to enable its transmitter unless the transmitter is in one of the following conditions:

- The transmitter is in the Failure state.

- The transmitter has been placed in the Open-Fibre state as a result of detection by FC-0 laser safety procedures of a laser safety condition (i.e., a condition that requires that the transmitter cease transmission).

If the transmitter is not already enabled and it is not in one of the above conditions, the FC-1 level concurrently issues a FC-1_Transmitter_State.Indication primitive to the FC-2 level with State = Working. Otherwise, receipt of this primitive with Control = Enable by a transmitter does not result in a transmitter state change. However, if a previously-detected laser safety condition is present at the time of receipt, the state transition which results when the laser safety condition is removed may be affected (see 6.4).

Receipt of this primitive with Control = Disable causes the FC-1 level to disable its transmitter if not previously disabled. If the transmitter is in the Working state at the time of receipt, the FC-1 level concurrently issues a FC-1_Transmitter_State.Indication primitive to the FC-2 level with State = Not_Enabled. Otherwise, receipt of this primitive with Control = Disable by a transmitter does not result in a transmitter state change. However, if a previously detected laser safety condition is present at the time of receipt (i.e., the transmitter is in the Open-Fibre state), the state transition which results when the laser safety condition is removed may be affected (see 6.4).

A.9.5 Additional comments

This primitive is illustrated by Figure A.2 (d).

A.10 FC-1_Loopback_Control.Request

A.10.1 Function

This primitive is used by the FC-2 level to request that the FC-1 Loopback function be enabled or disabled.

A.10.2 Semantics

FC-1_Loopback_Control.Request (Control)

Control The Loopback control request made by the FC-2 level. The allowable values of this parameter are Enable and Disable. The format of this parameter is not defined.

A.10.3 When generated

A FC-2 level issues the FC-1_Loopback_Control.Request primitive with Control = Enable to request that the FC-1 Loopback function be enabled.

A FC-2 level issues the FC-1_Loopback_Control.Request primitive with Control = Disable to request that the FC-1 Loopback function be disabled.

A.10.4 Effect on receipt

Receipt of this primitive with Control = Enable causes the FC-1 level to enable its Loopback function unless the FC-1 transmitter or receiver is in a state not compatible with the Loopback function. When the Loopback function is enabled, information passed to the FC-1 level via FC-1_Data.Request primitives is shunted to the FC-1 receiver and reflected in FC-1_Data.Indication primitives issued by the receiver after Synchronization is reestablished (if necessary), overriding any information received by the FC-1 receiver.

Receipt of this primitive with Control = Disable causes the FC-1 level to disable its Loopback function. When the Loopback function is disabled, information passed to the FC-1 level via FC-1_Data.Request primitives is encoded and transmitted by the FC-1 transmitter. Information

received and decoded by the FC-1 receiver is indicated to the FC-2 level by FC-1_Data.Indication primitives after Synchronization is reestablished (if necessary).

A.10.5 Additional comments

The FC-1 Loopback function is disabled upon completion of Initialization of an FC-1 transmitter and receiver.

Behavior of a transmitter in the Working state is unpredictable when the FC-1 Loopback function is enabled.

Loss of Synchronization may occur whenever the Loopback function is enabled or disabled.

This primitive is illustrated by the request portion of Figure A.2 (c).

A.11 FC-1_Loopback_Control.Confirm

A.11.1 Function

This primitive is used by the FC-1 level to accept or reject the request that the FC-1 Loopback function be enabled or disabled.

A.11.2 Semantics

FC-1_Loopback_Control.Confirm (Status)

Status	The results of the FC-1_Loopback_Control.Request primitive. The allowable values of this parameter are Accept and Reject. The format of this parameter is not defined.
--------	--

A.11.3 When generated

A FC-1 level issues the FC-1_Loopback_Control.Confirm primitive to a FC-2 level to indicate the acceptance or rejection of a previously issued FC-1_Loopback_Control.Request primitive. Acceptance is indicated only when the receiver via FC-1_Data.Indication primitives presents valid information.

Effect on receipt

Receipt of the FC-1_Loopback_Control. Confirm primitive with Status = Accept informs the FC-2 level that the enabling or disabling of the FC-1 Loopback function was successful or that the FC-1 Loopback function was already enabled or disabled.

Receipt of the FC-1_Loopback_Control. Confirm primitive with Status = Reject informs the FC-2 level that the enabling of the FC-1 Loopback function was unsuccessful. A request to enable the FC-1 Loopback function is rejected whenever the FC-1 receiver is not in the Synchronization-Acquired or Loss-Of-Synchronization state or the FC-1 transmitter is not in the Working, Open-Fibre, or Not-Enabled state. (a request to disable the FC-1 Loopback function is always accepted).

A.11.4 Additional comments

This primitive is illustrated by the confirm portion of Figure A.2 (c).

Annex B (informative) **Communication models**

B.1 Model 1 example

When one port of the channel only transmits Data frames and the other port only transmits Link level (Link_Control) responses, the channel is said to be performing half-duplex communication.

For example in Figure B.1,

- N_Port A(1) transmits a Data frame (command or data) on its outbound Fibre and N_Port B(1) receives the Data frame on its inbound Fibre.
- In response, N_Port B(1) transmits a Link_Control frame on its outbound Fibre and N_Port A(1) receives it on its inbound Fibre.
- Multiple Data frames from N_Port A(1) and the respective Link_Control frames from N_Port B(1) results in similar flow.

Figure B.1 depicts the FC-2 half-duplex communication model with its logical structure and components. The model is composed of the following components:

- Node A and B
- N_Ports A(1) and B(1)
- Link_Control_Facility (LCF) in each N_Port
- Originator and Originator status in one N_Port
- Responder and Responder status in the other N_Port

The Originator is the logical function in a N_Port A(1) that initiates an Exchange, whereas the Responder is the logical function in a N_Port, which is the destination of the Exchange. Either N_Port can be an Originator and the other N_Port a Responder in Figure B.1 at any given time. The role of N_Ports as Originator and Responder do not change during a given Exchange.

B.2 Model 2 example

When both ports of a channel transmit Data frames and Link_Control frames (responses to Data frames) in directions opposite to their respective Data frames, the channel is said to be performing duplex communication.

For example in Figure B.2,

- N_Port A(1) transmits a Data frame on its outbound Fibre and N_Port B(1) receives it on its inbound Fibre.
- Simultaneously N_Port B(1) transmits its Data frame to N_Port A(1) on its outbound Fibre and N_Port A(1) receives it on its inbound Fibre.
- In response to the Data frame from N_Port A(1), N_Port B(1) transmits a Link_Control response on its outbound Fibre and N_Port A(1) receives the response on its inbound Fibre.
- In response to the Data frame from N_Port B(1), N_Port A(1) transmits a Link_Control response on its outbound Fibre and N_Port B(1) receives the response on its inbound Fibre.
- Multiple Data frames from N_Port A(1), and N_Port B(1) and the respective responses from N_Port B(1), and N_Port A(1) results in similar flow.

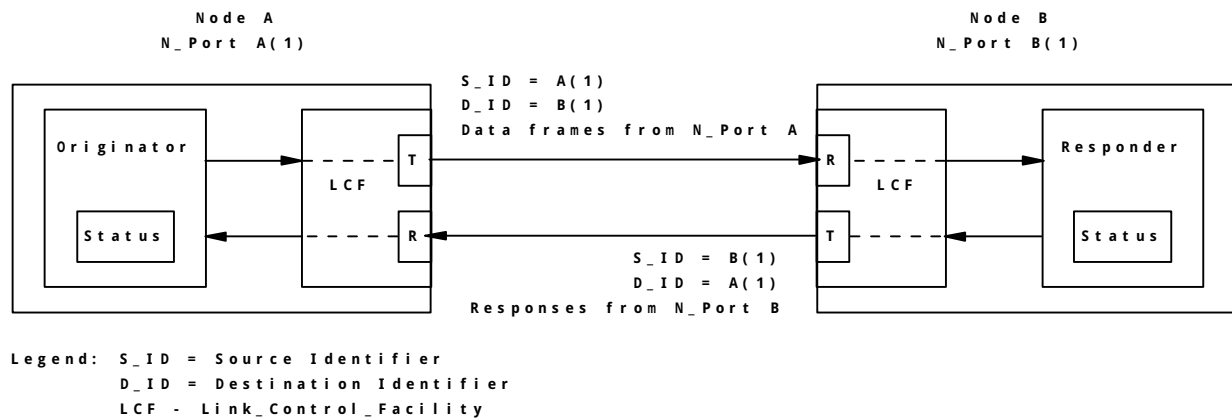


Figure B.1 - FC-2 half-duplex communication model example

Figure B.2 depicts the FC-2 duplex communication model with its logical structure and components. The FC-2 duplex communication model is bi-directionally symmetrical and both communicating N_Ports have equivalent abilities.

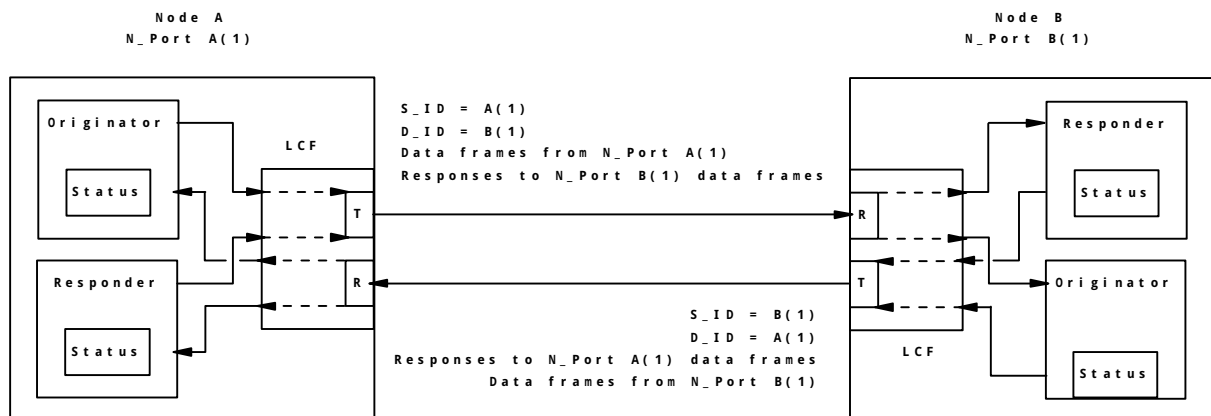


Figure B.2 - FC-2 duplex communication model example

The FC-2 duplex communication model consists of an Originator and a Responder in each of the communicating N_Ports.

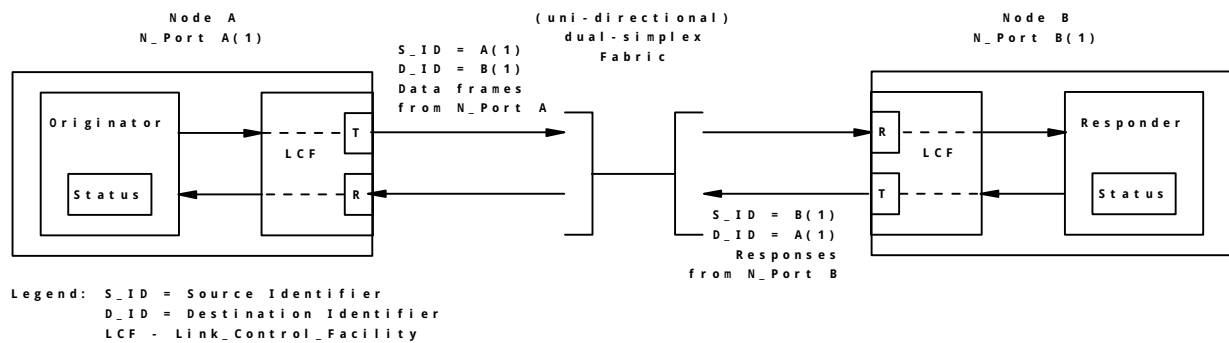


Figure B.3 - FC-2 dual-simplex communication model example

From the perspective of a given N_Port, the Outbound Fibre transmits a mixture of Data frames according to the activity generated by the N_Port and Link level responses to the Data frames received on the Inbound Fibre from the remote N_Port.

B.3 Model 3 example

When the channel transmits Data frames in one direction with the Link_Control frames arriving in the opposite direction, and limits the flow of Data frames and Link_Control frames at any given time to a single direction, the channel is said to be performing dual-simplex communication.

For example in Figure B.3,

- N_Port A(1) transmits a Data frame on its outbound Fibre and N_Port B(1) receives the Data frame on its inbound Fibre through the unidirectional Fabric. (simplex from A(1) to B(1)).
- In response to the Data frame, N_Port B(1) transmits a Link_Control response on its outbound Fibre and, due to unidirectional limitation of the Fabric, N_Port A(1) receives the response non-concurrently on its inbound Fibre. (simplex from B(1) to A(1)).
- Multiple Data frames from N_Port A(1) and the respective responses from N_Port B(1) results in similar flow.

Figure B.3 depicts the FC-2 dual-simplex communication model with its logical structure and components. The FC-2 dual-simplex communication model is functionally similar to the half-duplex model with the following difference. While the half-duplex communication model supports simultaneous bi-directional flow of frames, the dual-simplex model allows only unidirectional flow at any given time.

Annex C (informative) CRC generation and checking

C.1 Extract from FDDI

First part of this annex is an extract from Fiber Distributed Data Interface (FDDI) Media Access Control (MAC) (Document reference: ISO/IEC 9314-2:1989). FDDI's Frame Check Sequence (FCS) methodology, polynomials and equations are used by Cyclic Redundancy Check (CRC) in FC-2. The term FCS is unique to FDDI and not used by Fibre Channel. Note that CRC coverage of FC-2 fields is specified in FC-2 part of FC-FS.

4.3.6 Frame check sequence (FCS)

This annex specifies the generation and checking of the FCS field. This field is used to detect erroneous data bits within the frame as well as erroneous addition or deletion of bits to the frame. The fields covered by the FCS field include the FC, DA, SA, INFO, and FCS fields.

4.3.6.1 Definitions

$F(x)$ - A degree $k-1$ polynomial which is used to represent the k bits of the frame covered by the FCS sequence (see section 4.2.2). For the purposes of the FCS, the coefficient of the highest order term shall be the first bit transmitted.

$L(x)$ - A degree 31 polynomial with all of the coefficients equal to one, i.e.,

$$L(x) = X^{31} + X^{30} + X^{29} + \dots + X^2 + X^1 + 1$$

$G(x)$ - The standard generator polynomial

$$G(x) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

$R(x)$ - The remainder polynomial which is of degree less than 32

$P(x)$ - The remainder polynomial on the receive checking side which is of degree less than 32

FCS - The FCS polynomial which is of degree less than 32

$Q(x)$ - The greatest multiple of $G(x)$ in

$$[X^{32} F(x) + X^k L(x)]$$

$$Q^*(x) = X^{32} Q(x)$$

$M(x)$ - The sequence which is transmitted

$M^*(x)$ - The sequence which is received

$C(x)$ - A unique polynomial remainder produced by the receiver upon reception of an error free sequence. This polynomial has the value

$$C(X) = X^{32} L(X) / G(X)$$

$$C(x) = X^{31} + X^{30} + X^{26} + X^{25} + X^{24} + X^{18} + X^{15} + X^{14} + X^{12} + X^{11} + X^{10} + X^8 + X^6 + X^5 + X^4 + X^3 + X + 1$$

4.3.6.2 FCS generation equations

The equations, which are used to generate the FCS sequence from $F(x)$, are as follows:

Equation C.1 - FCS sequence equation 1

$$FCS = L(X) + R(X) = R\$ (X)$$

where $R\$ (X)$ is the one's complement of $R(X)$

Equation C.2 - FCS sequence equation 2

$$[X^{32} F(x) + X^k L(X)] / G(X) = Q(X) + R(X) / G(X)$$

Equation C.3 - FCS sequence equation 3

$$M(x) = x^{32} F(x) + FCS$$

NOTE - All arithmetic is modulo 2.

In Equation C.1, note that adding $L(x)$ (all ones) to $R(x)$ simply produces the one's complement of $R(x)$; this equation is specifying that the $R(x)$ is inverted before it is sent out.

Equation C.3 simply specifies that the FCS is appended to the end of $F(x)$.

4.3.6.3 FCS checking

The received sequence $M^*(x)$ may differ from the transmitted sequence $M(x)$ if there are transmission errors. The process of checking the sequence for validity involves dividing the received sequence by $G(x)$ and testing the remainder. Direct division, however, does not yield a unique remainder because of the possibility of leading zeros. Thus a term $L(x)$ is prepended to $M^*(x)$ before it is divided. Mathematically, the received checking is shown in Equation C.4.

Equation C.4 - Received checking

$$X^{32} [M^*(X) + X^k L(X)] / G(X) = Q^*(X) + P(X) / G(X)$$

In the absence of errors, the unique remainder is the remainder of the division

Equation C.5 - Checking remainder

$$P(X) / G(X) = X^{32} L(X) / G(X) = C(X)$$

C.2 CRC generation

The transmission bit order is defined as the bit ordering used by FC-1 prior to 8B/10B encoding (see clause 5), with bit A as the first bit transmitted and bit H as the last. The CRC field is calculated on the Frame_Header and the Data field content, using the defined transmission bit order (see 11.5), represented as the polynomial $F(x)$ in Figure C.1. The CRC field $R\$ (x)$ is transmitted sequentially starting with the highest order term ($X(31)$) and ending with the lowest order term ($X(0)$).

The Figure C.1 illustrates the flow from FC-2 to FC-0 using the FC-FS notation, including a reference to the mathematical notation used in N.1. The FC-2 word notation designates the most significant bit of the word as bit 31 and the least significant bit as 0. The FC-2 passes the words to FC-1 from left to right (i.e., Word 0 through n). The FC-1 transmits each byte starting with the least significant bit, namely bits A-E first followed by bits F-H. For example, in Figure C.1, the first byte of word 0 is transmitted with bit 24 first as bit A and bit 31 last as bit H. The

order of the polynomial $F(x)$ covered by the CRC and the order of the CRC polynomial $R(x)$ are illustrated in Figure C.1.

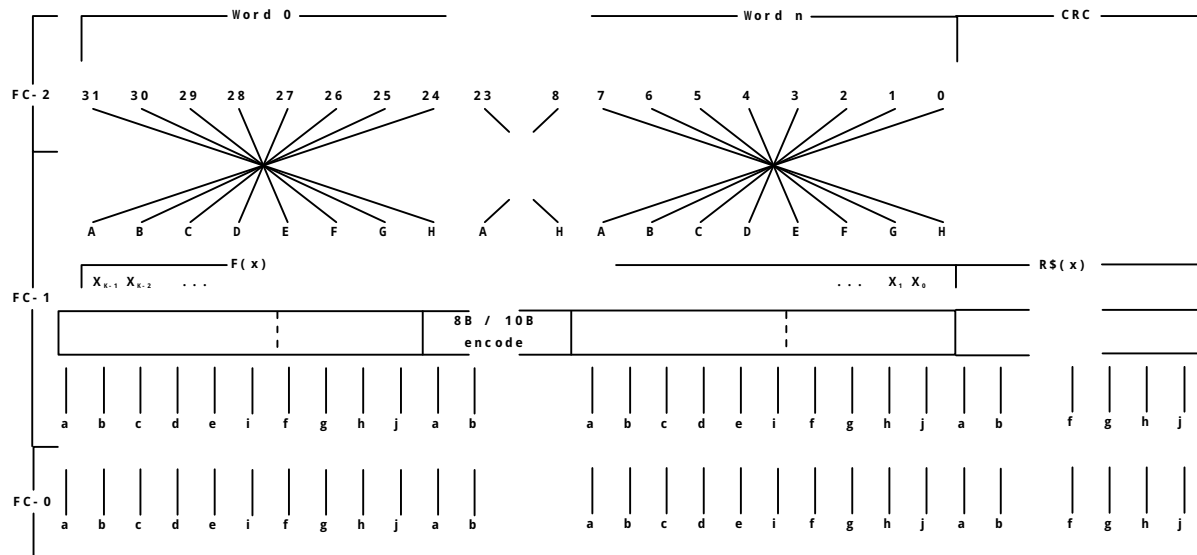


Figure C.1 - CRC coverage and bit ordering

C.3 Transmit order of a word

The transmit order of a FC-2 word in the example illustrated in Figure C.1 is expanded in Figure C.2. The top half of the figure shows the ordering of the first halfword sent and the bottom half shows the ordering of the second halfword.

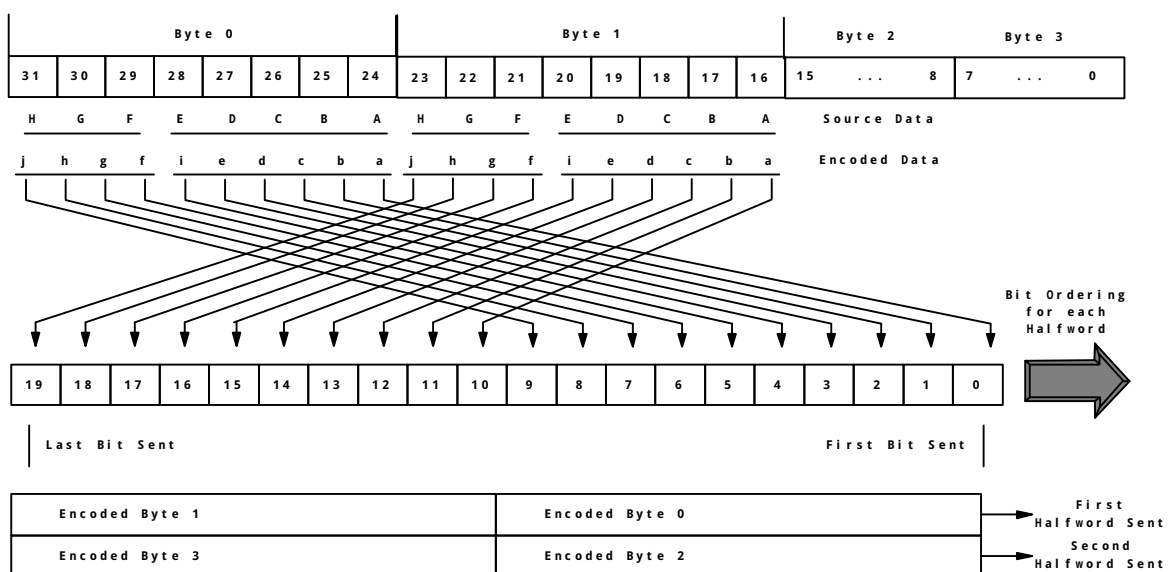


Figure C.2 - Word transmit order

C.4 CRC generation example for ACK_1 frame

An example of CRC generation for an ACK_1 frame is provided in a set of Table C.1 through Table C.8. Table C.1 shows an example of an ACK_1 fields without CRC and Table C.2 shows the hexadecimal values for each field. Table C.3 shows the transmit bit order (038040C...080 hex) with the bytes in Table C.2 transposed. Table C.4 shows the bit stream $X(32)F(x)+X(k)L(x)$ (FC7F...080 hex) for the sample. Table C.5 shows the generated remainder (649EOBF7 hex) for the sample. Table C.6 shows the one's complement of the remainder (9B61F408 hex) for the sample. The transmitted bit sequence for the sample with the CRC (038040C...F408 hex) is shown in Table C.7. The transmitted 10B stream for the sample with CRC is shown in Table C.8.

Table C.1 - Sample FC-2 frame		
Sample ACK_1 without CRC (frame header fields)		
R_CTL	D_ID	
rrrr rrrr	S_ID	
TYPE	F_CTL	
SEQ_ID	DF_CTL	SEQ_CNT
OX_ID		RX_ID
Parameter		

Table C.2 - Sample ACK_1 without CRC			
Sample Frame Header			
C0	01	02	03
00	04	05	06
00	C0	00	00
02	00	00	03
FF	FF	FF	FF
00	00	00	01

Table C.3 - F(x)			
Bytes in Table C.2 transposed			
03	80	40	C0
00	20	A0	60
00	03	00	00
40	00	00	C0
FF	FF	FF	FF
00	00	00	80

Table C.4 - $X^{*32} F(x) + X^{*k} L(x)$			
FC	7F	BF	3F
00	20	A0	60
00	03	00	00
40	00	00	C0
FF	FF	FF	FF
00	00	00	80

Table C.5 - $R(x)$			
64	9E	0B	F7

Table C.6 - $L(x) + R(x) = R$(x)$			
9B	61	F4	08

Table C.7 - $M(x)$			
03	80	40	C0
00	20	A0	60
00	03	00	00
40	00	00	C0
FF	FF	FF	FF
00	00	00	80
9B	61	F4	08

Table C.8 - $M(x) - (10B)$			
D0.6	D1.0	D2.0	D3.0
D0.0	D4.0	D5.0	D6.0
D0.0	D0.6	D0.0	D0.0
D2.0	D0.0	D0.0	D3.0
D31.7	D31.7	D31.7	D31.7
D0.0	D0.0	D0.0	D1.0
D25.6	D6.4	D15.1	D16.0

Annex D (informative) Data transfer protocols and examples

This annex provides Data transfer protocol examples.

D.1 Frame level protocol

Class dependencies of frame level protocols are shown.

D.1.1 Class 1 frame level protocol

Class 1 frame level protocols employ

- Data frame
- ACK
- R_RDY (response to SOFc1)

The Class 1 frame level protocol is illustrated in Figure D.1.

- The Originator initiates the Sequence with a Data frame embedded with SOFc1 (or SOFi1).
- SOFc1 is used to indicate a Connect-request and the Sequence initiation. The F_Port and the destination N_Port, as illustrated respond this Data frame with SOFc1 with a R_RDY. Next Data frame is not sent until the Connect-request is accepted by the destination, unless the Fabric supports stacked connect-requests. (With stacked connect-requests, the only Data frame that can be sent before the connect-request is accepted by the destination is another connect-request to a different destination.)
- Within an established Connection SOFi1 indicates the Sequence initiation.
- The Originator streams multiple Data frames and the Responder responds with ACK.
 - The ACK returns the following information unaltered contained in the F_CTL of Data frame to which it is responding:
 - First Sequence
 - Last Sequence
 - Last Data frame
 - Sequence transmit initiative
 - The ACK toggles following information contained in the F_CTL of the Data frame:
 - Exchange Context
 - Sequence Context

F_CTL usage for the Sequence is described in Table D.1.

- SOFn1 is used to indicate the Sequence in progress.
- The end of Sequence is indicated to the Sequence Recipient by the Last Data frame bit in F_CTL. The last ACK is embedded with EOFt or EOFdt that indicates the end of Sequence to the Sequence Initiator.
- The Last Sequence and the Last Data frame bits in F_CTL indicate the end of Exchange to the Sequence Recipient. The Last Sequence and the EOFt or EOFdt indicate to either N_Port the end of Exchange.

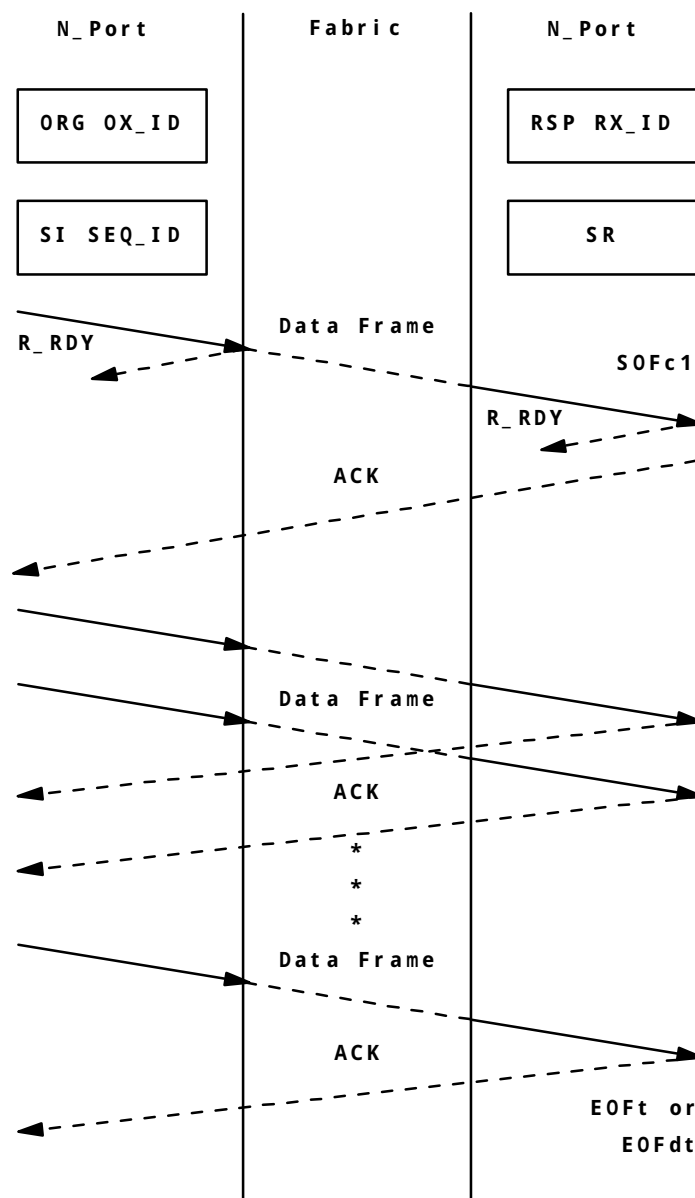


Figure D.1 - Class 1 frame level protocol

D.1.2 Class 2 frame level protocol

The Class 2 frame level protocol employs

1. Data frame
2. ACK
3. R_RDY

The Class 2 frame level protocol is illustrated in Figure D.2.

1. The Originator initiates the Sequence with a Data frame embedded with SOFi2.
2. The F_Port responds with an R_RDY and forwards the Data frame to the destination.

3. The destination responds with an R_RDY, in addition to ACK.
4. The F_Port and the N_Port respond each with R_RDY on receipt of ACK.
5. The Originator streams multiple Data frames and the Responder responds with ACK.
 - ACK returns some information contained in F_CTL of the Data frame to which it is responding unaltered.
 - First Sequence
 - Last Sequence
 - Last Data frame
 - Sequence transmit initiative
 - ACK toggles some information contained in F_CTL of the Data frame.
 - Exchange Context
 - Sequence Context

F_CTL usage for the Sequence is described in Table D.1.

1. For each of these frames received, each N_Port or F_Port returns a R_RDY.
2. SOFn2 is used to indicate the Sequence in progress.
3. The Sequence Initiator indicates the end of Sequence by the Last Data frame bit in F_CTL. However, the Sequence ends in the perspective of Sequence Recipient, only when all Data frames are received or accounted for.
4. The Sequence Recipient transmits EOFt only in the final ACK after all Data frames are received or accounted for.

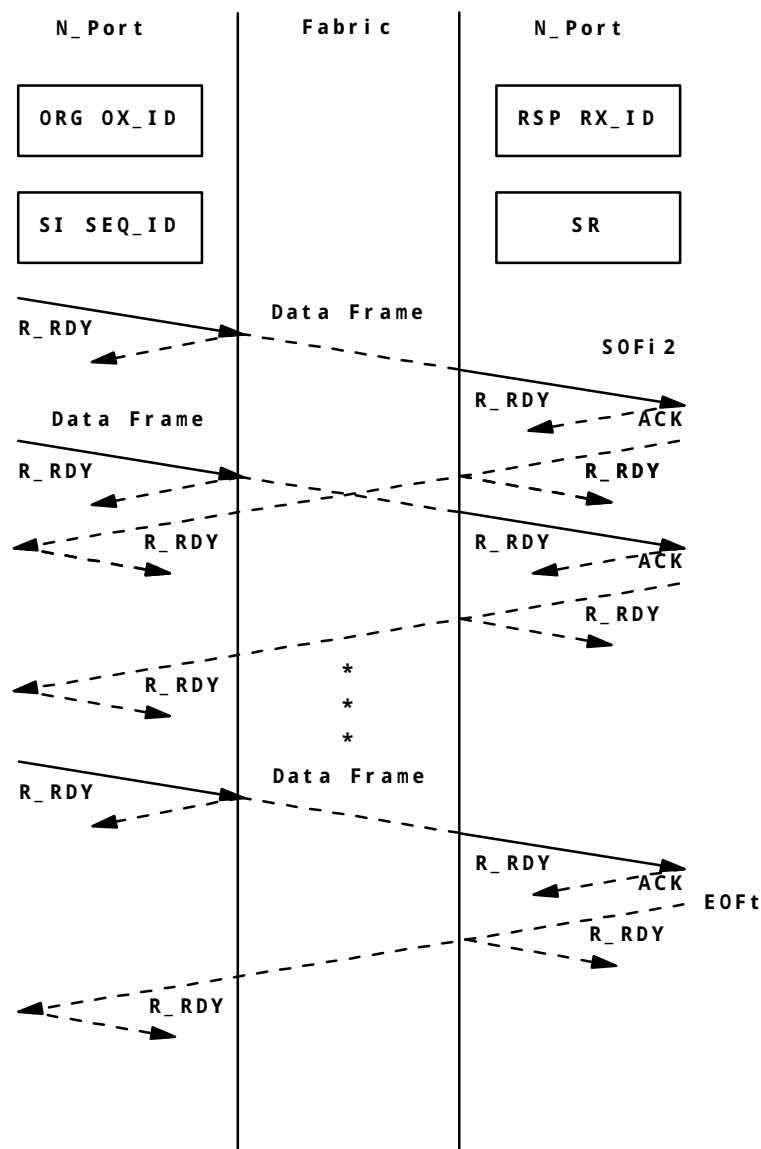


Figure D.2 - Class 2 frame level protocol

The Class 3 frame level protocol employs

- Data frame
- R_RDY

The Class 3 frame level protocol is illustrated in Figure D.3.

- The Originator initiates the Sequence with a Data frame embedded with SOF₃.
- The F_Port responds with an R_RDY and forwards the Data frame to the destination.
- The destination responds with an R_RDY.
- The Originator streams multiple Data frames. For each of these frames received, each N_Port or F_Port returns a R_RDY. F_CTL usage for the Sequence is described in Table D.2.

- SOF_{n3} is used to indicate the Sequence in progress.
- The end of Sequence is indicated to the Sequence Recipient by the Last Data frame bit in F_CTL and EOF_t.

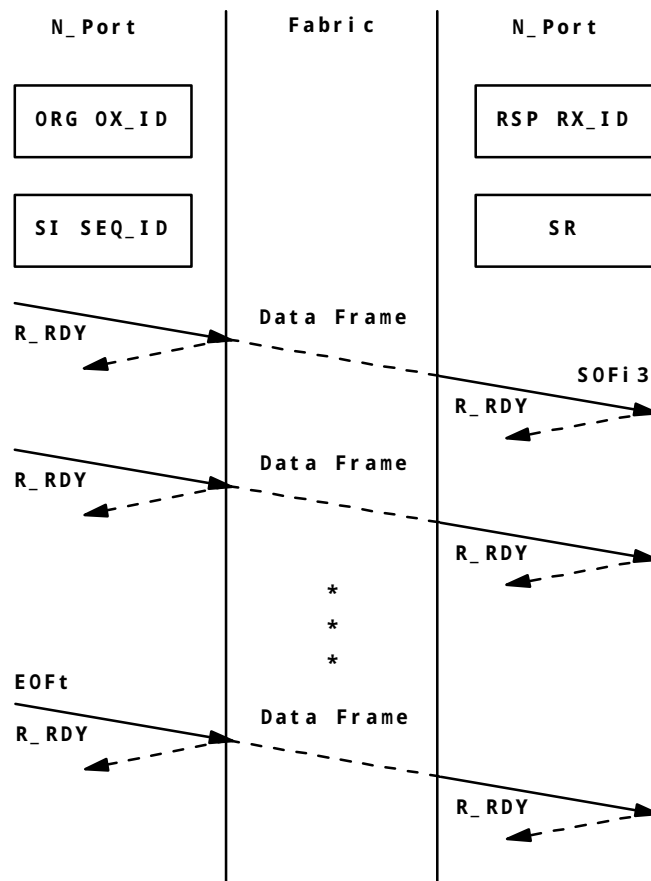


Figure D.3 - Class 3 frame level protocol

Table D.1 - F_CTL for Class 1 or Class 2 frame level protocol						
Description	Exchange Context	Sequence Context	First Sequence of Exchange	Last Sequence of Exchange	End Sequence	Sequence transmit initiative
F_CTL Bits	23	22	21	20	19	16
First Data frame	0 (ORG)	0 (SI)	1 (First)	0 (Sequence)	0	0 (NM)
ACK	1 (RSP)	1 (SR)	1 (First)	0 Sequence)	0	0 (NM)
Intermediate Data frame(s)	0	0	1	0	0	0 (NM)
ACK	1	1	1	0	0	0 (NM)
Last Data frame	0	0	1	0	1	0 (retain Sequence

						Initiative)
ACK	1	1	1	0	0	0 (NM)
Note - NM - Not Meaningful						

Table D.2 - F_CTL for Class 3 frame level protocol						
Description	Exchange Context	Sequence Context	First Sequence of Exchange	Last Sequence of Exchange	End Sequence	Sequence transmit initiative
F_CTL Bits	23	22	21	20	19	16
First Data frame	0 (ORG)	0 (SI)	1 (First)	0 (Sequence)	0	0 (NM)
Intermediate Data frame(s)	0	0	1	0	0	0 (NM)
Last Data frame	0	0	1	0	1	0 (retain Sequence Initiative)
Note - NM - Not Meaningful						

D.2 Sequence level protocol example

Sequence level protocol is illustrated with a three Sequence Exchange in Figure D.4. The first Sequence is a "read" request. The second Sequence transfers the "data". The third Sequence transfers "ending status" and ends the Exchange.

Frames 1,2, and 3 represent the first Sequence of an Exchange. In this example a Command Request for a Read operation is sent as a request Sequence. Note that Sequence Initiative is transferred to the Sequence Recipient.

Frames 4,5 and 6 represent the first, intermediate and last frames of the data transferred in response to the Read request. Note that the Sequence Initiative is retained in order to start a Sequence with ending status.

Frames 7,8 and 9 represent the ending status for the preceding data transfer and end the Exchange. Depending on the Upper Level

Protocol, the Responder may not be allowed to end the Exchange, but transfer the Sequence initiative to the Originator to complete the Exchange.

F_CTL usage

Use of F_CTL bits for these example Sequences are shown in Table D.3.

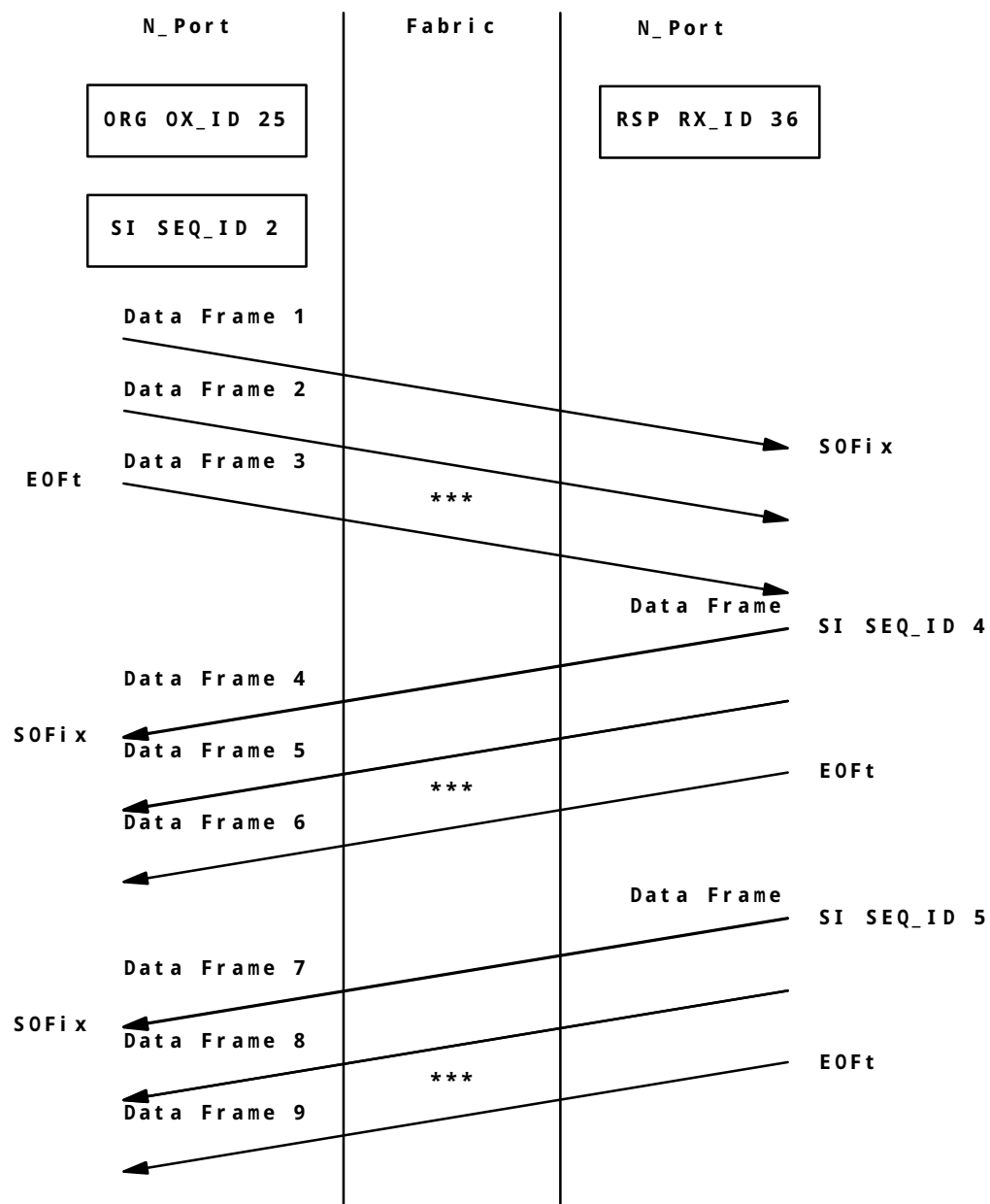


Figure D.4 - Sequence level protocol example

Table D.3 - Sequence level protocol example						
Description	Exchange Context	Sequence Context	First Sequence of Exchange	Last Sequence of Exchange	End Sequence	Sequence transmit initiative
F_CTL Bits	23	22	21	20	19	18
First Data frame (SOFix) of the Exchange and of the first Sequence (a Read Request Sequence)	0	0	1	0	0	0 (NM)
Intermediate Data frame of first	0	0	1	0	0	1

sequence						
Last Data frame of first Sequence	0	0	1	0	1	1
First Data frame (SOFix) of intermediate Sequence (Reply Sequence)	1	0	0	0	0	0 (NM)
Intermediate Data frame of intermediate Sequence	1	0	0	0	0	0 (NM)
Last Data frame of intermediate Sequence	1	0	0	0	1	0
First Data frame (SOFix) of the Last Sequence (Reply Status Sequence)	1	0	0	1	0	0 (NM)
Intermediate Data frame of the Last Sequence	1	0	0	1	0	0 (NM)
Last Data frame of the Last Sequence and of the Exchange	1	0	0	1	1	0

D.3 Class 1 frame level protocol example

N_Port_Login is used as an example to illustrate Class 1 frame flow.

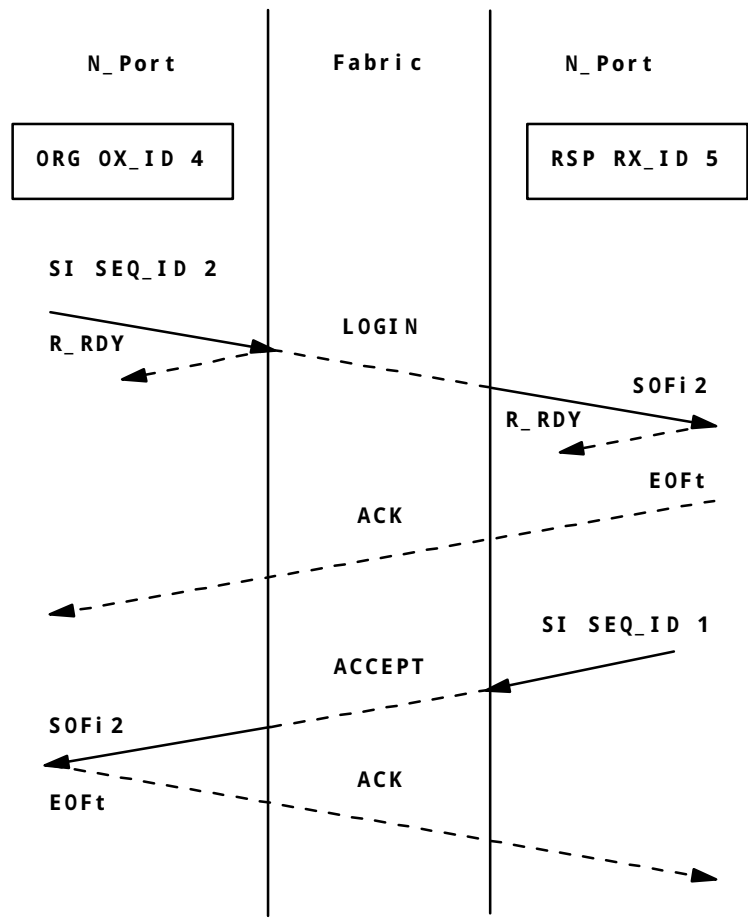


Figure D.5 - Class 1 frame level protocol - Login example

D.4 Class 2 frame level protocol example

N_Port_Login is used to illustrate Class 2 frame flow.

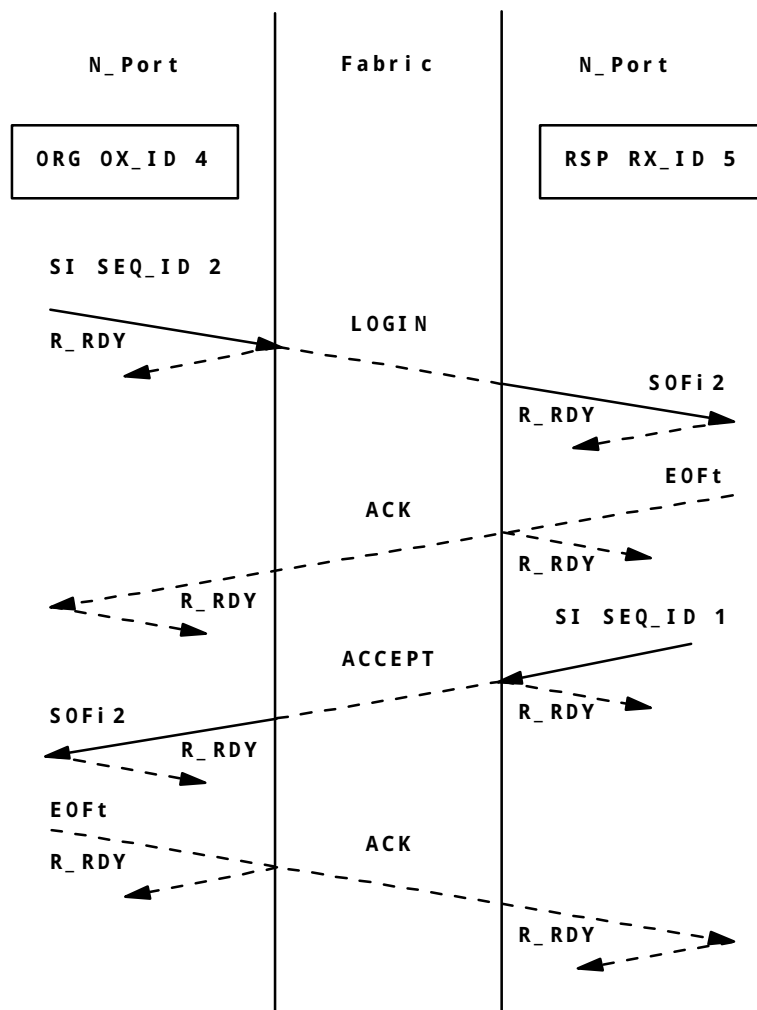


Figure D.6 - Class 2 frame level protocol - Login example

D.5 Class 3 frame level protocol example

N_Port_Login is used to illustrate Class 3 frame flow.

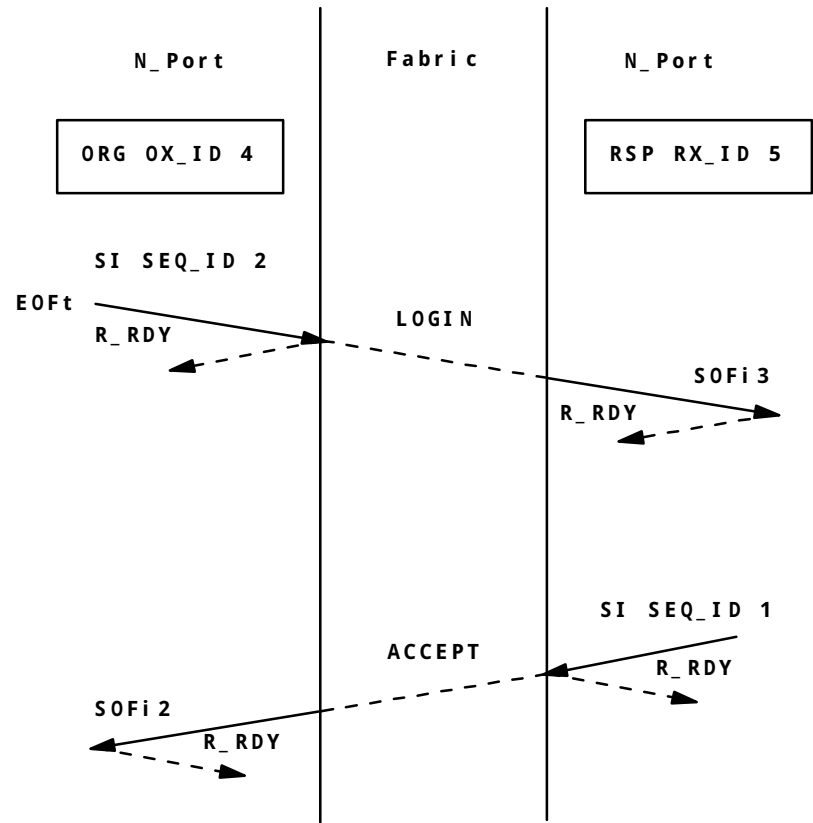


Figure D.7 - Class 3 frame level protocol - Login example

Annex E (informative) Connection management applications

E.1 Example cases

The following Cases represent examples of how a Class 1 Connection is removed using the Remove Connection procedure.

E.1.1 Case 1

Table E.1 shows a Case where N_Port (A) transmits its last Sequence without N_Port (B) transmitting any Sequences.

Table E.1 - Case 1		
xmit E_C (A)	recv E_C (A)	N_Port Actions
0	0	A transmitting Sequences
1	0	A transmits last Data frame of last Sequence
0	0	A receives the ACK_1 to the last Data frame with EOFdt - Connection Removed
Note - A = this N_Port B = other connected N_Port		

E.1.2 Case 2

Table E.2 shows a Case where N_Port (A) completes its last Sequence before N_Port (B).

Table E.2 - Case 2		
xmit E_C (A)	recv E_C (A)	N_Port Actions
0	0	Both A and B transmitting Sequences
1	0	A transmits last Data frame of last Sequence
0	0	B completes Sequences in progress, A responds with Link_Continues
0	1	A receives the last Data Frame of last Sequence from B
0	0	A transmits EOFdt on ACK_1 - Connection Removed
Note - A = this N_Port B = other connected N_Port		

E.1.3 Case 3

Table E.3 shows a Case where N_Port (B) transmits its last Sequence before N_Port (A).

Table E.3 - Case 3		
xmit E_C (A)	recv E_C (A)	N_Port Actions
0	0	Both A and B transmitting Sequences
0	1	A receives last Data frame of last Sequence from B.
0	0	A complete Sequences in progress
1	0	A transmits last Data frame of last Sequences
0	0	A receives EOF _{dt} on ACK_1 - Connection Removed
Note - A = this N_Port B = other connected N_Port		

E.1.4 Case 4

Table E.4 shows a Case where N_Port (A) transmits its last Data frame of its last Sequence simultaneously with receiving the last Data frame of N_Port (B)'s last Sequence.

Table E.4 - Case 4		
xmit E_C (A)	recv E_C (A)	N_Port Actions
0	0	Both A and B transmitting Sequences (A is Connection Initiator)
1	0	A transmits last Data frame of last Sequence.
0	1	A receives Data frame from B before receiving Link_Continue for its last Data frame with E_C = 1
0	0	A waits for its Link_Continue response
0	0	A receives its ACK_1/ACK from B with EOF SOFix)
0	0	A transmits ACK_1 with EOF SOFix - Connection Removed
Note - A = this N_Port B = other connected N_Port		

E.2 Ending sequence and Connection

Table E.5 shows an example of F_CTL Bit settings for key Data frames within an example Exchange.

The first Sequence is a "read" request. The second Sequence transfers the "data". The third Sequence transfers "ending status" and ends the Exchange and terminates a Class 1 Dedicated Connection.

Table E.5 - F_CTL for example Exchange							
Description	Exc Owner	Seq Owner	First Seq of Exc	Last Seq of Exc	Last Seq Data frame	E_C	Seq xmit initiative
Bits	23	22	21	20	19	18	16
First Data frame of Exchange (SOF _{ix}) - a Read request (xmit)	0	0	1	0	0	0	0 (NM)
Intermediate Data frame of first Sequence (xmit)	0	0	1	0	0	0	0 (NM)
Last Data frame of first Sequence (xmit)	0	0	1	0	1	0	1
First Data frame of reply "data" Sequence (SOF _{ix}) (recv)	1	0	0	0	0	0	0 (NM)
Last Data frame of reply data Sequence (recv)	1	0	0	0	1	0	0
First Data frame of reply Status Sequence (SO-F _{ix}) (recv)	1	0	0	1	0	0	0 (NM)
Last Data frame of Exchange (recv)	1	0	0	1	1	1	1
Last ACK_1 of Exchange (xmit) with EOF _{dt}	0	1	0	1	0	0	0

Frames 1,2, and 3 represent the first Sequence of an Exchange. In this example a Command Request for a Read operation is sent as a request Sequence. Note that Sequence Initiative is transferred to the Sequence Recipient.

Frames 4 and 5 represent the first and last frames of the data transfer associated with the Read operation. Note that the Sequence Initiative is retained in order to start a Sequence with ending status.

Frames 6 and 7 represent the ending status for the preceding data transfer and ends the Exchange. Frame 7 ends the Sequence, ends the Exchange, and requests termination of the Connection. The ACK_1 response to frame 7 (frame 8) removes the Dedicated Connection with EOF_{dt}.

E.3 Port states

A Port's behavior relative to its Receiver and Transmitter may be viewed as a number of separate and distinct states within a Port with respect to Primitive Sequences.

Active (AC)

A Port is in the Active state when it is not in any other states. In the Active state, a Port is operational and available to transmit and receive frames. The Port is transmitting and receiving Idles.

Link Recovery

When a N_Port is performing any portion of Link Recovery, it is considered "Busy" to other N_Ports attached to the Fabric, if present. Three substates exist during the Link Recovery State:

- LR Transmit (LR1)
The Port transmits the LR Primitive Sequence to reset the attached Port or F_Port.
- LR Receive (LR2)
The Port transmits the LRR Primitive Sequence as a result of receiving the LR Primitive Sequence.
- LRR Receive (LR3)
The Port transmits Idles as a result of receiving the LRR Primitive Sequence.

Link Failure

When a N_Port is in the Link Failure state, the N_Port is unable to process frames. The Link Failure state is an abnormal occurrence at the N_Port and the N_Port is considered "not available" to other N_Ports attached to the Fabric, if present.

Two substates exist:

- NOS Receive (LF1)
When NOS is received, the Port transmits OLS.
- NOS Transmit (LF2)
When a Port is not operational, it transmits NOS to indicate a Link failure.

Offline

The Offline state may be entered in response to receiving OLS on its link, or by action initiated by the Link_Control facility within the Port. The Link_Control facility enters the Offline state for four reasons:

- during initialization, to proceed from Not_Operational to Operational such as at power-up,
- during shutdown, to remove a Port from service, or
- to perform Port diagnostics.
- to indicate an internal failure.

Three substates exist:

- OLS Transmit (OL1)
The Port transmits OLS. After the Port receives the LR Primitive Sequence or exceeds a timeout period, it may perform any link activity including turning off its transmitter without errors being detected by the other attached Port. The Port exits this state by receiving LR and transmitting the LRR Primitive Sequence.
- OLS Receive (OL2)
When a Port receives OLS, it transmits the LR Primitive Sequence. It ignores link errors or other Primitive Sequences excluding LR or LRR. When the Port receives LR or LRR, it exits this substate.
- Wait for OLS (OL3)
When a Port is in either OLS Transmit or OLS Receive State and it detects Loss of Signal or Loss of Synchronization for a period of time greater than a timeout period, the Port en-

ters the Wait for OLS State and is Offline. It normally exits this state upon reception of OLS.

E.4 State change table

Table E.6 describes the state changes made based on the current state and the Primitive Sequence or Signal received. See 10.4.3 for specific additional considerations and conditions for entering and exiting the Offline State (OL1 or OL2).

Table E.6 - N_Port states - Primitive Sequences									
State	ACTIVE	LINK RECOVERY			LINK FAILURE		OFFLINE		
Input Event	IDLE RECV	LR XMIT	LR RECV	LR RECV	NOS RECV	NOS XMIT	OLS XMIT	OLS RECV	WAIT OLS
Substrate	(AC)	(LR1)	(LR2)	(LR3)	(LF1)	(LF2)	(OL1)	(OL2)	(OLS)
xmit >>	IDLE	LR	LRR	IDLE	OLS	NOS	OLS	LR	NOS
L >> LR	LR2	LR2	**	LR2	LR2	**	LR2 Note 2	LR2	LF2
L >> LRR	LR3 Note 3	LR3	LR3	**	**	**	**	LR3	LF2
L >> IDLES	**	**	AC	AC	**	**	**	**	**
L >> OLS	OL2	OL2	OL2	OL2	OL2	OL2	OL2 Note 2	**	OL2
L >> NOS	LF1	LF1	LF1	LF1	**	LF1	LF1 Note 2	LF1	LF1
Loss of Signal	LF2	LF2	LF2	LF2	LF2	** Note 1	OL3 Note 2	OL3 Note 1	**
Loss of Sync > Limit	LF2	LF2	LF2	LF2	LF2	**	OL3 Note 2	OL3	**
Event timeout (R_T_TOV)	N/A	LF2	LF2	LF2	LF2 Note 4	N/A	OL3 Note 2 Note 6	OL3 Note 5	N/A

LEGEND

L >> means receiving from the Link

N/A means not applicable

An ** entry means no change in state

NOTES

- Depending on Laser safety requirements, the transmitter may enter a "pulse" transmission mode of operation when Loss of Signal is detected.
- All events are ignored until the Port determines it is time to leave the OLS transmission state.
- A Primitive Sequence Protocol error is detected (An improper Primitive Sequence was received in this State). The Primitive Sequence Protocol error count in the LESB is incremented.
- The timeout period starts timing when NOS is no longer recognized and none of the other events occur which cause a transition out of the state.
- The timeout period starts timing when OLS is no longer recognized and none of the other events occur which cause a transition out of the state.
- The timeout period starts timing when the Port is attempting to go online transmits OLS, and none of the other events occur which cause a transition out of state.

L>>LR

The Port is receiving and recognizing the Link Reset Primitive Sequence.

L>>LRR

The Port is receiving and recognizing the Link Reset Response Primitive Sequence.

L>>Idles

The Port is receiving and recognizing Idles.

L>>OLS

The Port is receiving and recognizing the Offline Primitive Sequence.

L>>NOS

The Port is receiving and recognizing the Not Operational Primitive Sequence.

Loss of Signal

The Port has detected loss of signal.

Loss of Sync > Limit

The Port has detected loss of synchronization for a period of time greater than a timeout period.

Event Timeout (R_T_TOV)

The timeout for an expected event exceeds the Receiver_Transmitter Timeout Value (R_T_TOV).

Annex F

(informative)

FC-FS service interface

This Annex a possible implementation of some of the services provided by FC-FS and some of the services that may be required by FC-FS. The intent is to specify the services available to FC-3 and FC-4s for the support of Upper Level Protocols with FC-FS. How many of the services described in this annex are chosen for a given implementation is up to that implementer; however, a set of FC-FS services will be supplied sufficient to satisfy the Upper Level Protocol(s) being used. The services as defined in this standard do not imply any particular implementation, or any interface. Services described are:

- FC-FS services provided to the local FC-4 (indicated by FC_FS_ prefix) for support of Upper Level Protocol data transfer.
- Login Services provided by FC-FS for Fabric and N_Port_Login.
- Link Services provided to the local FC-4 entities for data services management.

Link Services, such as Login, may be performed by a common process or by individual FC-4s in an implementation specific manner.

Note - Throughout this service interface, confirmation primitives are not used with Class 3 services since Acknowledgements are not sent to confirm successful delivery.

F.1 FC_FS to ULP Data Services

The following service primitives are provided by FC-FS for use by FC-3 and FC-4 to support ULP data transfer:

- FC_FS_SEQUENCE.request
- FC_FS_SEQUENCE_TAG.indication
- FC_FS_SEQUENCE.indication
- FC_FS_SEQUENCE.confirmation

In the absence of a FC-3, the transfer of an Information Unit by an FC-4 corresponds to the transfer of a Sequence by FC-FS.

An example of a typical data transfer is shown in Figure F.1.

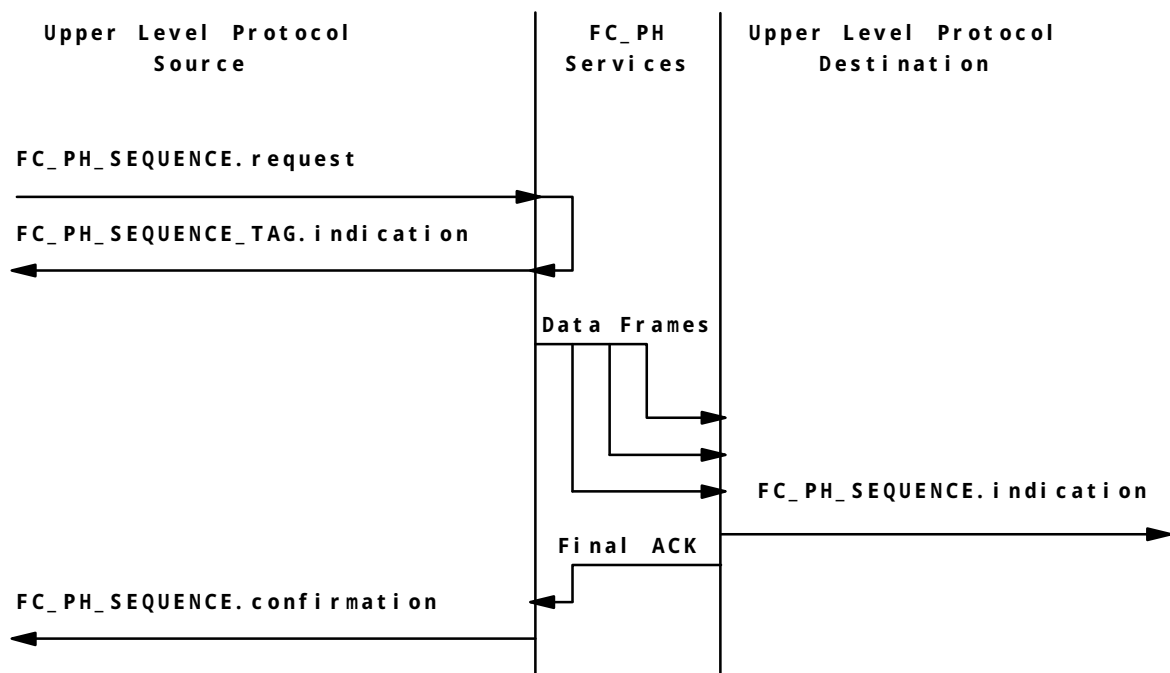


Figure F.1 - Data Transfer Service Primitives Example

F.1.1 FC_FS_SEQUENCE.request

This primitive defines the transfer of one or more Data Blocks within a single Sequence from a local ULP entity to a single peer ULP entity or to multiple peer ULP entities in the case of a group destination address.

F.1.1.1 Semantics of the Primitive

```

FC_FS_SEQUENCE.request (
    Type,
    Exchange_Tag,
    Sequence_Tag,
    Allowed_Class,
    Routing_Bits,
    D_ID,
    S_ID,
    First_Sequence,
    Last_Sequence,
    Chained_Sequence,
    Sequence_Initiative,
    Continue_Sequence_Condition,
    Exchange_Error_Policy,
    Exchange_Reassembly,
    Data_Field_Control,
    Network_Header,
    Association_Header,
    Device_Header,
    Information_Category(1),
    Block_Offset(1),
    Data_Block(1),

```



```

Information_Category(2),
Block_Offset(2),
Data_Block(2),
...
Information_Category(n),
Block_Offset(n),
Data_Block(n)
)

```

The Type will identify the Upper Level Protocol (see 12.5).

The Exchange_Tag will provide a local identifier of the Exchange that includes this Sequence. The Exchange_Tag parameter is optional on the first Sequence to allow assignment by FC-FS. The Exchange_Tag will be assigned by the ULP, the FC-4 or FC-FS to provide a unique identifier for the Exchange that includes this Sequence.

Note - The Exchange_Tag may use the Exchange_Context and Exchange_ID or any other identifier to uniquely identify the Exchange.

The Sequence_Tag may optionally be provided with this primitive. If provided in Classes 1 or 2, the Sequence Tag will be a unique identifier of the Sequence. If provided in Class 3, the Sequence_Tag shall be used by FC-FS as the Sequence ID. If the Sequence_Tag is not assigned by the ULP or FC_4 and included in this primitive, it shall be assigned by FC-FS and indicated in the FC_FS_SEQUENCE_TAG.indication.

Note - This requirement implies that the FC-4 is responsible for controlling reuse of Sequence IDs by FC-FS in Class 3. If a Sequence_Tag is reused on a second IU before R_A_TOV has expired on the last frame of the previous IU, data integrity errors may result if the Sequence_Count wraps.

The Allowed_Class will indicate the set of classes of service allowed for all the frames in this Sequence (see clause 16). The frames for the Sequence will all be transmitted using the same class. Sequences using Class 3 will not be sent in the same Exchange with Sequences using Class 1 or 2.

The Routing_Bits will indicate the routing for all the frames in this Sequence and will indicate an allowable values specified in 12.1.2.

The D_ID will indicate the identifier of the desired destination of this Sequence (see 12.3). For Class 1 and 2, the D_ID will be an individual address; for Class 3, the D_ID may be either a individual address or a group address.

The S_ID will indicate the address identifier source of the Sequence (see 12.3.2).

The First_Sequence will indicate when the Sequence is the first in a new Exchange (see 12.6).

The Last_Sequence will indicate when the Sequence is the last in an Exchange (see 12.6).

The Chained_Sequence will indicate that a Reply Sequence is expected and will be included within this Dedicated Connection The Chained_Sequence will only be set in accordance with the conditions specified in 12.6.

The Sequence_Initiative will indicate when the Sequence Initiative is to be transferred to the Sequence Recipient on the last Data frame of this Sequence (see 12.6).

The Continue_Sequence_Condition will indicate the expected time before the next Sequence in the Exchange is presented and will take on a value specified in 12.6. The Continue_Sequence_Condition provides advisory information to FC-FS used to terminate connections and invalidate X_IDs.

The `Exchange_Error_Policy` will specify the error recovery policy to be used in the Exchange and is specified when the First Sequence is indicated. It will be set to a value as allowed by the Abort Sequence Condition bits in 12.6.

The `Exchange_Reassembly` is reserved and will be set to zero.

The `Data_Field_Control` will specify the complete set of optional Headers that are to be included in frames of this Sequence (see 12.8)

The `Network_Header` may optionally specify network addresses for the Source and Destination of the Sequence (see 13.3).

The `Association_Header` may optionally specify a identifier to be used to track an Exchange (see 13.4).

The `Device_Header` may optionally specify a Upper Level header (see 13.5).

Each set of `Information_Category`, `Block_Offset`, and `Data_Block` specifies a block or subblock for transmission and is referenced as a subrequest. A Data Block is a unit of data with a single `Information_Category` and a single associated offset.

The `Information_Category` may be used to provide additional control information as defined in 12.2. The number of different `Information_Categories` passed in a single Sequence will not exceed the value communicated by the `N_Port_Login Categories per Sequence` parameter. The `Information_Category` may be the same for multiple `Data_Blocks` in a Sequence.

The `Block_Offset` may optionally specify the Relative Offset (see 12.12) for the first byte of this block or subblock of data.

The `Data_Block` will specify the Payload to be passed for this block or subblock of data (see 4.12.5).

F.1.1.2 When Generated

This primitive is generated by an FC-4 to request an Upper Level Protocol data transfer by FC-FS.

F.1.1.3 Effect of Receipt

Upon receipt of a `FC_FS_SEQUENCE.request` primitive, the source FC-FS performs the following actions:

- a) Receives the indicated unique `Sequence_Tag` or may indicate to the FC-4 a unique `Sequence_Tag` using the `FC_FS_SEQUENCE_TAG.indication` primitive.
- b) Validates the parameters and verifies that the requested operation is possible, e.g., checks against login parameters, exchange initiative, etc.
- c) If this is the first request for an Exchange, then enables an Exchange to be started with the first frame transmitted.
- d) If this is a request that requires Exchange resources, then enables the `X_ID` to be assigned on the first frame of the Sequence.
- e) If this is a request to release Exchange resources, then enables the `X_ID` to be invalidated at the end of the Sequence.
- f) If this is the last request for an Exchange, then enables the Exchange to be ended with the last frame transmitted.
- g) Assigns a `Sequence_ID` to this transfer request.

- h) Segments a block or subblock into frames for transmission. FC-FS determines frame boundaries in addition to those produced by the passing of multiple Data Blocks from the FC-4. If Relative Offset is supported, the Initial Relative Offset is used in the first frame of the block or subblock and the Relative Offset is computed for each succeeding frame for the block or subblock.
- i) Appends an appropriate frame header and trailer to each frame.
- j) Transfers the frames to the destination.
- k) Maintains the Exchange Status Block and Sequence Status Blocks.
- l) Uses FC_FS_SEQUENCE.confirmation to notify FC-4s as Sequences are successfully completed or abnormally terminated.

F.1.2 FC_FS_SEQUENCE_TAG.indication

This primitive provides an indication of the Exchange_Tag and Sequence_Tag for Class 1 and 2 Sequences. This primitive is optional for Class 3 Sequences.

F.1.2.1 Semantics of the Primitive

```
FC_FS_SEQUENCE_TAG.indication    (
                                   Type,
                                   Exchange_Tag,
                                   Sequence_Tag
                                   )
```

The Type will identify the Upper Level Protocol (see 12.5).

The Exchange_Tag will provide a local identifier of the Exchange that includes this Sequence. The Exchange_Tag may be assigned by the ULP, FC-4 or FC-FS to provide a unique identifier for the Exchange that includes this Sequence.

NOTE - The Exchange_Tag may use the Exchange_Context and Exchange_ID to uniquely identify the Exchange.

The Sequence_Tag may provide a unique identifier of the Sequence. The Sequence_Tag may be assigned by the ULP, FC-4 or FC-FS to provide a unique identifier for the Sequence.

F.1.2.2 When Generated

This primitive is atomically generated in response to the FC_FS_SEQUENCE.request primitive.

F.1.2.3 Effect of Receipt

The effect of receipt of this primitive by the FC-4 entity is unspecified.

F.1.3 FC_FS_SEQUENCE.indication

This primitive defines the transfer of a single Sequence of one or multiple blocks of data from FC-FS to the local Upper Level Protocol entity.

F.1.3.1 Semantics of the Primitive

```
FC_FS_SEQUENCE.indication    (
                                   Type,
                                   Exchange_Tag,
                                   Class,
                                   Routing_Bits,
                                   D_ID,
                                   S_ID,
                                   First_Sequence,
```

```

Last_Sequence,
Chained_Sequence,
Continue_Sequence_Condition,
Exchange_Error_Policy,
Exchange_Reassembly,
Data_Field_Control,
Network_Header,
Association_Header,
Device_Header,
Information_Category(1),
Block_Offset(1),
Data_Block(1),
Information_Category(2),
Block_Offset(2),
Data_Block(2),
...
Information_Category(n),
Block_Offset(n),
Data_Block(n),
Sequence_Valid
)

```

The Type will identify the Upper Level Protocol (see 12.5).

The Exchange_Tag will provide a local identifier of the Exchange that includes this Sequence.

Note - The Exchange_Tag may use the Exchange_Context and Exchange_ID or any other identifier to uniquely identify the Exchange.

The Class will indicate the class of service for all the frames in this Sequence (see 16).

The Routing_Bits will indicate the routing for all the frames in this Sequence and will indicate an allowable values specified in 18.1.2.

The D_ID will indicate the identifier of the desired destination of this Sequence (see 12.3). For Class 1 and 2, the D_ID will be an individual address; for Class 3, the D_ID may be either a individual address or a group address.

The S_ID will indicate the address identifier source of the Sequence (see 12.3.2).

The First_Sequence will indicate when the Sequence is the first in a new Exchange (see 12.6).

The Last_Sequence will indicate when the Sequence is the last in an Exchange (see 12.6).

The Chained_Sequence will indicate that a Reply Sequence is expected and will be included within this Dedicated Connection The Chained_Sequence will only be set in accordance with the conditions specified in 12.6.

The Sequence_Initiative will indicate when the Sequence Initiative is to be transferred to the Sequence Recipient on the last Data frame of this Sequence (see 12.6).

The Continue_Sequence_Condition will indicate the expected time before the next Sequence in the Exchange is presented and will take on a value specified in 12.6.

The Exchange_Error_Policy will specify the error recovery policy to be used in the Exchange and is specified when the First Sequence is indicated. It will be set to a value as allowed by the Abort Sequence Condition bits in 12.6.

The Exchange_Reassembly is reserved and will be set to zero.

The Data_Field_Control will specify the complete set of optional Headers that are to be included in frames of this Sequence (see 12.8).

The Network_Header may optionally specify network addresses for the Source and Destination of the Sequence (see 13.3)

The Association_Header may optionally specify a identifier to be used to track an Exchange (see 13.4).

The Device_Header may optionally specify a Upper Level header (see 12.5).

Each set of Information_Category, Block_Offset, and Data_Block specifies one block or sub-block received and is referenced as a subrequest. A Data Block is a unit of data with a single Information_Category and a single associated offset.

The Information_Category may be used to provide additional control information as defined in 12.2. The number of different Information_Categories passed in a single Sequence will not exceed the value communicated by the N_Port_Login Categories per Sequence parameter.

The Block_Offset may optionally specify the Relative Offset (see 12.12) for the first byte of this block of data.

The Data_Block will specify the Payload to be passed for this block or subblock of data (see 4.12.5).

Sequence_Valid will be set for Sequences with all frames received and valid according to the frame validity criteria in 11.8.1.

F.1.3.2 When Generated

This primitive is generated upon the successful completion of a Sequence reception. When the Exchange_Error_Policy is not Abort_Discard_Single_Sequence, within the same Exchange, the ordering of FC_FS_SEQUENCE.indications will occur in the same order as the FC_FS_SEQUENCE.requests. Upon receipt of data frames sent by the source, the destination FC-FS performs the following actions:

- a) Maintains the state of the Class 1 Dedicated Connection.
- b) Assembles the received block(s) and sub-blocks, recording any errors.
- c) Maintains the Exchange Status Block and Sequence Status Blocks.
- d) Issues ACK responses as required.
- e) Uses the FC_FS_SEQUENCE.indication primitive to pass the completed Sequence to the appropriate FC-4.

F.1.3.3 Effect of Receipt

The effect of receipt of this primitive by the FC-4 or ULP entity is unspecified.

F.1.4 FC_FS_SEQUENCE.confirmation

This primitive will provide an appropriate response to a FC_FS_SEQUENCE.request primitive issued for Class 1 or 2 indicating the success or failure of the request.

F.1.4.1 Semantics of the Primitive

FC_FS_SEQUENCE.confirmation (

Type,
Exchange_Tag,

```

Sequence_Tag,
Transmission_Status,
Reject_Reason
)

```

The Type will identify the Upper Level Protocol (see 12.5).

The Exchange_Tag will provide a local identifier of the Exchange that includes this Sequence. The Exchange_Tag is the local identifier, which was previously assigned to the Exchange.

The Sequence_Tag will provide a unique identifier of the Sequence. The Sequence_Tag is the local Sequence identifier, which was previously assigned to the Sequence.

The Transmission_Status will indicate status as one of the following:

- Successful - Sequence delivered completely to Recipient
- Unsuccessful - Sequence was not delivered completely due to abort or frame transfer error.
- Stopped_by_Recipient - Recipient stopped Sequence as indicated in ACK
- Rejected_Request - The Sequence was not sent by the Initiator due to the specified Reject_Reason.
- Rejected_by_Fabric - Reject frame received from Fabric
- Rejected_by_N_Port - Reject frame received from N_Port

When the Transmission_Status is Rejected_Request, Rejected_by_Fabric or Rejected_by_N_Port, the Reject_Reason will indicate one of the Reject reason codes given in Table 40.

F.1.4.2 When Generated

This primitive is generated upon the completion of the attempt to transmit the Sequence. This occurs after the last ACK is received indicated successful delivery of the Sequence, or after one of the Transmission_Status conditions occurs that indicates failure to deliver the Sequence.

F.1.4.3 Effect of Receipt

The effect of receipt of this primitive by the FC-4 or ULP entity is unspecified.

F.2 Login Services

Login requests are issued by a common service entity when multiple ULPs are present. The following service primitives are provided by FC-FS to support Login:

- FABRIC_LOGIN.request
- FABRIC_LOGIN.indication
- FABRIC_LOGIN.confirmation
- IMPLICIT_FABRIC_LOGIN.request
- N_PORT_LOGIN.request
- N_PORT_LOGIN.indication
- N_PORT_LOGIN.response
- N_PORT_LOGIN.confirmation
- IMPLICIT_N_PORT_LOGIN.request

- N_PORT_LOGOUT.request
- N_PORT_LOGOUT.indication
- N_PORT_LOGOUT.confirmation

F.2.1 Fabric Login Primitive Flows

An example of the usage of these service primitives in a typical Fabric Login is shown in Figure F.2. Fabric Login primitive flows when no Fabric is present and a point-to-point connections exists are shown in Figure F.3.

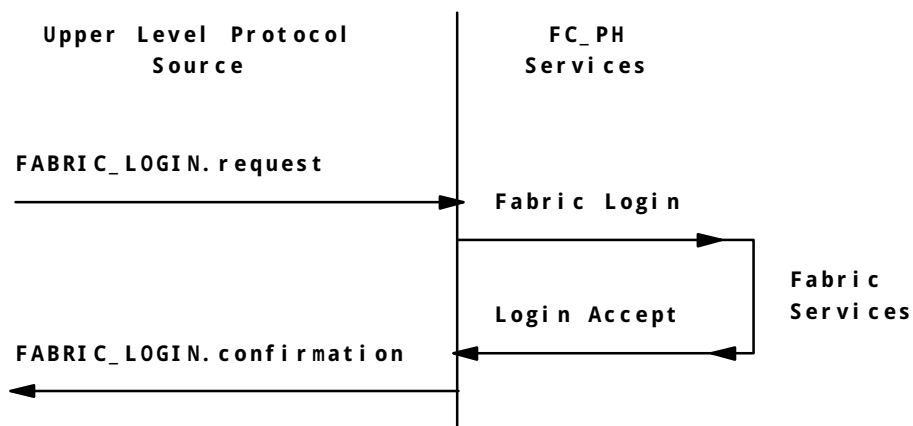


Figure F.2 - Fabric Login Service Primitives Example with a Fabric

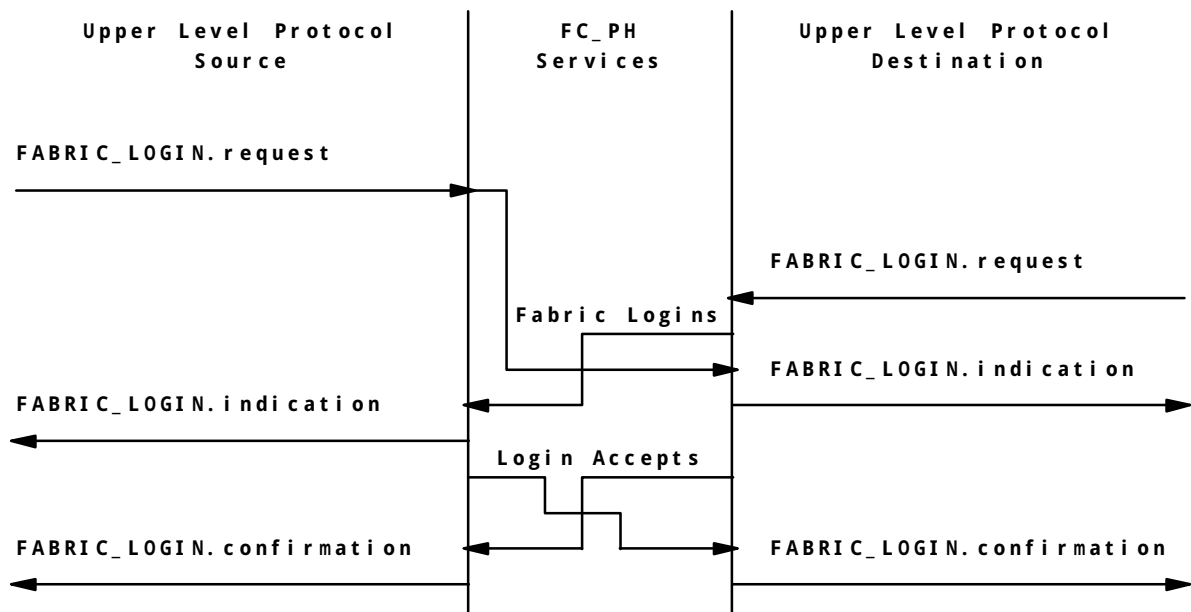


Figure F.3 - Fabric Login Service Primitives in Point-to-Point Configuration

F.2.2 Fabric Login Service Parameters

In the Fabric Login process, the Login Service Parameters for the N_Port are optionally provided to FC-FS with the FABRIC_LOGIN.request Service Primitive and are collectively referenced as My_Fabric_Service_Parameters. Additionally, the Login Service Parameters provided by the Fabric are optionally provided by FC-FS in the FABRIC_LOGIN.confirmation and are collectively referenced as F_Port_Fabric_Service_Parameters. Both My_Fabric_Service_Parameters and Other_Port_Fabric_Service_Parameters are of Fabric_Service_Parameters format as follows:

```
Fabric_Service_Parameters (
fabric_common_service_parameters (
    FC_FS_Highest_Version,
    FC_FS_Lowest_Version,
    Buffer_to_Buffer_Credit,
    Buffer_to_Buffer_Receive_Data_Size,
    R_A_TOV,
    E_D_TOV
),
    Port_Name,
    Node/Fabric_Name,
    class_1_service_parameters (
        Class_Velocity,
        Intermix,
        Stacked_Connect_Requests
    ),
    class_2_service_parameters (
        Class_Velocity,
        Sequential_Delivery
    ),
    class_3_service_parameters (
        Class_Velocity,
        Sequential_Delivery
    )
)
```

The FC_FS_Highest_Version and FC_FS_Lowest_Version will optionally specify the supported version range (see 17.6.2.1, 17.7.1.1).

The Buffer_to_Buffer_Credit will optionally specify the buffers available as defined in 17.6.2.2 and 17.7.1.2.

The Buffer_to_Buffer_Receive_Data_Size will optionally specify the largest data frame that can be received as defined in 17.6.2.5 and 17.7.1.5.

The R_A_TOV and E_D_TOV will optionally specify appropriate timeout values as defined in 17.7.1.6 and 17.7.1.7.

The Port_Name will be specified as defined in 17.6.4 and 17.7.2.

The Node/Fabric_Name will be specified as defined in 17.6.5 and 17.7.3.

The Class_Velocity will optionally be specified as per 17.6.7.1 and 17.7.4.1.

The Intermix parameter will optionally be specified as per 17.6.7.2 and 17.7.4.2.

The Stacked_Connect_Requests parameter will optionally be specified as per 17.6.7.2 and 17.7.4.2.

The Sequential_Delivery parameter will optionally be specified as per 17.6.7.2 and 17.7.4.2.

F.2.3 FABRIC_LOGIN.request

This primitive is used to provide Fabric Login parameters and to request a login with the Fabric.

F.2.3.1 Semantics of the Primitive

```
FABRIC_LOGIN.request    (
                        My_ID,
                        My_Fabric_Service_Parameters
                        )
```

The My_ID will specify the SID to be used in the Sequence that delivers the Fabric Login.

My_Fabric_Service_Parameters will optionally specify the parameters to be used in the payload of the Fabric Login.

F.2.3.2 When Generated

A level above FC-FS will generate this primitive to provide operating parameters to FC-FS and to request a Fabric Login.

F.2.3.3 Effect of Receipt

The receipt of this primitive will cause FC-FS to attempt Link Initialization (see 10.6.2) if the Link is not Active and to transmit a Fabric Login Sequence with Class as specified by 17.3.3.

F.2.4 FABRIC_LOGIN.indication

This primitive is used to indicate that a Fabric Login was received from another N_Port.

F.2.4.1 Semantics of the Primitive

```
FABRIC_LOGIN.indication (Other_Port_Fabric_Service_Parameters)
```

The Other_Port_Fabric_Service_Parameters will optionally specify the parameters from the payload of the received Fabric Login.

F.2.4.2 When Generated

This primitive will be generated upon the reception of a Fabric Login Sequence.

F.2.4.3 Effect of Receipt

Actions upon the receipt of this primitive are unspecified.

F.2.5 FABRIC_LOGIN.confirmation

This primitive will provide an appropriate response to the FABRIC_LOGIN.request primitive signifying the success of the primitive and if a Fabric is present will provide the parameters returned by the Fabric.

F.2.5.1 Semantics of the Primitive

```
FABRIC_LOGIN.confirmation    (
                        My_ID,
                        Request_Status,
```

```

Reject_Reason,
Fabric_Status,
Other_Port_Fabric_Service_Parameters
)

```

The My_ID will reflect the D_ID returned in the Fabric Login Accept Frame.

The Request_Status will indicate status as one of the following:

- Successful - Fabric Login completed
- Unsuccessful - Sequence was not delivered completely due to reason other than reject
- Rejected_Request - The Request was not sent by the Initiator due to the specified Reject_Reason.
- Rejected_by_Fabric - Reject frame received from Fabric
- Rejected_by_N_Port - Reject frame received from N_Port
- Rejected_by_Link_Services - Link Services Reject frame received from N_Port

When the Request_Status is Rejected_Request, Rejected_by_Fabric or Rejected_by_N_Port, the Reject_Reason will indicate one of the Reject reason codes given in Table 40.

The Fabric_Status will indicate status as one of the following:

- isolated - Link is not connected
- no_fabric - N_Port is connected point-to-point with another N_Port
- fabric - N_Port is connected to a Fabric

Other_Port_Fabric_Service_Parameters will optionally specify the parameters to be used for the F_Port in the operation of the Fabric when a Fabric is present as indicated by Fabric_Status, or will optionally specify the parameters to be used for the other N_Port when no_fabric is indicated by Fabric_Status.

F.2.5.2 When Generated

This primitive is generated upon the completion of a Fabric Login attempt.

F.2.5.3 Effect of Receipt

The effect of receipt of this primitive by the FC-4 entity is unspecified.

F.2.6 IMPLICIT_FABRIC_LOGIN.request

This primitive is used to perform implicit Fabric Login and provides for the specification of the complete set of Fabric parameters.

F.2.6.1 Semantics of the Primitive

```

IMPLICIT_FABRIC_LOGIN.request (
    My_ID,
    Fabric_Status,
    My_Fabric_Service_Parameters,
    Other_Port_Fabric_Service_Parameters
)

```

The My_ID will specify my own address.

The Fabric_Status will indicate status as one of the following:

- isolated - Link is not connected
- no_fabric - N_Port is connected point-to-point with another N_Port
- fabric - N_Port is connected to a Fabric

My_Fabric_Service_Parameters will optionally specify the parameters to be used in the operation of the N_Port.

Other_Port_Fabric_Service_Parameters will optionally specify the parameters to be used for the F_Port in the operation of the Fabric when a Fabric is present as indicated by Fabric_Status, or will optionally specify the parameters to be used for the other N_Port when no_fabric is indicated by Fabric_Status.

F.2.6.2 When Generated

FC-4 will generate this primitive to provide operating parameters to FC-FS implicitly.

F.2.6.3 Effect of Receipt

The receipt of this primitive will cause FC-FS to initialize with the specified parameters.

F.2.7 N_Port_Login Primitive Flows

An example of the usage of the N_Port_Login service primitives in a typical N_Port_Login is shown in Figure F.4.

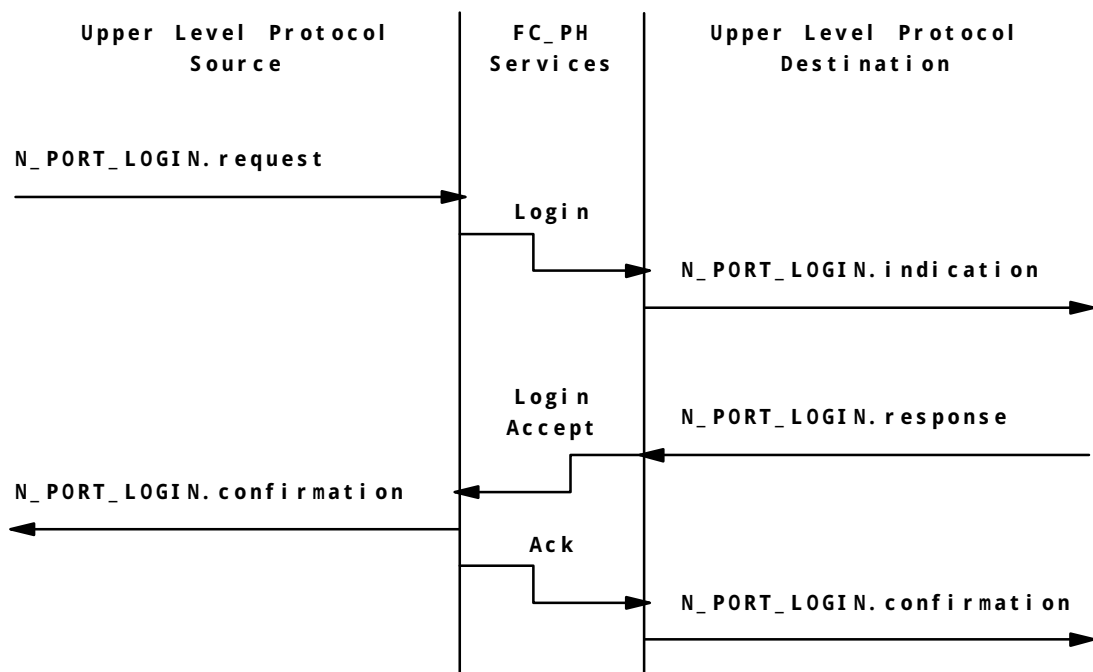


Figure F.4 - N_Port_Login Service Primitives Example

F.2.8 N_Port_Login Service Parameters

In the N_Port_Login process, the Login Service Parameters for the N_Port are optionally provided to the originating FC-FS with the N_PORT_LOGIN.request service primitive and are collectively referenced at the originating N_Port as My_N_Port_Service_Parameters. Additionally, the Login Service Parameters provided by the responding N_Port are optionally provided by FC-FS in the FABRIC_LOGIN.confirmation and are collectively referenced at the originating N_Port as Other_N_Port_Service_Parameters.

Similarly, the Login Service Parameters passed with the N_PORT_LOGIN.response are collectively referenced at the responding N_Port as My_N_Port_Service_Parameters and the Login Service Parameters indicated with the N_PORT_LOGIN.confirmation are collectively referenced at the responding N_Port as Other_N_Port_Service_Parameters.

Both My_N_Port_Service_Parameters and Other_N_Port_Service_Parameters are of N_Port_Service_Parameters format as follows:

```

N_Port_Service_Parameters (
    n_port_common_service_parameters (
        FC_FS_Highest_Version,
        FC_FS_Lowest_Version,
        Buffer_to_Buffer_Credit,
        Continuously_Increasing,
        Random_Relative_Offset,
        Buffer_to_Buffer_Receive_Data_Size,
        Total_Concurrent_Sequences,
        Relative_Offset_by_Information_Category,
        E_D_TOV
    ),
    N_Port_Name,
    Node_Name,
    class_1_service_parameters (
        Class_Validity,
        Intermix,
        Initial_Responder_Process_Associator,
        ACK_0_Initiator_Support,
        ACK_0_Recipient_Support,
        XID_Interlock,
        Error_Policy_Support,
        Categories_per_Sequence,
        Receive_Data_Field_Size,
        Concurrent_Sequences,
        N_Port_End_to_End_Credit,
        Open_Sequences_per_Exchange
    ),
    class_2_service_parameters (
        Class_Validity,
        Initial_Responder_Process_Associator,
        Sequential_ACK_Transmission,
        ACK_0_Initiator_Support,
        ACK_0_Recipient_Support,
        XID_Interlock,
        Error_Policy_Support,
        Categories_per_Sequence,
        Concurrent_Sequences,
        N_Port_End_to_End_Credit,
        Open_Sequences_per_Exchange
    ),

```

```

class_3_service_parameters (
    Class_Validity,
    Initial_Responder_Process_Associator,
    Error_Policy_Support,
    Categories_per_Sequence,
    Concurrent_Sequences,
    Open_Sequences_per_Exchange
)

```

The FC_FS_Highest_Version and FC_FS_Lowest_Version will optionally specify the supported version range and will equal the value specified for Fabric Login(see 17.6.2.1).

The Buffer_to_Buffer_Credit will optionally specify the buffers available point-to-point as defined in 17.6.3.2.

The Continuously_Increasing will specify relative offset properties as defined in 17.6.3.3.

The Random_Relative_Offset will specify relative offset properties as defined in 17.6.3.3.

The Buffer_to_Buffer_Receive_Data_Size will optionally specify the largest data frame that can be received as defined in 17.6.2.5.

The Total_Concurrent_Sequences will be specified as defined in 17.6.3.5.

The Relative_Offset_by_Information_Category will be specified as defined in 17.6.3.5.

The E_D_TOV will be specified as defined in 17.6.3.6.

The N_Port_Name will be specified as defined in 17.6.5.

The Node_Name will be specified as defined in 17.6.5.

The Class_Validity will optionally be specified as per 17.6.8.1.

The Intermix parameter will optionally be specified as per 17.6.8.2.

The Initial_Responder_Process_Associator parameter will optionally be specified as per 17.6.8.3.

The ACK_0_Initiator_Support parameter will optionally be specified as per 17.6.8.4.

The ACK_0_Initiator_Support parameter will optionally be specified as per 17.6.8.4.

The XID_Interlock parameter will optionally be specified as per 17.6.8.4.

The Error_Policy_Support parameter will optionally be specified as per 17.6.8.4.

The Categories_per_Sequence parameter will optionally be specified as per 17.6.8.4.

The Receive_Data_Field_Size parameter will optionally be specified as per 17.6.8.5.

The Concurrent_Sequences parameter will optionally be specified as per 17.6.8.6.

The N_Port_End_to_End_Credit parameter will optionally be specified as per 17.6.8.7.

The Open_Sequences_per_Exchange parameter will optionally be specified as per 17.6.8.8.

F.2.9 N_PORT_LOGIN.request

This primitive is used to provide N_Port_Login parameters and to request a login with another N_Port.

F.2.9.1 Semantics of the Primitive

```
N_PORT_LOGIN.request (
    My_ID,
    Other_ID,
    My_N_Port_Service_Parameters
)
```

The My_ID will specify the SID to be used in the Sequence that delivers the N_Port_Login.

The Other_ID will specify the DID of the N_Port_Login.

My_N_Port_Service_Parameters will optionally specify the parameters to be used in the payload of the N_Port_Login. N_Port_Login Parameters will be consistent with the parameters used in Fabric Login.

F.2.9.2 When Generated

FC-4 will generate this primitive to request a N_Port_Login.

F.2.9.3 Effect of Receipt

The receipt of this primitive will cause FC-FS to transmit a N_Port_Login Sequence with a Class as specified by 17.4.3.

F.2.10 N_PORT_LOGIN.indication

This primitive is used to provide N_Port_Login parameters from a requesting N_Port.

F.2.10.1 Semantics of the Primitive

```
N_PORT_LOGIN.indication (
    My_ID,
    Other_ID,
    Other_N_Port_Service_Parameters
)
```

The My_ID will specify the DID in the received N_Port_Login Sequence.

The Other_ID will specify the SID of the N_Port_Login.

Other_N_Port_Service_Parameters will optionally specify the parameters that were received in the N_Port_Login Sequence.

F.2.10.2 When Generated

This primitive is generated upon the receipt of a N_Port_Login Sequence.

F.2.10.3 Effect of Receipt

Upon receipt of this primitive, FC-4 will optionally generate a N_Port_LOGIN.response to cause an Accept Sequence to be sent in response to the N_Port_Login.

F.2.11 N_PORT_LOGIN.response

This primitive is used to provide N_Port_Login parameters for accepting a request for a Login with another N_Port.

F.2.11.1 Semantics of the Primitive

```
N_PORT_LOGIN.response (
    My_ID,
    Other_ID,
    My_N_Port_Service_Parameters
)
```

The My_ID will specify the SID to be used in the Sequence that delivers the N_Port_Login Accept.

The Other_ID will specify the DID of the N_Port_Login Accept.

My_N_Port_Service_Parameters will optionally specify the parameters to be used in the payload of the N_Port_Login Accept. N_Port_Login Parameters will be consistent with the parameters used in Fabric Login.

F.2.11.2 When Generated

FC-4 will generate this primitive to respond to a N_Port_Login with an Accept.

F.2.11.3 Effect of Receipt

The receipt of this primitive will cause FC-FS to transmit a N_Port_Login Accept Sequence.

F.2.12 N_PORT_LOGIN.confirmation

This primitive will provide an appropriate response to the N_PORT_LOGIN.request primitive signifying the success or failure of the primitive.

F.2.12.1 Semantics of the Primitive

```
N_PORT_LOGIN.confirmation (
    My_ID,
    Other_ID,
    Request_Status,
    Reject_Reason,
    Other_N_Port_Service_Parameters
)
```

The My_ID will specify my own address used in the N_Port_Login.

The Other_ID will specify the SID of the other N_Port.

The Request_Status parameter will indicate status as one of the following:

- Successful - N_Port_Login completed
- Unsuccessful - Sequence was not delivered completely due to reason other than reject
- Rejected_Request - The Request was not sent by the Initiator due to the specified Reject_Reason.
- Rejected_by_Fabric - Reject frame received from Fabric
- Rejected_by_N_Port - Reject frame received from N_Port

- Rejected_by_Link_Services - Link Services Reject frame received from N_Port

When the Request_Status is Rejected_Request, Rejected_by_Fabric or Rejected_by_N_Port, the Reject_Reason will indicate one of the Reject reason codes given in Table 40.

Other_N_Port_Service_Parameters will optionally specify the parameters that were received from the other N_Port during Login.

F.2.12.2 When Generated

N_PORT_LOGIN.confirmation is generated by FC-FS in response to a N_Port_LOGIN.request.

F.2.12.3 Effect of Receipt

The effect of receipt of this primitive by the FC-4 entity is unspecified.

F.2.13 IMPLICIT_N_PORT_LOGIN.request

This primitive is used to perform an Implicit N_Port_Login.

F.2.13.1 Semantics of the Primitive

```
IMPLICIT_N_PORT_LOGIN.request      (
                                    My_ID,
                                    Other_ID,
                                    My_N_Port_Service_Parameters,
                                    Other_N_Port_Service_Parameters
                                    )
```

The My_ID will specify my own address.

The Other_ID will specify the N_Port_Login destination address.

My_N_Port_Service_Parameters will optionally specify the parameters to be used in the operation of the N_Port.

Other_N_Port_Service_Parameters will optionally specify the parameters to be used for the Other_N_Port for operation.

F.2.13.2 When Generated

This primitive is generated to provide operating parameters to FC-FS implicitly.

F.2.13.3 Effect of Receipt

The receipt of this primitive will cause FC-FS to initialize with the specified parameters.

F.2.14 N_PORT_LOGOUT.request

This primitive is used to request that a Login be terminated with the specified N_Port.

F.2.14.1 Semantics of the Primitive

```
N_PORT_LOGOUT.request (
                            My_ID,
                            My_N_Port_Name,
                            Other_ID,
                            Class
                            )
```


The My_ID will specify my own address.

The My_N_Port_Name is the N_Port_Name for my Port and will be specified as defined in 17.6.4.

Other_ID will specify the D_ID for the Logout Sequence.

Class will specify the Class for the Logout Sequence.

F.2.14.2 When Generated

This primitive will be generated to request a Logout Sequence.

F.2.14.3 Effect of Receipt

Receipt of this primitive will cause FC-FS to generate a Logout Sequence.

F.2.15 N_PORT_LOGOUT.indication

This primitive is used to indicate the receipt of a Logout Sequence.

F.2.15.1 Semantics of the Primitive

```
N_PORT_LOGOUT.indication    (
                                Other_ID
                                Other_N_Port_Name,
                                )
```

Other_ID will specify the S_ID of the received Logout Sequence.

The Other_N_Port_Name is the N_Port_Name for the other Port and will be specified as defined in 17.6.4.

F.2.15.2 When Generated

This primitive will be generated to in response to the receipt of a Logout Sequence.

NOTE - Logout may cause Sequences in other Active Exchanges with this N_Port to be aborted.

F.2.15.3 Effect of Receipt

The effect of receipt of this primitive is unspecified.

F.2.16 N_PORT_LOGOUT.confirmation

This primitive is used to confirm the completion of a Logout

F.2.16.1 Semantics of the Primitive

```
N_PORT_LOGOUT.confirmation  (
                                Other_ID
                                Request_Status,
                                Reject_Reason
                                )
```

Other_ID will specify the D_ID for the Logout Sequence.

The Request_Status parameter will indicate status as one of the following:

- Successful - N_Port_Login completed

- Unsuccessful - Sequence was not delivered completely due to reason other than reject
- Rejected_Request - The Request was not sent by the Initiator due to the specified Reject_Reason.
- Rejected_by_Fabric - Reject frame received from Fabric
- Rejected_by_N_Port - Reject frame received from N_Port
- Rejected_by_Link_Services - Link Services Reject frame received from N_Port

When the Request_Status is Rejected_Request, Rejected_by_Fabric or Rejected_by_N_Port, the Reject_Reason will indicate one of the Reject reason codes given in Table 40.

F.2.16.2 When Generated

This primitive will be generated to confirm completion of a Logout Sequence.

F.2.16.3 Effect of Receipt

The effect of receipt of this primitive is unspecified.

F.3 Link Services

Other Link Services in addition to Login are provided. These services include:

- Abort_Exchange
- Read_Connection_Status
- Read_Exchange_Status_Block
- Read_Sequence_Status_Block
- Request_Sequence_Initiative
- Establish_Streaming
- Estimate_Credit
- Advise_Credit
- Read_Timeout_Value
- Read_Link_Status
- Echo
- Test
- Reinstate_Recovery_Qualifier
- Link_Credit_Reset
- Remove_Connection
- Abort_Sequence
- Link_Reset
- N_Port_Reset
- Link_Offline

F.3.1 LS_CONTROL.request

This primitive is used to perform Link Service actions.

F.3.1.1 Semantics of the Primitive

```

LS_CONTROL.request      (
                          Control_Action,
                          My_ID,
                          Other_ID,
                          Request_Tag,
                          Target_Exchange_Tag,
                          Target_Sequence_Tag,
                          Payload
                          )

```

The Control_Action parameter will include the following:

- Abort_Exchange
- Read_Connection_Status
- Read_Exchange_Status_Block
- Read_Sequence_Status_Block
- Request_Sequence_Initiative
- Establish_Streaming
- Estimate_Credit
- Advise_Credit
- Read_Timeout_Value
- Read_Link_Status
- Echo
- Test
- Reinstate_Recovery_Qualifier
- Link_Credit_Reset
- Remove_Connection
- Abort_Sequence
- Link_Reset
- N_Port_Reset

The My_ID and Other_ID optionally provide address identifiers for the request.

The Request_Tag parameter will provide a local identifier of this request.

The Target_Exchange_Tag parameter will provide a local identifier of the target Exchange when the Control_Action is Abort_Sequence, Abort_Exchange, Read_Exchange_Status_Block, or Read_Sequence_Status_Block.

The Target_Sequence_Tag parameter will provide the Sequence_Tag of the target Sequence when the Control_Action parameter is Abort_Sequence or Read_Sequence_Status_Block.

The Payload will optionally provide a payload for the requested Link Service Sequence. The payload will optionally provide additional information sufficient to identify and process the request according to Table 194.

F.3.1.2 When Generated

This primitive is generated by FC-4 to request a Link Service.

F.3.1.3 Effect of Receipt

Upon receipt of this primitive, FC-FS issues the associated Link Service Sequence.

F.3.2 LS_CONTROL.confirmation

This primitive is used to provide a confirmation of the completion of a Link Services request.

F.3.2.1 Semantics of the Primitive

```
LS_CONTROL.confirmation (
    Control_Action,
    My_ID,
    Other_ID,
    Request_Tag,
    Target_Exchange_Tag,
    Target_Sequence_Tag,
    Request_Status,
    Reject_Reason,
    Payload
)
```

The Control_Action is the requested Link Service.

The My_ID and Other_ID optionally are the address identifiers from the request.

The Exchange_Tag parameter will provide a local identifier of this request.

The Target_Exchange_Tag parameter will provide a local identifier of the target Exchange when the Control_Action is Abort_Sequence, Abort_Exchange, Read_Exchange_Status_Block, or Read_Sequence_Status_Block.

The Target_Sequence_Tag parameter will provide the Sequence_Tag of the target Sequence when the Control_Action parameter is Abort_Sequence or Read_Sequence_Status_Block.

The Request_Status parameter will indicate status as one of the following:

- Successful - N_Port_Login completed
- Unsuccessful - Sequence was not delivered completely due to reason other than reject
- Rejected_Request - The Request was not sent by the Initiator due to the specified Reject_Reason.
- Rejected_by_Fabric - Reject frame received from Fabric
- Rejected_by_N_Port - Reject frame received from N_Port
- Rejected_by_Link_Services - Link Services Reject frame received from N_Port

When the Request_Status is Rejected_Request, Rejected_by_Fabric or Rejected_by_N_Port, the Reject_Reason will indicate one of the Reject reason codes given in Table 40.

The Payload will optionally provide the returned payload for the Link Service Accept.

F.3.2.2 When Generated

This primitive will be generated upon the completion of a Link Service request.

F.3.2.3 Effect of Receipt

The effect of receipt of this primitive by the FC-4 entity is unspecified.

Annex G

(informative)

Out of order characteristics

This annex describes some of the implications of out of order transfer. There are two cases considered:

- Out of order transfer of Data frames due to the inability of a Fabric to maintain order.
- Out of order transmission of ACKs by a N_Port due to its buffer availability algorithms.

G.1 Out of order Data frame delivery

Based on F_Port service parameters, the delivery of frames during Class 2 service may occur as:

- "Misordered Delivery". The Destination N_Port receives frames in an order different than a Source N_Port sent them, i.e., the Fabric does not maintain the ordering of the frames.
- "Ordered Delivery". The Destination N_Port receives frames in the same order as the Source N_Port sent them, i.e., the Fabric maintains the ordering of the frames.

The implications of misordered delivery and Class 2 Sequence recovery will be discussed.

Misordered frame delivery can occur whenever there are multiple routes, within the Fabric, between two communicating N_Ports. When a Sequence is initiated, the individual frames of the Sequence are independently routed by the Fabric and, therefore, may take different routes through the Fabric, with some routes being longer or shorter than others. This may cause the misordered delivery of frames to the destination N_Port. Also, since each frame is independently routed, it is very difficult for the Fabric to purge, or flush from the Fabric, all the frames for a Sequence.

Because of the above, this standard has provided the following functions to aid in the detection and recovery of Sequences abnormally terminated due to timeout, e.g., because a frame was lost:

- A timeout, R_A_TOV, specified by the Fabric, which is twice the longest amount of time that a frame may stay in the Fabric before it is either delivered, or it will never be delivered.
- Establishment of a Recovery_Qualifier range for the duration of the R_A_TOV time. If a Sequence has been aborted, the Sequence Recipient will discard any received Data frames that fall within this range, for a period of R_A_TOV. Likewise, the Sequence Initiator will discard any received Link_Control frames that fall within this range, for a period of R_A_TOV.

These functions have several implications:

- When a N_Port is initialized, it may not have knowledge of Sequences initiated prior to initialization. For example, a N_Port may be powered off after sending a Sequence, and then powered back on. Some (or all) frames of this prior Sequence may still be traversing the Fabric after the N_Port has been initialized. Therefore, after initialization, a N_Port must wait the R_A_TOV time before it initiates any Sequences. This ensures that there will be no duplicate frames in the Fabric.
- The establishment of Recovery_Qualifiers requires a N_Port to maintain a list of Recovery_Qualifiers. Entries must be added to this list when a Sequence is abnormally terminated, and entries must be deleted from this list when R_A_TOV has expired for the entry. The list must be referenced prior to Sequence initiation to ensure that a Data frame that falls within a Recovery_Qualifier range is not transmitted.

- If a subset of the entire Sequence_Qualifier (e.g., X_ID) is used to route and store incoming frames, a frame falling within the Recovery_Qualifier range may not be detected until after the frame is placed in a receive buffer and the frame header is validated. This has implications on credit and buffer management.

Note - the Sequence to which this frame belongs was abnormally terminated and all the credit for the Sequence was recovered. As a result, this frame is an "unexpected" frame that is not accounted for by the current credit management within the N_Port. Therefore, it may be occupying a buffer that a source N_Port believes is available. This may cause another frame to receive a P_BSY, even though the sender of the busied frame obeyed the credit rules. Note that this can happen to a Class 1 frame if the receiving N_Port supports intermix. Therefore, a Class 1 frame, other than SOFc1, may receive a P_BSY in violation of the Class 1 rules.

G.2 Out of order ACK transmission

The transmission of ACK frames in either Class 1 or Class 2 service may occur as:

- "Misordered Transmission". In this case, the Sequence Recipient is not acknowledging Data frames in the SEQ_CNT order. That is, the corresponding ACK frames are not being sent in SEQ_CNT order.
- "Ordered Transmission". In this case, the Sequence Recipient is acknowledging Data frames in the SEQ_CNT order. That is, the corresponding ACK frames are being sent in SEQ_CNT order.

The implications of misordered transmission of ACKs and ordered transmission of ACKs are:

- With misordered transmission, the credit for a lost ACK cannot be recovered until after a Sequence timeout is detected. That is, the credit is lost until the E_D_TOV time has expired.
- With ordered transmission, the reception of an ACK recovers the credit for all Data frames with that SEQ_CNT or lower, regardless of whether previous ACKs were received. This is true regardless of whether the Fabric supports misordered delivery or ordered delivery.

Annex H (informative) Link Error Status Block

In this annex, guidelines are provided to manage the Link Error Status Block (see 23.8).

H.1 Link failure counters

Four types of Link failures are recorded in individual counters in LESB. The Link failure counters are:

- Link Failure Count (Word 0) (miscellaneous link errors)
- Loss of Synchronization Count (Word 1) (confirmed and persistent sync. loss)
- Loss of Signal Count (Word 2)
- Primitive Sequence Protocol Error Count (Word 3)

Link failure counters summary

The conditions under which individual counter increments are summarized in Table J.1. (See Table E.6 and 10.4.3 for specific state changes, related nomenclature, considerations and conditions.)

H.2 Invalid Transmission Word

The Invalid Transmission Word Counter (Word 4) increments, once for every Invalid Transmission Word received (see 5.2.2.2 and 6.1.3.1), except during the following conditions:

- No Transmission Word errors are counted if the receiver is in the Loss-of-Synchronization state (see 6.1.3)
- No Transmission Word errors are counted if the Port is in the OLS Receive state (OL2) or the Wait for OLS state (OL3) (see 10.6.3)

H.3 Invalid CRC Count

The Invalid CRC Count (Word 5) increments, once for every received frame which meets one of the following conditions:

1. The Port is in the Active State (AC) and the received frame's CRC is in error and the frame is either missing an EOF delimiter or the EOF delimiter is an EOF_n, EOF_t, or EOF_{dt} (see note under 11.6.2)
2. The Port is in the Active State (AC) and the received frame's CRC is in error (see note under 11.6.2)

NOTE - The frames received with EOF_{ni}, EOF_d_{ti}, or EOF_a may be excluded from consideration.

Table J.1 - Link Failure Counters and management

State	ACTIVE	LINK RECOVERY			LINK FAILURE		OFFLINE		
Input Event	IDLE RECV	LR XMIT	LR RECV	LRR RECV	NOS RECV	NOS XMIT	OLS XMIT	OLS RECV	WAIT OLS
Substate	(AC)	(LR1)	(LR2)	(LR3)	(LF1)	(LF2)	(OL1)	(OL2)	(OLS)

L >> LR	-	-	-	-	-	-	-	-	(1) Wd 3↑ (Ref. 17)
L >> LRR	(1) Wd 3↑ (Ref. 1)	-	-	-	-	-	-	-	(1) Wd 3↑ (Ref. 17)
L >> IDLES	-	-	-	-	-	-	-	-	-
L >> OLS	-	-	-	-	-	-	-	-	-
L >> NOS	Wd 0↑ (Ref. 1)	Wd 0↑ (Ref. 2)	Wd 0↑ (Ref. 5)	Wd 0↑ (Ref. 8)	- (Ref. 12)	(2) - (Ref. 11)	(3) Wd 3↑ (Ref. 20)	Wd 0↑ (Ref. 15)	(2) - (Ref. 17)
Loss of Signal	Wd 2↑ (Ref. 21)	Wd 2↑ (Ref. 4)	Wd 2↑ (Ref. 7)	Wd 2↑ (Ref. 10)	Wd 2↑ (Ref. 13)	- (Ref. 11)	- (Ref. 18)	- (Ref. 16)	- (Ref. 17)
Loss of Sync > Limit	Wd 1↑ (Ref. 21)	(4) Wd 1↑ (Ref. 4)	(4) Wd 1↑ (Ref. 7)	(4) Wd 1↑ (Ref. 10)	(4) Wd 1↑ (Ref. 13)	- (Ref. 11)	- (Ref. 18)	- (Ref. 16)	- (Ref. 17)
Event timeout (R_T_TOV)	-	Wd 0↑ (Ref. 3)	Wd 0↑ (Ref. 6)	Wd 0↑ (Ref. 9)	Wd 0↑ (Ref. 14)	-	- (Ref. 19)	-	-
LEGEND: L >> means receiving from the Link An - entry means no change to counter Ref.N means refer to the section listed near bottom of this table Wd 0↑ means increment Link Failure Counter (Word 0) Wd 1↑ means increment Loss of Synchronization Counter (Word 1) Wd 2↑ means increment Loss of Signal Counter (Word 2) Wd 3↑ means increment Primitive Sequence Protocol Error Counter (Word 3)									
Ref.1 = 10.5.1 Ref.2 = 10.5.2.1 (a) Ref.3 = 10.5.2.1 (b) Ref.4 = 10.5.2.1 (c) Ref.5 = 10.5.2.2 (a) Ref.6 = 10.5.2.2.(b) Ref.7 = 10.5.2.2 (c) Ref.8 = 10.5.2.3 (a) Ref.9 = 10.5.2.3 (b) Ref.10 = 10.5.2.3 (c) Ref.11 = 10.5.3.1 Ref.12 = 10.5.3.2 (a) Ref.13 = 10.5.3.2 (b) Ref.14 = 10.5.3.2 (c) Ref.15 = 10.5.4.2 (a) Ref.16 = 10.5.4.2 (b) Ref.17 = 10.5.4.3 Ref.18 = 10.6.2 (a) Ref.19 = 10.6.2 (b) Ref.20 = 10.6.3 Ref.21 = 10.6.4									
Notes: 1) Abnormal Link_Response from the attached Port 2) A normal event if the Port is in loopback, or if the attached Port is in the Wait for OLS state. 3) Only increments if the condition occurs while performing the Online to Offline protocol. 4) This condition will not occur, since the Event Timeout occurs first.									

Annex I (informative) Available BB_Credit management example

The following is an example implementation using BB_Credit and available BB_Credit (see Figure I.1). Assume two Ports, A and B, and both are either N_Ports or L_Ports.

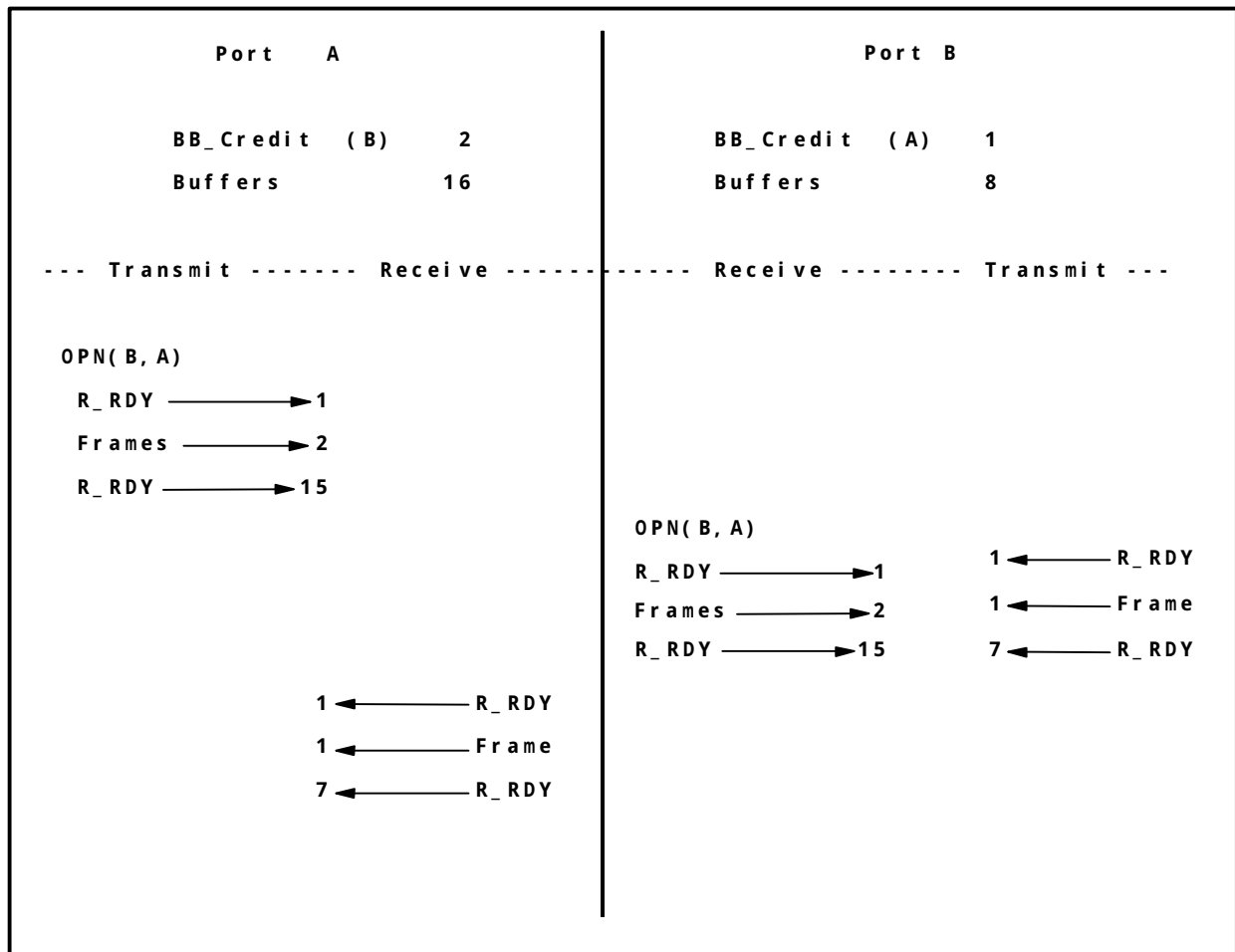


Figure I.1 - Alternate_Credit Use Example

Port A has sixteen receive buffers available and a BB_Credit Login value of two for Port B. Port B has eight receive buffers available and a BB_Credit Login value of one for Port A.

If they are L_Ports, assume that L_Port A has arbitrated and won and is going to open B; full-duplex is used; and, both A and B have frames to transmit.

Port A:

- a) looks up the BB_Credit Login value for Port B (two).
- b) checks to see how many receive buffers it has available (sixteen).

- c) transmits if in the Loop, OPN(B,A), two Fill Words and one R_RDY and three Fill Words. (If N_Port, step 3 is not applicable.)
- d) transmits two frames (BB_Credit Login value for B). Note that had BB_Credit been zero (0), Port A could have sent no frames (prior to receiving R_RDY). The remaining fifteen R_RDYs are transmitted after the first frame.
- e) receives and counts the R_RDYs sent by Port B (eight). Once Port A has received and discounted two R_RDYs (to account for the frames already shipped against Login BB_Credit in step 4), Port A has an available BB_Credit of six that it may use to transmit up to six additional frames.
- f) transmits one frame for each available BB_Credit (and decrement the available BB_Credit by one).
- g) counts the number of R_RDYs received from Port B.
- h) receives frames sent by Port B into the receive buffer(s).
- i) transmits one R_RDY for each receive buffer that becomes available.
- j) repeats steps 6 through 9 until all frames have been sent.
- k) transmits CLS, if in the Loop.
- l) continues to receive frames from B, but transmits no R_RDYs or frames.
- m) receives CLS, if in the Loop and closes its end of the Loop.

Port B:

- a) receives, if in Loop, OPN(B,A) and opens the Loop. (If N_Port, step 1 is not applicable.)
- b) looks up the BB_Credit for Port A (one) and checks to see how many receive buffers it has available (eight).
- c) transmits if in the Loop, two Fill Words and one R_RDY and three Fill Word. (If N_Port, step 4 is not applicable.)
- d) transmits one frame (BB_Credit Login value for Port A). Note that had BB_Credit been zero (0), B could have sent no frames. The remaining seven R_RDYs are transmitted after the first frame.
- e) receives and counts the number of R_RDYs sent by Port A. Once Port B has received and discounted one R_RDY (to account for the frame already shipped against BB_Credit in step 5), Port B has an available BB_Credit of fifteen that it may use to transmit up to fifteen additional frames.
- f) transmits one frame for each available BB_Credit (and decrement the available BB_Credit by one).
- g) counts the number of R_RDYs received from Port A.
- h) receives the number of frames sent by Port A into the receive buffer(s).
- i) transmits an R_RDY for each receive buffer that becomes available.
- j) may repeat steps 6 through 9 until the Available BB_Credit is exhausted.
- k) transmits a CLS, if in the Loop.
- l) Since the CLS has already been received, Port B closes its end of the Loop.

There are several variations on the previous example.

- a) Data frame transfer is only from A to B even though OPN(B,A)(full-duplex) is used. Port A transmits one R_RDY followed by the number of frames represented by the BB_Credit Login value. In this case, B has no Data frames to transmit, but transmits one R_RDY for each available receive buffer and FC-FS Link_Control frames (e.g., ACKs) to A. Note that B may limit the number of frames that A can transmit by transmitting one R_RDY for each

available receive buffer, followed by CLS. Once Port A has received the CLS, it can then only transmit frames until its available BB_Credit is exhausted before it must close its end of the Loop.

- b) BB_Credit is zero (0) on both ends. In this case, neither A nor B can transmit any frames until at least one R_RDY is received. For each R_RDY received, a frame may be sent. Note that when BB_Credit is zero (0), a Loop turn-around delay is required at A before transmitting the first (and possibly the only) frame. By guaranteeing a minimum of receive buffers (as indicated by BB_Credit), this turn-around delay may be eliminated.
- c) Port A transmits an OPN(B,A) (full-duplex), followed by at least one R_RDY, followed by two frames (the BB_Credit Login value for Port B). Port B does not look up the BB_Credit for A, and sends at least one R_RDY (one for each available receive buffer). Port B waits for the first R_RDY from A. When at least one R_RDY is received (i.e., available BB_Credit is one (1)), B can transmit one FC-FS frame.
- d) Port A transmits an OPN(B,B) (half-duplex). In this case, Port B cannot identify A and must wait for the first frame from A to transmit any frames (Note - since this is an half-duplex OPN, no data frames may be transmitted by B). Once a frame is received, the low-order byte of the S_ID is the AL_PA of A. B can then establish the BB_Credit for A. If B is using a minimum Loop value for BB_Credit, the AL_PA of A is not required.
- e) Port A transmits an OPN(B,B) (half-duplex). In this case, Port B must wait for the first R_RDY from A. Once at least one R_RDY is received (i.e., available BB_Credit is one (1)), B can transmit one FC-FS Link_Control frame to A.

Annex J

(informative)

48-bit Global Identifier Format

J.1 General

The IEEE defined 48-bit global identifier (GID_ 48) is assigned by a manufacturer who has been assigned a company_id value by the IEEE Registration Authority. The 48-bit identifier is a concatenation of the 24-bit company assigned by the IEEE Registration Authority and a 24-bit extension identifier assigned by the organization with that company_id assignment.

The IEEE administers the assignments of 24-bit company_id values. The assignments of these values are public, so that a user of a GID-48 value can identify the organization, which provided that value. The IEEE/RAC has no control over the assignments of 24-bit extension identifiers and assumes no liability for assignments of duplicate GID-48 identifiers.

J.2 Format

You may have a 48-bit global identifier (GID-48) provided by an authorized manufacturer of these values (which sell electronically readable chips). The most significant 24 bits of this value are the company_id value assigned to the manufacturer by the IEEE Registration Authority. The least-significant 24-bit extension identifier assigned by the manufacturer.

For example, assume that manufacturer’s IEEE-assigned company_id value is ACDE48₁₆ and the manufacturer’s extension identifier is 234567₁₆. The GID-64 value generated from these two numbers is ACDE48234567₁₆, whose byte and bit representations are illustrated in Figure J.1.

most-significant byte				least-significant byte		
AC	DE	48	23	45	67	(hex)
1010	1101	0100	0010	0100	0110	(binary)
1100	1110	1000	0011	0101	0111	
A+0	A+1	A+2	A+3	A+4	A+5	original byte address

Figure J.1 - 48-bit global identifier example

If provided in byte-addressable media, the original byte-address is specified: the most through least significant bytes of the GID-48 value are contained within the lowest through highest byte addresses, as illustrated.

When transferred to other standard-specific locations (within a disk file or network packet, for example) the relative ordering of the bytes may be changed, as specified within the applicable standard.

Annex K

(informative)

Fibre Channel use of IEEE Global Identifiers

The highest level architecture of Fibre Channel (FC) consists of multiple Nodes (workstations, disk arrays, supercomputers etc.) interconnected by an active intelligent interconnection scheme called a Fabric. Allowance is made for Nodes, which have multiple ports (N_Ports), and each N_Port is connected to a separate Fabric port (F_Port). In this document, the N_Port to F_Port interface is defined explicitly but the operation of the Fabric is only defined in terms of a functional model. However other FC standards define Fabrics using specific technologies, and based on a specific topology.

The Fabric model defines a single, homogenous, 24-bit address space, and those "short" addresses are the basis for routing information within the Fabric. The protocol supported by the N_Port/F_Port interface is also capable of transporting "long" addresses of up to 60 bits in length. These addresses may be globally unique types defined by a number of standards bodies worldwide. FC-FS generically refers to these addresses as WorldWide Names (WWNs). Methods of address resolution are provided between the WWNs and the FC "short" addresses.

These WWNs have two different uses. First, they are used to provide unique identification of items within a FC configuration. Second, they are used to facilitate routing in situations where the FC configuration is a small part of a larger wide area network, or where the Upper Layer Protocols being transported by FC assume the use of those long addresses. A prime example of such an Upper Layer Protocol is the IEEE 802.2 Logical Link Layer protocols mapped to FC as defined by the FC Link Encapsulation (FC-LE) scheme.

For the first use:

- It is acceptable that a single WWN be provided for each Fabric and each Node, and this is then used with a local qualifier to identify the specific port within the Node or Fabric;
- It is also acceptable that a separate WWN be provided with each N_Port or F_Port.

The situation for the second use is more complex. Where:

- an application designed for an IEEE 802 LAN is moved with minimal modification into a FC environment;
- communication occurs between an FC-LE Entity and an IEEE 802 station on a LAN; or
- FC provides an intermediate part of a path between two IEEE 802 stations on LANs (which is handled by IEEE 802.1G remote bridging);

It is strongly suggested that each N_Port and F-Port be assigned its own WWN.

In addition, the use of bridging techniques to communicate between Entities implementing FC-LE and systems on the IEEE 802 LAN to:

- allow the FC-LE entity be able to address the system on the IEEE 802 LAN; and
- allow the system on the IEEE 802 LAN to use ULAs to address the FC-LE Entity requires that each N_Port and F_Port be assigned its own WWN.

One of the formats defined as usable for a WWN is an IEEE 48-bit address format. This is specified to contain a 48-bit IEEE Standard 802.1A Universal LAN MAC Address (ULA). The ULA is represented as an ordered string of six bytes numbered from 0 to 5. The least significant bit of byte 0 is identified as the Individual/Group Address (I/G) bit. The next least significant bit is identified as the Universally or Locally Administered Address (U/L) bit. It is further specified that layout of the bytes in two FC standard 32 bit words is as follows:

Table K.1 - Fibre Channel use of IEEE Global Identifiers								
Bits 63	55	47		39	31	23	15	07 00
		ULA Byte 0	UI // LG	ULA Byte 1	ULA Byte 2	ULA Byte 3	ULA Byte 4	ULA Byte 5

NOTE - All fields in the FC-FS standard are defined as having the most significant bit in the highest numbered bit position, and the most significant byte in the position closest to the start of the FC-FS Packet.

The FC-FS standard identifies no further details of the contents of the ULA.

Annex L (informative) Class 4 Examples

Two ladder diagrams illustrating the operation of a single Class 4 circuit is shown in Figure L.1 and Figure L.2. Both figures include details on how End-to-end and VC credit are managed. Figure L.1 shows how a N_Port request setup of a Class 4 circuit, the subsequent circuit establishment, followed by an activation and deactivation cycle. Figure L.2 shows the non-destructive collision of two activation requests, normal Class 4 frame and acknowledgment flow ending with the removal of the Class 4 circuit.

L.1 Setup, activate and deactivate

The following describes in detail the Frame and Class 4 Primitive Signal flow shown in Figure L.1:

- a) The Circuit Initiator, N_Port 'A', enters the Pending state and transmits a Quality of Service request (a) (QoSR) to the Quality of Service Facilitator (QoSF). The QoSF sets F_Port 'A' to the Pending state and builds two counter-directional virtual circuits across the Fabric, between F_Port 'A' and F_Port 'B', satisfying the QoS parameters.
- b) The QoSF acknowledge (b) the receipt of the QoSR frame, if either Class 2 or 4 is used to request the Class 4 circuit setup.
- c) F_Port 'B' enters the Pending state before the updated QoSR (c) frame is sent to N_Port 'B'. The QoSF determine which Service Class to use to propagate the setup request. N_Port 'B' enters the Pending state and initialize the service parameters, including the credit environment when it processes the QoSR.
- d) N_Port 'B' acknowledge (d) reception of the QoSR frame if the QoSF uses Class 2 or 4 to continue the Class 4 circuit setup process.
- e) N_Port 'B' transmits an ACC (e) frame, in response to the QoSR frame (c), once it is ready to proceed. The QoSF initialize the environment for the Class 4 circuit in response to the ACC frame.
- f) The QoSF acknowledge (f) reception of the ACC if Class 2 or 4 was selected by the QoSF to complete setup of the Class 4 circuit.
- g) When the Fabric environment have been initialized the QoSF return an ACC (g) frame to N_Port 'A' in response to the initial setup request (a). N_Port 'A' initialize the service parameters, including the credit environment when it processes the ACC.
- h) N_Port 'A' acknowledge (h) receipt of the ACC frame if N_Port 'A' selected Class 2 or 4 to request setup of the Class 4 circuit.
- i) F_Port 'A' transmits a VC_RDY (i) and enters the Dormant state, when it is ready to accept an activation request. N_Port 'A' enters the Dormant state when it receives the VC_RDY and increases the VC credit limit and VC_Credit.
- j) F_Port 'B' transmits a VC_RDY (j) and enters the Dormant state, when it is ready to accept an activation request. N_Port 'B' enters the Dormant state when it receives the VC_RDY and increases the VC credit limit and VC_Credit.
- k) F_Port 'A' transmits a VC_RDY (k) advertising excess bandwidth. However, N_Port 'A' does not increase its VC_Credit when it receives the VC_RDY since VC_Credit already is at the VC_Credit limit.
- l) N_Port 'B' transmits an activation request (l), consuming one EE_Credit and one VC_Credit. Transmission of the SOFc4 delimited causes N_Port to enter the Live state. This also change the VC credit limit to 2, the Live VC credit limit. F_Port 'B' enters the Live state when it receives the SOFc4 delimited activation frame. The frame then travers-

es the Fabric and is forwarded by F_Port 'A'. F_Port 'A' enters the Live state when it transmits the SOFc4 delimiter. N_Port 'A' enters the Live state when it receives the activation frame. This also changes the VC credit limit to the Live VC credit limit.

- m) F_Port 'B' transmit a VC_RDY (m) to replenish VC credit. N_Port 'B' increases VC_Credit when it receives the VC_RDY.
- n) F_Port 'B' transmit a VC_RDY (n) to replenish VC credit. N_Port 'B' increases VC_Credit when it receives the VC_RDY.
- o) F_Port 'A' transmit a VC_RDY (o) to replenish VC credit. N_Port 'A' increases VC_Credit when it receives the VC_RDY.
- p) F_Port 'A' transmit a VC_RDY (p) to replenish VC credit. However, N_Port 'A' does not increase VC_Credit since VC_Credit already is at the VC credit limit.
- q) N_Port 'B' transmits a Data frame (q) consuming one EE_Credit and one VC_Credit. The frame is received by F_Port 'B', traverses the Fabric, is forwarded by F_Port 'A' and is received by N_Port 'A'. N_Port 'A' later acknowledge (w) the frame reception.
- r) N_Port 'B' transmits its last Data frame (r) to the Sequence initiated by (l) requesting deactivation of the Class 4 circuit. This consumes one EE_Credit and one VC_Credit. The frame is received by F_Port 'B', traverses the Fabric, is forwarded by F_Port 'A' and is received by N_Port 'A'. N_Port 'A' later acknowledge (x) and honor the deactivation request.
- s) N_Port 'A' acknowledge reception of the activation frame (l) by transmitting an ACK frame (s), consuming one VC_Credit. The ACK frame is received by F_Port 'A', traverses the Fabric, is forwarded by F_Port 'B' and is received by N_Port 'B'. N_Port 'B' reclaim one EE_Credit on receipt of the ACK frame.
- t) F_Port 'A' transmit a VC_RDY (f) to replenish VC credit. N_Port 'A' increases VC_Credit when it receives the VC_RDY.
- u) F_Port 'B' transmit a VC_RDY (u) to replenish VC credit. N_Port 'B' increases VC_Credit when it receives the VC_RDY.
- v) F_Port 'B' transmit a VC_RDY (v) to replenish VC credit. N_Port 'B' increases VC_Credit when it receives the VC_RDY.
- w) N_Port 'A' acknowledge reception of the Data frame (q) by transmitting an ACK frame (w), consuming one VC_Credit. The ACK frame is received by F_Port 'A', traverses the Fabric, is forwarded by F_Port 'B' and is received by N_Port 'B'. N_Port 'B' reclaim one EE_Credit on receipt of the ACK frame.
- x) N_Port 'A' acknowledge reception of Data frame (r) by transmitting an ACK frame (x), consuming one VC_Credit. N_Port 'A' elects to honor the deactivation request made by N_Port 'B'. The EOFdt delimited ACK frame is received by F_Port 'A', traverses the Fabric, is forwarded by F_Port 'B' and is received by N_Port 'B'. N_Port 'A' enters the Dormant state when it transmits the EOFdt delimiter, F_Port 'A' enters the Dormant state when it receives the EOFdt delimiter, F_Port 'B' enters the Dormant state when it transmit the EOFdt delimiter and N_Port 'B' enters the Dormant state when it receives the EOFdt delimiter. N_Port 'B' re-initializes EE_Credit on receipt of the ACK frame since the Class 4 virtual connection is now Dormant.
- y) F_Port 'B' transmit a VC_RDY (y) to replenish VC credit. However, N_Port 'B' does not increase VC_Credit since VC_Credit already is at the VC credit limit.
- z) F_Port 'A' transmit a VC_RDY (z) to replenish VC credit. N_Port 'A' increases VC_Credit when it receives the VC_RDY.

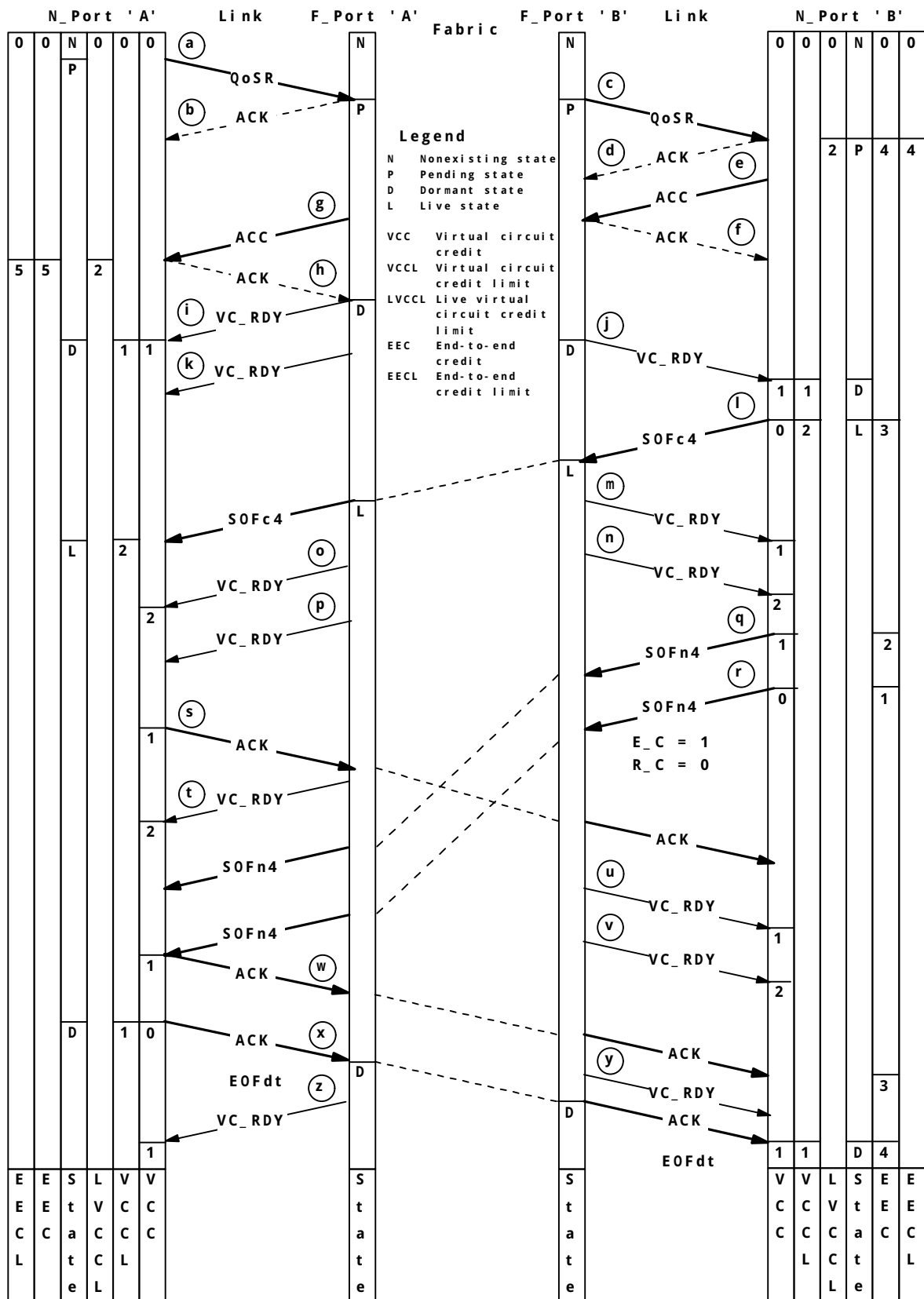


Figure L.1 - Class 4 setup, activation and deactivation

L.2 Activation collision and removal

The following describes in detail the Frame and Class 4 Primitive Signal flow shown in Figure L.2:

- a) N_Port 'A' transmits an activation request (a), consuming one EE_Credit and one VC_Credit, causing N_Port 'A' to enter the Live state from the Dormant state. This also changes the VC_Credit limit to the Live VC credit limit. F_Port 'A' enters the Live state when it receives the SOFc4 delimited activation frame. The frame then traverses the Fabric and is forwarded by F_Port 'B'. However, F_Port 'B' is already in the Live state, due to the reception of an SOFc4 delimited frame (b) transmitted by N_Port 'B'. N_Port 'B' receives the SOFc4 delimited activation frame and later acknowledge it (m).
- b) N_Port 'B' consumes one EE_Credit and one VC_Credit when it transmits the activation frame (b). Transmitting the activation frame causes N_Port 'B' to enter the Live state and increases the VC credit limit. F_Port 'B' enters the Live state when it receives the activation frame. This frame then transverse the Fabric and is forwarded by F_Port 'A' to N_Port 'A'. N_Port 'A' later acknowledge (j) reception of the SOFc4 delimited activation frame. Both F_Port 'A' and N_Port 'A' remain in the Live state.
- c) F_Port 'A' transmits a VC_RDY (c) to replenish VC credit. N_Port 'A' increases VC_Credit when it receives the VC_RDY.
- d) F_Port 'B' transmits a VC_RDY (d) to replenish VC credit. N_Port 'B' increases VC_Credit when it receives the VC_RDY.
- e) F_Port 'A' transmits a VC_RDY (e) to replenish VC credit. N_Port 'A' increases VC_Credit when it receives the VC_RDY.
- f) F_Port 'B' transmits a VC_RDY (f) to replenish VC credit. N_Port 'B' increases VC_Credit when it receives the VC_RDY.
- g) N_Port 'A' transmits its last Data frame (g) requesting removal of the Class 4 circuit, consuming one EE_Credit and one VC_Credit. The frame is received by F_Port 'A', traverses the Fabric, is forwarded by F_Port 'B' and is received by N_Port 'B'. N_Port 'B' later honor the removal request when it acknowledges (r) the Data frame.
- h) N_Port 'B' transmits its last Data frame (h) to the Sequence initiated by (b). This consumes one EE_Credit and one VC_Credit. The frame is received by F_Port 'B', traverses the Fabric, and is forwarded by F_Port 'A' before N_Port 'A' receives it. N_Port 'A' later acknowledge the frame reception (l).
- i) F_Port 'B' transmits a VC_RDY (i) to replenish VC credit. N_Port 'B' increases VC_Credit when it receives the VC_RDY.
- j) N_Port 'A' acknowledge reception of Data frame (b) by transmitting an ACK frame (j), consuming one VC_Credit. The ACK frame is received by F_Port 'A', traverses the Fabric, is forwarded by F_Port 'B' and is received by N_Port 'B'. N_Port 'B' reclaim EE_Credit on receipt of the ACK frame.
- k) F_Port 'A' transmits a VC_RDY (k) to replenish VC credit. N_Port 'A' increases VC_Credit when it receives the VC_RDY.
- l) N_Port 'A' acknowledge reception of Data frame (h) by transmitting an ACK frame (l), consuming one VC_Credit. The ACK frame is received by F_Port 'A', traverses the Fabric, is forwarded by F_Port 'B' and is received by N_Port 'B'. N_Port 'B' reclaim EE_Credit on receipt of the ACK frame.
- m) N_Port 'B' acknowledge reception of Data frame (a) by transmitting an ACK frame (m), consuming one VC_Credit. The ACK frame is received by F_Port 'B', traverses the Fabric,

is forwarded by F_Port 'A' and is received by N_Port 'A'. N_Port 'A' reclaim EE_Credit on receipt of the ACK frame.

- n) F_Port 'A' transmits a VC_RDY (n) to replenish VC credit. N_Port 'A' increases VC_Credit when it receives the VC_RDY.
- o) F_Port 'A' transmits a VC_RDY (o) to replenish VC credit. N_Port 'A' increases VC_Credit when it receives the VC_RDY.
- p) F_Port 'B' transmits a VC_RDY (p) to replenish VC credit. N_Port 'B' increases VC_Credit when it receives the VC_RDY.
- q) F_Port 'B' transmits a VC_RDY (q) to replenish VC credit. However, N_Port 'B' does not increase its VC_Credit when it receives the VC_RDY since VC_Credit is already at the VC credit limit.
- r) N_Port 'B' acknowledge reception of Data frame (g) by transmitting an ACK frame (r), consuming one VC_Credit. N_Port 'B' elects to honor the removal request made by N_Port 'A'. The EOFrt delimited ACK frame is received by F_Port 'B', traverses the Fabric, is forwarded by F_Port 'A' and is received by N_Port 'A'. N_Port 'B' enters the Nonexisting state when it transmits the EOFrt delimiter, F_Port 'B' enters the Nonexisting state when it receives the EOFrt delimiter, F_Port 'A' enters the Nonexisting state when it transmit the EOFrt delimiter and N_Port 'A' enters the Nonexisting state when it receives the EOFrt delimiter. N_Port 'A' do not reclaim EE_Credit on receipt of the ACK frame since the Class 4 virtual connection is now nonexistent. However, N_Port 'A' final Sequence still completes successfully.
- s) F_Port 'A' transmits a VC_RDY (s) to replenish VC credit. However, N_Port 'A' does not increase its VC_Credit when it receives the VC_RDY since VC_Credit is already at the VC credit limit.

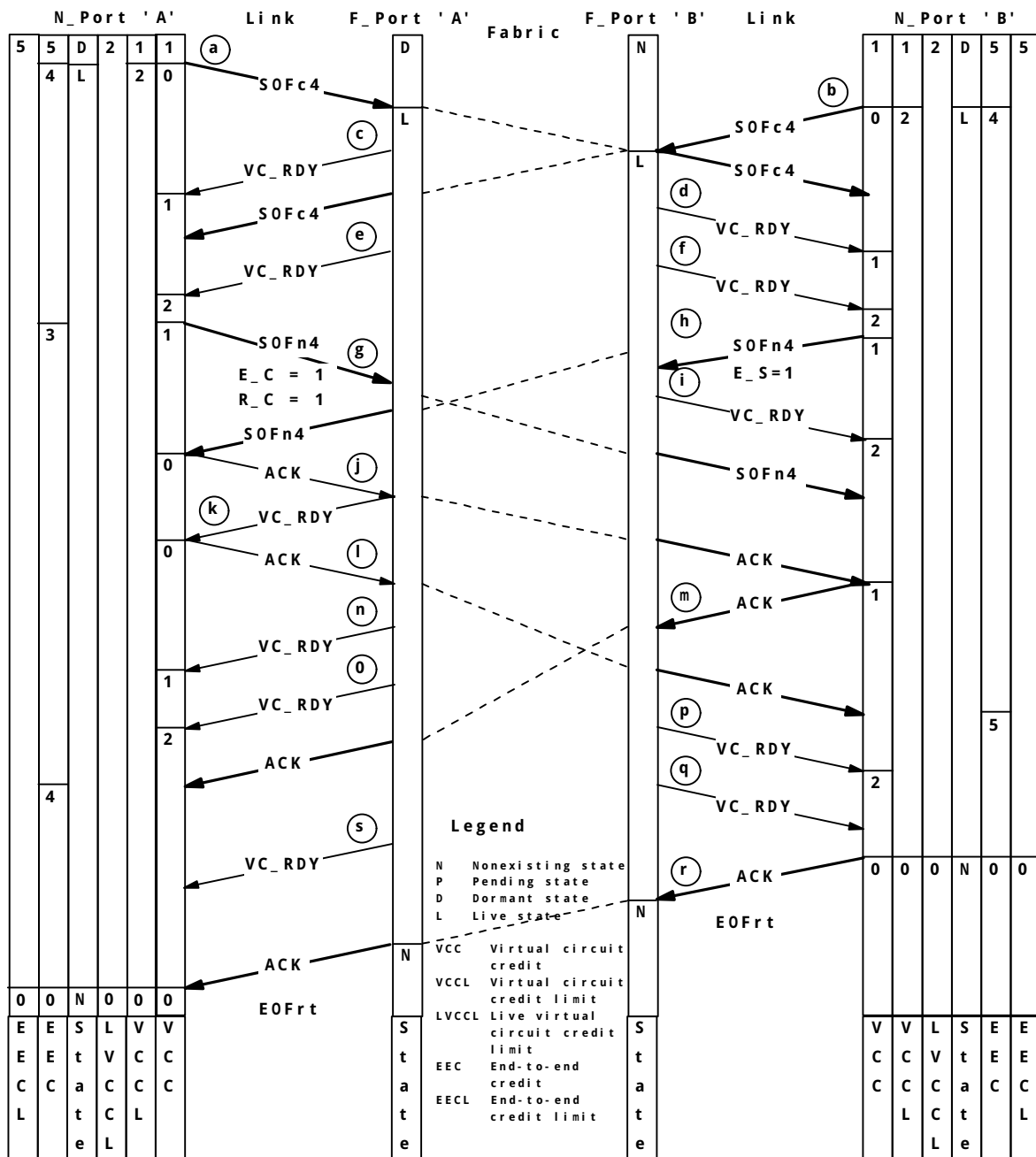


Figure L.2 - Class 4 activation collision and circuit removal

Annex M (Informative) Priority and Preemption

M.1 Overview

Real-time systems require interconnect networks that provide guaranteed bandwidth, guaranteed in-order delivery, and guaranteed latency. To meet these requirements, an interconnect network that resolves access contention using priority and preemption is desired.

A priority value in the Fibre Channel frame header is defined (Word 1 Bits 31-25, see clause 12) and can be used by a Fabric to resolve contention between N_Ports requesting a Dedicated Connection to a single resource N_Port. A 7-bit field is defined since this size enables efficient implementations of the Rate Monotonic Scheduling Algorithm.

Preemption in the context of a Fibre Channel Fabric is the act of establishing a connection to a resource or resources (if multicast is used) that is/are already a part of a dedicated connection. The act of preempting an existing dedicated connection is completed ensuring that complete frames are transmitted and received with link integrity maintained.

M.2 Preemption Process

A preemption is initiated when a preemption request frame (SOFc1 with preemption flag set = 1, defined in clause 12) is forwarded to the Fabric by an unconnected N_Port. The preemption request will result in:

- the preemption of an existing dedicated connection and the establishment of a new dedicated connection between the preemption requesting N_Port (the preemptor) and the desired destination(s) N_Port(s) (the preemption destination(s)),
- the establishment of a dedicated connection between the preemptor the preemption destination(s), or
- rejection of the preemption request by the Fabric. In a real-time application, rejection by the Fabric might be due to a priority value that is too low.

The process for performing a preemption is as follows:

Upon receipt of a request submitted by an FC-4 or other upper level protocol to initiate a connection using preemption, the FC-2 protocol attempts to establish a connection using the SOFc1, with the preemption flag set = 1, preemption request frame. In Figure M.1, the preemptor P is sending a preemption request frame (SOFc1 with preemption flag set = 1) to the Fabric.

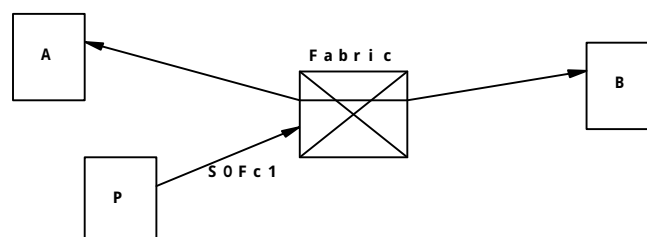


Figure M.1 - Pre-emption Request

If the Fabric denies the request the Fabric returns a F_RJT Link_Response frame with a “pre-empt request rejected” reason code (see clause 14) to the preemptor with no other effects on the Fabric (see Figure M.2). No connections are changed if the Fabric rejects a preemption request.

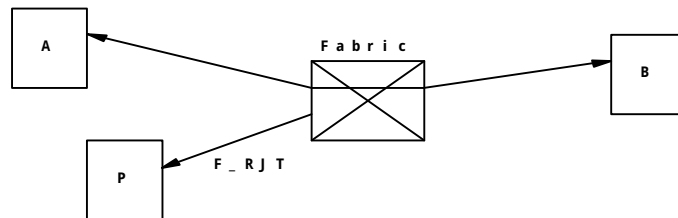


Figure M.2 - Connection Rejection

If the Fabric accepts the request, the Fabric terminates the connection(s) being preempted (A to B) and forwards a PRMT basic link service command (see 15.2.6) to both the preempted connection initiator and recipient (Ports A and B now know why they have abnormally terminated sequences that must be managed.). To facilitate connection termination, the Fabric may initiate a Link Reset (see 10.6.5) to both the preempted and connection initiator and recipient (A and B) after forwarding the PRMT, as shown in Figure M.3.

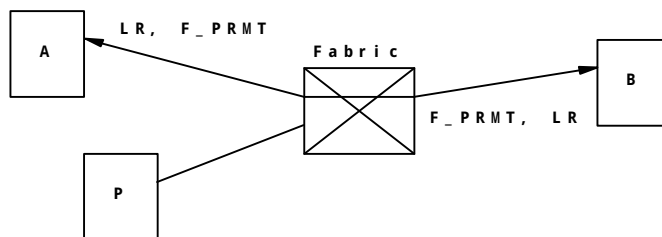


Figure M.3 - Pre-emption Accepted

The Fabric then establishes a new connection (P to B) by forwarding the SOFc1 preemption request frame to the preemption destination (B) as shown in Figure M.4. The new dedicated connection is established when the preemptor (P) receives the acknowledge from the preemption destination N_Port (B).

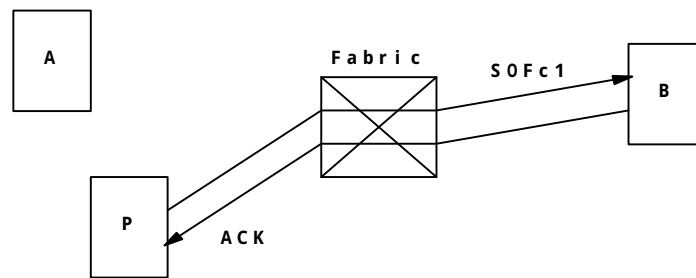


Figure M.4 - Connection Established

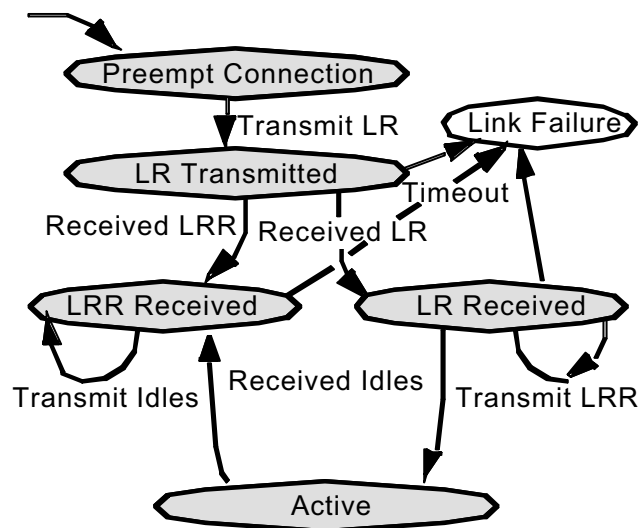
M.3 Link Reset Protocol

The link reset protocol, as defined in 10.6.5 follows the process outlined in the preceding figure when used for preemption. The “Timeout” period is specified as the R_T_TOV value (see 23.2.1.1) for detection of loss of synchronization (see 6.1.2). To guarantee timely progress through the preemption sequence the following delays should be specified by the application layer profile.

- P_T_LR, Preempt Connection state to Transmit LR
- R_T_LRR, Received LR to Transmit LRR
- R_T_ID, Received LRR to Transmit Idles

Reasonable values for these event delays are dependent on the application that is using preemption. A rule of thumb that could be useful might be:

The sum of P_T_LR, R_T_LRR, and R_T_ID should be equal to n times the class 1 connection setup latency. Where the class 1 connection setup latency is the time from when the connection initiator forwarded the SOFc1 to when the connection recipient detects reception of the SOFc1. The value of n is dependent on the application environment. High performance environments will want a small value of n (~1 or 2), other environments may function with a value much larger.

**Figure M.5 - Link Reset Diagram**

Annex N (Informative) Tutorial for IEEE Registered

N.1 Introduction

The IEEE Registration Authority for a fee provides a registered number that is guaranteed to be unique. The unique number may be provided in either of two formats, depending on the requirements of the manufacturer. The number is provided as a 6 hexadecimal number value as the IEEE company_id. The number is provided as three hexadecimal-digit pairs in cononical form representing the 3 octets of the 24-bit number as the IEEE Organizationally Unique Identifier (OUI). A manufacturer for all its products that use an IEEE registration uses the same number. A manufacturer shall base all its identifiers on the same number, even if the identifiers have different formats. A manufacturer shall not purchase a new company_id until at least one of the identifier spaces using the company_id is substantially exhausted. Other identifier spaces shall continue using the original company_id until they are also exhausted.

The IEEE Registration Authority assigned “company_id” is the “IEEE company identifier” defined in 13.3.2. Two formats are defined within FC-FS, while two additional formats are defined in the ANSI X3T11 document (X3T11/96-467) dated December 12, 1996, entitled “New identifier formats based on IEEE registration” (see M.3 for details on how to obtain copies).

In the context of Fibre Channel, a set of unique worldwide identifiers is exchanged as part of the login process. A unique identifier is determined for each communicating node and for the port on each node through which the login is performed. Fibre Channel uses the Name Address Authority (NAA) field to select among several alternative registration authorities and to identify the format of the worldwide identifier. The NAA field, defined in Table 26, occupies the high order 4 bits (bits 60-63) of the unique world-wide identifier, assuring that the identifier is unique even if the 60-bit patterns derived using the registered IEEE company_id happen to be identical. The complete 64-bit or 128-bit pattern, including the NAA bits, shall be unique worldwide.

The FC-FS formats shown in Figure N.1 use the IEEE company identifier as the direct or indirect basis for their unique values.

Table M.1 - FS Identifier Types		
NAA bits 63-60	FC-FS format name	IEEE basis for format
0 0 0 1	IEEE	IEEE 803.2 48-bit ID
0 0 1 0	IEEE extended	IEEE 803.2 48-bit ID
0 1 0 1	IEEE registered name	IEEE company_id
0 1 1 0	IEEE registered extended	IEEE company_id

Caution

The IEEE administers the assignments of 24-bit company_id values. The assignments of these values are public, allowing a user of an identifier to identify the manufacturer that provided the value. The IEEE/RAC has no control over the assignments of the vendor-specified fields and assumes no liability for assignments of duplicate identifiers.

A company shall use the same IEEE company_id for all its IEEE Registration Authority based identifiers until the identifier spaces using that company_id are substantially exhausted.

N.2 Formats of FC-FS identifiers

The first two examples use the IEEE 802.3 48-bit identifier based on the IEEE company_id value. The second two examples use the IEEE company_id value more directly.

An example is provided for each format, showing how all fields are derived.

IEEE 48-bit identifier format

This format is defined in 13.3.2.1. The name represents a historical artifact from the FC document and does not imply that the 48-bit identifier is actually generated by IEEE.

The FC-FS IEEE 48-bit worldwide identifier uses the 48-bit IEEE 802.1 universal LAN MAC address (ULA), which in turn is constructed from the IEEE company_id. This value is typically used to uniquely identify a Fibre Channel node.

The format for the FC-FS IEEE 48-bit identifier is shown in Figure N.1.

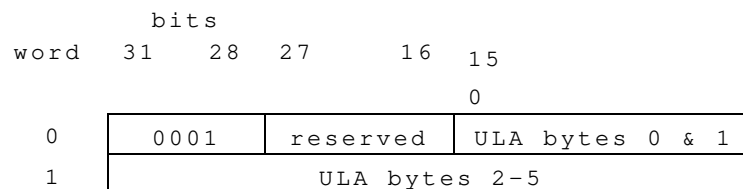


Figure N.1 - FC-FS IEEE 48-bit identifier

ULA Bytes 0, 1, and 2 are generated using the IEEE company_id. ULA Bytes 3, 4, and 5 represent a unique value provided by the identified company.

With this mapping, the portion of the world-wide name that correspond to the special Ethernet address bits (if the ULA was to be used in that context) are as follows:

- Bit 40 is the Individual/Group ID bit. The bit shall be zero when used with the FC-FS IEEE 48-bit address.
- Bit 41 is the Universally Administered / Locally Administered Address bit. The bit shall be zero when used with the FC-FS IEEE 48-bit address.

This format is in common usage in Fibre Channel devices as a node identifier.

Figure N.2 illustrates an example in which the IEEE company_id value used is:

hex 'AC DE 48'

which has a binary representation of:

1010 1100 1101 1110 0100 1000

this value is combined with a value generated by the identified company of hex '00 00 80' to create a ULA of:

hex 'AC DE 48 00 00 80'

The bit order is not changed for the Fibre Channel identifier as it would be if transmitting a LAN address on an Ethernet-based LAN link.

Using this ULA, the following 64-bit Fibre Channel IEEE 48-bit identifier format is created.

hex '10 00 AC DE 48 00 00 80'

which would have a binary representation (in the format of Figure N.1) shown in Figure N.2.

	byte			
word	0	1	2	3
0	0001 0000	0000 0000	1010 1100	1101 1110
1	0100 1000	0000 0000	0000 0000	1000 0000

Figure N.2 - FC-FS IEEE 48-bit identifier example

IEEE Extended identifier format

The IEEE extended identifier format allows the high order 12 bits that are unused in the IEEE 48-bit format to be used as a vendor specified field that extends the vendor specified field contained in the ULA. The identified company shall ensure that the combination of the 12-bit vendor specified field and the 3-byte vendor specified portion of the ULA is unique company-wide. Together with the NAA and IEEE company_id portion of the ULA, this guarantees that the overall FC-FS IEEE extended identifier shall be unique worldwide.

Fibre Channel may use the vendor-specified field to distinguish different ports to a node. The node may have an IEEE 48-bit FC-FS format and each port may be distinguished by an IEEE Extended format that uses the same ULA value as the node, but a unique vendor specified value for each port.

The vendor specified field may also be used by the identified company to extend the IEEE 48-bit ULA to a 60-bit field. The identified company shall manage the 12-bit vendor specified fields and ULA bytes 0 through 5 such that every 64-bit value IEEE extended identifier is unique world wide.

The format is shown in Figure N.3.

	Bits				
word	31	28	27	16	15
					0
0	0010	Vendor Specific		ULA bytes 0 & 1	
1	ULA bytes 2-5				

Figure N.3 - FC-FS IEEE extended identifier

ULA Bytes 0, 1, and 2 are generated using the IEEE company_id. ULA Bytes 3, 4, and 5 represent a unique value provided by the identified company. With this mapping, the portions of the world-wide name that correspond to the special Ethernet address bits (if the ULA was to be used in that context) are as follows:

- Bit 40 is the Individual/Group ID bit. The bit shall be zero when used with the FC-FS IEEE 48-bit address.
- Bit 41 is the Universally Administered / Locally Administered Address bit. The bit shall be zero when used with the FC-FS IEEE 48-bit address.

This format is in common usage in Fibre Channel devices as a port identifier. Some applications also use this format as a node identifier.

Figure N.4 illustrates an example in which the IEEE company_id value used is:

hex 'AC DE 48'

which has a binary representation of

1010 1100 1101 1110 0100 1000

this value is combined with a value generated by the identified company of hex '00 00 80' to create a ULA of:

hex 'AC DE 48 00 00 80'

The bit order is not changed for the Fibre Channel identifier as it would be in transmitting a LAN address on a LAN link.

Using this ULA, the following 64-bit Fibre Channel IEEE extended identifier can be created. In this example, the vendor specified value selected by the identified company is B17 hexadecimal, which has a binary representation of 1011 0001 0111. The resulting Fibre Channel IEEE extended identifier is:

Hex '2B 17 AC DE 48 00 00 80'

which would have a binary representation (in the format of Figure N.3) shown in Figure N.4.

	byte			
word	0	1	2	3
0	0010 1011	0001 0111	1010 1100	1101 1110
1	0100 1000	0000 0000	0000 0000	1000 0000

Figure N.4 - FC-FS IEEE extended identifier example

IEEE Registered identifier format

The IEEE Registered identifier format provides a worldwide unique identifier typically used to identify any Fibre Channel object, including ports, nodes, and fabrics. This value is based directly on the IEEE company_id, avoiding the requirement for the identified company to maintain a 48-bit ULA registry. Instead, the identified company shall maintain a registry of vendor specified identifier values that guarantees that all identifiers are unique worldwide. The regis-

try mechanism should guarantee that the identified company uses substantially all of the identifiers that can be created with a single IEEE company_id before a new IEEE company_id is purchased and used.

When the NAA value indicates that the format is IEEE registered, the identifier shall contain the 24-bit IEEE company_id followed by a 36-bit vendor specified identifier (VSID) which uniquely indicates a Node, a N_Port, a F_Port, a Fabric, or other object.

The format for the IEEE registered identifier is shown in Figure N.5.

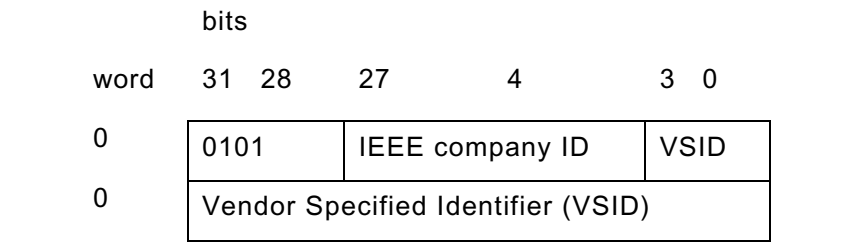


Figure N.5 - FC-FS IEEE registered identifier

This format is a new format specifically designed for identifying Fibre Channel objects, including ports and nodes. This format should be used except where historical usage makes the IEEE 48-bit identifier or the IEEE extended identifier more appropriate.

Figure N.6 illustrates an example in which the IEEE company_id value used is:

hex 'AC DE 48'

which has a binary representation of

1010 1100 1101 1110 0100 1000

In this example, the vendor-specified value selected by the identified company is:

B 17 34 F6 2D (hexadecimal)

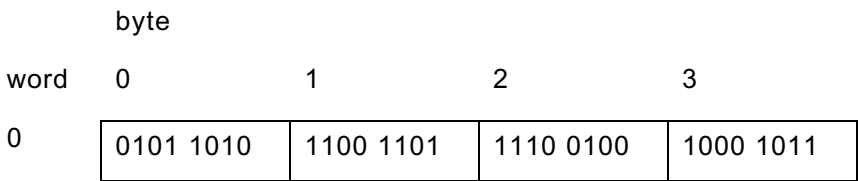
which has a binary representation of

1011 0001 0111 0011 0100 1111 0110 0010 1101

The resulting IEEE registered format is:

Hex '5A CD E4 8B 17 34 F6 2D'

which would have a binary representation (in the format of Figure N.5) shown in Figure N.6.



1	0001 0111	0011 0100	1111 0110	0010 1101
---	-----------	-----------	-----------	-----------

Figure N.6 - FC-FS IEEE registered identifier example

IEEE registered extended identifier format

The IEEE Registered Extended format is defined by the FC-FS documents, but is at present used only for SCSI objects attached by Fibre Channel. For information about the proper use of this format, see the “IEEE Tutorial for SCSI use of IEEE company_id”.

IEEE FC-FS Qualified FC-4 TYPE format

The IEEE FC-FS Qualified FC-4 TYPE format is used to distinguish conflicting uses of a vendor unique FC-4 TYPE code (reference RNFT). This value is based directly on the IEEE company_id, avoiding the requirement for the identified company to maintain a 48-bit ULA registry. Instead, the identified company shall maintain a registry of vendor specified Qualified FC-4 TYPE identifier values that guarantees that all such identifiers are unique worldwide. The registry mechanism should guarantee that the identified company uses substantially all such identifiers that can be created with a single IEEE company_id before a new IEEE company_id is purchased and used.

The IEEE FC-FS Qualified FC-4 TYPE identifier shall contain the 8-bit FC-4 TYPE encoded value followed by a 24-bit IEEE company_id of the company that defined the FC-4 protocol.

N.3 Availability of references

Copies of the following T10 draft documents are available for purchase from Global Engineering Documents. For further information, contact Global Engineering Documents at 800-854-7179 (phone) or 303-792-2181 (phone) or by mail at 15 Inverness Way East, Englewood, CO 80122-5704. The document is also available on the SCSI web page at <http://www.t10.org>.

- New identifier formats based on IEEE registration, Bob Snively, Feb. 24, 1997, X3T11/96-467, revision 2.
- Tutorial for SCSI use of IEEE registered company_id, Bob Snively, X3T10/97-101, revision 2 or higher.

Annex O **(informative)** **Clock Synchronization**

O.1 Introduction

The goal of the Clock Synchronization Service described in clause 32 is to provide each participating node with a continuously-running counter that, at all times, contains exactly the same value that is found in the counter in every other participating node. Clause 32 provides the message definitions and formats required to accomplish this goal in an interoperable way. But the extent to which the value in a given node's counter actually matches the value in any other node's counter is dependent on the techniques used to implement the elements described in clause 32.

For systems with low accuracy requirements, the CSU ELS frames could be handled in software with no special hardware/firmware support. The client software could use any existing timer resources to maintain its local version of the counter. For systems that require the higher levels of accuracy, dedicated hardware assistance would be needed.

It is the purpose of this annex to present several possible hardware implementations and to discuss the sources of error in each of them.

Clause 32 defines two separate mechanisms for transfer of the synchronizing information -- the Extended Link Service (ELS) method and the Primitive Signal method. This annex addresses only the ELS method.

O.2 Discussion

The approach used will be to first present a basic model of an NL_Port, in order to give a context for the rest of the discussion. Then basic hardware-based implementations for each topology (point-to-point, fabric, and loop) will be presented, followed by a discussion of the various sources of error. Finally, approaches for reducing these errors will be given.

O.2.1 A Model of an NL_Port

Figure O.1 presents a model of a generic NL_Port that will be used as the basis for the discussions in this annex. The elasticity buffer in the receive path and the multiplexer in the transmit path exist to support the operation of the port in an Arbitrated Loop topology. The remainder of the components supports all topologies. For purposes of this annex, the interaction of the host with the port logic occurs entirely through data structures in the port's Memory, which the host accesses via the Host Bus.

As transmission words are received, they pass through a deserializer. Decoder (Des/Decode), and are checked for validity and for the various types of frame delimiters (CRC/Valid). Valid frames destined for the local node are pushed onto the Rec FIFO. From the Rec FIFO, frames are stored as data structures in the Memory. The host is informed of the presence of incoming frames via an unspecified mechanism, and the data is then transferred to the host via the host bus.

For outgoing data, the host and the port cooperate (in an unspecified manner) to cause the outgoing frames to be placed into the port's Memory. From there, the frames are transferred into the Trans FIFO. The frames are sent through the CRC logic, the multiplexer, and the encoder/serializer logic and onto the link. The CRC logic calculates the CRC value, which is placed in the outgoing frame at the appropriate location.

For Arbitrated Loop operation, a port that is in neither the OPEN nor the OPENED state, incoming transmission words are sent directly from the Elasticity buffer to the multiplexer and out onto the link via the encoder/serializer logic.

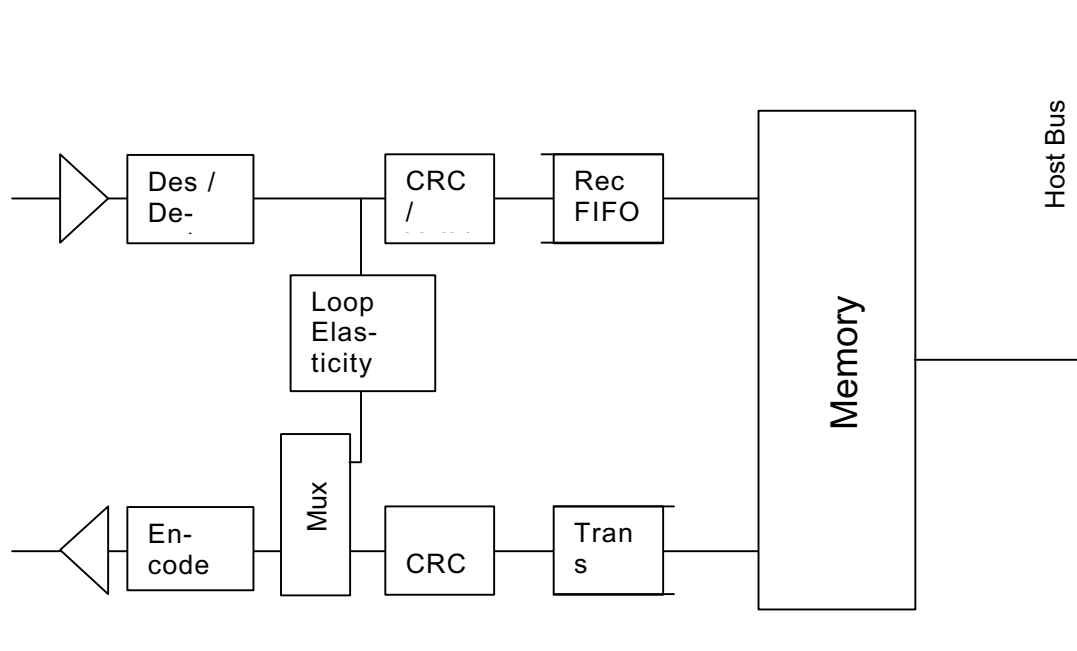


Figure O.1 - Generic NL_Port

O.2.2 Hardware-Assisted Clock Synchronization

Figure O.2 shows the location of the Clock Sync circuitry that supports the Server. Figure O.3 shows the location of the corresponding circuitry that supports the Client.

For the Server, the Host Bus connection allows the loading of an initial value into the master clock. The Server periodically sends the master clock value to the clients in a CSU ELS frame. A multiplexer at the input to the CRC logic allows the CSU frame to bypass the Transmit FIFO, thereby eliminating unnecessary delays caused by other traffic.

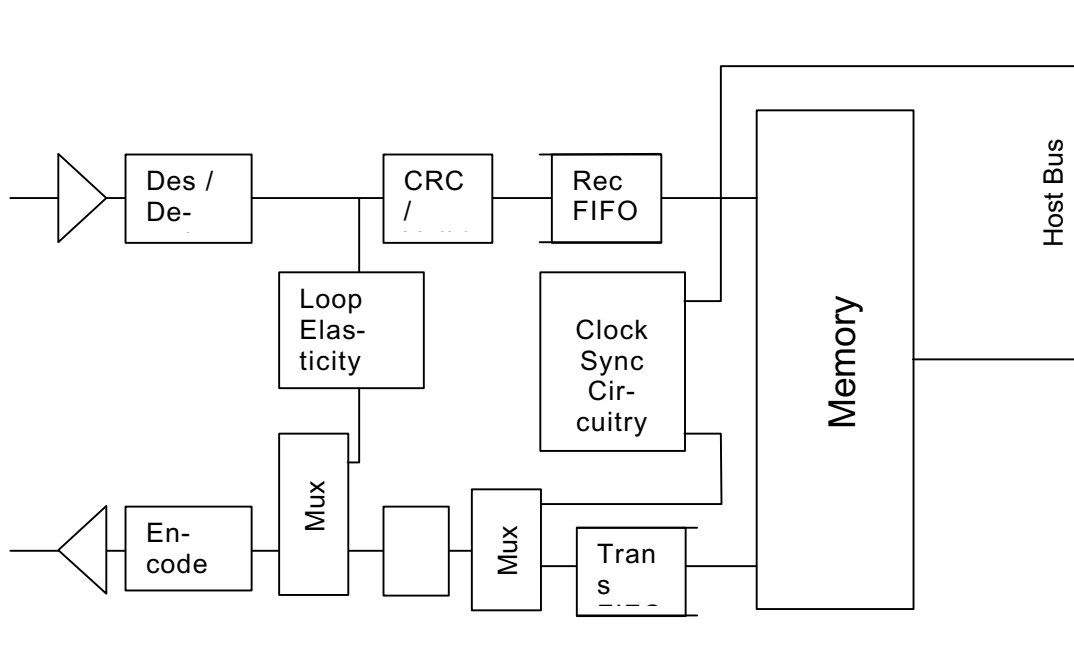


Figure O.2 - Server NL_Port Clock Sync Context

For the Client (see Figure O.3), the Clock Sync circuitry receives the CSU ELS frame prior to the Receive FIFO, thereby eliminating unnecessary delays caused by other traffic. The Host Bus connection allows application software to access the clock sync value. Note that for highest accuracy in applying time tags, the clock synch value should be accessed directly by hardware, i.e., without software intervention.

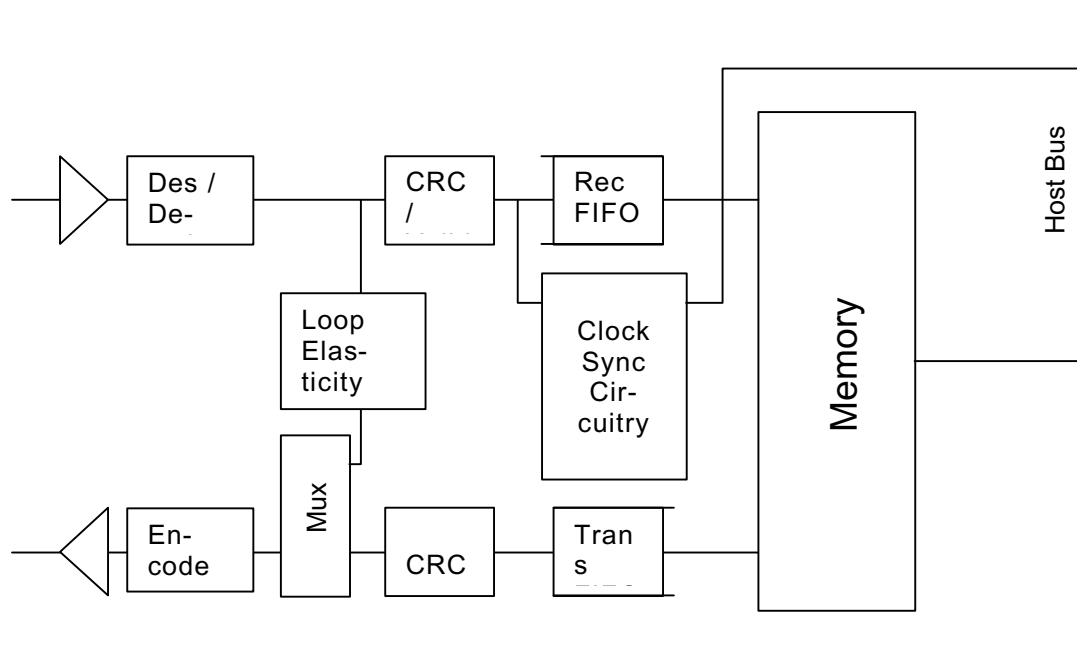


Figure O.3 - Client NL_Port Clock Sync Context

Figure O.4 and Figure O.5 show an implementation of the Clock Sync logic for the Server and the Client, respectively. These represent a very basic implementation.

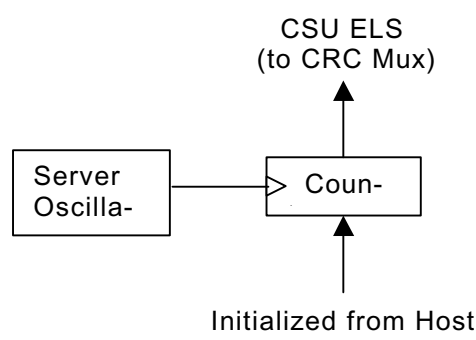


Figure O.4 - Server Clock Sync Implementation (Basic Approach)

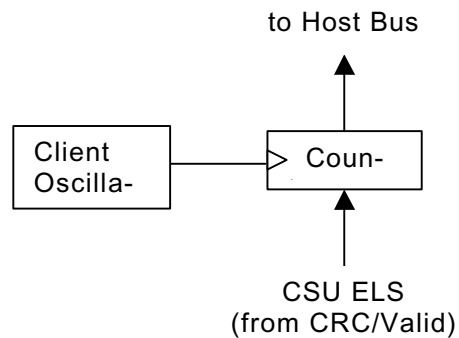


Figure O.5 - Client Clock Sync Implementation (Basic Approach)

O.2.3 A Point-to-Point System

Although a simple point-to-point topology may not be of great practical interest, it will be discussed first because it simplifies the discussion of the errors involved. All of the errors discussed in this section are applicable to all topologies. For reference in the following discussions, Figure O.6 shows the Clock Synchronization model from 32.2 with the Fabric removed.

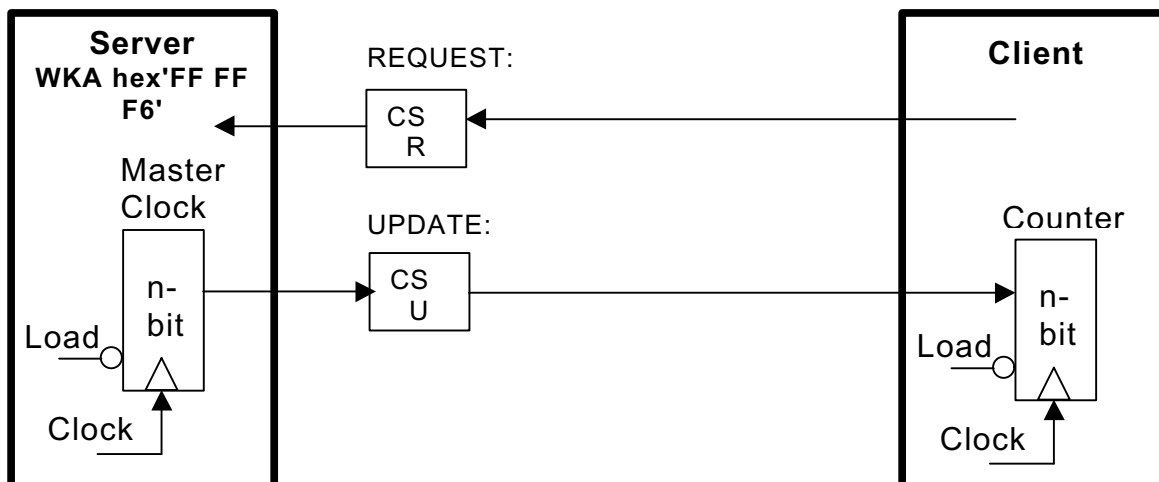


Figure O.6 - ELS Clock Sync Model - Point-to-Point

O.2.3.1 Discussion of Errors

It should be noted that there will generally be a deterministic component of each of these errors, as well as a non-deterministic component. If extremely accurate clock information is needed, a system designer can generally measure or calculate the deterministic components of the errors and adjust the observed clock value to account for them. But the non-deterministic component is, by its nature, not subject to adjustment in the same way by the system designer.

O.2.3.1.1 Client Oscillator Frequency Error

Even though the counters in the server and the client nominally count at the same frequency, the oscillators that drive them are independently subject to the tolerances specified in the Fibre Channel standard. So even if they were to contain the exact same value at some point in time (e.g., just after receipt of the CSU ELS at the client), the values would slowly drift apart as time passes, until the next CSU ELS arrives. Figure O.7 illustrates this effect. The client oscillator is assumed to be of slightly higher frequency than that of the server. Near the center of the figure, it is assumed that another CSU ELS is received at the client. This results in the value of the client clock being corrected so that it again matches the server clock.

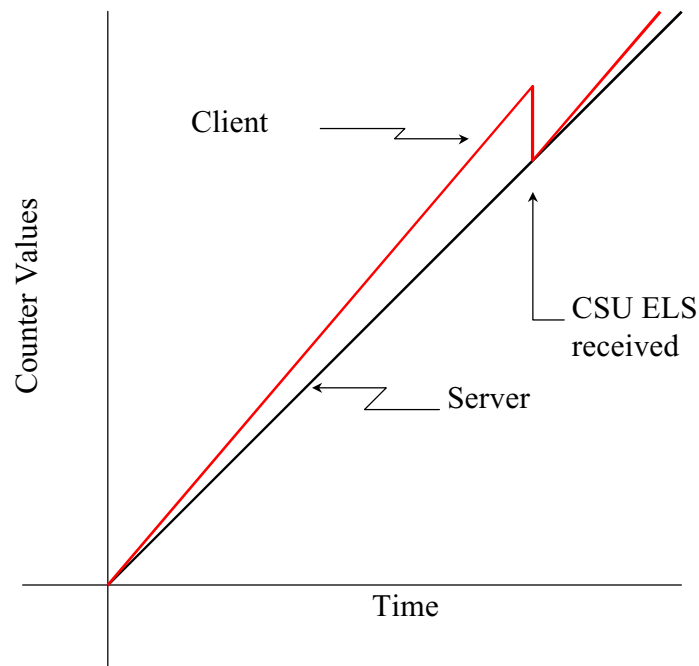


Figure O.7 - Client Clock Drift

The correction of the client's clock when it receives the CSU ELS limits the maximum error as seen by the user of the client's clock. However, it may also result in that user seeing time appear to run backwards. (Reading the clock just prior to receipt of the CSU ELS will return a value that is larger than the value returned if the clock is read just after receipt of the CSU ELS.) This non-monotonic behavior may cause difficulties with some algorithms that are intended to interpret these values.

The error due to oscillator frequency differences is essentially deterministic. A given client can determine the degree to which its oscillator frequency exceeds (or falls behind) that of the server by observing the time between receipt of CSU ELS frames, and the degree to which its clock value exceeds (or lags) the value in the received CSU ELS. This error can then be largely compensated for, either by hardware or by software algorithms.

Analysis:

Symbol	Definition
T_CSU	The period of the CSU ELS frames. I.e., the time between successive CSU frames.
f_server	The frequency of the oscillator in the Clock Synchronization server.
f_client	The frequency of the oscillator in the Clock Synchronization client.
f_tol	The allowed tolerance of the Fibre Channel transmission frequency.
f_nom	The nominal frequency of Fibre Channel transmission.
freq_error	The maximum client clock error due to mis-match of client vs. server oscillator frequencies.

The maximum error will occur just prior to the receipt of a CSU ELS frame. Specifically,

$$\text{freq_error} = T_CSU * [f_client / f_server] - T_CSU$$

The worst mis-match will occur when one oscillator is at the fast end of the allowable range, and the other is at the slow end. So assume that:

$$f_client = f_nom * (1 + f_tol), \text{ and}$$

$$f_server = f_nom * (1 - f_tol)$$

Then, since $f_tol = \pm 100 \text{ ppm}$,

$$\text{freq_error} \sim T_CSU * (2 * 10^{-4})$$

Example:

T_CSU	freq_error
100 m	20 μs
1 s	200 μs

O.2.3.1.2 Link Propagation Delay Error

In the preceding discussion, it was assumed that the CSU ELS that was sent from the server was received instantaneously at the client. In general this is not exactly true, since the frame will need to traverse the link that connects the two nodes. Since the value in the CSU ELS is not updated as it travels down the link, the value received by the client will represent the value of the server's clock at some time in the past. For a given system, with fixed cable lengths, this error, too, is deterministic. For many systems of interest, the error will be negligible. If it is not, its magnitude can be determined by the system designer and can be compensated for. This assumes, of course, that the cable lengths are known and fixed.

Analysis:

Symbol	Definition
link_delay_error	The error caused by the fact that the Clock Count value in the CSU ELS frame does not update as the frame travels down the cable from the transmitter to the receiver.

The magnitude of this error depends on the properties of the specific cable involved. Nominal estimates of delay are:

Electrical cables: 3.5 ns / meter

Optical cables: 5 ns / meter

Example:

For a 33-meter electrical cable:

link_delay_error ~ 33 m * 3.5 ns / m, or

link_delay_error ~ 116 ns

For a 10-Km optical cable,

link_delay_error ~ 10 Km * 5 ns / m, or

link_delay_error ~ 50 us

O.2.3.1.3 Unload Error

Another assumption that was made in the preceding discussions was that the value in the CSU ELS exactly represented the content of the server's counter at the time the most significant bit of that value was placed on the wire (see 32.2.4.4). If a given implementation of a server fails to achieve this, the result will be observed by the client as an error. Depending on the design, this error may contain both deterministic and non-deterministic errors. Non-deterministic errors may result, for example, if the design is such that the CSU ELS frame is placed into a FIFO behind other frames. Since it cannot be known ahead of time what, if any, other frames are ahead of the CSU ELS in the FIFO, the errors will appear to be non-deterministic. Deterministic errors could result from a failure of the design to account for transmission delays from the time the value is taken from the counter until it actually appears on the wire.

It is possible to deal with the deterministic portion of unload error by simply defining it to not exist in a particular system. Note that the server's deterministic unload error affects all client clocks by the same amount. If all references to time in the system are made through client clocks (i.e., if no reference is made directly to the clock in the server), then one could simply define the objective standard to be the server's counter value plus the server's unload error as defined above. By this definition, there is no remaining deterministic unload error at the system level. One should still be conscious of the non-deterministic portion of the error, which could be much larger than the deterministic portion.

Analysis:

Symbol	Definition
t_full_frame	Time to transmit a maximum-size Fibre Channel frame at full-speed Fibre Channel rate, including SOF, EOF, CRC, inter-frame gap, and payload.

Symbol	Definition
unload_error_D	The deterministic portion of the error caused by delays in the Clock Synchronization server logic between the time the counter value is read and the time the most significant bit of the clock count value in the CSU ELS frame is placed on the link.
unload_error_ND	The non-deterministic portion of the error caused by delays in the Clock Synchronization server logic between the time the counter value is read and the time the most significant bit of the clock count value in the CSU ELS frame is placed on the link.

There is very little useful analysis that can be done regarding the unload error outside the context of a specific design. It can be noted, however, that the non-deterministic component of the error has the potential to be very large if it is not addressed in the design of the server's logic. The CSU ELS frame might be queued up in the server's Transmit FIFO behind some number of maximum-length frames. If the other end of the link has no buffer space to receive frames (BB_Credit = 0), then additional delays may occur beyond that needed to transmit the frames ahead of the CSU ELS.

Example:

Without justification, assume that unload_error_D is equivalent to the transmission time of 5 full-speed Fibre Channel Transmission Words. Then

$$\text{unload_error_D} = 5 * 37.65 \text{ ns} = 188 \text{ ns}$$

Regarding the non-deterministic portion of the unload error, assume that the CSU frame does not bypass the Transmit FIFO. Also assume that the FIFO can hold up to four full-size Fibre Channel frames; and that the design of the server does not ensure the FIFO is empty when the CSU ELS frame is pushed onto the FIFO. Assume, however, that BB_Credit > 0 so that no additional wait occurs beyond the time to transmit the frames in the FIFO. Then since

$$\begin{aligned} t_{\text{full_frame}} &= ((15 + (2112 / 4)) \text{ Transmission Words}) * (37.65 \text{ ns} / \text{Transmission Word}), \text{ or} \\ t_{\text{full_frame}} &\sim 20 \mu\text{s} \end{aligned}$$

Then

$$\begin{aligned} \text{unload_error_ND} &= t_{\text{full_frame}} * 3, \text{ or} \\ \text{unload_error_ND} &\sim 60 \mu\text{s} \end{aligned}$$

O.2.3.1.4 Load Error

The link propagation delay error was discussed previously. That error dealt with the fact that the CSU ELS clock value was not updated as the ELS made its way from the server's transmitter to the client's receiver. But the client's clock synchronization counter is separated from its receiver by some amount of logic, the details of which depend on the specific design of the client. The time from the arrival of the CSU ELS at the client's receiver until the client's counter is updated will be perceived by the client as an error. Similarly to the unload error discussed above, the load error may contain both deterministic and non-deterministic components.

Analysis:

Symbol	Definition
<code>t_full_frame</code>	Time to DMA a full Fibre Channel frame into host memory.
<code>load_error_D</code>	The deterministic portion of the error caused by delays in the Clock Synchronization client logic between the time the most significant bit of the clock count value in the CSU ELS frame is received from the link and the time that value is actually loaded into the client's counter.
<code>load_error_ND</code>	The non-deterministic portion of the error caused by delays in the Clock Synchronization client logic between the time the most significant bit of the clock count value in the CSU ELS frame is received from the link and the time that value is actually loaded into the client's counter.

There is little useful analysis that can be done regarding the unload error outside the context of a specific design. Figure O.3 indicated that the CSU ELS frame went directly from the CRC/Validation logic into the client's clock synchronization circuitry. If instead, the frame is allowed to languish in the Receive FIFO, the non-deterministic portion of load error could become fairly large.

Example:

Without justification, assume the deterministic portion of load error is on the order of the time to transmit 6 full-speed Fibre Channel Transmission Words. Then

$$\text{load_error_D} = 6 * 37.65 \text{ ns} \sim 226 \text{ ns}$$

Regarding the non-deterministic portion of the load error, assume that the CSU frame does not bypass the Receive FIFO. Also assume that the FIFO can hold up to four full-size Fibre Channel frames. Arbitrarily assume that the design of the client logic is such that it can empty the FIFO exactly as fast as the link can fill it. Then by assumption,

$$\text{t_full_frame} \sim 20 \text{ } \mu\text{s}$$

Then

$$\text{load_error_ND} = \text{t_full_frame} * 3, \text{ or}$$

$$\text{load_error_ND} \sim 60 \text{ } \mu\text{s}$$

O.2.3.1.5 R/T Clock Domain Error

As defined above, the R/T clock domain error is actually a component of the load error. It is dealt with separately because of its unique nature. It is a non-deterministic error that arises from the (assumed) fact that not all of the logic in the client's port operates from the same clock oscillator. It is assumed that most of the logic is operated from the same oscillator that drives the transmitter. But there is a small amount that is operated from the clock recovered from the received serial bit stream. Specifically, the deserialize/decode logic and the front end of the elasticity buffer of Figure O.3 are assumed to operate from the recovered clock. Passing information from one clock domain to another requires re-synchronizing to the receiving domain's clock. Standard methods for accomplishing this generally result in a delay of 1-2 cycles of the receiving domain's clock. This difference (i.e., zero to one cycle of the receiving domain's clock) is non-deterministic. The R/T Clock Domain Error can be treated as a deter-

ministic delay of one-and-a-half clock cycles, and a non-deterministic value of +/- one-half of a clock cycle.

Analysis:

Symbol	Definition
logic_clk_period	The period of the clock that drives the logic in which the client's clock synchronization counter resides.
clk_domain_error_D	The deterministic portion of the client clock error due to crossing between the receiver clock domain and the clock domain in which the client's clock synchronization counter resides.
clk_domain_error_N D	The non-deterministic portion of the client clock error due to crossing between the receiver clock domain and the clock domain in which the client's clock synchronization counter resides.

Using the assumptions stated in the preceding discussion,

$\text{clk_domain_error_D} = 1.5 * \text{logic_clk_period}$, and

$\text{clk_domain_error_ND} = 0.5 * \text{logic_clk_period}$

Example:

Assume that the logic_clk_period is the same as the time to transmit one Fibre Channel Transmission Word. i.e.,

Assume $\text{logic_clk_period} = (40 \text{ bits} / (1.0625 * 10^9 \text{ bits/sec})) = 37.56 \text{ ns}$. Then

$\text{clk_domain_error_D} = 56 \text{ ns}$, and

$\text{clk_domain_error_ND} = 19 \text{ ns}$

O.2.3.1.6 Server Oscillator Error

An assumption in the preceding discussions is that the server's oscillator frequency is correct by definition. Recall that the stated goal of the clock synchronization service is to faithfully reproduce at each client node, an exact copy of the server's counter, which is counting cycles of the server's oscillator. If the goal is, instead, to provide each client with a value that represents some other, independent clock value, then the extent to which the server's oscillator fails to match the update rate of that other clock will be seen as another error. A discussion of this error is outside the scope of this annex, and will not be addressed further.

O.2.3.2 Techniques for Reducing Deterministic Errors

O.2.3.2.1 A Fix for Differences in Oscillator Frequencies

Shown in Figure O.8 is a model of logic that could be used in place of Figure O.5 to correct for errors due to the difference in the frequency of the client's oscillator as compared to that of the server. This figure is intended as a model only, not as a specific implementation. For example, multipliers and dividers can take up a considerable amount of logic, and can be replaced by an appropriate series of adds; or by some techniques such as skipping counts (if the client's oscillator is too fast), or inserting counts (if the client's oscillator is too slow).

In the figure, it is assumed that the counter is simply reset to zero upon receipt of the CSU ELS frame, rather than being loaded with the value in the CSU ELS. At that same time, the value from the CSU ELS frame is captured in the ELS_value_n register, the previous value from the ELS_value_n register is captured in the ELS_value_{n-1} register, and the value in the counter just prior to its being reset is captured in the ELS_arrived_{n-1} register. These values are then used as shown in the figure to calculate the client's local clock value.

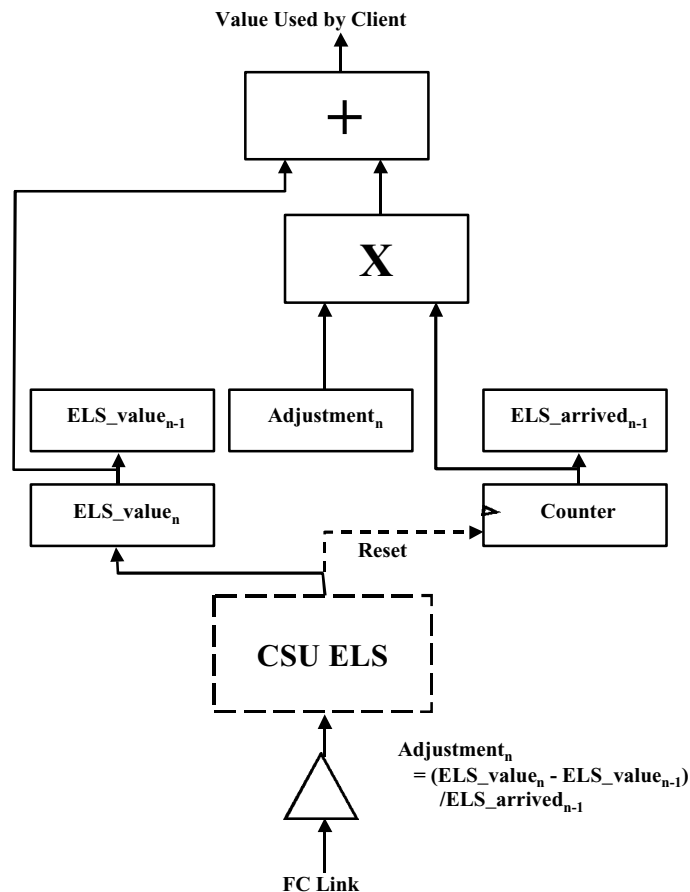


Figure O.8 - Client Clock Sync Logic Model (Rate Adjusted)

Figure O.9 shows a minimal set of hardware assists needed to implement the model of Figure O.8. Upon the receipt of the CSU ELS, host software would read the ELS_value_n and ELS_arrived_{n-1} registers. Since the ELS_value_{n-1} register is nothing more than the old value of the ELS_value_n register, host software would maintain this value internally. The calculation of the Adjustment_n factor and the corrected "valued used by client" would be calculated by host software using the algorithms indicated in Figure O.8.

The Raw Time Tag tuple from the counter / ELS_value_n register is shown in the figure as being available directly, and not going through the host bus interface. This is to emphasize the problem in allowing software delays to corrupt the attachment of the time tag value to data sets. The implication of the figure is that the Raw Time Tag value would be available directly to some hardware that could attach the value directly to the data set with minimal delay. A simple addition of the counter value to the ELS_value_n value would result in an unadjusted,

non-monotonic time value as shown in Figure O.7. But for more accurate results, host software could apply the adjustment factor from Figure O.8.

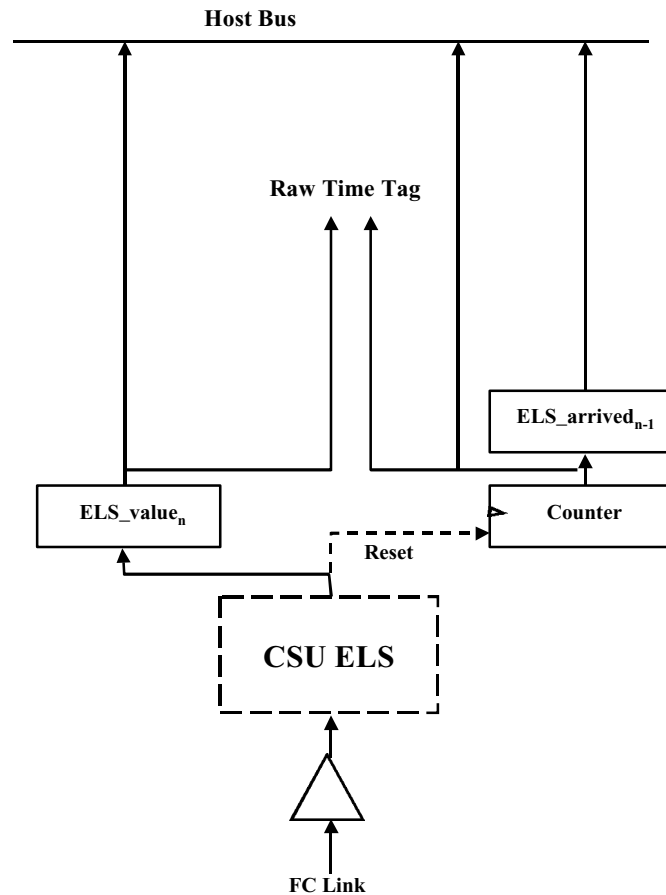


Figure O.9 - Rate Adjustment Hardware Assists for Client Clock Sync

Moving the calculation of the adjustment factor to software has additional benefits. The model of Figure O.8 implicitly assumes that the only error involved is that due to differences in the oscillator frequencies of the server and its clients. In a real implementation, of course, all of the sources of error will contribute to the total error. The host software algorithms can apply filtering algorithms to the data in addition to simply calculating the adjustment factor. This will result in better estimates of the true value of the clock.

O.2.3.2.2 A Fix for Link Propagation Delay Error

Simply adding it to the value received in the CSU ELS frame can compensate for the deterministic portion of the link propagation delay error. (See Figure O.10).

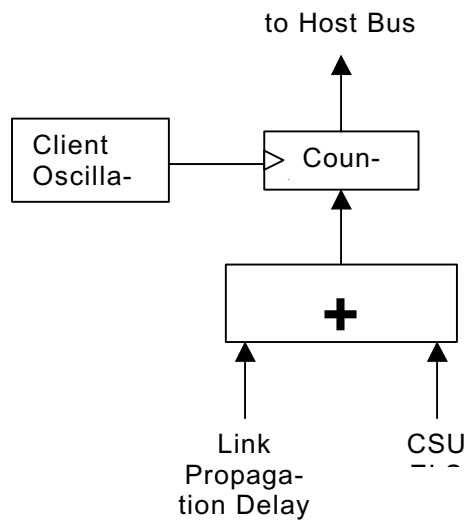


Figure O.10 - Client Clock Sync Implementation (Link Delay Fix)

O.2.3.2.3 A Fix for Load Error

The fix for link propagation delay error can also be used to correct the deterministic portion of the load error by simply replacing the link propagation delay error in Figure O.10 by the load error. Of course, both errors can be corrected simultaneously by simply adding them together before applying them to the adder.

O.2.3.2.4 A Fix for Unload Error

O.2.3.1.3 indicated that it was possible under some conditions to define the deterministic portion of the unload error into non-existence. If this is not possible or desirable for some system, another approach for correcting it is shown in Figure O.11. It will be noted that if this fix is combined with that of O.2.3.2.2. (the fix for link propagation delay error), one will have two adders in series. While it would be possible to combine the two adders by combining the Load Error of the client with the Unload Error of the server, this is not recommended. Doing so would violate the concept of information hiding and would also violate at least the spirit of the standard, since the standard requires that the value in the CSU ELS correctly represent the time in the server's clock as the CSU ELS exits the server port.

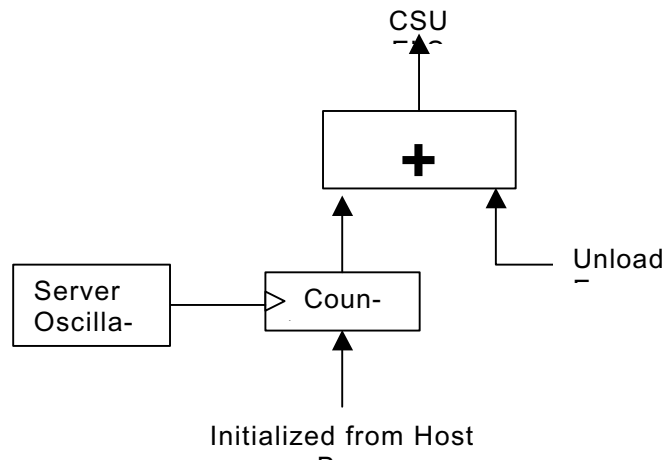


Figure O.11 - Server Clock Sync Implementation (Unload Error Fix)

O.2.3.3 Dealing With Non-Deterministic Error

On the server side, the fix for the non-deterministic component of unload error is to eliminate as many sources of non-deterministic delay as possible. Some design elements to consider include the following:

- Transmit FIFO control. Assuming that the CSU ELS frame is entered into the Transmit FIFO of Figure O.2, ensure that the FIFO is empty at the time the Clock Count value is pushed onto the FIFO.
- BB_Credit. Before the CSU ELS frame is entered into the Transmit FIFO, ensure that BB_Credit is greater than zero. This will ensure that the frame can be transmitted onto the link without delay.

On the client side, some design elements to consider include the following:

- Special CSU frame handling. The CSU ELS frame has a unique R_CTL Information Category value. This may be of use in quickly recognizing the incoming CSU frame so that it can be given special handling, such as bypassing the normal Receive FIFO.

On either the server or the client side, some design elements to consider include the following:

- Priority. One could use high priority for minimizing delays in processing the CSU ELS frame.

O.2.3.4 Dealing With Non-Monotonicity

As discussed in O.2.3.1, if the client oscillator frequency error is not corrected, the client's counter may be set to an earlier time value when the CSU ELS is received. If the proposed fix for this error source is not implemented, it may still be desirable to have a monotonically increasing client clock value in order to avoid difficulties with some algorithms that use that value. If the client's oscillator is slower than that of the server, non-monotonicity does not occur -- the value of the client's counter jumps when the CSU ELS is received, but the jump is in the positive direction. So the problem only occurs when the client's oscillator is faster than that of the server. In this case, when the CSU ELS is received, rather than simply loading the

CSU ELS value into the counter as was done in Figure O.5 and continuing to count from there, one could stop the counter for a number of clock cycles. The number of cycles to stop could be calculated as the difference between the client counter value at the time the CSU ELS is received, and the value in the CSU ELS. The result of this would be as shown Figure O.12.

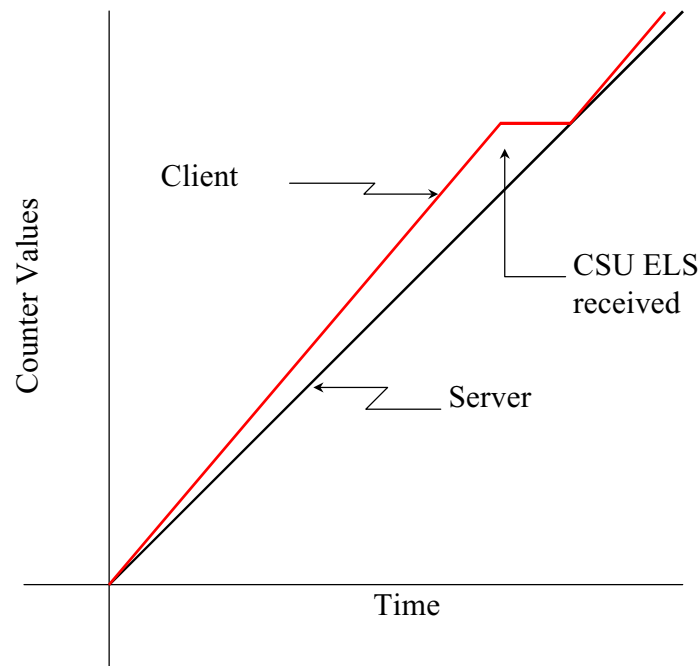


Figure O.12 - Client Clock Drift (Monotonic)

O.2.4 Fabric Considerations

For reference, Figure O.13 reproduces the model from 32.2 for the Clock Synchronization Service in a Fabric-based system. This is the basis for the discussions in the sub-clauses that follow. In order to illustrate more of the possible sources of error, the discussions will assume that the Clock Sync Server is implemented in a separate node outside of any switch element in the Fabric. It should be noted, however, that implementing this server inside of the Fabric might allow for eliminating some of these errors altogether. For simplicity, a single-switch Fabric will be assumed in all of the examples.

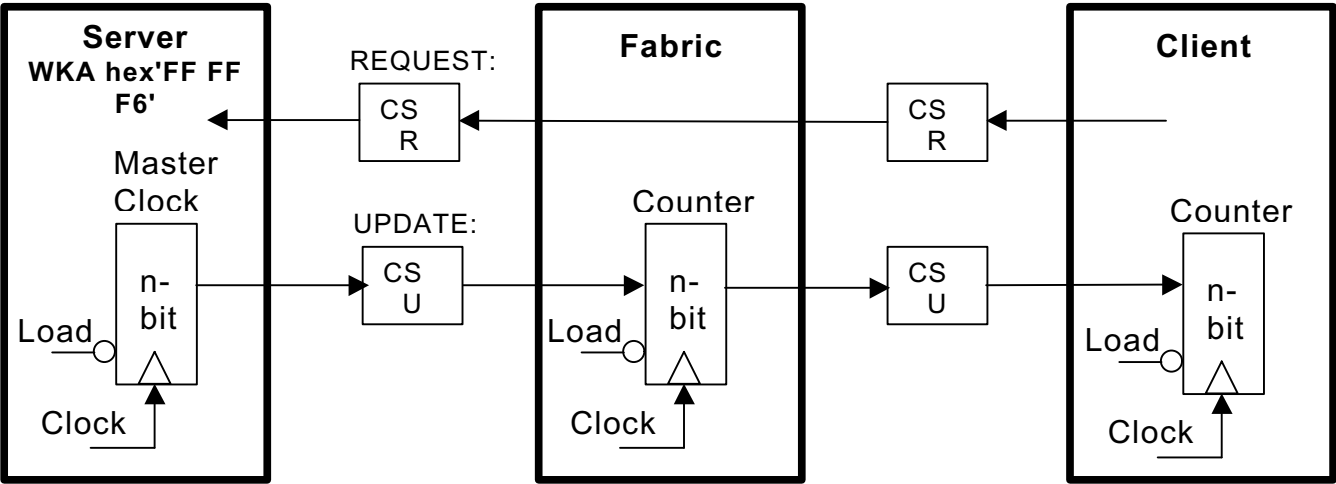


Figure O.13 - ELS Clock Sync Model – Fabric

The insertion of the Fabric between the server and the client results in additional errors being introduced into the client's counter. In terms of error analysis, the Fabric acts as a client of the server, and as a server to the ultimate client.

O.2.4.1 Discussion of Errors

The general nature of these errors is the same as discussed in O.2.3.1. Here, we will discuss only the differences between the point-to-point case and the fabric case.

O.2.4.1.1 Client Oscillator Frequency Error

In the Fabric case, there are at least two oscillators that may introduce errors -- the one in the ultimate client, and the one in the Fabric, in its role as a client. For the best results, both errors should be considered. The design of the specific Fabric in question should be analyzed to determine the exact number of oscillators in the Fabric that need to be considered.

However, it should be noted that the Fabric oscillator(s) will only affect the end result for the period of time between the arrival of the CSU ELS at the Fabric (from the original server), and the time the Fabric sends the CSU ELS to the ultimate client. In a well-designed system, this time will be very small, and the resulting error can probably be ignored.

Analysis:

Symbol	Definition
T_CSU_Fabric	The time between receipt of a CSU ELS frame by the Fabric and the time it transmits the CSU ELS frame to the ultimate client.
f_server	The frequency of the oscillator in the Clock Synchronization server.
f_fabric	The frequency of the oscillator in the Fabric.

Symbol	Definition
f_{tol}	The allowed tolerance of the Fibre Channel transmission frequency.
f_{nom}	The nominal frequency of Fibre Channel transmission.
$freq_error_fabr$ ic	The maximum client clock error due to mis-match of fabric vs. server oscillator frequencies.

The error will accumulate during the time the CSU ELS frame is resident in the Fabric. This error will be in addition to the similar error that occurs at the client for the time between CSU ELS frames. Specifically,

$$freq_error_fabric = T_CSU_Fabric * [f_fabric / f_server] - T_CSU_Fabric$$

The worst mis-match will occur when one oscillator is at the fast end of the allowable range, and the other is at the slow end. So assume that:

$$f_fabric = f_nom * (1 + f_tol), \text{ and}$$

$$f_server = f_nom * (1 - f_tol)$$

Then, since $f_{tol} = 100$ ppm,

$$freq_error_fabric \sim T_CSU_Fabric * (2 * 10^{-4})$$

Another way to look at this is that the worst case total error due to both Fabric and Client oscillator frequency differences (as compared to the Server) will be:

$$freq_error_total = freq_error + freq_error_fabric, \text{ or}$$

$$freq_error_total = (T_CSU + T_CSU_Fabric) * (2 * 10^{-4})$$

Example:

T_CSU_Fabric	$freq_error_fabric$
80 μs	16 ns
320 μs	64 ns

Note that even with these rather large values of T_CSU_Fabric this component is quite small compared to T_CSU that was calculated in O.2.3.1.1 and can therefore be ignored.

O.2.4.1.2 Link Propagation Delay Error

In the case of the Fabric, there are two links that contribute to the error -- one from the original server to the Fabric, and one from the Fabric to the ultimate client. These errors can be expected to be commensurate with each other.

O.2.4.1.3 Unload Error

There are two sources of unload error -- one in the original server, and one in the Fabric as it acts as a server for the ultimate client. These errors can be expected to be commensurate with each other.

Regarding the defining away of the deterministic portion of unload error, a caution is in order. The unload error associated with the server itself can still be defined away as before. The unload error associated with the Fabric can only be defined away if it is known that the path from the server to each client goes through the same number of Fabric elements; and that the Fab-

ric elements all have identical unload errors. If this is true, though, the unload error of the Fabric can be treated the same as that of the server.

The considerations of O.2.3.2.3 can be applied to lessen the non-deterministic portion of unload error in the Fabric.

Analysis:

The presence of the Fabric has two potential effects. First, and most obviously, the circuitry in the Fabric that maintains the counter and that acts as the surrogate server for the client, has its own unload error. This error simply adds to the unload error of the original Server. Secondly, contention in the Fabric can affect the unload error of the original Server if care is not taken in the design of the Server. Specifically, if the Server design takes the value from the counter and puts it in the CSU ELS frame before ensuring that the BB_Credit is greater than zero, then contention in the Fabric will cause a delay in getting the CSU ELS onto the wire. This will effectively worsen the Server's unload error.

Example:

Regarding the non-deterministic portion of the unload error, assume in the Fabric case that the Transmit FIFO can hold up to four full-size Fibre Channel frames; and that the design of the server does not ensure the FIFO is empty when the CSU ELS frame is pushed onto the FIFO. Again without justification, assume that each of the frames (including the CSU ELS frame) must wait for delivery of, say, four full-size Fibre Channel frames before it receives BB_Credit so that it can proceed. Then since

$$t_{\text{full_frame}} = ((15 + (2112 / 4)) \text{ Transmission Words}) * (37.65 \text{ ns} / \text{Transmission Word}), \text{ or}$$

$$t_{\text{full_frame}} \sim 20 \mu\text{s}$$

Then

$$\text{unload_error_ND} = t_{\text{full_frame}} * (3 * (4+1) + (4)), \text{ or}$$

$$\text{unload_error_ND} \sim 380 \mu\text{s}$$

O.2.4.1.4 Load Error

There are two sources of load error -- one in the ultimate client, and one in the Fabric as it acts as a client of the original server. These errors can be expected to be commensurate with each other.

O.2.4.1.5 R/T Clock Domain Error

The Fabric can be expected to contain internal clock boundaries that will need to be crossed as the CSU ELS information passes through the Fabric. The number of such crossings will depend on the internal design of the Fabric.

O.2.4.1.6 Server Oscillator Error

The Fabric will not introduce any additional errors in this area. The effect of the Fabric oscillator frequency was covered as part of the client oscillator frequency error above.

O.2.4.2 Fixes for Fabric Errors

Since the nature of the errors introduced by the Fabric is the same as those discussed in the point-to-point case, the same fixes can generally be applied to the design of the Fabric.

It should be emphasized that 32.2.2.3 includes rules for Fabrics that are designed to minimize the effect of delays through the Fabric. The Fabric maintains its own counter. It loads this

counter with the value received in the incoming CSU ELS frame. When the CSU frame is to be forwarded on the Client link, the Fabric modifies the CSU frame to contain the *current value* from the counter in the Fabric. If these rules are followed, the effect of delay through the Fabric is essentially eliminated.

O.2.5 Loop Considerations

For reference, Figure O.14 reproduces the model from Clause 32.2 for the Clock Synchronization Service in a Loop-based system. This is the basis for the discussions in the sub-clauses that follow.

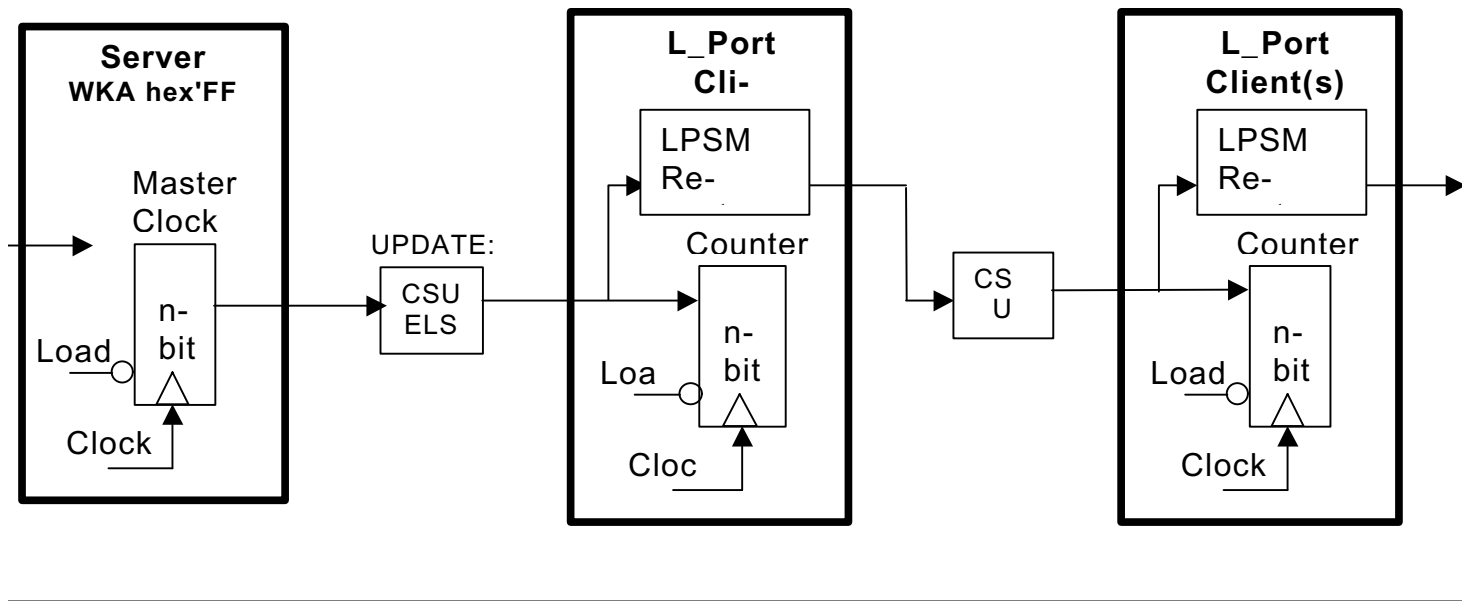


Figure O.14 - ELS Clock Sync Model – Loop

The diagram assumes that one of the L_Ports acts as the server while the other nodes on the Loop are clients. However, there is no requirement that all nodes on the loop be clients. The insertion of n L_Ports between the server and the client(s) results in additional errors being introduced into the client's counter. In terms of error analysis, it doesn't matter if the nodes between the server and a given client are clients or not since the delay through the LPSM repeater will be the same.

O.2.5.1 Discussion of Errors

The general nature of these errors is the same as discussed in O.2.3.1. Here, we will discuss only the differences between the point-to-point case and the loop case. There is, however, one unique aspect of the loop configuration which is the delay that occurs as transmission words are passed from one node to the next (node delay).

O.2.5.1.1 Node Delay

The Arbitrated Loop standard (FC-AL-2) allows up to 6 transmission word times to elapse between the time a transmission word is received until it is forwarded on to the next node in the loop. This delay is largely deterministic. There is a non-deterministic component of the error due to clock skew management.

Analysis:

634

Symbol	Definition
node_delay_error_D	The deterministic portion of the error caused by the fact that the Clock Count value in the CSU ELS frame is not updated as the frame is passed from one node to the next.
node_delay_error_ND	The non-deterministic portion of the error caused by the fact that the Clock Count value in the CSU ELS frame is not updated as the frame is passed from one node to the next.

Example:

In order to calculate the cumulative deterministic node delay, the system designer needs to know the number and type of nodes that lie between the server and the client. This will be different for each client on the loop.

Assume there are 5 nodes from Vendor A and 5 nodes from Vendor B between the server and the client. Also assume the specific node delays are as follows:

Vendor_A_node_delay = 6 transmission word times

Vendor_B_node_delay = 5 transmission word times

Then the deterministic node delay is as follows:

$$\text{node_delay_error_D} = 5 * \text{Vendor_A_node_delay} + 5 * \text{Vendor_B_node_delay}$$

$$\text{node_delay_error_D} = 5 * (6 * 37.65 \text{ ns}) + 5 * (5 * 37.65 \text{ ns})$$

$$\text{node_delay_error_D} = 2.07 \text{ usec}$$

For estimating the non-deterministic error, consider the discussion in FC-AL-2 Annex A (L_Port Elasticity buffer management). In that annex, it is stated that no more than 4 transmission words will be deleted between frames. Using this assumption, the worst case non-deterministic error would be:

$$\text{node_delay_error_ND} = 4 * 37.56 \text{ ns}$$

$$\text{node_delay_error_ND} = 150.24 \text{ ns}$$

This shows that even under worst case conditions the non-deterministic node delay error will probably be small compared to the deterministic error, depending on the size of the loop. The larger the loop the smaller it will be.

O.2.5.1.2 Client Oscillator Frequency Error

This error is the same as discussed in O.2.3.1.1. Only the server's oscillator and the oscillator of the client under consideration need to be considered. The effect of oscillators in other nodes is considered as part of the non-deterministic component of node delay error.

O.2.5.1.3 Link Propagation Delay Error

The nature of this error is the same as discussed in O.2.3.1.2. In the case of the loop, of course, the number of links to consider is generally larger. The links to consider are all of the links that lie between the server's transmitter and the client's receiver. This will be different for each client on the loop.

O.2.5.1.4 Unload Error

This error is the same as discussed in O.2.3.1.3. Even in the loop configuration, there is only one unload error, which is due to the server. There is no unload error in intermediate nodes because the counter value is simply transferred from input to output without being loaded into a counter and then unloaded from the counter.

O.2.5.1.5 Load Error

This error is the same as discussed in O.2.3.1.4. Even in the loop configuration, there is only one load error that applies to any given client. That is the load error in that client. There is no load error in intermediate nodes because the counter value is simply transferred from input to output without being loaded into a counter and then unloaded from the counter.

O.2.5.1.6 R/T Clock Domain Error

This error is the same as discussed in O.2.3.1.5. Even in the loop configuration, there is only one R/T clock domain error that applies to any given client. That is the R/T clock domain error in that client. The effect of clock domain crossings in other nodes is considered as part of the non-deterministic component of node delay error.

O.2.5.1.7 Server Oscillator Error

The Loop will not introduce any additional errors in this area.

O.2.5.2 Fixes for Loop Errors

Since the nature of the errors introduced in a loop is generally the same as those discussed in the point-to-point or Fabric cases, the same fixes can be applied to the design of the loop.

There is one source of error that is unique to loops, that being the node delay. The deterministic portion of the node delay error can simply be subtracted out at the client, as was done for other deterministic errors. Another approach to minimizing node delay error is to position the most time-sensitive nodes as close as possible to the server (on the downstream side).

O.3 An Example

Figure O.15 shows a hypothetical example of the application of clock synchronization to a tactical avionics system. The system contains two independent sensor subsystems, a processing subsystem, and a weapon delivery subsystem. The sensor subsystems receive energy from their environment, convert it to a series of digital samples, and send the sample set to the processing subsystem. Based on the combined information from the two sensor subsystems, the processing subsystem determines whether potential targets are present and if so, their tracking information. This data is presented to the pilot. The processing subsystem then computes data for attacking a target identified by the pilot. This data is sent to the weapon delivery subsystem, which causes the weapon to be targeted and released at the appropriate time for accurate delivery.

The figure does not explicitly show the Clock Synchronization server. Each of the four subsystems, though, is presumed to be a client of the same server so that they share a common sense of time.

The sensor subsystems attach a time tag (Time A1, Time B1) to the set of digitized samples from their respective receivers. Assuming that successive samples in a data set are taken at regular intervals, tagging the data set with the time of arrival of the energy at the sensor for the first sample will allow the determination of the time of arrival of all samples in the set.

The information available to the algorithm in the processing subsystem includes:

- Digitized samples of the energy received at Sensor A
- The time of arrival of the sampled energy at Sensor A
- Characteristics of Sensor A, including the orientation of the receiving aperture
- Digitized samples of the energy received at Sensor B
- The time of arrival of the sampled energy at Sensor B
- Characteristics of Sensor B, including the orientation of the receiving aperture
- The current time

Note that once the time tag has been attached to the samples by the sensor subsystems, the processing subsystem has no need to further tag them. It is not necessary, for example, for it to note the time at which the frames containing the samples arrived at its FL_Port. What is important is the time at which the energy from potential targets arrived at the sensors. The current time may be important so that the processing subsystem does not present stale data to the pilot. It is also important so that any computed targeting information be prepared for a time that is still in the future. (It would do little good to tell the weapon delivery subsystem what it should have done at some time in the past.) The sense of shared time between the processing subsystem and the weapon delivery subsystem ensures that the weapon is triggered at the time most appropriate for precise delivery of the weapon.

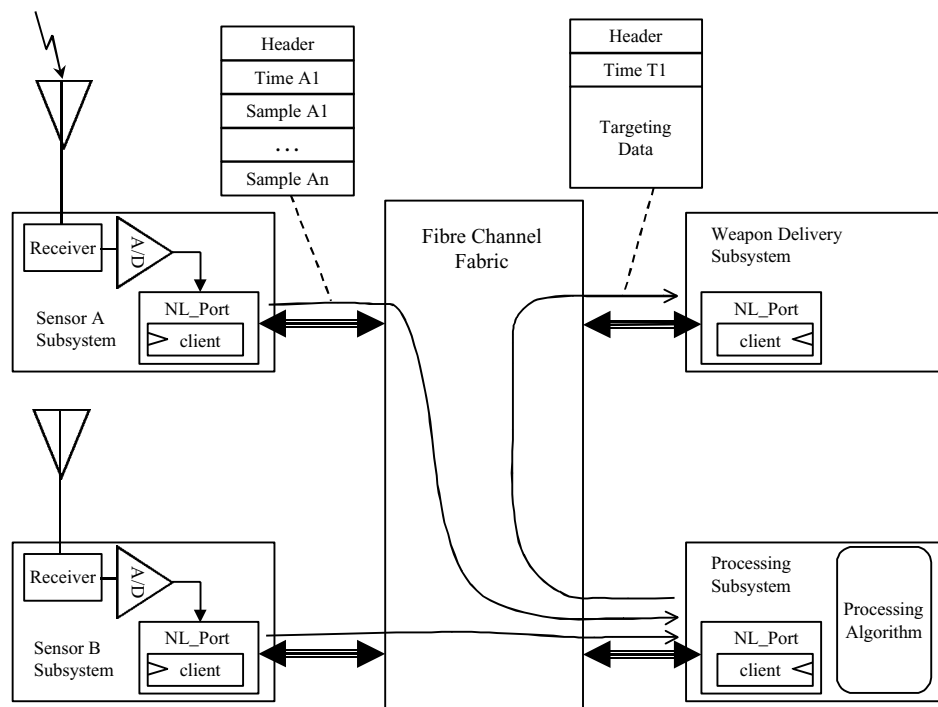


Figure O.15 - Application of Clock Synchronization to Tactical Avionics

In this example, the critical associations of time and data occur in the sensor subsystems and in the weapon delivery subsystem. If software is involved in reading the time counter and attaching it to a data set, the accuracy of the time tag will probably be worse by at least one, and probably two orders of magnitude as compared to a hardware-based time tagging. So for maximum precision, the sensor subsystems would use hardware to capture a time value from the counter at the precise time that the sample comes from the Analog-to-Digital converter.

(Even greater inaccuracy would result if the samples were to travel through the Fabric and be time-tagged when they arrive at the processing subsystem.)

Similarly, in the weapon delivery subsystem, the actual triggering of the weapon would be accomplished by hardware directly linked to the synchronized time counter.

In contrast to these considerations, the software in the processing subsystem has a more relaxed need for knowledge of time. Its primary need is to ensure that the information it presents to the pilot represents the current situation; and that the targeting data that it computes for the weapon delivery subsystem is for some time in the future. But the time that is used in the algorithm itself as part of the interpretation of the sample data is the time attached to that data by the sensor subsystems. So the processing subsystem probably has no need for a direct hardware-based tagging of information.

As a final comment, if the adjustment for oscillator frequencies (see O.2.3.2) is desired, but the sensor nodes have no embedded processor to apply the adjustment factors, the simple "Time A1" value indicated in Figure O.15 could be replaced by the following tuple:

- Counter
- ELS_value_n
- ELS_value_{n-1}
- ELS_arrived_{n-1}

Of course, the hardware in the sensor that attaches this tuple to the data set should be able to do an atomic reading of all components of the tuple so that values in the tuple are coherent.

Algorithms in the processing subsystem could then apply the adjustment algorithms separately for each sensor before using the time tag in its tracking and targeting algorithms.

Annex P Speed negotiation details (informative)

This annex contains informative details relating to the development of the speed negotiation algorithm and requirements and to speed agile Ports.

P.1 Basic assumptions

The Speed Negotiation method is based on the following set of assumptions:

- a) The objective is to find the highest common speed that actually operates for all elements in the FC link involved in the speed negotiation.
Functionality is demanded from the entire link at the speed selected including all cables, connectors, hubs, transceivers, serdes, and conversion devices. The design capabilities of the components are not sufficient criteria for acceptance – actual hardware must perform.
- b) Error free Word Sync for 1000 Word times is an adequate quality measure for Speed Negotiation purposes. This is not the same as verifying operation at the Fibre Channel bit error rate.
- c) Link quality issues detected after the speed has been determined are addressed by other means.
- d) Once a speed has been negotiated, it is permissible that the speed not be changed after conditions are improved such that operation at a higher speed would now be possible. Forcing a re-negotiation is done with higher level protocols or out-of-band.
- e) Speed Negotiation shall not hang unless it is physically impossible for any common speed to exist.
- f) Point-to-point topology (E-E, N-N, F-N, and FL-NL Port connections) is required. Loop configurations that negotiate speeds shall present a single port to the other negotiating Port for speed negotiation purposes.
- g) Ports capable of Speed Negotiation are not required to support a common 1Gb/s. speed.
- h) The transmitter and receiver shall be capable of working at different speeds at the same time during Speed Negotiation.
- i) Each Port can support up to a maximum of any 4 speeds.
- j) The Speed Negotiation method shall be independent of and compatible with the link protocol (e.g. OLD_PORT vs FC-AL)
- k) The same Speed Negotiation method is required to support both copper and optical Ports.
- l) The algorithm shall be realizable in a host driver or in Port firmware.
- m) Hubs are set to a single speed by some out-of-band means. This does not preclude intelligent hubs from being able to operate at multiple speeds.
- n) The algorithm works with loop infrastructure (e.g. hub) that has a Port attachment scheme that supports sensing of the operating speed of the infrastructure by the attached Port receiver while preventing the attached Port transmitter from connecting into the existing infrastructure unless the proper speed is established.
- o) Once a particular speed has been established Speed Negotiation is not attempted unless a LINK FAILURE is detected or by means outside the scope of this section.
- p) The algorithm is required to work with Ports that operate only at any single speed or with loops that have been set to a single speed by means not specified in this standard

P.2 Supported configurations

There are three cases supported by the algorithm as shown in Figure P.1:

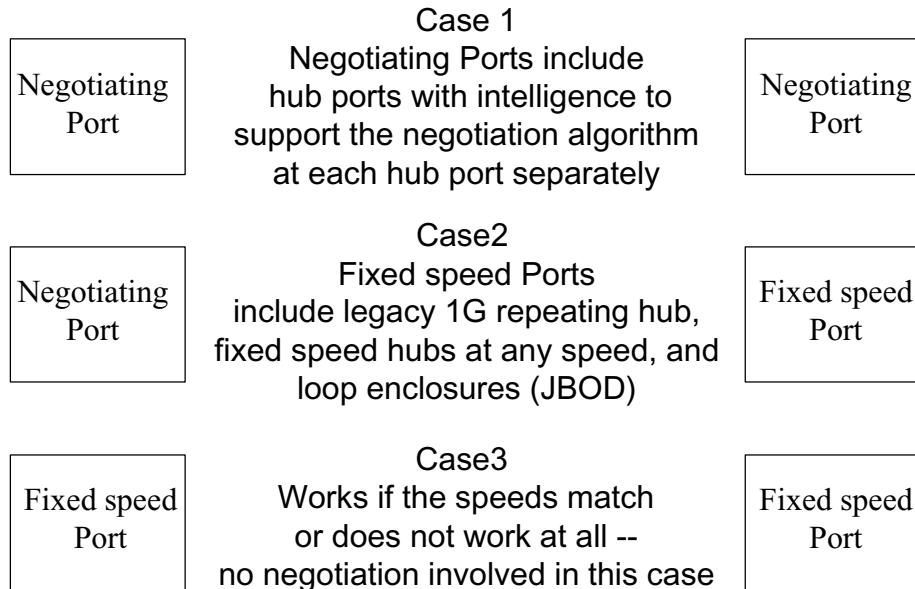


Figure P.1 - Three configurations supported by the Speed Negotiation requirements

Note that only a single pair of Ports can be involved in the Speed Negotiation. This means that multi Port entities such as hubs and JBOD's have significant restrictions when used with the Speed Negotiation algorithm. Specifically, hub ports either shall be capable of executing the Speed Negotiation algorithm independently for every hub Port or the hub speed is fixed for all Ports. For JBOD's the entire enclosure shall be presented to the attached loop Port as a single Speed Negotiating loop Port or the entire population of devices within the JBOD enclosure shall be fixed speed.

P.3 Derivation of timing requirements

In this section the derivation of the timing requirements is shown

Receiver Cycle Time, t_{rxcycl}

This timing value is based on firmware capabilities present in current implementations and ranges from 16 to 30 ms.

Wait_Transmitter Cycle Time, $t_{waitcycl}$

$= (\text{number of speeds}) * \text{Maximum Receiver Cycle Time}$

$$= 4 * 30 \text{ ms} = 120 \text{ ms}$$

The $t_{waitcycl}$ time is increased by 2 ms to allow for the worst case alignment of the other transmitter's send time with the four 30 ms worst case receiver sampling intervals. The

;minimum $t_{waitcycl}$ is therefore 122 ms. The maximum $t_{waitcycl}$, 152 ms, is found by adding 30 ms (maximum t_{rxcycl})

Master_Transmitter cycle time, t_{txcycl}

As stated above for $t_{waitcycl}$, the total time that the transmitter must send a particular speed is 122 ms. Only 3 different speeds need be tested for during t_{txcycl} since the speed being transmitted will be sent for a duration of t_{rxcycl} in State 27 in Figure 96. Therefore:

Minimum $t_{txcycl} = 122\text{ms} - 16\text{ ms (minimum } t_{rxcycl}) = 106\text{ ms}$

Maximum $t_{txcycl} = \text{Minimum } t_{txcycl} + 30\text{ ms (maximum } t_{rxcycl}) = 136\text{ ms}$

Speed stability time , t_{stbl}

t_{stbl} must be of sufficient duration to insure that the other transmitter is no longer changing speeds. The longest transmitter time for a particular speed is 5 maximum Receiver Cycle Times or 150 ms + 2 ms for worst case misalignment of receiver cycle time with respect to the transmitter's send time . After an additional maximum Receiver Cycle Time, the speed will have either changed or will be stable (in the Negotiate_follow stage). Therefore:

Minimum $t_{stbl} = 152\text{ ms}$

Maximum $t_{stbl} = \text{Minimum } t_{stbl} + 30\text{ ms(maximum } t_{rxcycl}) = 182\text{ ms}$

Longest stage convergence time , t_{fail}

t_{fail} is used as a watch dog timer to time out the Negotiate_master and Negotiate_follow stages in the event that convergence does not occur. The longest t_{fail} was derived by a trace through the algorithm for the case where both devices are capable of 4 speeds, the infrastructure in one direction is capable of 4 speeds, but the infrastructure in the other direction is only capable of the lowest speed. This longest time conservatively fits within 10 instances of maximum transmission times. Therefore t_{fail} is given by:

> 10 * maximum transmission time

> 10 * 152 = 1,520 ms (1500 is used in the algorithm)

Algorithm convergence time:

The longest convergence time, including Wait_for_signal, is approximately 11* maximum transmitter times of 152ms + t_{stbl} of 182 ms for a total of 1,854 ms. The longest time is with both ports capable of negotiating at all 4 speeds and where the infrastructure only supports the lowest speed. Based on this calculation a maximum convergence time of 2 seconds was chosen for the Speed Negotiation algorithm.

P.4 Ports using separate PMD components

This section describes the issues with using separate PMD components in a speed agile application. Figure P.2 shows the general relationship of the two Ports and the physical architecture within one of the Ports.

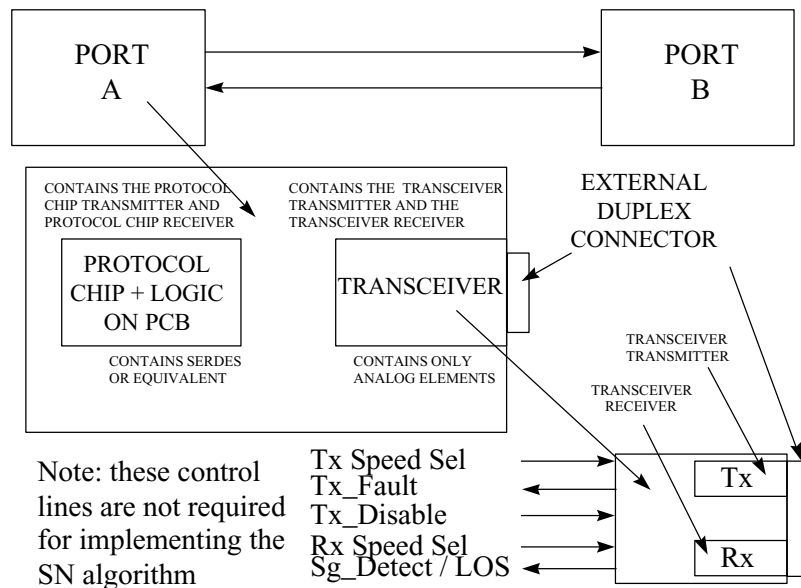


Figure P.2 - Physical architecture of a Port with a separate transceiver component

If a Port uses a separate PMD component (and especially a removable PMD component such as a GBIC) care is required to ensure that both the Port supplier and the PMD component supplier clearly understand what is required to achieve the speed agility and to execute the Speed Negotiation algorithm. Although timings are formally measured at the external duplex Port connector for the Speed Negotiation algorithm this is only possible because the granularity of the timing requirements for the algorithm are orders of magnitude more coarse than the signal propagation time through the removable PMD component and the logic. In practice, only the protocol chip and other board logic are capable of enforcing accurate timings so if the separate PMD has time delays of the order of the Speed Negotiation algorithm timing granularity the assumption of timing values applying at the Port connector is rendered invalid.

Other features are listed below:

- The protocol chip and other board logic may be supplied from different sources than the transceiver. This means that the protocol chip and other board logic cannot be generally treated as a unit with the transceiver. Specifications shall be placed specifically on one or the other (or both separately) and the use of any control signals shall be noted.
- There are effectively two transmitters and two receivers in each Port. The receiver in the transceiver is termed the transceiver receiver and the receiver in the protocol chip or on the board, but not part of the transceiver is termed the protocol chip receiver. Similarly: transceiver transmitter and protocol chip transmitter.
- The speed of the transceiver transmitter is controlled by the protocol chip and other board logic by changing the speed of the data signals driven from the protocol chip. However, the launch amplitude and /or other properties of the transceiver transmitter either needs to be (1) common to all supported speeds, (2) a control signal to the transceiver is used to set the amplitude of the transceiver transmitter or (3) internal circuitry within the transceiver senses the incoming bit rate and adjusts the amplitude accordingly.

Note: The requirements for full speed and double speed FC are not mutually exclusive. That is, it is possible to design a transceiver transmitter that meets both the full speed and the double speed requirements without any change.

- The speed of the transceiver receiver is similarly controlled by either (1) having the transceiver receiver specifications common to all supported speeds, (2) a control signal to the transceiver is used to set the properties of the transceiver receiver or (3) internal circuitry within the transceiver senses the incoming bit rate and adjusts the receiver parameters accordingly.

Note: The requirements for full speed and double speed FC are not mutually exclusive. That is, it is possible to design a transceiver receiver that meets both the full speed and the double speed requirements without any change. For any speeds higher than double speed the transceiver receiver needs to change its properties in order to meet the transceiver receiver requirements.

P.5 Implementation notes

Since the Speed Negotiation algorithm requires some cycles from the processor in the Port when executing it may be desirable to temporarily change or disable the speed negotiation capability of ports that remain unconnected or unused for extended periods to eliminate the need for frequent unproductive polling of the Port. There are several methods that could be used for changing or disabling the function:

- Commands from an out of band management system
- Hardware jumpers
- Using a much longer polling period for incoming signal sync possibilities before actually starting speed negotiation than used during active Speed Negotiation
- Using a signal detect function (Rx_LOS) to detect when a connection is made (may not be a reliable indication that the Tx side is connected and requires that the opposite connected port be transmitting something)
- Using an automatic sensing of connector retention engagement or the presence of a removable PMD device to sense plausible device attachment (does not guarantee that the opposite end of the link is connected)

Transceivers may be able to transmit and detect error free bit streams even though they and other link elements were not designed or specified for operation at the speed being used. This condition may allow links to achieve word sync and satisfactory error rates but with degraded margin. It is up to the implementer to ensure that the elements of the physical plant are designed to comply with the requirements specified for operation at the set speed.

Index

3

32-bit IP address, 154

4

48-bit IEEE, 153, 154, 637, 649

A

Abort Exchange, 201
 Abort Exchange (ABTX), 199, 203, 204, 240, 252, 318, 421
 Abort Sequence (ABTS), 188, 199, 240, 252, 407
 Abort Sequence Condition, 142, 188
 Abort Sequence Protocol, 479
 Accept (ACC), 318, 322

ACK_0, 166, 167, 169, 376, 378, 424, 427, 533

ACK_1, 166, 167, 168, 424, 427, 533, 586

ACK_Form, 140, 184, 185

Acknowledged Class, 49

Active, 49

Active Sequence, 205, 219, 220, 407, 456, 462, 470, 480

Active Virtual Circuit Credit limit, 49

address identifier, 49, 71, 131

Advise Credit, 201

Advise Credit (ADVC), 199, 205, 318

Alias, 49

alias address identifier, 49, 52, 58, 286, 287, 289, 290, 292, 296, 491, 501, 532

Alias Group Service Parameters, 286, 289, 501

Alias registration, 501

Alias Routing, 501

Alias Server, 49, 392, 489, 490, 501

Alias Token, 49

Alternate BB_Credit management, 438

Application data mapping, 78, 446

arbitrated loop topology, xlv, 68

Arbitrated Loop topology, 49, 52, 53, 67, 68, 549
 Association Header, 420, 502
 Association_Header, 54, 77, 157, 158, 159, 160
Available BB_Credit, 49
 available credit, 515, 539, 540
 Available Credit, 439

B

Bandwidth Allocation, 200, 202, 301, 320
 Bandwidth Deallocation, 200, 202, 302, 320
 Basic Accept (BA_ACC), 191, 193, 195
 Basic Link Service, 130, 135, 148, 160, 162, 186, 187, 188, 191, 193, 195, 197, 198, 199, 221, 400, 407, 451, 456, 462, 536
 Basic Link Service commands, 186, 187, 197
 Basic Reject (BA_RJT), 191, 193, 195
BB_buffer, 49, 486
 BB_Credit allocation, 432
 BB_Credit management, 360, 363, 431, 438, 633
 BB_Credit_CNT management, 432
 BB_Credit_Count reset, 438
Beginning Running Disparity, 49
 bidirectional Connection, 141, 454, 456, 460, 463
 bit synchronization, 94, 108, 577
Block, 49, 56, 59, 60, 61, 76, 77, 78, 214, 215, 216, 416, 418, 446, 483, 488, 631
 Broadcast, 132, 384, 500, 503
 BSY / RJT in flow control, 440
buffer, 49
 Buffered Class 1, 133, 351, 352, 367, 369, 371, 374, 389, 549
 Buffer-to-buffer Class dependency, 432
 buffer-to-buffer Credit, 168, 333, 345, 360, 362, 363, 383, 393, 423, 438, 468, 469, 476, 486
 Buffer-to-buffer Credit_Count, 59, 425
 buffer-to-buffer flow control, 49, 74, 162, 167, 329, 330
 Buffer-to-buffer flow control model, 431, 433
 Buffer-to-buffer management rules, 431
 Buffer-to-buffer Receive Size, 361, 365, 385
 Building Blocks, 64, 74
 Busy (F_BSY, P_BSY), 163

C

Camp-On, 371, 374, 389, 542
 CCITT, 153, 156, 157
 Chained_Sequence (C_S), 415, 456, 461
character, 49
circuit, 49
 circuit initiator, 291, 513, 517
 circuit recipient, 291, 513, 517
 Class 1, 327
Class 1 service, 50
 Class 2, 330
Class 2 service, 50
 Class 3, 332

644

Class 3 service, 50
Class 4 bidirectional circuits, 50
Class 4 Circuit Initiator, 50
Class 4 Circuit Recipient, 50
Class 4 end-to-end credit limit, 50
Class 4 service, 50
 Class dependent frame flow, 432
 Class Specific Control (CS_CTL), 133, 504
Classes of service, 50, 64, 69, 71, 327, 329, 331, 333, 517
 Clock Synchronization Request, 200, 202, 304, 321
 Clock Synchronization Update, 200, 202, 305, 321
code bit, 50
code violation, 50
 Code Violation, 88, 97, 575
comma, 50
 Comma, 85, 94, 96
comma character, 50
 communication model, 327
 Communication model, 66, 67, 580
 Communication Model, 489, 514, 560
concatenation, 50, 61
 concurrent Sequences, 51, 76, 169, 401
 congestion, 74, 513, 540
Connection Initiator, 50
 Connection management, 451, 452, 453, 600
Connection Recipient, 50
 Connection Recovery, 112, 114, 181, 459, 462, 475
Connectionless buffers, 50
Connectionless frames, 50
Connectionless service, 51
Connection-oriented frames, 50
 connect-request, 111
 Continue Sequence Condition, 138, 142, 404, 405, 415
Continuously Increasing Relative Offset, 51
 COR, 59, 133, 511, 542, 543, 544
 CR_TOV, 386, 391
 CRC generation and checking, 583
Credit, 51
 Credit_Count, 425
current running disparity, 51
 Cyclic Redundancy Check (CRC), 57, 123

D

D_NAA, 152, 153, 154, 155
Data Character, 51
 Data Field Control (DF_CTL), 148
Data frame, 51
 data integrity, 123, 149, 484, 608
 Datagram, 70, 327, 332
Decoding, 51
Dedicated Connection, 51, 327
Dedicated Duplex, 51
Dedicated Simplex, 51
delimiter, 51
 Destination Identifier (D_ID), 132, 346, 352

destination N_Port, 51, 68
Destination_ID (D_ID), 132
Destination_Identifier (D_ID), 51
Device_Header, 149, 151, 152, 159, 165
DF_CTL, 123, 148
Diagnostic mode, 108
discard policy, 51, 406, 488
Discard policy, 477, 485
Disconnect_Terminate_Invalid (EOFdti), 124
disconnection, 51
Discover Address, 200, 202, 298, 320
Discover F_Port Service parameters, 320
Discover F_Port Service Parameters, 200, 202
Discover N_Port Service parameters, 320
Discover N_Port Service Parameters, 200, 202
Discover N_Port/F_Port Service Parameters, 297
disparity, 51
Dynamic Half Duplex, vi, 358, 361, 364, 384

E

E_D_TOV, 59, 163, 167, 186, 187, 189, 217, 218, 219, 358, 363, 365, 383, 384, 385, 402, 404, 406, 411, 419, 427, 459, 467, 468, 469, 470, 471, 472, 486, 487, 512, 538, 542, 544, 547, 615, 616, 619, 621, 630
Echo, 199, 201, 207, 318, 626
ECHO, 207, 208
EE_buffer, 51, 486
encoding, 51
End_Connection (E_C), 139, 452
End_of_Frame (EOF), 110, 123
End_Of_Frame (EOF), 528
End_of_Frame Abort (EOFa), 125
End_of_Frame Disconnect_Terminate (EOFdt), 124
End_of_Frame Invalid (EOFni), 125
End_of_Frame Normal (EOFn), 124
End_of_Frame Terminate (EOft), 124
End_Sequence, 139, 404, 405, 407
end-to-end credit, 182, 292, 295
end-to-end Credit, 114, 115, 161, 169, 181, 205, 206, 294, 296, 345, 350, 351, 355, 356, 363, 381, 383, 393, 394, 395, 396, 403, 407, 423, 425, 429, 468, 471, 476, 478, 480, 486
End-to-end Credit, 59, 329, 333, 366, 367, 368, 373, 381, 386, 391, 406, 425, 443, 483
End-to-End Credit, 428, 506
End-to-end Credit allocation, 430
end-to-end Credit management, 169
end-to-end Credit_CNT, 181, 423, 455, 465
End-to-end Credit_Count, 59, 329
end-to-end flow control, 51, 162, 167, 169, 170, 181, 182, 293, 328, 330, 331, 332, 423, 425, 427, 428, 440, 442, 521, 538
End-to-end flow control, 426, 443, 539
End-to-end flow control model, 428, 429
End-to-end management rules, 426
EOFa, 93, 125, 126, 182, 335, 336, 484, 485, 506, 508, 528, 531, 535, 551, 631
EOFdt, 93, 124, 462, 528

EOFdti, 124, 125, 126, 182, 452, 485, 506, 508, 522, 528, 535, 631
EOFn, 93, 124, 174
EOFni, 93, 125, 126, 182, 506, 508, 528, 631
EOft, 93, 124, 508, 528, 535
Error policy, 368, 379
Error Policy, 302, 303, 304, 320, 407, 408, 409, 410, 417, 477, 479, 488, 536
Establish Streaming, 201
Establish Streaming (ESTS), 199, 209, 318, 394
Estimate Credit, 201
Estimate Credit (ESTC), 199, 208, 318, 395
Estimate Credit procedure, 205, 394, 396, 428
Exchange, 51, 54, 55, 61, 64, 74, 75, 76, 77, 137, 149
Exchange Context, 137, 139, 147, 593, 594, 595
Exchange reassembly, 138, 144
Exchange Status Block, 199, 201, 214, 416
Expiration_Security_Header, 151, 152, 160
Explicit Login, 344

F

F_BSY, 59, 165, 171, 172, 347, 349, 352, 535
F_CTL, 59, 125, 128, 132, 135, 137, 139, 140, 144, 145, 146, 147, 150, 163, 165, 170, 171, 173, 175, 180, 184, 185, 375
F_Port, 52, 73
F_Port Login, 199, 201, 210
F_Port_Name, 52
F_RJT (Fabric Reject), 164, 165, 534
Fabric, 52, 67, 68
Fabric Activate Alias_ID, 200, 201, 287, 319
Fabric Address Notification, 200, 202, 224, 320
Fabric Deactivate Alias_ID, 200, 201, 288, 319
Fabric Login, 75, 338, 345, 346, 347, 348, 357, 358, 359, 367, 368, 383, 386, 549, 614, 615
Fabric model, 71, 72, 453, 637
Fabric topology, 67
Fabric Topology, 68, 553
Fabric_Frame, 122, 127
Fabric_Name, 52, 211, 386
FC-0, xlv, 63, 108, 112, 568
fibre, 52
Fibre Channel, 52
Fibre Channel Address Resolution Protocol Reply, 200, 202, 262, 320
Fibre Channel Address Resolution Protocol Request, 200, 202, 258, 320
Fibre Channel Name, 156
Fibre Channel structure, 62
Fill Data Bytes, 139, 144
First_Sequence, 137, 139, 145, 147
FLOGI, 185, 201, 210, 318, 324, 344, 347, 348, 357
flow control, 73, 111, 164, 167, 330, 332, 334, 336, 340, 343
Flow control, 423
frame, 52

Frame, 93, 110, 121
 frame content, 123, 124
Frame Content, 52
 Frame field order, 125
 frame format, 121
 frame header, 52
 Frame Header, 128
 Frame identification, 128
 Frame reception, 126
 frame retransmission, 166
 Frame transmission, 121
 Frame validity, 126
 FT_0 frame, 122
 FT_1 frame, 123, 361, 380, 385

G

Get Alias_ID, 200, 201, 286, 319

H

half-duplex, 580, 581, 635
 hierarchy, 74, 75, 82
 History bit, 169, 403, 426, 427
 History Bit, 170
Hunt Group, 52, 59, 66, 384, 489, 501, 502

I

identifiers, 49
Idle, 59, 94, 111
 IEC, 47, 136, 583
 IEEE 48-bit address, 153, 637, 649
 IEEE extended, 154, 156, 157, 648, 650, 651
 IEEE Extended, 650
 IEEE extended address, 154
 IEEE registered, 154, 155, 648, 652, 653
 IEEE Registered, 153, 155, 156, 157, 648, 651, 653
ignored, 52
 image pair, 238
Implicit Login, 344
 Implicit Logout, 356
infinite buffer, 52
Information Category, 52
 Information Category bits, 130, 166, 179, 201
information transfer, 53, 66
 information unit, 237
Information Unit, 53, 237, 606
initialization, 53
 Initialization, 118
 Integrated Class 2 flow control, 442, 443, 444
 interface, 564
 intermix, 339, 539
Intermix, 53, 70, 214, 334, 335, 336, 367, 369, 374, 387, 443, 506, 551
 interoperability, 55, 165
 invalid frame, 474, 485
 Invalid frame, 124, 127
 invalid Transmission Word, 95, 99
Invalid Transmission Word, 97

646

invalidate X_ID, 609
 Invalidate X_ID, 138, 140

L

laser, 101, 103, 109, 281
 last Sequence, 137
 Last_Sequence, 137, 139, 145, 147
 LCR in flow control, 440
 Link Credit Reset, 59, 166, 181
 Link Error Status Block, 215, 483, 631
 link errors, 70, 631
 Link errors, 474
 Link Errors, 215, 473
 Link failure, 631
 Link Failure, 115, 116, 119, 281, 282, 469, 474, 487, 603, 631
 LINK FAILURE, 604
 Link Incident Record Registration, 200, 202, 283, 321
 Link Initialization, 77
 Link integrity, 468
 Link Reset, 59, 112, 120, 475, 486, 647
 Link Service Reject, 201, 242, 250, 256, 297, 308, 322, 356
 Link timeout, 120, 462, 469
 Link_Continue, 167, 533
Link_Control, 53, 55, 56, 59, 66, 75, 145, 161, 162, 164, 165, 166, 167, 179, 181, 182, 184, 535
 Link_Response, 162, 164, 170, 184, 534
local F_Port, 53, 333
 Locally assigned Network Header Address, 153, 154
 lock-down mode, 460
 Lock-Down Mode, 388
 Login, 59
 Login (LOGI), 367
 Login Control List Management, 200, 201, 305, 321
 Logout, 201, 212, 318
 Logout (LOGO), 199
 Loop Initialize, 200, 202, 225, 321
 Loop Port Control, 200, 202, 227, 321
 Loop Status, 200, 202, 229, 321
loopback, 53
 Loopback, 95, 107
 LR Receive state, 529
 LR Receive State, 114
 LR Transmit state, 529
 LR Transmit State, 113
 LRR Receive state, 530
 LRR Receive State, 115

M

mandatory, 53, 57
 media, xlv
 missing frame, 51, 54, 379, 406
 multiplex, 65
 Multiplex, 69, 330

N

N_Port, 54, 60
N_Port Activate Alias_ID, 200, 202, 289, 319
N_Port Deactivate Alias_ID, 200, 202, 290, 319
N_Port identifier, 66, 71, 73, 163, 212
N_Port Identifier, 54
N_Port_Login, 184, 201, 210, 325, 350
N_Port_Login (PLOGI), 199
N_Port_Name, 54, 366
Name_Identifier, 53, 156, 157
Network_Address_Authority, 53, 60, 152
Network_Destination_ID, 153
Network_Header, 152, 159
Network_Source_ID, 153
No Operation, 60, 188
No Operation (NOP), 197
Node, 53
node identifier, 649, 651
Node_Name, 53
non-repeating ordered set, 53
Non-Repeating Ordered Set, 571, 572
NOS Receive state, 530
NOS Receive State, 116
NOS Transmit state, 530
NOS Transmit State, 115
Not Operational, 54, 95, 101

O

OFC, 60
OLS, 60
OLS Receive state, 531
OLS Receive State, 117
OLS Transmit state, 530
OLS Transmit State, 117
OLS transmit timeout, 471
Online to Offline Protocol, 119
open fibre control, 60
Open Sequence, 366, 373, 382, 391, 413
Operation, 54
Operation_Associator, 159
Operational, 54
optional, 54
Optional headers, 151
Originator, 54, 66, 75, 158, 350, 355, 412, 492
Originator Exchange_ID, 51, 412
OX_ID, 54, 60, 77

P

P_BSY, 60
P_RJT, 60, 126, 165
Parameter field, 150, 170, 172, 173, 175, 176, 178, 180
Parameter Field, 126, 188
Payload, 54, 67, 125, 130, 131, 163, 165
physical model, 64, 65
PLOGI, 60, 210
point-to-point, 353, 354
Point-to-point, 67, 365, 443

Port identifier, 66
Port Identifier, 54
Port_Name, 54
Primitive Sequence, 54, 77
Primitive Sequence protocol error, 474
Primitive Sequences, 94, 110, 111, 118, 529
Process login, 238
Process Login, 199, 201, 319
Process logout, 247
Process Logout, 199, 201, 319

Q

Quality of Service Request, 200, 202, 291, 319

R

Read Connection Status, 199, 201, 213, 318
Read Exchange Concise, 199, 201, 222, 319
Read Exchange Status, 318
Read Link Error Status, 318
Read Link Error Status Block, 199, 201, 215
Read Port List, 200, 202, 314, 321
Read Port Status Block, 200, 202, 311, 321
Read Sequence Status, 318
Read Sequence Status Block, 201, 216
Read Sequence Status Block (RSS), 199
Read Timeout Value, 201, 217, 318
Read Timeout Value (RTV), 199
Read Virtual Circuit Status, 200, 202, 296, 320
recognized Ordered Set, 576
Registered Link Incident Record, 200, 202, 278, 321
Registered State Change Notification, 200, 202, 231, 320
Reinstate Recovery Qualifier, 199, 201, 219, 319
Reject (P_RJT, F_RJT), 175, 534
Relative Offset present, 138, 144, 449
Remove Connection (RMC), 187, 198, 199, 456
repeating ordered set, 55
Repeating Ordered Set, 571, 572
Report Node Capability, 202, 224
Report Node FC-4 Types, 200, 202, 316, 321
Request Node Identification Data, 200, 202, 271, 321
Request Sequence Initiative, 200, 201, 220, 319
Request Topology Information, 200, 202, 265, 321

S

Sequence, 593
Sequence context, 418
Sequence Context, 137, 139, 147, 594, 595
Sequence initiative, 414
Sequence Initiative, 138, 140, 145, 147, 399, 405, 414, 417
Sequence integrity, 333, 334, 468, 476, 478
Source_ID (S_ID), 132
State change notification, 252

State Change Notification, 200, 201, 319
 State Change Registration, 200, 202, 235, 320
 Stop Sequence, 138, 143, 170, 482, 487

T

Test, 200, 201, 222, 319
 Test process login state, 255
 Test Process Login State, 200, 201, 237, 319
 Third Party Process Logout, 200, 201, 299, 319

U

unrecognized ordered set, 56
 unrecognized Ordered Set, 110
 Unrecognized Ordered Set, 574, 575

X

X_ID reassigned, 138, 140