



ISO/IEC 14165-251

Edition 1.0 2008-01

INTERNATIONAL STANDARD

Information technology – Fibre channel –
Part 251: Framing and signalling (FC-FS)

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2008 ISO/IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester.

If you have any questions about ISO/IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland
Email: inmail@iec.ch
Web: www.iec.ch

About the IEC

The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications

The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigenda or an amendment might have been published.

- Catalogue of IEC publications: www.iec.ch/searchpub

The IEC on-line Catalogue enables you to search by a variety of criteria (reference number, text, technical committee,...). It also gives information on projects, withdrawn and replaced publications.

- IEC Just Published: www.iec.ch/online_news/justpub

Stay up to date on all new IEC publications. Just Published details twice a month all new publications released. Available on-line and also by email.

- Electropedia: www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 20 000 terms and definitions in English and French, with equivalent terms in additional languages. Also known as the International Electrotechnical Vocabulary online.

- Customer Service Centre: www.iec.ch/webstore/custserv

If you wish to give us your feedback on this publication or need further assistance, please visit the Customer Service Centre FAQ or contact us:

Email: csc@iec.ch
Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00



ISO/IEC 14165-251

Edition 1.0 2008-01

INTERNATIONAL STANDARD

Information technology – Fibre channel –
Part 251: Framing and signalling (FC-FS)

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

PRICE CODE

XC

ICS 35.200

ISBN 2-8318-9483-2

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

Contents

FOREWORD	37
INTRODUCTION	40
1 Scope	41
2 Normative references	41
2.1 Approved references	41
2.2 References under development	41
2.3 Other references	41
3 Definitions, abbreviations, conventions and keywords	43
3.1 Definitions	43
3.2 Editorial conventions	50
3.3 Abbreviations, acronyms and symbols	51
3.3.1 Data rate abbreviations	51
3.3.2 Acronyms and other abbreviations	51
3.3.3 Symbols	55
3.4 Keywords	55
4 Structure and Concepts	57
4.1 Introduction	57
4.2 FC-1 general description	58
4.3 FC-2 general description	59
4.4 FC-FS physical model	60
4.5 Communication models	60
4.5.1 Introduction	60
4.5.2 Hunt Group	61
4.5.3 Fractional bandwidth	62
4.6 Bandwidth	62
4.7 Topology	62
4.7.1 Types	62
4.7.2 Point-to-point topology	62
4.7.3 Fabric topology	62
4.7.4 Arbitrated Loop topology	63
4.8 Classes of service	63
4.8.1 General	63
4.8.2 Class 1 service - dedicated connection	64
4.8.3 Class 2 service - multiplex	64
4.8.4 Class 3 service - datagram	64
4.8.5 Class 4 service - fractional bandwidth	65
4.8.6 Class 6 - multicast connection	65
4.9 Intermixing other classes with Class 1 or Class 6	65
4.10 General Fabric model	66
4.10.1 General	66
4.10.2 Fabric Ports (F_Ports)	66
4.10.3 Connection Service	68
4.10.4 Connectionless Service	68
4.11 Fibre Channel services	69
4.12 Building Blocks	69
4.12.1 Building block hierarchy	69
4.12.2 Frame	70
4.12.3 Sequence	70
4.12.3.1 Introduction	70
4.12.3.2 Sequence Identifier (SEQ_ID)	70
4.12.3.3 Sequence Status Blocks	70
4.12.4 Exchange	71
4.12.4.1 Introduction	71

4.12.4.2	Exchange_Identifiers (OX_ID and RX_ID)	71
4.12.4.3	Association_Header	71
4.12.4.4	Exchange Status Blocks	71
4.12.5	Exchange of service parameters	71
4.13	Segmentation and reassembly	72
4.13.1	General	72
4.13.2	Application data mapping	72
4.13.3	Relative offset	72
4.13.4	Sending end mapping	72
4.13.5	Capability	72
4.13.6	FC-2 mapping	73
4.13.7	Segmentation	73
4.13.8	Reassembly	73
4.14	Error detection and recovery	73
5	FC-1 8B/10B transmission code	74
5.1	Introduction	74
5.2	Notation conventions	74
5.3	Character encoding and decoding	75
5.3.1	Introduction	75
5.3.2	Transmission order	75
5.3.3	Valid and invalid Transmission Characters	75
5.3.3.1	Definitions	75
5.3.3.2	Generating Transmission Characters	81
5.3.3.3	Validity of received Transmission Characters	81
5.4	Word encoding and decoding	82
5.5	Ordered Sets	82
5.5.1	General	82
5.5.2	Frame delimiters	83
5.5.3	Primitive Signals	83
5.5.3.1	Introduction	83
5.5.3.2	Idle	83
5.5.3.3	Receiver_Ready (R_RDY)	83
5.5.3.4	Virtual Circuit Ready (VC_RDY)	85
5.5.3.5	BB_SCs	85
5.5.3.6	BB_SCr	86
5.5.3.7	SYNx, SYNy, SYNz	86
5.5.3.8	ARByx, ARB(val)See [2]	86
5.5.3.9	CLS	86
5.5.3.10	DHD	86
5.5.3.11	MRKtx	86
5.5.3.12	OPNyx	86
5.5.3.13	OPNyy	86
5.5.3.14	OPNyr	86
5.5.3.15	DHD	86
5.5.4	Primitive Sequences	86
5.5.4.1	Introduction	86
5.5.4.2	Not_Operational (NOS)	87
6	FC-1 Receiver and Transmitter State Diagrams	88
6.1	Receiver	88
6.1.1	Introduction	88
6.1.2	State Diagram Overview	88
6.1.3	Operational and Not Operational conditions	88
6.1.4	Word Synchronization Procedure	89
6.1.4.1	Bit Synchronization	89
6.1.4.2	Transmission Word synchronization	90
6.1.4.2.1	Introduction	90

6.1.4.2.2	Achieving Word Synchronization	90
6.1.4.2.3	Word alignment methods	90
6.1.4.2.3.1	Continuous-mode alignment	90
6.1.4.2.3.2	Explicit-mode alignment	90
6.1.5	Loss of Word Synchronization	90
6.1.5.1	Introduction	90
6.1.5.2	Detection of loss of Signal	91
6.1.5.3	Detection of an invalid Transmission Word	91
6.1.6	State transitions	91
6.1.6.1	Default State	91
6.1.6.2	State A (Loss of Synchronization)	91
6.1.6.3	State B (Word Synchronization Acquired)	91
6.1.6.3.1	State B.1 (No Invalid Transmission Word Detected State)	92
6.1.6.3.2	State B.2 (First Invalid Transmission Word Detected State)	92
6.1.6.3.3	State B.3 (Second Invalid Transmission Word Detected State)	92
6.1.6.3.4	State B.4 (Third Invalid Transmission Word Detection State)	92
6.1.6.4	State C (Reset)	92
6.2	Transmitter	93
6.2.1	State Diagram	93
6.2.2	Operational condition	93
6.2.3	State Transitions	93
6.2.3.1	Not Enabled State	93
6.2.3.2	Working State	94
6.2.3.3	Failure State	94
7 FC_Port state machine		95
7.1	State diagram	95
7.2	Active State (AC)	95
7.3	Link Recovery	95
7.3.1	Link recovery hierarchy	95
7.3.2	LR Transmit State (LR1)	95
7.3.2.1	General (while in the LR1 state)	95
7.3.2.2	Class 1 (while in the LR1 state)	95
7.3.2.3	Class 2 and Class 3 (while in the LR1 state)	96
7.3.2.4	Class 4 (while in the LR1 state)	97
7.3.2.5	Class 6 (while in the LR1 state)	97
7.3.3	LR Receive State (LR2)	97
7.3.3.1	General (while in the LR2 state)	97
7.3.3.2	Class 1 (while in the LR2 state)	97
7.3.3.3	Class 4 (while in the LR2 state)	97
7.3.3.4	Class 6 (while in the LR2 state)	98
7.3.4	LRR Receive State (LR3)	98
7.3.4.1	General (while in the LR3 state)	98
7.3.4.2	Class 1 and Class 6 behavior (while in the LR3 state)	98
7.3.4.3	Class 4 behavior (while in the LR2 state)	98
7.4	Link Failure	99
7.4.1	NOS Receive State (LF1)	99
7.4.1.1	General	99
7.4.1.2	Class 4 behavior	99
7.4.2	NOS Transmit State (LF2)	99
7.4.2.1	General	99
7.4.2.2	Class 4 behavior	99
7.5	Offline	99
7.5.1	General	99
7.5.2	OLS Transmit State (OL1)	100
7.5.2.1	Actions applicable to all classes	100
7.5.2.2	Class 4 behavior	100
7.5.3	OLS Receive State (OL2)	100

7.5.3.1	General	100
7.5.3.2	Class 4 behavior	100
7.5.4	Wait for OLS State (OL3)	100
7.5.4.1	Actions applicable to all classes	100
7.5.4.2	Class 4 behavior	101
7.6	Primitive Sequence protocols	101
7.6.1	Functions	101
7.6.2	Link Initialization protocol	101
7.6.3	Link Reset protocol	101
7.6.4	Link Failure protocol	101
7.6.5	Online-to-offline protocol	102
8	Frame formats	103
8.1	General frame format	103
8.2	Frame transmission	103
8.3	Start-of-Frame (SOF) delimiter	103
8.3.1	Introduction	103
8.3.2	SOF Connect Class 1 or 6 (SOF _{c1})	104
8.3.3	SOF Circuit Activate Class 4 (SOF _{c4})	104
8.3.4	SOF Initiate (SOF _{ix})	104
8.3.4.1	Applicability	104
8.3.4.2	SOF Initiate Class 1 or 6 (SOF _{i1})	104
8.3.4.3	SOF Initiate Class 2 (SOF _{i2})	104
8.3.4.4	SOF Initiate Class 3 (SOF _{i3})	104
8.3.4.5	SOF Initiate Class 4 (SOF _{i4})	104
8.3.5	SOF Normal (SOF _{nx})	104
8.3.5.1	Applicability	104
8.3.5.2	SOF Normal Class 1 or 6 (SOF _{n1})	104
8.3.5.3	SOF Normal Class 2 (SOF _{n2})	104
8.3.5.4	SOF Normal Class 3 (SOF _{n3})	104
8.3.5.5	SOF Normal Class 4 (SOF _{n4})	105
8.3.6	SOF Fabric (SOF _f)	105
8.4	Frame_Header	105
8.5	Data Field	105
8.6	CRC	105
8.7	End-of-Frame (EOF) delimiter	106
8.7.1	Introduction	106
8.7.2	Valid frame content	106
8.7.2.1	EOF Normal (EOF _n)	106
8.7.2.2	EOF Terminate (EOF _t)	106
8.7.2.3	EOF Disconnect Terminate (EOF _{dt}) (Class 1 or Class 6)	106
8.7.2.4	EOF Deactivate Terminate (EOF _{dt}) (Class 4)	107
8.7.2.5	EOF Remove Terminate (EOF _{rt})	107
8.7.3	Invalid frame content	107
8.7.3.1	General	107
8.7.3.2	End of Frame Abort (EOF _a)	107
8.7.3.3	EOF Disconnect Terminate Invalid (EOF _{dti}) (Class 1 and Class 6)	108
8.7.3.4	EOF Deactivate Terminate Invalid (EOF _{dti}) (Class 4)	108
8.7.3.5	EOF Remove Terminate Invalid (EOF _{rti})	108
8.7.3.6	EOF Invalid (EOF _{ni})	108
8.8	Frame field order	108
8.9	Frame reception	110
8.9.1	Rules	110
8.9.2	Frame validity	110
8.9.3	Invalid frame processing	110
9	Frame_Header	111
9.1	Introduction	111

9.2	Identification	111
9.2.1	Frame identification	111
9.2.2	Sequence identification	111
9.3	Routing Control (R_CTL)	112
9.3.1	Introduction	112
9.3.2	ROUTING Field	112
9.3.3	INFORMATION Field	112
9.4	Address identifiers (D_ID, S_ID)	114
9.4.1	General	114
9.4.2	Reserved address identifiers	114
9.4.3	Destination_ID (D_ID)	114
9.4.4	Source_ID (S_ID)	114
9.5	Class Specific Control (CS_CTL)/Priority	115
9.5.1	Introduction	115
9.5.2	CS_CTL	115
9.5.2.1	General	115
9.5.2.2	Class 1 and Class 6	115
9.5.2.3	Class 2	116
9.5.2.4	Class 3	116
9.5.2.5	Class 4	117
9.5.3	Priority	118
9.5.3.1	Introduction	118
9.5.3.2	Class 1 and Class 6	118
9.5.3.3	Class 2 and Class 3	119
9.5.3.4	Class 4	119
9.6	Data structure type (TYPE)	120
9.7	Frame Control (F_CTL)	122
9.7.1	Introduction	122
9.7.2	Exchange Context	122
9.7.3	Sequence Context	124
9.7.4	First_Sequence	125
9.7.5	Last_Sequence	125
9.7.6	End_Sequence	125
9.7.7	End_Connection (E_C) (Class 1 or 6) or Deactivate Class 4 circuit	125
9.7.8	CS_CTL/Priority Enable	125
9.7.9	Sequence Initiative	126
9.7.10	ACK_Form	126
9.7.11	Retransmitted Sequence	126
9.7.12	Unidirectional Transmit or Remove_Connection	126
9.7.13	Continue Sequence Condition	127
9.7.14	Abort Sequence Condition	127
9.7.15	Relative offset present	128
9.7.16	Exchange reassembly	129
9.7.17	Fill Data Bytes	129
9.7.18	F_CTL bits on Data frames	129
9.7.19	F_CTL bits on Link_Control frames	130
9.8	Sequence_ID (SEQ_ID)	132
9.9	Data Field Control (DF_CTL)	132
9.10	Sequence count (SEQ_CNT)	133
9.11	Originator Exchange_ID (OX_ID)	133
9.12	Responder Exchange_ID (RX_ID)	134
9.13	Parameter	134
10	Optional headers	135
10.1	Introduction	135
10.2	ESP_Header	136
10.3	Network_Header	138

10.4	Association_Header	139
10.4.1	Introduction	139
10.4.2	Process_Associators	140
10.4.2.1	Originator and Responder Process_Associators	140
10.4.2.2	Multicast Process_Associator	140
10.4.2.3	Operation_Associators	141
10.5	Device_Header	141
11	Data frames and responses	142
11.1	Data frames	142
11.1.1	Introduction	142
11.1.2	Frame Delimiters	142
11.1.3	Addressing	142
11.1.4	Data Field	143
11.1.5	Payload size	143
11.1.6	Responses	143
11.1.6.1	R_RDY response	143
11.1.6.2	Data frame responses	143
11.1.6.2.1	Introduction	143
11.1.6.2.2	ACK frames - successful Data frame delivery	143
11.1.6.3	Link_Response frames - Unsuccessful Data frame delivery	144
11.2	Link_Control Frames	144
11.2.1	Introduction	144
11.2.2	Link_Continue function	146
11.2.2.1	Introduction	146
11.2.2.2	Receiver Ready (R_RDY)	146
11.2.2.3	Acknowledge (ACK)	147
11.2.2.3.1	General	147
11.2.2.3.2	ACK_1	148
11.2.2.3.3	ACK_0	148
11.2.2.3.4	Header definition for all ACK forms	148
11.2.2.3.4.1	Addressing	148
11.2.2.3.4.2	F_CTL	148
11.2.2.3.4.3	SEQ_ID	148
11.2.2.3.4.4	SEQ_CNT	148
11.2.2.3.4.5	Parameter field	149
11.2.2.3.5	Responses	149
11.2.3	Link_Response	149
11.2.3.1	Introduction	149
11.2.3.2	Fabric Busy (F_BSY)	149
11.2.3.2.1	Description	149
11.2.3.2.2	Responses	150
11.2.3.3	N_Port Busy (P_BSY)	150
11.2.3.3.1	Description	150
11.2.3.3.2	Responses	152
11.2.3.4	Reject (P_RJT, F_RJT)	152
11.2.3.4.1	Introduction	152
11.2.3.4.2	Class 4	153
11.2.3.4.3	Parameter field	153
11.2.3.4.3.1	Reject Code format	153
11.2.3.4.3.2	Invalid D_ID	156
11.2.3.4.3.3	Invalid S_ID	156
11.2.3.4.3.4	Nx_Port not available, temporary	156
11.2.3.4.3.5	Nx_Port not available, permanent	156
11.2.3.4.3.6	Class not supported	156
11.2.3.4.3.7	Delimiter usage error	156
11.2.3.4.3.8	TYPE not supported	157

11.2.3.4.3.9	Invalid Link_Control	157
11.2.3.4.3.10	Invalid R_CTL field	157
11.2.3.4.3.11	Invalid F_CTL field	157
11.2.3.4.3.12	Invalid OX_ID	157
11.2.3.4.3.13	Invalid RX_ID	157
11.2.3.4.3.14	Invalid SEQ_ID	157
11.2.3.4.3.15	Invalid DF_CTL	157
11.2.3.4.3.16	Invalid SEQ_CNT	157
11.2.3.4.3.17	Invalid Parameter field	157
11.2.3.4.3.18	Exchange Error	157
11.2.3.4.3.19	Protocol Error	157
11.2.3.4.3.20	Incorrect length	157
11.2.3.4.3.21	Unexpected ACK	158
11.2.3.4.3.22	Class of service not supported by entity at hex 'FF FF FE'	158
11.2.3.4.3.23	Login Required	158
11.2.3.4.3.24	Excessive Sequences attempted	158
11.2.3.4.3.25	Unable to Establish Exchange	158
11.2.3.4.3.26	Fabric path not available	158
11.2.3.4.3.27	Invalid VC_ID (Class 4)	158
11.2.3.4.3.28	Invalid CS_CTL Field	158
11.2.3.4.3.29	Insufficient resources for VC (Class 4)	158
11.2.3.4.3.30	Invalid class of service	158
11.2.3.4.3.31	Preemption request rejected	158
11.2.3.4.3.32	Preemption not enabled	158
11.2.3.4.3.33	Multicast error	159
11.2.3.4.3.34	Multicast error terminate	159
11.2.3.4.3.35	Vendor Specific Reject	159
11.2.3.4.3.36	Responses	159
11.2.4	Link_Control commands	159
11.2.4.1	Introduction	159
11.2.4.2	Link Credit Reset (LCR)	159
11.2.4.2.1	Description	159
11.2.4.2.2	Protocol	160
11.2.4.2.3	Request Sequence	160
11.2.4.2.4	Responses	160
11.2.4.3	End (END)	160
11.2.4.3.1	Description	160
11.2.4.3.2	Protocol	161
11.2.4.3.3	Request Sequence	161
11.2.4.3.4	Reply Sequence	161
11.3	ACK generation assistance	161
11.3.1	Introduction	161
11.3.2	N_Port Login	161
11.3.2.1	Capability Indicator	161
11.3.3	Applicability	161
11.3.4	F_CTL bits	161
11.3.5	Login rules	161
11.3.6	ACK_Form errors	162
12	Link Services	163
12.1	Sequence and Exchange management	163
12.2	Basic Link Service commands	163
12.2.1	Introduction	163
12.2.2	Abort Sequence (ABTS)	164
12.2.2.1	Overview	164
12.2.2.2	Aborting Sequences using ABTS	165
12.2.2.2.1	Introduction	165

12.2.2.2.2	ABTS Initiator	165
12.2.2.2.3	ABTS Recipient	165
12.2.2.2.4	Recovery Qualifier	166
12.2.2.2.5	Protocol	166
12.2.2.2.6	Request Sequence	166
12.2.2.2.7	Reply Sequence	167
12.2.2.3	Aborting Exchanges using ABTS	167
12.2.2.3.1	Introduction	167
12.2.2.3.2	ABTS sent by the last Sequence Initiator in an open Sequence	168
12.2.2.3.3	ABTS sent by the last Sequence Initiator in a new Sequence	168
12.2.2.3.4	ABTS sent in an open or new Sequence	168
12.2.2.3.5	ABTS by the last Sequence Recipient	168
12.2.2.3.6	Request Sequence	168
12.2.2.3.7	Reply Sequence	169
12.2.3	Basic Accept (BA_ACC)	170
12.2.3.1	Description	170
12.2.3.2	Protocol	170
12.2.3.3	Request Sequence	170
12.2.3.4	Reply Sequence	170
12.2.4	Basic Reject (BA_RJT)	170
12.2.4.1	Description	170
12.2.4.2	Protocol	170
12.2.4.3	Request Sequence	170
12.2.4.4	Reply Sequence	171
12.2.5	No Operation (NOP)	172
12.2.5.1	Description	172
12.2.5.2	Protocol	172
12.2.5.3	Request Sequence	172
12.2.5.4	Reply Sequence	172
12.2.6	Remove Connection (RMC)	172
12.2.6.1	Description	172
12.2.6.2	Protocol	173
12.2.6.3	Request Sequence	173
12.2.6.4	Reply Sequence	173
12.2.7	Dedicated connection preempted (PRMT)	173
12.2.7.1	Description	173
12.2.7.2	Protocol	173
12.2.7.3	Request Sequence	173
12.2.7.4	Reply Sequence	173
12.3	Extended Link Services	173
12.3.1	Introduction	173
12.3.2	Extended Link Service requests	174
12.3.2.1	Introduction	174
12.3.2.2	Abort Exchange (ABTX)	178
12.3.2.2.1	Description	178
12.3.2.2.2	Protocol	179
12.3.2.2.3	Request Sequence	179
12.3.2.2.4	Reply Sequence	180
12.3.2.3	Advise Credit (ADVC)	180
12.3.2.3.1	Description	180
12.3.2.3.2	Protocol	180
12.3.2.3.3	Request Sequence	180
12.3.2.3.4	Reply Sequence	181
12.3.2.4	Echo (ECHO)	182
12.3.2.4.1	Description	182
12.3.2.4.2	Protocol	182
12.3.2.4.3	Request Sequence	183

12.3.2.4.4	Reply Sequence	183
12.3.2.5	Estimate Credit (ESTC)	183
12.3.2.5.1	Description	183
12.3.2.5.2	Protocol	183
12.3.2.5.3	Request Sequence	184
12.3.2.5.4	Reply Sequence	184
12.3.2.6	Establish Streaming (ESTS)	184
12.3.2.6.1	Description	184
12.3.2.6.2	Protocol	184
12.3.2.6.3	Request Sequence	184
12.3.2.6.4	Reply Sequence	185
12.3.2.7	Login (FLOGI/PLOGI)	186
12.3.2.7.1	Description	186
12.3.2.7.2	Protocol	186
12.3.2.7.3	Request Sequence	186
12.3.2.7.4	Reply Sequence	186
12.3.2.8	Logout (LOGO)	186
12.3.2.8.1	Description	186
12.3.2.8.2	Protocol	187
12.3.2.8.3	Request Sequence	187
12.3.2.8.4	Reply Sequence	187
12.3.2.9	Read Connection Status (RCS)	187
12.3.2.9.1	Description	187
12.3.2.9.2	Protocol	187
12.3.2.9.3	Request Sequence	188
12.3.2.9.4	Reply Sequence	188
12.3.2.10	Read Exchange Status Block (RES)	189
12.3.2.10.1	Description	189
12.3.2.10.2	Protocol	190
12.3.2.10.3	Request Sequence	190
12.3.2.10.4	Reply Sequence	190
12.3.2.11	Read Link Error Status Block (RLS)	191
12.3.2.11.1	Description	191
12.3.2.11.2	Protocol	191
12.3.2.11.3	Request Sequence	191
12.3.2.11.4	Reply Sequence	191
12.3.2.12	Read Sequence Status Block (RSS)	191
12.3.2.12.1	Description	191
12.3.2.12.2	Protocol	192
12.3.2.12.3	Request Sequence	192
12.3.2.12.4	Reply Sequence	192
12.3.2.13	Read Timeout Value (RTV)	193
12.3.2.13.1	Description	193
12.3.2.13.2	Protocol	193
12.3.2.13.3	Request Sequence	193
12.3.2.13.4	Reply Sequence	193
12.3.2.14	Reinstate Recovery Qualifier (RRQ)	194
12.3.2.14.1	Description	194
12.3.2.14.2	Protocol	194
12.3.2.14.3	Request Sequence	194
12.3.2.14.4	Reply Sequence	195
12.3.2.15	Request Sequence Initiative (RSI)	195
12.3.2.15.1	Description	195
12.3.2.15.2	Protocol	196
12.3.2.15.3	Request Sequence	196
12.3.2.15.4	Reply Sequence	196
12.3.2.16	Test (TEST)	196

12.3.2.16.1	Description	196
12.3.2.16.2	Protocol	197
12.3.2.16.3	Request Sequence	197
12.3.2.16.4	Reply Sequence	197
12.3.2.17	Report node Capability Information (RNC)	197
12.3.2.18	Fabric Address Notification (FAN)	197
12.3.2.18.1	Description	197
12.3.2.18.2	Protocol	198
12.3.2.18.3	Request Sequence	198
12.3.2.18.4	Reply Sequence	198
12.3.2.19	Loop Initialize (LINIT)	198
12.3.2.19.1	Description	198
12.3.2.19.2	Protocol	198
12.3.2.19.3	Request Sequence	198
12.3.2.19.4	Reply Sequence	199
12.3.2.20	Loop Port Control (LPC) (Obsolete)	200
12.3.2.21	Loop Status (LSTS)	200
12.3.2.21.1	Description	200
12.3.2.21.2	Protocol	200
12.3.2.21.3	Request Sequence	200
12.3.2.21.4	Reply Sequence	200
12.3.2.22	Registered State Change Notification (RSCN)	202
12.3.2.22.1	Introduction	202
12.3.2.22.2	RSCNs issued by the Fabric Controller	203
12.3.2.22.3	RSCN issued by the affected Nx_Port	203
12.3.2.22.4	RSCN initiative	203
12.3.2.22.5	RSCN registration	203
12.3.2.22.6	Protocol	203
12.3.2.22.7	Request Sequence	204
12.3.2.22.8	Reply Sequence	205
12.3.2.23	State Change Registration (SCR)	206
12.3.2.23.1	Description	206
12.3.2.23.2	Protocol	206
12.3.2.23.3	Request Sequence	206
12.3.2.23.4	Reply Sequence	207
12.3.2.24	Process login (PRLI)	207
12.3.2.24.1	Introduction	207
12.3.2.24.2	Protocol	207
12.3.2.24.3	Request Sequence	207
12.3.2.24.4	Reply Sequence	209
12.3.2.25	Process logout (PRLO)	211
12.3.2.25.1	Description	211
12.3.2.25.2	Protocol	211
12.3.2.25.3	Request Sequence	212
12.3.2.25.4	Reply sequence	213
12.3.2.26	State change notification (SCN)	216
12.3.2.27	Test Process Login State (TPLS)	216
12.3.2.27.1	Description	216
12.3.2.27.2	Protocol	216
12.3.2.27.3	Request Sequence	216
12.3.2.27.4	Reply sequence	218
12.3.2.28	Fibre Channel Address Resolution Protocol Request (FARP_REQ)	220
12.3.2.28.1	Description	220
12.3.2.28.2	Protocol	220
12.3.2.28.3	Request Sequence	220
12.3.2.28.4	Reply Sequence	224
12.3.2.29	Fibre Channel Address Resolution Protocol Reply (FARP_REPLY)	224

12.3.2.29.1	Description	224
12.3.2.29.2	Protocol	225
12.3.2.29.3	Request Sequence	225
12.3.2.29.4	Reply Sequence	227
12.3.2.30	Request Node Identification Data (RNID)	227
12.3.2.30.1	Introduction	227
12.3.2.30.2	Protocol	228
12.3.2.30.3	Request Sequence	228
12.3.2.30.4	Reply Sequence:	229
12.3.2.31	Registered Link Incident Report (RLIR)	234
12.3.2.31.1	Description	234
12.3.2.31.2	Link Incident reporting procedure	234
12.3.2.31.3	Protocol	235
12.3.2.31.4	Request Sequence	235
12.3.2.31.5	Reply Sequence	241
12.3.2.32	Link Incident Record Registration (LIRR)	241
12.3.2.32.1	Description	241
12.3.2.32.2	Registration for Link Incident Records	241
12.3.2.32.3	Responsibilities of Valid-Registered Recipients	242
12.3.2.32.4	Protocol	242
12.3.2.32.5	Request Sequence	242
12.3.2.32.6	Reply Sequence	243
12.3.2.33	Get Alias_ID (GAID)	244
12.3.2.33.1	Description	244
12.3.2.33.2	Protocol	244
12.3.2.33.3	Request Sequence	244
12.3.2.33.4	Reply Sequence	245
12.3.2.34	Fabric Activate Alias_ID (FACT)	245
12.3.2.34.1	Description	245
12.3.2.34.2	Protocol	245
12.3.2.34.3	Request Sequence	245
12.3.2.34.4	Reply Sequence	246
12.3.2.35	Fabric Deactivate Alias_ID (FDACT)	246
12.3.2.35.1	Description	246
12.3.2.35.2	Protocol	246
12.3.2.35.3	Request Sequence	246
12.3.2.35.4	Reply Sequence	247
12.3.2.36	N_Port Activate Alias_ID (NACT)	247
12.3.2.36.1	Description	247
12.3.2.36.2	Protocol	247
12.3.2.36.3	Reply Sequence	248
12.3.2.36.4	Reply Sequence	248
12.3.2.37	N_Port Deactivate Alias_ID (NDACT)	249
12.3.2.37.1	Description	249
12.3.2.37.2	Protocol	249
12.3.2.37.3	Request Sequence	249
12.3.2.37.4	Reply Sequence	249
12.3.2.38	Quality of Service Request (QoSR)	250
12.3.2.38.1	Description	250
12.3.2.38.2	Protocol	250
12.3.2.38.3	Request Sequence	250
12.3.2.38.4	Reply Sequence	252
12.3.2.39	Read Virtual Circuit Status (RVCS)	254
12.3.2.39.1	Description	254
12.3.2.39.2	Protocol	254
12.3.2.39.3	Request Sequence	254
12.3.2.39.4	Reply Sequence	255

12.3.2.40 Discover N_Port/Service Parameters (PDISC)	255
12.3.2.40.1 Description	255
12.3.2.40.2 Protocol	255
12.3.2.40.3 Request Sequence	255
12.3.2.40.4 Reply Sequence	256
12.3.2.41 Discover F_Port Service Parameters (FDISC)	256
12.3.2.41.1 Description	256
12.3.2.41.2 Protocol	256
12.3.2.41.3 Request Sequence	256
12.3.2.41.4 Reply Sequence	256
12.3.2.42 Discover Address (ADISC)	256
12.3.2.42.1 Description	256
12.3.2.42.2 Protocol	256
12.3.2.42.3 Request Sequence	257
12.3.2.42.4 Reply Sequence	257
12.3.2.43 Third Party Process Logout (TPRLO)	258
12.3.2.43.1 Description	258
12.3.2.43.2 Protocol	258
12.3.2.43.3 Request Sequence	258
12.3.2.44 Clock Synchronization Request (CSR)	260
12.3.2.44.1 Description	260
12.3.2.44.2 Protocol	260
12.3.2.44.3 Request Sequence	260
12.3.2.44.4 Reply Sequence	260
12.3.2.45 Clock Synchronization Update (CSU)	260
12.3.2.45.1 Description	260
12.3.2.45.2 Protocol	261
12.3.2.45.3 Request Sequence	261
12.3.2.45.4 Reply Sequence	261
12.3.2.46 Login Control List Management (LCLM)	261
12.3.2.46.1 Description	261
12.3.2.46.2 Protocol	263
12.3.2.46.3 Request Sequence	263
12.3.2.46.4 Reply sequence	264
12.3.2.47 Read Port Status Block (RPS)	266
12.3.2.47.1 Description	266
12.3.2.47.2 Protocol	266
12.3.2.47.3 Request Sequence	266
12.3.2.47.4 Reply Sequence	267
12.3.2.48 Read Port List (RPL)	270
12.3.2.48.1 Description	270
12.3.2.48.2 Protocol	270
12.3.2.48.3 Request Sequence	270
12.3.2.48.4 Reply Sequence	271
12.3.2.49 Report Port Buffer Conditions (RPBC)	271
12.3.2.49.1 Description	271
12.3.2.49.2 Protocol	272
12.3.2.49.3 Request Sequence	272
12.3.2.49.4 Reply Sequence	272
12.3.2.50 Report node FC-4 Types (RNFT)	273
12.3.2.50.1 Description	273
12.3.2.50.2 Protocol	273
12.3.2.50.3 Request Sequence	273
12.3.2.50.4 Reply Sequence	274
12.3.2.51 Scan Remote Loop (SRL)	275
12.3.2.51.1 Description	275
12.3.2.51.2 Protocol	275

12.3.2.51.3	Request Sequence	275
12.3.2.51.4	Reply Sequence	276
12.3.2.52	Set Bit-error Reporting Parameters (SBRP)	276
12.3.2.52.1	Description	276
12.3.2.52.2	Protocol	277
12.3.2.52.3	Request Sequence	277
12.3.2.52.4	Reply Sequence	278
12.3.2.53	Report Port Speed Capabilities (RPSC)	279
12.3.2.53.1	Description	279
12.3.2.53.2	Protocol	279
12.3.2.53.3	Request Sequence	279
12.3.2.53.4	Reply Sequence	280
12.3.2.54	Read Exchange Concise (REC)	280
12.3.2.54.1	Description	280
12.3.2.54.2	Protocol	281
12.3.2.54.3	Request Sequence	281
12.3.2.54.4	Reply Sequence	281
12.3.3	Extended Link Service Reply Sequences	282
12.3.3.1	Overview	282
12.3.3.2	LS_ACC	282
12.3.3.3	Reply Sequence	283
12.3.3.4	Link Service Reject (LS_RJT)	283
12.3.3.4.1	Description	283
12.3.3.4.2	Payload	283
12.3.3.4.3	Reply Sequence	285
12.4	FC-4 Link Service	286
13	Classes of service	288
13.1	Introduction	288
13.2	Class 1 - Dedicated connection	288
13.2.1	Function	288
13.2.2	Rules	289
13.2.3	Delimiters	290
13.2.4	Frame size	290
13.2.5	Flow control	290
13.2.6	Stacked connect-requests	290
13.3	Class 2 - Multiplex	290
13.3.1	Function	290
13.3.2	Rules	291
13.3.3	Delimiters	292
13.3.4	Frame size	292
13.3.5	Flow control	292
13.4	Class 3 - Datagram	292
13.4.1	Function	292
13.4.2	Rules	293
13.4.3	Delimiters	293
13.4.4	Frame size	294
13.4.5	Flow control	294
13.4.6	Sequence integrity	294
13.5	Intermix	294
13.5.1	Introduction	294
13.5.2	Rules	294
13.5.3	Frame size	295
13.5.4	Flow control	295
13.6	Class 4 - Fractional	295
13.6.1	Function	295
13.6.2	Procedures	296

13.6.3	Login	297
13.6.4	Circuit Setup	297
13.6.5	Circuit Activation	297
13.6.6	Circuit Deactivation	297
13.6.7	Circuit Removal	298
13.6.8	Rules	298
13.6.9	Interleaving other classes of service in Class 4	299
13.6.10	Class 4 delimiters	299
13.6.11	Frame size	300
13.6.12	End-to-end flow control	300
13.6.13	Buffer-to-buffer flow control	300
13.7	Class 6 - Connected Multicast	301
13.7.1	Function	301
13.7.2	Rules	301
13.7.3	Delimiters	303
13.7.4	Frame size	303
13.7.5	Flow control	303
13.7.6	Stacked Connect-requests	303
14	Name Identifier Formats	304
14.1	Introduction	304
14.2	IEEE 48-bit address	304
14.3	IEEE extended	305
14.4	Locally assigned	305
14.5	32-bit IP address	306
14.6	IEEE registered	306
14.7	IEEE registered extended	307
14.8	Other uses of IEEE registered Company_ID	307
14.9	EUI-64 Mapped	307
14.9.1	General	307
14.9.2	EUI-64 to WWN Mapping Rules	308
14.9.3	Encapsulated MAC-48 and EUI-48 translation	308
15	Login and Service Parameters	309
15.1	Introduction	309
15.2	Default Login values	309
15.3	Fabric Login	310
15.3.1	Introduction	310
15.3.2	Explicit Fabric Login	310
15.3.2.1	Introduction	310
15.3.2.2	Explicit Fabric Login Request	310
15.3.2.3	Responses to Explicit Fabric Login	310
15.3.2.4	Relogin with the Fabric	311
15.3.3	SOF delimiters	312
15.3.4	Frequency	312
15.3.5	Fabric Login completion - Originator	312
15.3.6	Fabric Login completion - Responder	312
15.4	N_Port Login	313
15.4.1	Introduction	313
15.4.2	Explicit N_Port Login	314
15.4.2.1	Introduction	314
15.4.2.2	N_Port Login - Fabric present	314
15.4.2.3	Responses to N_Port Login - Fabric present	314
15.4.2.4	N_Port Login - No Fabric present	315
15.4.2.5	Responses to N_Port Login - No Fabric present	315
15.4.3	SOF delimiters	316
15.4.4	Frequency	316
15.4.5	N_Port Login completion - Originator	316

15.4.6	N_Port Login completion - Responder	316
15.4.7	N_Port Login frame flow	316
15.5	Logout	317
15.5.1	Introduction	317
15.5.2	Explicit N_Port Logout	317
15.5.3	Implicit Logout	317
15.6	Service Parameters	317
15.6.1	ELS and LS_ACC Payload	317
15.6.2	Common Service Parameters	319
15.6.2.1	Applicability	319
15.6.2.2	Payload	320
15.6.2.3	Buffer-to-buffer Credit	321
15.6.2.4	Common Features	321
15.6.2.4.1	Continuously increasing relative offset	321
15.6.2.4.2	Clean Address	322
15.6.2.4.3	Random relative offset	322
15.6.2.4.4	Valid Vendor Version Level	322
15.6.2.4.5	Multiple N_Port_ID Assignment	322
15.6.2.4.6	N_Port/F_Port	322
15.6.2.4.7	BB_Credit Management	322
15.6.2.4.8	E_D_TOV Resolution	323
15.6.2.4.9	Multicast	323
15.6.2.4.10	Broadcast	323
15.6.2.4.11	Hunt Group	323
15.6.2.4.12	Query Buffer Conditions	323
15.6.2.4.13	Clock Synchronization Primitive Capable	323
15.6.2.4.14	R_T_TOV value	324
15.6.2.4.15	Dynamic Half Duplex Supported	324
15.6.2.4.16	SEQ_CNT	324
15.6.2.4.17	Payload Bit	324
15.6.2.5	BB_SC_N	325
15.6.2.6	Buffer-to-buffer Receive Data_Field size	325
15.6.2.7	Total Concurrent Sequences	325
15.6.2.8	Relative offset by category	325
15.6.2.9	R_A_TOV	325
15.6.2.10	E_D_TOV	325
15.6.3	N_Port_Name	326
15.6.4	Node_or Fabric_Name	326
15.6.5	Class Service Parameters	326
15.6.5.1	Applicability	326
15.6.5.2	Payload	328
15.6.5.3	Class validity	329
15.6.5.4	Service options	329
15.6.5.4.1	Introduction	329
15.6.5.4.2	Intermix Mode	329
15.6.5.4.3	Stacked Connect-requests	330
15.6.5.4.4	Sequential delivery	330
15.6.5.4.5	Priority/Preemption	331
15.6.5.4.6	Preference	331
15.6.5.4.6.1	Nx_Port	331
15.6.5.4.6.2	Fx_Port	332
15.6.5.4.7	DiffServ QoS	333
15.6.5.4.7.1	N_Port Login	333
15.6.5.4.7.2	F_Port Login	333
15.6.5.5	Initiator control	334
15.6.5.5.1	Introduction	334
15.6.5.5.2	Initial Process_Associator	334

15.6.5.5.3	ACK_0 capability	334
15.6.5.5.4	ACK generation assistance	335
15.6.5.5.5	Clock synchronization ELS capable	335
15.6.5.6	Recipient control	336
15.6.5.6.1	Introduction	336
15.6.5.6.2	ACK_0 capability	336
15.6.5.6.3	X_ID interlock	337
15.6.5.6.4	Error policy supported	337
15.6.5.6.5	Categories per Sequence	338
15.6.5.6.6	Clock synchronization ELS capable	338
15.6.5.7	Receive Data_Field Size	338
15.6.5.8	Concurrent Sequences	338
15.6.5.9	End-to-end Credit	339
15.6.5.10	Open Sequences per Exchange	340
15.6.5.11	CR_TOV	340
15.6.6	Vendor Version Level	340
15.6.7	Services Availability	340
15.6.7.1	Introduction	340
15.6.7.2	Multicast Server	340
15.6.7.3	Clock Synchronization Server	340
15.6.7.4	Security Key Distribution Server	340
15.6.7.5	Alias Server	340
15.6.7.6	Quality of Service Facilitator	341
15.6.7.7	Management Server	341
15.6.7.8	Time Server	341
15.6.7.9	Directory Server	341
15.6.8	Login Extension	341
15.6.8.1	General	341
15.6.8.2	Login Extension Data Length	341
15.6.8.3	Login Extension format	341
15.6.9	Clock Synchronization Quality of Service	342
15.6.9.1	N_Port Login	342
15.6.9.1.1	Applicability	342
15.6.9.1.2	CS_QoS_Request	343
15.6.9.1.3	CS_Accuracy (Mantissa and Exponent)	343
15.6.9.1.4	Clock Synchronization Implemented MSB	344
15.6.9.1.5	Clock Synchronization Implemented LSB	344
15.6.9.1.6	Clock Synchronization Update Period	344
15.6.9.2	Fabric Login	344
15.6.9.2.1	Applicability	344
15.6.9.2.2	CS_Transfer_Accuracy	345
15.6.9.2.3	Clock Synchronization Implemented MSB	345
15.6.9.2.4	Word 0, Bits 7 to 0 Clock Synchronization Implemented LSB	346
16	Process Login/Logout	347
16.1	Process Login	347
16.1.1	Introduction	347
16.1.2	PRLI/PRLO Relationships	349
16.1.2.1	Introduction	349
16.1.2.2	PA not supported	349
16.1.2.3	PA required by originator, supported by responder	349
16.1.2.4	PA required by responder, supported by originator	349
16.1.2.5	PA required by originator and responder	350
16.1.3	Mode of operation	350
16.1.3.1	Informative mode	350
16.1.3.2	Binding mode	350
16.1.4	Protocol	350

16.1.4.1	PA required by originator and responder	350
16.1.4.2	PA required by originator, supported by responder	351
16.1.4.3	PA supported by originator, required by responder	351
16.2	Process Logout	351
17	Exchange, Sequence and sequence count management	353
17.1	Introduction	353
17.1.1	Data frame transfer	353
17.1.2	Sequence	353
17.1.3	Streamed Sequences	353
17.1.4	SEQ_CNT	353
17.1.5	Exchange	353
17.1.6	Sequence Initiative	355
17.2	Applicability	355
17.3	Exchange rules	356
17.3.1	Exchange management	356
17.3.2	Exchange origination	356
17.3.3	Sequence delimiters	357
17.3.4	Sequence initiation	357
17.3.5	Sequence management	357
17.3.6	SEQ_CNT	358
17.3.7	Normal ACK processing	358
17.3.8	Normal Sequence completion	359
17.3.9	Detection of missing frames	360
17.3.10	Sequence errors - Class 1, 2, 4 and 6	361
17.3.10.1	Rules common to all Discard policies	361
17.3.10.2	Discard multiple Sequences Error Policy	362
17.3.10.3	Discard a single Sequence Error Policy	363
17.3.10.4	Process with infinite buffers Error Policy	363
17.3.11	Sequence errors - Class 3	364
17.3.11.1	Rules common to all discard policies	364
17.3.11.2	Process with infinite buffers Error Policy	364
17.3.12	Sequence Status Rules	364
17.3.13	Exchange termination	365
17.3.14	Exchange Status Rules	365
17.4	Exchange management	365
17.5	Exchange origination	365
17.5.1	Introduction	365
17.5.2	Exchange Originator	366
17.5.3	Exchange Responder	366
17.5.4	X_ID assignment	367
17.5.5	X_ID interlock	367
17.6	Sequence management	367
17.6.1	Open and active Sequences	367
17.6.2	Sequence_Qualifier management	368
17.6.3	Sequence initiative and termination	368
17.6.4	Transfer of Sequence Initiative	368
17.6.5	Sequence Termination	368
17.6.5.1	Introduction	368
17.6.5.2	Class 1 and Class 6	369
17.6.5.3	Class 2	369
17.6.5.4	Class 3	369
17.6.5.5	Class 4	369
17.6.5.6	Continue Sequence Condition	369
17.6.5.7	End_Sequence	369
17.7	Exchange termination	369
17.7.1	Normal termination	369

17.7.2	Abnormal termination	370
17.8	Status blocks	370
17.8.1	Exchange Status Block	370
17.8.2	Sequence Status Block	371

18 Flow control management- 374

18.1	Introduction	374
18.2	Physical flow control model for Classes 1, 2, 3 and 6	375
18.3	Credit and Credit_Count	376
18.3.1	Introduction	376
18.3.2	Credit_Count management	376
18.3.3	Management by increasing the Credit_Count	376
18.3.4	Management by decreasing the Credit_Count	376
18.3.5	Credit_Count types	376
18.3.6	Usage	376
18.4	End-to-end flow control	376
18.4.1	End-to-end management rules	376
18.4.2	Sequence Initiator	377
18.4.3	Sequence Recipient	378
18.4.3.1	General	378
18.4.3.2	ACK_0	378
18.4.3.3	ACK_1	378
18.4.3.4	Last ACK timeout	378
18.4.3.5	Streamed Sequences	379
18.4.4	EE_Credit	379
18.4.5	EE_Credit_CNT	379
18.4.6	EE_Credit management	379
18.4.7	End-to-end flow control model	380
18.4.8	End-to-end class dependency	380
18.4.8.1	End-to-end Credit allocation	380
18.4.8.2	EE_Credit_CNT management	380
18.4.9	EE_Credit recovery	381
18.4.10	Class 4	381
18.4.11	Procedure to estimate end-to-end Credit	382
18.4.11.1	Introduction	382
18.4.11.2	Procedure steps	382
18.4.11.2.1	General	382
18.4.11.2.2	Establish Streaming Sequence	383
18.4.11.2.3	Estimate Credit Sequence	384
18.4.11.2.4	Advise Credit Sequence	384
18.5	Buffer-to-buffer flow control	385
18.5.1	Introduction	385
18.5.2	Buffer-to-buffer management rules	385
18.5.3	BB_Credit	386
18.5.4	BB_Credit_Count	386
18.5.5	BB_Credit management	386
18.5.6	Buffer-to-buffer flow control model	386
18.5.7	Class dependent frame flow	386
18.5.8	R_RDY	386
18.5.9	BB_Credit_Count reset	387
18.5.10	Alternate buffer-to-buffer Credit management	391
18.5.11	BB_Credit Recovery	391
18.6	VC_RDY	392
18.7	BSY / RJT in flow control	393
18.8	LCR in flow control	393
18.9	Integrated Class 2 flow control	393
18.10	Intermix	393

18.11	Point-to-point topology	394
19	Segmentation and reassembly	398
19.1	Introduction	398
19.2	Sending end	398
19.2.1	Introduction	398
19.2.2	Relative offset space	398
19.2.3	Data block	398
19.2.4	Sequence	398
19.2.5	Relationship between Sequences	398
19.3	FC-2	399
19.3.1	Mechanisms	399
19.3.2	Relative offset	399
19.3.3	SEQ_CNT	399
19.4	Login	399
19.5	Segmentation rules	400
19.6	Reassembly rules	400
20	Connection management	402
20.1	Introduction	402
20.1.1	Establishing a Connection	402
20.1.2	Removing a Connection	402
20.1.3	Preempting a dedicated connection	402
20.1.4	Frame processing precedence	403
20.2	Applicability	403
20.3	Topology models	404
20.3.1	Introduction	404
20.3.2	Fabric model	404
20.3.3	Point-to-point model	404
20.4	Connect/disconnect rules	405
20.4.1	Connect-request rules	405
20.4.1.1	Source of connect-request	405
20.4.1.2	Destination of connect-request	406
20.4.2	Connection Rules	406
20.4.3	Remove Connection rules	407
20.5	Establishing a Connection	407
20.5.1	Introduction	407
20.5.2	Connection Initiator	407
20.5.3	Stacked connect-requests	409
20.5.4	Unidirectional dedicated connection	410
20.5.5	Destination of connect-request	410
20.6	Connected	411
20.7	Removing a Connection	411
20.7.1	Introduction	411
20.7.2	When to remove a Connection	411
20.7.3	End_Connection bit	411
20.7.4	EOFdt transmission	412
20.8	Connection Recovery	412
20.8.1	Introduction	412
20.8.2	Link timeout	412
20.8.3	Corrupted connect-request	412
20.9	Connection Preemption	413
20.9.1	Applicability	413
20.9.2	Topology Model	413
20.9.3	Rules for Preemption	413
20.9.3.1	Preemptor	413
20.9.3.2	Preempted Source	413
20.9.3.3	Preempted Destination(s)	414

20.9.3.4	Preemption Destination(s)	414
20.10	Establishing a Connection Using Preemption	414
20.10.1	Introduction	414
20.10.2	Connection Initiator	414
20.10.3	Preemption Destination	416
21	Error detection/recovery	417
21.1	Introduction	417
21.2	Timeouts	417
21.2.1	Timeout periods	417
21.2.1.1	General	417
21.2.1.2	R_T_TOV	417
21.2.1.3	E_D_TOV	417
21.2.1.4	R_A_TOV	418
21.2.2	Link Failure timeouts	418
21.2.3	Link timeout	418
21.2.4	Sequence timeout	419
21.2.4.1	Introduction	419
21.2.4.2	Classes 1, 2 and 6	419
21.2.4.3	Class 3	419
21.2.4.4	End-to-end Class 2 Credit loss	420
21.2.5	OLS transmit timeout	420
21.2.6	Timer Enhancements for Classes 1 and 6	420
21.2.6.1	Introduction	420
21.2.6.2	Applicability	421
21.2.6.3	Login	421
21.2.6.4	Value	421
21.2.6.5	Stacked Connect-request	421
21.2.6.6	Rules	421
21.2.6.6.1	Connection Initiator	421
21.2.6.6.2	Connection Recipient	422
21.2.6.6.3	Fabric	422
21.2.7	Bit-Error-Rate Thresholding	422
21.2.7.1	Introduction	422
21.2.7.2	Types of Link Errors Caused by Bit Errors	422
21.2.7.3	Error Bursts	422
21.2.7.4	Bit-Error-Rate-Thresholding Measurement	422
21.3	Link error detection	423
21.3.1	Link Failure	423
21.3.2	Code violations	423
21.3.3	Primitive Sequence protocol error	423
21.4	Link error recovery	423
21.5	Link recovery - secondary effects	423
21.5.1	Class 1 and Class 6	423
21.5.2	Class 2	424
21.5.3	Class 3	425
21.5.4	Class 4	425
21.6	Exchange Integrity	425
21.6.1	Applicability	425
21.6.2	Exchange management	425
21.6.3	Exchange Error Policies	425
21.6.3.1	Introduction	425
21.6.3.2	Discard multiple Sequences	425
21.6.3.3	Discard a single Sequence	426
21.6.3.4	Process with infinite buffering	426
21.6.3.5	Discard multiple Sequences with retransmission	426
21.6.4	Sequence integrity	426

21.6.5	Sequence error detection	426
21.6.6	X_ID processing	427
21.7	Sequence recovery	427
21.7.1	Introduction	427
21.7.2	Abnormal Sequence termination	427
21.7.2.1	Introduction	427
21.7.2.2	Abort Sequence Protocol	427
21.7.2.2.1	General Case	427
21.7.2.2.2	Special case - new Exchange	428
21.7.2.3	Class 1 or 6 Sequence retransmission	428
21.7.2.4	Recipient abnormal termination	430
21.7.2.5	End_Sequence	430
21.7.3	Stop Sequence Protocol	430
21.7.4	End-to-end Credit loss	430
21.8	Link Error Status Block	431
21.9	Detailed Error Detection/Actions	431
21.9.1	Errors detected	431
21.9.2	Actions by Initiator or Recipient	432
22	Hunt Group	435
22.1	Introduction	435
22.2	Function	435
22.3	Communication Model	435
22.4	Applicability	435
22.4.1	Classes	435
22.4.2	Class 4 Addressing	435
22.5	Formation	436
22.5.1	General	436
22.5.2	Registration/deregistration	436
22.5.3	Inquiry	437
22.5.4	HG_ID Removal	437
22.6	N_Port Login	437
22.7	Addressing Protocol	437
22.7.1	address identifier Protocol	437
22.7.2	Originator	437
22.7.3	Responder	437
22.7.4	Address Resolution	438
22.7.5	Class	438
22.7.6	Class 4	438
22.8	Resource management	438
22.8.1	Introduction	438
22.8.2	HG_ID use	439
22.8.3	N_Port_ID use	439
22.9	Rotary Group	439
23	Multicast	441
23.1	Applicability	441
23.2	Class 3 Multicast	441
23.2.1	Introduction	441
23.2.2	Registration and De-registration	441
23.2.3	Multicast Routing	441
23.2.4	Class 3 Multicast Rules	442
23.3	Class 6 Multicast	442
23.3.1	Introduction	442
23.3.2	Class 6 Multicast Routing	443
23.3.3	Class 6 Multicast Rules	443
23.3.4	Class 6 Multicast Server	444
23.3.5	Class 6 Multicast Recovery	444

23.4	Broadcast	444
23.5	Other	445
24	Alias_IDs	446
24.1	Introduction	446
24.2	Alias Server	446
24.3	Alias Service protocol	446
24.4	Alias_ID Routing	446
24.5	Function Flow	447
24.6	PA Considerations	447
24.6.1	Hunt Groups	447
24.6.2	Multicast Groups	447
24.6.3	Broadcast	448
25	Class 4 – Fractional	449
25.1	Introduction	449
25.2	Default N_Port Login service parameters	450
25.3	Communication Model	450
25.3.1	Introduction	450
25.3.2	In-order delivery	451
25.3.3	Guaranteed delivery	451
25.3.4	Fractional bandwidth management	452
25.3.5	Class 4 circuit	452
25.3.5.1	Introduction	452
25.3.5.2	Setup	453
25.3.5.3	Activation	456
25.3.5.4	Deactivation	457
25.3.5.5	Removal	458
25.4	Deactivation, removal contention and error recovery	460
25.4.1	CTI has requested removal of Class 4 circuit	460
25.4.2	CTR has requested removal of a Class 4 circuit	460
25.4.3	CTI has requested deactivation of a Class 4 circuit	460
25.4.4	CTR has requested deactivation of a Class 4 circuit	461
25.4.5	CTI has no outstanding removal or deactivation request	461
25.4.6	CTR has no outstanding removal or deactivation request	461
25.5	Ordered Sets	461
25.6	Quality of Service	463
26	Stacked Connect-request	465
26.1	Introduction	465
26.2	FC-2 Mechanisms	465
26.3	Applicability	465
26.4	Communications Model	465
26.5	Requirements	465
26.5.1	Connection Initiator	465
26.5.2	Connection Recipient	465
26.5.3	Fabric	466
26.6	Login	466
26.7	Stacked Connect-request Invoked	466
26.8	Timer	466
26.8.1	General	466
26.8.2	CR_TOV	466
26.8.3	E_D_TOV	466
26.9	Rules	466
26.9.1	Connection Initiator	466
26.9.2	Connection Recipient	466
26.9.3	Fabric	467
27	Clock synchronization service	468

27.1	Introduction	468
27.1.1	References	468
27.1.2	Applicability	468
27.1.3	Function	468
27.1.4	Assumptions	468
27.1.5	Clock Synchronization Quality of Service	468
27.2	ELS Command Service	468
27.2.1	ELS Commands	468
27.2.2	Fabric Topology	469
27.2.2.1	Model	469
27.2.2.2	Clock Synchronization Server Rules	469
27.2.2.3	Fabric Rules	469
27.2.2.4	Fabric Options	470
27.2.2.5	Client Rules	470
27.2.2.6	Client Options	470
27.2.3	Loop Topology	470
27.2.3.1	Model	470
27.2.3.2	L_Port Server Rules	470
27.2.3.3	L_Port Server Options	471
27.2.3.4	L_Port Client Rules	471
27.2.3.5	Client Options	471
27.2.4	Use of FC-FS Constructs	471
27.2.4.1	Login/Logout	471
27.2.4.2	Clock Synchronization Request (CSR) ELS	472
27.2.4.2.1	Command Payload	472
27.2.4.2.2	LS_ACC Payload	473
27.2.4.3	Clock Synchronization Update (CSU) ELS Command	474
27.3	Primitive Signal Service	475
27.3.1	Topology compatibility	475
27.3.2	Communication Model	475
27.3.3	Requirements	476
27.3.3.1	Introduction	476
27.3.3.2	Clock Synchronization Server Rules	478
27.3.3.3	Fabric Rules	479
27.3.3.4	Client Rules	479
28	Link Speed Negotiation	480
28.1	Speed Negotiation overview	480
28.2	Link physical architecture and requirements	481
28.3	Speed Negotiation requirements on L_Ports	481
28.4	State machines	482
28.5	Primitives	482
28.6	Speed Negotiation algorithm	482
28.6.1	Algorithm overview	482
28.6.2	Stage 1 - Wait_for_signal	485
28.6.3	Stage 2 - Negotiate_master and Watchdog timer	486
28.6.4	Stage 3 - Negotiate_follow	488
28.6.5	Stage 4 - Normal operation	490
28.6.6	Optional Stage 5 - Slow_wait	490
28.6.7	Timing requirements	492
Annex A	(informative) CRC generation and checking	494
A.1	Extract from FDDI	494
A.2	Frame check sequence (FCS)	494
A.3	Definitions	494
A.3.1	FCS generation equations	495
A.3.2	FCS checking	495
A.4	CRC generation	495

A.5	Transmit order of a word	496
A.6	CRC generation example for ACK_1 frame	496
Annex B	(informative) Data transfer protocols and examples	500
B.1	General	500
B.2	Frame level protocol	500
B.2.1	Class 1 and 6 frame level protocol	500
B.2.2	Class 2 frame level protocol	501
B.2.3	Class 3 Frame Level Protocol	503
B.3	Sequence level protocol example	505
B.4	Class 1 and 6 frame level protocol example	507
B.5	Class 2 frame level protocol example	508
B.6	Class 3 frame level protocol example	509
Annex C	(informative) Connection management applications	511
C.1	Example cases	511
C.1.1	Introduction	511
C.1.2	Case 1	511
C.1.3	Case 2	511
C.1.4	Case 3	512
C.1.5	Case 4	512
C.2	Ending sequence and Connection	512
Annex D	(informative) Out of order characteristics	514
D.1	Introduction	514
D.2	Out of order Data frame delivery	514
D.3	Out of order ACK transmission	515
Annex E	(informative) Link Error Status Block	516
E.1	Introduction	516
E.2	Link failure counters	516
E.3	Invalid Transmission Word	516
E.4	Invalid CRC Count	516
Annex F	(informative) Class 4 Examples	518
F.1	Introduction	518
F.2	Setup, activate and deactivate	518
F.3	Activation collision and removal	519
Annex G	(informative) Priority and Preemption	524
G.1	Overview	524
G.2	Preemption Process	524
G.3	Link Reset Protocol	526
Annex H	(informative) Clock Synchronization	527
H.1	Introduction	527
H.2	Discussion	527
H.2.1	Introduction	527
H.2.2	A Model of an NL_Port	527
H.2.3	Hardware-Assisted Clock Synchronization	528
H.2.4	A Point-to-Point System	528
H.2.4.1	Introduction	528
H.2.4.2	Discussion of Errors	531
H.2.4.2.1	Introduction	531
H.2.4.2.2	Client Oscillator Frequency Error	531
H.2.4.2.3	Link Propagation Delay Error	532
H.2.4.2.4	Unload Error	533
H.2.4.2.5	Load Error	534
H.2.4.2.6	R/T Clock Domain Error	535
H.2.4.2.7	Server Oscillator Error	536

H.2.4.3	Techniques for Reducing Deterministic Errors	536
H.2.4.3.1	A Fix for Differences in Oscillator Frequencies	536
H.2.4.3.2	A Fix for Link Propagation Delay Error	537
H.2.4.3.3	A Fix for Load Error	537
H.2.4.3.4	A Fix for Unload Error	540
H.2.4.4	Dealing With Non-Deterministic Error	540
H.2.4.5	Dealing With Non-Monotonicity	540
H.2.5	Fabric Considerations	541
H.2.5.1	Introduction	541
H.2.5.2	Discussion of Errors	542
H.2.5.2.1	Client Oscillator Frequency Error	542
H.2.5.2.2	Link Propagation Delay Error	544
H.2.5.2.3	Unload Error	544
H.2.5.2.4	Load Error	544
H.2.5.2.5	R/T Clock Domain Error	544
H.2.5.2.6	Server Oscillator Error	545
H.2.5.3	Fixes for Fabric Errors	545
H.2.6	Loop Considerations	545
H.2.6.1	Introduction	545
H.2.6.2	Discussion of Errors	545
H.2.6.3	Introduction	545
H.2.6.3.1	Node Delay	546
H.2.6.3.2	Client Oscillator Frequency Error	547
H.2.6.3.3	Link Propagation Delay Error	547
H.2.6.3.4	Unload Error	547
H.2.6.3.5	Load Error	547
H.2.6.3.6	R/T Clock Domain Error	547
H.2.6.3.7	Server Oscillator Error	547
H.2.6.4	Fixes for Loop Errors	547
H.3	An Example	547
Annex I (informative)	Speed Negotiation details	550
I.1	Scope	550
I.2	Basic assumptions	550
I.3	Supported configuration	551
I.4	Derivation of timing requirements and characteristics	551
I.4.1	Introduction and diagram conventions	551
I.4.2	Receiver cycle time, t_{rxcycl}	552
I.4.3	Master transmitter cycle time, t_{txcycl}	552
I.4.4	Speed stability time, t_{stbl}	552
I.4.5	Watchdog timer threshold, t_{fail}	552
I.4.6	Watchdog Timer test delay, t_{wddly}	553
I.4.7	Speed recording time, t_{ncycl}	553
I.4.8	Speed recording time initial value, t_{ncinit}	554
I.4.9	Parameters relating to the optional slow_wait stage:	555
I.4.9.1	Low processing load sleep time, t_{sleep}	555
I.4.9.2	Slow_wait cycle transmit cycle delay, t_{txdly}	555
I.4.9.3	Periodic sync search wake time, t_{wake}	555
I.4.10	Duration of disruption to single loops caused by connecting speed negotiating ports to hubs	556
I.4.10.1	Introduction	556
I.4.10.2	Maximum single disruption in Wait_for_signal stage	557
I.4.10.3	Maximum single disruption in Slow_wait stage	558
I.4.10.4	Maximum single disruption in Negotiate_master stage	558
I.4.10.5	Maximum single disruption in Negotiate_follow stage	559
I.4.10.6	Maximum disruption group - Wait_for_signal	559
I.4.10.7	Maximum disruption group - Slow_wait	560

I.4.10.8	Maximum disruption group - Negotiate_master	561
I.4.10.9	Maximum disruption group - Negotiate_follow	562
I.4.10.10	Maximum single disruption overall	562
I.4.10.11	Maximum disruption group overall	563
I.4.10.12	Summary of loop disruption	563
I.4.11	Algorithm convergence time:	564
I.5	Ports using separate PMD components	564
I.6	Implementation notes	566
Annex J (informative)	IEEE company_ID	567
Annex K (informative)	WWN-to-EUI-64 Mapping	568
K.1	Background	568
K.2	Solution	568
K.3	Case Study	569
Bibliography		571

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

Figure 1 - Fibre Channel Structure	57
Figure 2 - Node functional configuration	59
Figure 3 - FC-FS physical model	61
Figure 4 - Point-to-point topology	62
Figure 5 - Fabric topology	63
Figure 6 - Examples of the Arbitrated Loop Topology	64
Figure 7 - Informative general Fabric model	67
Figure 8 - FC-2 building block hierarchy	69
Figure 9 - Receiver State Diagram	89
Figure 10 - Transmitter State Diagram	93
Figure 11 - FC-2 frame format	103
Figure 12 - CIVC_ID and CRVC_ID management	118
Figure 13 - Frame structure when ESP_Header is not used	135
Figure 14 - Frame structure with ESP_Header and ESP_Trailer	136
Figure 15 - QoS Sequence flow	250
Figure 16 - Illustration of parameters	277
Figure 17 - LS_RJT format	283
Figure 18 - Class 4 circuit – Management	296
Figure 19 - Image/Group of Related Processes	347
Figure 20 - Image pairs	348
Figure 21 - Exchange - Sequence relationship	354
Figure 22 - Exchange origination	366
Figure 23 - Physical flow control model for Classes 1, 2, 3 and 6	375
Figure 24 - End-to-end flow control model	381
Figure 25 - Procedure to estimate end-to-end Credit	383
Figure 26 - Buffer-to-buffer flow control model	387
Figure 27 - Class 1 or 6/SOFc1 frame flow with delivery or non-delivery to the fabric	388
Figure 28 - Class 1 or 6/SOFc1 frame flow with delivery or non-delivery to an Nx_Port	388
Figure 29 - Buffer-to-buffer - Class 2 frame flow with delivery or non-delivery to a Fabric	389
Figure 30 - Buffer-to-buffer - Class 2 frame flow with delivery or non-delivery to an Nx_Port	390
Figure 31 - Buffer-to-buffer - Class 3 frame flow	391
Figure 32 - LCR frame flow and possible responses	394
Figure 33 - LCR flow control model	395
Figure 34 - Integrated Class 2 flow control	397
Figure 35 - Frame Flow Timers	421
Figure 36 - Link recovery hierarchy	424
Figure 37 - Relationship of a Non-Participating to a Participating Nx_Port and switch	440
Figure 38 - Class 3 Multicast Routing	442
Figure 39 - Class 6 Multicast Routing	443
Figure 40 - Function Flow	447
Figure 41 - Class 4 Circuit hierarchy	449
Figure 42 - Basic Class 4 circuit	451
Figure 43 - Class 4 circuits – Example	451
Figure 44 - Class 4 circuit - Port level state diagram	453
Figure 45 - Class 4 circuit – Setup	455
Figure 46 - Class 4 circuit – Activation	457
Figure 47 - Class 4 - Example of frame flow	457
Figure 48 - Class 4 circuit - CTI or CTR initiated	458
Figure 49 - Class 4 circuit - Fabric initiated deactivation	458
Figure 50 - Class 4 circuit - CTI or CTR initiated removal	459
Figure 51 - Class 4 circuit - Fabric initiated removal	459
Figure 52 - Class 4 circuit - Data frame responses	463
Figure 53 - ELS Clock Sync Model – Fabric	469
Figure 54 - ELS Clock Sync Model – Loop	471

Figure 55 - Clock Synchronization Data Distribution	476
Figure 56 - Synchronization Primitive Substitution for IDLE Primitives in Inter-frame Interval	476
Figure 57 - Clock Synchronization Primitive Ordered Sets	477
Figure 58 - Physical architecture of the Speed Negotiating Link	481
Figure 59 - Delay / Test operations	483
Figure 60 - Overview of the Speed Negotiation algorithm	484
Figure 61 - Wait_for_signal flowchart	485
Figure 62 - Negotiate_master and Watchdog timer flowchart	487
Figure 63 - Negotiate_follow flowchart	489
Figure 64 - Normal operation flowchart	490
Figure 65 - Slow_wait flowchart	491
Figure A.1 - CRC coverage and bit ordering	496
Figure A.2 - Word transmit order	496
Figure B.1 - Class 1 and 6 frame level protocol	501
Figure B.2 - Class 2 frame level protocol	503
Figure B.3 - Class 3 frame level protocol	504
Figure B.4 - Sequence level protocol example	506
Figure B.5 - Class 1 and 6 frame level protocol - Login example	508
Figure B.6 - Class 2 frame level protocol - Login example	509
Figure B.7 - Class 3 frame level protocol - Login example	510
Figure F.1 - Class 4 setup, activation and deactivation	521
Figure F.2 - Class 4 activation collision and circuit removal	523
Figure G.1 - Preemption Request	524
Figure G.2 - Connection Rejection	525
Figure G.3 - Preemption Accepted	525
Figure G.4 - Connection Established	525
Figure G.5 - Link Reset Diagram	526
Figure H.1 - Generic NL_Port	528
Figure H.2 - Server NL_Port Clock Sync Context	529
Figure H.3 - Client NL_Port Clock Sync Context	529
Figure H.4 - Server Clock Sync Implementation (Basic Approach)	530
Figure H.5 - Client Clock Sync Implementation (Basic Approach)	530
Figure H.6 - ELS Clock Sync Model - Point-to-Point	530
Figure H.7 - Client Clock Drift	531
Figure H.8 - Client Clock Sync Logic Model (Rate Adjusted)	538
Figure H.9 - Rate Adjustment Hardware Assists for Client Clock Sync	539
Figure H.10 - Client Clock Sync Implementation (Link Delay Fix)	539
Figure H.11 - Server Clock Sync Implementation (Unload Error Fix)	540
Figure H.12 - Client Clock Drift (Monotonic)	541
Figure H.13 - ELS Clock Sync Model – Fabric	542
Figure H.14 - ELS Clock Sync Model – Loop	545
Figure H.15 - Application of Clock Synchronization to Tactical Avionics	548
Figure I.1 - Three configurations supported by the speed negotiation requirements	551
Figure I.2 - Example worst case timing for t_fail	553
Figure I.3 - Example worst case timing for t_ncycl using Rx_LOS	554
Figure I.4 - Example worst case timing for t_ncinit using Pass sync_test	555
Figure I.5 - Example worst case timing for t_wake	556
Figure I.6 - Example of maximum single disruption, Wait_for_signal	557
Figure I.7 - Example of maximum single disruption, Slow_wait	558
Figure I.8 - Example of maximum single disruption, Negotiate_master	559
Figure I.9 - Example where hub is at maximum port speed	559
Figure I.10 - Example of maximum disruption group - Wait_for_signal	560
Figure I.11 - Example of maximum disruption group - Slow_wait	561
Figure I.12 - Example of maximum disruption group - Negotiate_master	562
Figure I.13 - Example of maximum single disruption overall	563
Figure I.14 - Physical architecture of a port with a separate transceiver component	565
Figure K.1 - Case Study	570

Table 1 -	Comparison of ISO and American numbering conventions	51
Table 2 -	Data rate abbreviations	51
Table 3 -	Symbols	55
Table 4 -	Bit designations	74
Table 5 -	Conversion Example	75
Table 6 -	Valid Data Characters	77
Table 7 -	Valid Special Characters	81
Table 8 -	Delayed Code Violation example	82
Table 9 -	Frame Delimiters	84
Table 10 -	Primitive Signals	85
Table 11 -	Primitive Sequences	87
Table 12 -	FC_Port states	96
Table 13 -	Frame byte order	109
Table 14 -	Frame_Header	111
Table 15 -	R_CTL - Type Code Summary	112
Table 16 -	Device_Data Information Categories	113
Table 17 -	Data Descriptor Payload	113
Table 18 -	FC-4 Link_Data Information Categories	113
Table 19 -	Video_Data Information Categories	114
Table 20 -	Extended Routing Information Categories	114
Table 21 -	Domain Controller and Well-known address identifiers	115
Table 22 -	CS_CTL field - Class 1	116
Table 23 -	CS_CTL field - Class 2	116
Table 24 -	CS_CTL field - Class 3	116
Table 25 -	CS_CTL field - Class 4	117
Table 26 -	Priority Field - Class 1 and Class 6	118
Table 27 -	Priority Field	119
Table 28 -	TYPE codes - Link Service	120
Table 29 -	TYPE codes - Video_Data	120
Table 30 -	TYPE codes - FC-4 (Device_Data and Link_Data)	121
Table 31 -	Exchange/Sequence Control (F_CTL)	123
Table 32 -	Continue Sequence Condition Bits Definition	127
Table 33 -	Abort Sequence Condition Bits Definition by Sequence Initiator	128
Table 34 -	Abort Sequence Condition Bits Definition by Sequence Recipient	129
Table 35 -	F_CTL bit interactions on Data frames	130
Table 36 -	F_CTL bit interactions on ACK, BSY or RJT	131
Table 37 -	DF_CTL bit definition	132
Table 38 -	ESP_Header and ESP_Trailer in a frame	138
Table 39 -	Network_Header	138
Table 40 -	Association_Header	139
Table 41 -	Association_Header Validity bits (Word 0, Bits 31 to 24)	140
Table 42 -	Allowable Data frame delimiters	142
Table 43 -	ACK Frames by Class	144
Table 44 -	Link_Response Frames by Class	144
Table 45 -	Link_Control Information Categories	145
Table 46 -	Link_Control frame delimiters	146
Table 47 -	ACK precedence	147
Table 48 -	F_BSY Reason Codes	150
Table 49 -	P_BSY code format	151
Table 50 -	P_BSY action codes	151
Table 51 -	P_BSY Reason Codes	152
Table 52 -	Reject Code format	153
Table 53 -	Reject Action Codes	154
Table 54 -	Reject Reason Codes	154

Table 55 - Basic Link Service Information Categories	164
Table 56 - ABTS Parameter Field Definitions	164
Table 57 - BA_ACC Payload	169
Table 58 - BA_RJT Payload Format	171
Table 59 - BA_RJT reason codes	171
Table 60 - BA_RJT Reason Code Explanation	172
Table 61 - Extended Link Routing Bits and Information Categories	174
Table 62 - ELS_Command codes	175
Table 63 - Responses to Received ELSS	177
Table 64 - ABTX Payload	179
Table 65 - Recovery Qualifier Values	180
Table 66 - ABTX LS_ACC Payload	180
Table 67 - ADVC Payload	181
Table 68 - ADVC LS_ACC Payload	182
Table 69 - ECHO Payload	183
Table 70 - ECHO LS_ACC Payload	183
Table 71 - ESTC Payload	184
Table 72 - ESTS Payload	185
Table 73 - ESTS LS_ACC Payload	185
Table 74 - LOGO Payload	187
Table 75 - LOGO LS_ACC Payload	187
Table 76 - RCS Payload	188
Table 77 - RCS LS_ACC Payload	188
Table 78 - Connection Status Codes	189
Table 79 - RES Payload	190
Table 80 - RES LS_ACC Payload	190
Table 81 - RLS Payload	191
Table 82 - RLS LS_ACC Payload	191
Table 83 - RSS Payload	192
Table 84 - RSS LS_ACC Payload	192
Table 85 - RTV Payload	193
Table 86 - RTV LS_ACC Payload	193
Table 87 - RRQ Payload	195
Table 88 - RRQ LS_ACC Payload	195
Table 89 - RSI Payload	196
Table 90 - RSI LS_ACC Payload	196
Table 91 - TEST Payload	197
Table 92 - FAN Payload	198
Table 93 - LINIT Payload	199
Table 94 - Initialization Function	199
Table 95 - LINIT LS_ACC Payload	199
Table 96 - LINIT Status	199
Table 97 - LSTS Payload	200
Table 98 - LSTS LS_ACC Payload	200
Table 99 - FC-FLA Compliance Level	201
Table 100 - Loop State	202
Table 101 - RSCN Payload	204
Table 102 - Generic affected Port_ID page	204
Table 103 - RSCN Event Qualifier values	205
Table 104 - Address Format	205
Table 105 - RSCN LS_ACC Payload	206
Table 106 - SCR Payload	206
Table 107 - Registration Function	207
Table 108 - SCR LS_ACC Payload	207
Table 109 - PRLI Payload	208
Table 110 - PRLI service parameter page format	208
Table 111 - PRLI LS_ACC Payload	209

Table 112 - PRLI LS_ACC service parameter response page format	210
Table 113 - PRLI accept response code	211
Table 114 - PRLO Payload	212
Table 115 - PRLO logout parameter page format	213
Table 116 - PRLO LS_ACC Payload	214
Table 117 - PRLO LS_ACC logout parameter response page format	215
Table 118 - PRLO accept response code	216
Table 119 - TPLS Payload	217
Table 120 - TPLS image pair ID page format	217
Table 121 - TPLS LS_ACC Payload	218
Table 122 - TPLS response page format	219
Table 123 - TPLS accept response code	220
Table 124 - FARP_REQ Request Payload Item	221
Table 125 - FARP_REQ Match Address Code Points	223
Table 126 - FARP_REQ Address Fields	224
Table 127 - Responder Action	224
Table 128 - FARP_REPLY Request Payload	225
Table 129 - FARP_REPLY Address Fields	227
Table 130 - FARP_REPLY LS_ACC Payload	227
Table 131 - RNID Payload	228
Table 132 - Node Identification Data Format	228
Table 133 - RNID Accept Payload	229
Table 134 - Node Identification Data Format	229
Table 135 - Common Identification Data	230
Table 136 - General Topology Specific Identification Data	231
Table 137 - Associated Type	232
Table 138 - Multi-function device bit definitions	233
Table 139 - Node Management	233
Table 140 - IP Version	234
Table 141 - RLIR Payload	236
Table 142 - Link Incident Record Format	236
Table 143 - Common Link Incident Record Data	237
Table 144 - Time Stamp Format values	238
Table 145 - Incident Qualifier	239
Table 146 - Incident Code values	240
Table 147 - RLIR LS_ACC Payload	241
Table 148 - LIRR Payload	242
Table 149 - Registration Function	243
Table 150 - Link Incident Record-Registration Format	243
Table 151 - LIRR LS_ACC Payload	243
Table 152 - Get Alias_ID Payload	244
Table 153 - Get Alias_ID LS_ACC Payload	245
Table 154 - Fabric Activate Alias_ID Payload	246
Table 155 - Fabric Activate Alias_ID LS_ACC Payload	246
Table 156 - Fabric Deactivate Alias_ID Payload	247
Table 157 - Fabric Deactivate Alias_ID LS_ACC Payload	247
Table 158 - N_Port Activate Alias_ID Payload	248
Table 159 - N_Port Activate Alias_ID LS_ACC Payload	249
Table 160 - N_Port Deactivate Alias_ID Payload	249
Table 161 - N_Port Deactivate Alias_ID LS_ACC Payload	250
Table 162 - QoSR Payload	251
Table 163 - QoSR LS_ACC Payload	253
Table 164 - RVCS Payload	255
Table 165 - RVCS LS_ACC Payload	255
Table 166 - Class 4 VC Status Block entry	255
Table 167 - ADISC Payload	257
Table 168 - ADISC LS_ACC Payload	257

Table 169 - Response summary to FDISC/PDISC	258
Table 170 - TPRLO Payload	259
Table 171 - TPRLO logout parameter page format	259
Table 172 - TPRLO LS_ACC Payload	259
Table 173 - LCLM Payload	263
Table 174 - LCLM Commands	264
Table 175 - Global capabilities	264
Table 176 - N_Port and node Name Field Values	264
Table 177 - LCLM LS_ACC Payload	265
Table 178 - Global capabilities	265
Table 179 - RPS Payload	266
Table 180 - Flag field bit definitions	267
Table 181 - RPS LS_ACC Payload	267
Table 182 - Flag field bit definitions	267
Table 183 - Port Status	268
Table 184 - L_Port_Extension	268
Table 185 - L_Port Status	269
Table 186 - RPL Payload	270
Table 187 - Maximum Size values	270
Table 188 - RPL LS_ACC Payload	271
Table 189 - Port Number Block	271
Table 190 - RPBC Payload	272
Table 191 - ELS Buffer Parameters Field	272
Table 192 - RPL LS_ACC Payload	273
Table 193 - RNFT Payload	273
Table 194 - RNFT LS_ACC Payload	274
Table 195 - RNFT FC-4 Entry	274
Table 196 - SRL Payload	275
Table 197 - Flag field definitions	276
Table 198 - SRL LS_ACC Payload	276
Table 199 - SBRP Payload	277
Table 200 - SBRP LS_ACC Payload	279
Table 201 - RPSC Payload	280
Table 202 - RPSC LS_ACC Payload	280
Table 203 - REC Payload	281
Table 204 - REC LS_ACC Payload	282
Table 205 - LS_RJT Reason Codes	284
Table 206 - LS_RJT Reason Code Explanation	285
Table 207 - NAA identifiers	304
Table 208 - IEEE 48-bit address format	304
Table 209 - IEEE extended format	305
Table 210 - Locally assigned format	306
Table 211 - 32-bit IP address format	306
Table 212 - IEEE Registered	306
Table 213 - IEEE Registered Extended	307
Table 214 - EUI-64 Mapped Name_Identifier Format	308
Table 215 - Bit Position Map	308
Table 216 - FLOGI, PLOGI or LS_ACC Payload	318
Table 217 - Common Service Parameter applicability	319
Table 218 - Common Service Parameters - FLOGI	320
Table 219 - Common Service Parameters - PLOGI and PLOGI LS_ACC	321
Table 220 - Common Service Parameters - FLOGI LS_ACC	321
Table 221 - Clock Synchronization Applicability	324
Table 222 - Class Service Parameters Applicability	326
Table 223 - Class Service Parameters - FLOGI	328
Table 224 - Class Service Parameters - PLOGI and PLOGI LS_ACC	329
Table 225 - Class Service Parameters - FLOGI LS_ACC	329

Table 226 - Intermix Mode Support	330
Table 227 - Stacked Connect-request support Login Bits	330
Table 228 - Sequential delivery support	331
Table 229 - Fabric Login Priority and Preemption Support	331
Table 230 - Class 2 and 3 Preference Bit Function	332
Table 231 - Relationship between Preferred delivery and sequential delivery	333
Table 232 - DiffServ QoS bit definition	334
Table 233 - Initial Process_Associator Bits Definition	334
Table 234 - ACK_0 Support Conditions (Initiator Control)	335
Table 235 - ACK_0 Support Conditions (Recipient Control)	336
Table 236 - Error Policy Bits Definition	337
Table 237 - Categories per Sequence Bits Definition	338
Table 238 - Concurrent Sequences field meaning	339
Table 239 - End-to-end Credit Field Meaning	339
Table 240 - Login Extension Page format	342
Table 241 - Page Code Definitions	342
Table 242 - Vendor Specific Page format	342
Table 243 - N_Port Clock Synchronization QoS	343
Table 244 - FLOGI/PLOGI CS_QoS_Request	343
Table 245 - Fx_Port Clock Synchronization QoS	345
Table 246 - Exchange Status Block	370
Table 247 - Sequence Status Block	372
Table 248 - Flow control applicability	374
Table 249 - Buffer participation	375
Table 250 - End-to-end flow control management	377
Table 251 - Buffer-to-buffer flow control management	385
Table 252 - Integrated Class 2 flow control management	396
Table 253 - Segmentation and reassembly rules summary	401
Table 254 - Responses to connect-request (SOFc1)	408
Table 255 - Responses to Preemption Requests	415
Table 256 - Link Error Status Block format for RLS command	431
Table 257 - Detailed errors and actions	432
Table 258 - Address Resolution	438
Table 259 - Bandwidth Allocation Example	453
Table 260 - Frame delimiter usage	461
Table 261 - CSR Payload	472
Table 262 - CSR Clock Sync Mode Meaning	472
Table 263 - CSR LS_ACC Payload	473
Table 264 - CSU Clock Sync Mode Meaning	473
Table 265 - CSU Payload	474
Table 266 - Clock Count Field Meaning	475
Table 267 - Neutral Disparity Character Values	477
Table 268 - Timing parameters with a range	493
Table 269 - Constant timing parameters	493
Table A.1 - Sample FC-2 frame	497
Table A.2 - Sample ACK_1 without CRC	497
Table A.3 - $F(x)$	497
Table A.4 - $X^{*32} F(x) + X^{*k} L(x)$	498
Table A.5 - $R(x)$	498
Table A.6 - $L(x) + R(x) = R\$(x)$	498
Table A.7 - $M(x)$	498
Table A.8 - $M(x) - (10B)$	499
Table B.1 - F_CTL for Class 1, Class 2, and Class 6 frame level protocols	505
Table B.2 - F_CTL for Class 3 frame level protocol	505
Table B.3 - Sequence level protocol example	507
Table C.1 - Case 1	511
Table C.2 - Case 2	511

Table C.3 - Case 3	512
Table C.4 - Case 4	512
Table C.5 - F_CTL for example Exchange	513
Table E.1 - Link Failure Counters and management	517
Table H.1 - Parameters used in analysis	532
Table H.2 - Example of analysis results	532
Table H.3 - Parameters used in analysis	533
Table H.4 - Parameters used in analysis	534
Table H.5 - Parameters used in analysis	535
Table H.6 - Parameters used in analysis	536
Table H.7 - Parameters used in analysis	543
Table H.8 - Example of analysis results	543
Table H.9 - Parameters used in analysis	546
Table K.1 - IEEE 48 bit address WWN format	568
Table K.2 - Mapped EUI-64 address	568
Table K.3 - IEEE Extended WWN format	569
Table K.4 - Mapped EUI-64 address	569
Table K.5 - IEEE Registered WWN format	569
Table K.6 - Mapped EUI-64 address	569

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

INFORMATION TECHNOLOGY – FIBRE CHANNEL –

Part 251: Framing and signalling (FC-FS)

FOREWORD

- 1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards. Their preparation is entrusted to technical committees; any ISO and IEC member body interested in the subject dealt with may participate in this preparatory work. International governmental and non-governmental organizations liaising with ISO and IEC also participate in this preparation.
- 2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.
- 3) The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC and ISO member bodies.
- 4) IEC, ISO and ISO/IEC publications have the form of recommendations for international use and are accepted by IEC and ISO member bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC, ISO and ISO/IEC publications is accurate, IEC or ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 5) In order to promote international uniformity, IEC and ISO member bodies undertake to apply IEC, ISO and ISO/IEC publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any ISO/IEC publication and the corresponding national or regional publication should be clearly indicated in the latter.
- 6) ISO and IEC provide no marking procedure to indicate their approval and cannot be rendered responsible for any equipment declared to be in conformity with an ISO/IEC publication.
- 7) All users should ensure that they have the latest edition of this publication.
- 8) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO member bodies for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon, this ISO/IEC publication or any other IEC, ISO or ISO/IEC publications.
- 9) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

IEC and ISO draw attention to the fact that it is claimed that compliance with this document may involve the use of patents as indicated below.

Cisco Systems, Inc. has informed IEC and ISO that it has patent applications or granted patents.

ISO and IEC take no position concerning the evidence, validity and scope of this putative patent right. The holder of this putative patent right has assured IEC and ISO that they are willing to negotiate free licences or licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this putative patent right is registered with IEC and ISO. Information may be obtained from:

Cisco Systems, Inc
Dan Lang Legal
170 West Tasman Drive
San Jose, CA95134
USA
e-mail: standards-ipr@cisco.com

IBM Corporation has informed IEC and ISO that it has patent applications or granted patents as listed below and possibly others not yet identified:

4,486,739 / US; 4,995,056 / US; 5,136,410/US; 5,151,977 / US; 5,260,933 / US; US2003/0103504A1; US 2003/0135620A1

ISO and IEC take no position concerning the evidence, validity and scope of this putative patent right. The holder of this putative patent right has assured IEC and ISO that they are willing to negotiate free licences or licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this putative patent right is registered with IEC and ISO. Information may be obtained from:

IBM Corporation
Brian Hinman
Vice President licensing intellectual property systems
North Castle Drive
Armonk, NY, 10504
USA
e-mail: hinman@us.ibm.com

Brocade Communication Systems, Inc. has informed IEC and ISO that it has patent applications or granted patents as listed below:

ISO and IEC take no position concerning the evidence, validity and scope of this putative patent right. The holder of this putative patent right has assured IEC and ISO that they are willing to negotiate free licences or licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this putative patent right is registered with IEC and ISO. Information may be obtained from:

Brocade Communication Systems Inc.
Office of the General Counsel
1745 Technology Drive
San Jose, CA 95110
USA
e-mail: generalcounsel@brocade.com

International Standard ISO/IEC 14165-251 was prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

The list of all currently available parts of the ISO/IEC 14165 series, under the general title *Information technology – Fibre channel*, can be found on the IEC web site.

This International Standard has been approved by vote of the member bodies, and the voting results may be obtained from the address given on the second title page.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

INTRODUCTION

This International Standard combines the following Fibre Channel standards (for full reference, see Bibliography):

ANSI INCITS 230-1994 (R1999)

ANSI INCITS 230-1994/Amendment 1-1996 (R2001)

ANSI INCITS 230-1994/Amendment 2-1999 (R2001)

ANSI INCITS 297-1996 (R2002)

ANSI INCITS 303-1998 (R2002)

This International Standard includes the following changes with respect to the above publications:

- definitions of existing services have been clarified and/or improved based on experience with existing implementations;
- outdated functions and features have been deleted;
- additional link services in support of new functions defined by the Fibre Channel family of documents have been included;
- definition of other capabilities which enhance the performance of existing Fibre Channel products and fit them for new applications.

INFORMATION TECHNOLOGY – FIBRE CHANNEL –

Part 251: Framing and signalling (FC-FS)

1 Scope

This part of ISO/IEC 14165 describes the framing and signalling interface of a high-performance serial link for support of FC-4s associated with upper level protocols (for example SCSI, IP, SBCCS, VI).

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

2.1 Approved references

- [1] ISO/IEC 14165-116, *Information technology – Fibre channel – Part 116: 10 Gigabit fibre channel (10GFC)*
- [2] ISO/IEC 14165-122, *Information technology – Fibre channel – Part 122: Arbitrated loop-2 (FC-AL-2)*
- [3] ISO/IEC 9314-2:1989, *Information processing systems – Fibre distributed data interface (FDDI) – Part 2: Token ring media access control (FDDI-MAC)*
- [4] ISO/IEC TR 8802-1, *Information technology – Telecommunications and information exchange between systems - Local and metropolitan area networks – Specific requirements – Part 1: Overview of Local Area Network Standards*
- [5] ISO/IEC 8802-2, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 2: Logical link control*
- [6] ISO/IEC 14165-414, *Information technology – Fibre channel – Part 414: Generic services – 4 (FC-GS-4)* [ANSI INCITS 387-2004]

2.2 References under development

- [7] ISO/IEC 14165-133, *Information technology – Fibre channel – Part 133: Switch fabric-3 (FC-SW-3)*, [ANSI INCITS 384-2004]
- [8] ISO/IEC 14165-142, *Information technology – Fibre channel – Part 142: - Physical interfaces-2 (FC-PI-2)*, [ANSI INCITS 404-2006]
- [9] ISO/IEC 14776-223, *Information technology – Small computer system interface (SCSI) – Part 223: Fibre channel protocol for SCSI, Version 3 (FCP-3)* [ANSI INCITS 416-2006]
- [10] ISO/IEC 14776-453, *Information technology – Small computer system interface (SCSI) – Part 453: Primary commands-3 (SPC-3)* [ANSI INCITS 408-2005]

2.3 Other references

All references in this subclause were correct at the time of approval of this International Standard. The provisions of the referenced specifications, as identified in this subclause, are valid within the context of this International Standard. The reference to a specification within this International Standard does not give it any further status

within ISO/IEC; in particular, it does not give the referenced specification the status of an International Standard.

IETF Requests for Comments (RFCs) may be obtained directly from the IETF web site at <http://www.ietf.org/rfc.html>.

- [11] INCITS TR-20:1998, *Information technology - Fibre Channel – Fabric Loop Attach (FC-FLA)*
- [12] RFC 2625, *IP and ARP over Fibre Channel*
- [13] RFC 2597, *Assured Forwarding PHB Group*, June 1999
- [14] RFC 2598, *An Expedited Forwarding PHB*, June 1999
- [15] RFC 768, *User Datagram Protocol*, August 1980.
- [16] RFC 791, *Internet Protocol*, September 1981.
- [17] RFC 793, *Transmission Control Protocol*, September 1981.
- [18] RFC 854, *Telnet Protocol Specification*, May 1983.
- [19] RFC 1157, *A Simple Network Management Protocol (SNMP)*, May 1990.
- [20] RFC 1901, *Introduction to Community-based SNMPv2*, January 1996
- [21] RFC 2373, *IP Version 6 Addressing Architecture*, July 1998.
- [22] RFC 2460, *Internet Protocol, Version 6 (IPv6) Specification*, December 1998.
- [23] RFC 2616, *Hypertext Transfer Protocol -- HTTP/1.1*, June 1999.
- [24] RFC 2818, *HTTP Over TLS*, May 2000.
- [25] RFC 2406, *IP Encapsulating Security Payload (ESP)*, November 1998.

3 Definitions, abbreviations, conventions and keywords

3.1 Definitions

3.1.1 acknowledged class: Any class of service that acknowledges a transfer.

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 17.6.1).

3.1.3 address identifier: An address value used to identify source (S_ID) or destination (D_ID) of a frame (see 9.4).

3.1.4 Alias_ID: An address identifier recognized by one or more Nx_Ports or the Alias Server, if the Nx_Port has registered with the Alias Server as a member of a group. An Alias_ID may be common to multiple Nx_Ports (see clause 24).

3.1.5 Alias-Token: A 12-byte field to indicate the type of Alias_ID and certain properties associated with the Alias_ID (see 12.3.2.33).

3.1.6 Arbitrated Loop topology: A Fibre Channel topology where L_Ports use arbitration to gain access to the loop (See FC-AL-2).

3.1.7 BB_Buffer: The buffer associated with buffer-to-buffer flow control (see 18.5).

3.1.8 buffer-to-buffer credit (BB_Credit): The controller parameter for buffer-to-buffer flow control (see 18.5.3).

3.1.9 B_Port: A bridge port is a fabric inter-element port used to connect bridge devices with E_Ports on a switch. The B_Port provides a subset of the E_Port functionality (see [7], FC-SW-3.).

3.1.10 buffer: A logical construct that holds a single frame.

3.1.11 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, unless specifically indicated otherwise.

3.1.12 character: An 8B10B encoding of a data byte or control value transmitted and interpreted by the FC-1 layer of Fibre Channel (see clause 5).

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

3.1.14 circuit-oriented frames: Frames sent through a Class 4 circuit.

3.1.15 classes of service: Type of frame delivery services used by the communicating Nx_Ports that may also be supported through a fabric (see 4.8 and clause 13).

3.1.16 Class 1 service: A service that establishes a dedicated connection between two communicating Nx_Ports (see 4.8.2 and 13.2).

3.1.17 Class 2 service: A service that multiplexes frames at frame boundaries to or from one or more Nx_Ports with acknowledgement provided (see 4.8.3 and 13.3).

3.1.18 Class 3 service: A service that multiplexes frames at frame boundaries to or from one or more Nx_Ports without acknowledgement (see 4.8.4 and 13.4).

3.1.19 Class 4 circuit: A pair of unidirectional Virtual Circuits between two communicating Nx_Ports (see 25.3.5).

3.1.20 Class 4 service: A fabric service that establishes Virtual Circuits to provide fractional bandwidth service

between communicating Nx_Ports. The fabric service multiplexes frames at frame boundaries using in-order delivery to or from one or more Nx_Ports with acknowledgment provided (see 4.8.5, 13.6 and clause 25).

3.1.21 Class 4 Circuit Initiator: The Nx_Port that initiates a Class 4 Circuit setup to establish a Class 4 Circuit (see 25.3.5).

3.1.22 Class 4 Circuit Recipient: The Nx_Port that receives a Class 4 Circuit setup request from a Class 4 Circuit Initiator and accepts the setup request by transmitting a valid response (see 25.3.5).

3.1.23 Class 6 service: A service that allows an Nx_Port to establish simultaneous dedicated connections with multiple Nx_Ports (see 4.8.6 and 13.7).

3.1.24 Class F service: A service that multiplexes frames at frame boundaries with acknowledgement provided. The service is used for control and coordination of the internal behavior of the Fabric (see [7], FC-SW-3).

3.1.25 Class N service: A class of service other than Class F (see [7], FC-SW-3).

3.1.26 code violation: An error condition that occurs when a received Transmission Character is not able to be decoded to a valid data byte or special code using the validity checking rules specified by the transmission code (see 5.3.3).

3.1.27 comma: The seven-bit sequence '0011111' b or '1100000' b in an 8B10B encoded stream (see 5.5.1).

3.1.28 Common Controlling Entity : The entity that controls and manages the resources for a Hunt Group (see clause 22)

3.1.29 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 (for example, S_ID||X_ID).

3.1.30 Connection Initiator: The Nx_Port that initiates a Class 1 or 6 Connection with a destination Nx_Port through a connect-request and also receives a valid response from the destination Nx_Port to complete the Connection establishment (see clause 20).

3.1.31 Connection Recipient: The destination Nx_Port that receives a Class 1 or 6 connect-request from the Connection Initiator and accepts establishment of the Connection by transmitting a valid response (see clause 20).

3.1.32 connectionless buffers: Receive buffers participating in connectionless service and capable of receiving connectionless frames (see clause 18).

3.1.33 connectionless frames: Frames participating in connectionless service (i.e., Class 1 or Class 6 frames with SOF_{c1}, Class 2, Class 3 and Class 4 frames referred to individually or collectively) (see clause 18).

3.1.34 connectionless service: Communication between two Nx_Ports performed without a dedicated connection.

3.1.35 continuously increasing relative offset: The condition of operation that requires frames ordered by SEQ_CNT within a Sequence to have a larger relative offset value in each frame (see 15.6.2.4.1).

3.1.36 credit: The maximum number of buffers available at a recipient to receive frames from a transmitting FC_Port (see 18.3).

3.1.37 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 (see 5.3.3).

3.1.38 data block: An ordered string of application data contained in a single Information Category.

3.1.39 data character: Any Transmission Character associated by the transmission code with a valid data byte.

(see 5.1).

3.1.40 data frame: An FC-4 Device_Data frame, an FC-4 Video_Data frame, or a Link_Data frame (see clause 11).

3.1.41 decoding: Validity checking of received Transmission Characters and generation of valid data bytes and special codes from those characters (see 5.3).

3.1.42 dedicated connection: A communicating circuit guaranteed and retained by the Fabric for two given Nx_Ports.

3.1.43 delimiter: An Ordered Set used to indicate a frame boundary (see 5.5, 8.3 and 8.7).

3.1.44 Destination_Identifier (D_ID): The address identifier used to indicate the targeted destination Nx_Port of the transmitted frame (see 9.4).

3.1.45 destination Nx_Port: The Nx_Port to where a frame is targeted.

3.1.46 discard policy: An error handling policy where a Sequence Recipient is able to discard Data frames received following detection of a missing frame in a Sequence (see 21.6.3).

3.1.47 disparity: The difference between the number of ones and zeros in a Transmission Character (see 5.5.3).

3.1.48 Domain Controller: The entity that controls activity within a given domain. Each Domain Controller is allocated an address (see [7], FC-SW-3.).

3.1.49 Domain_ID: The highest or most significant hierarchical level in the three-level addressing hierarchy (i.e., the most significant byte of the address identifier) (see 9.4.2 and [7], FC-SW-3).

3.1.50 EE_Buffer: Buffer(s) associated with end-to-end flow control (see 18.4).

3.1.51 encoding: Generation of Transmission Characters from valid data bytes and special codes (see 5.3).

3.1.52 E_Port: A Fabric expansion port that connects to another E_Port or B_Port to create an Inter-Switch Link (see [7], FC-SW-3).

3.1.53 Exchange: The unit of protocol activity that transfers information between a specific Originator Nx_Port and specific Responder Nx_Port using one or more related non-concurrent Sequences that may flow in the same or opposite directions. The Exchange is identified by an OX_ID and a RX_ID (see clause 17).

3.1.54 Exchange_Identifier (X_ID): A collective reference to OX_ID and RX_ID (see 9.2.1).

3.1.55 Exchange Status Block: A logical construct that contains the status of an Exchange. An Originator Nx_Port has an Originator Exchange Status Block and the Responder Nx_Port has a Responder Exchange Status Block for each active Exchange (see 4.12.4.4 and 17.8.1).

3.1.56 exclusive connection: A Class 1 or Class 6 dedicated connection without Intermix (see 4.8.2 and 13.5).

3.1.57 F_Port: The LCF within the Fabric that attaches to an N_Port through a link. An F_Port is addressable by the N_Port attached to it, with a common well-known address identifier (hex 'FF FF FE') (see 9.4.2).

3.1.58 Fabric: The entity that interconnects Nx_Ports attached to it and is capable of routing frames by using the D_ID information in a FC-2 frame header (see 4.7.3).

3.1.59 Fabric_Name: A Name_Identifier associated with a Fabric (see clause 14 and 15.6.4).

3.1.60 FC_Port: A port that is capable of transmitting or receiving Fibre Channel frames according to the requirements defined in this standard. FC_Ports include N_Ports, NL_Ports, Nx_Ports, L_Ports, F_Ports, FL_Ports, Fx_Ports, E_Ports, and B_Ports.

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

3.1.62 fractional bandwidth: A portion of the total bandwidth available on a path (see clause 25).

3.1.63 frame: An indivisible unit of information used by FC-2 (see 8.1).

3.1.64 F_Port_Name: A Name_Identifier associated with an F_Port (see 9.4.2).

3.1.65 frame content: The information contained in a frame between its SOF and EOF delimiters, excluding the delimiters (see 8.3.1).

3.1.66 Frame Type 0: A type of Data frame used for Link Control (see clause 8.5).

3.1.67 Frame Type 1: A type of Data frame used for data exchange (see clause 8.5).

3.1.68 Fx_Port: A switch port capable of operating as an F_Port or FL_Port (see FC-AL-2).

3.1.69 Hunt Group: A set of Nx_Ports with a common Alias_ID managed by a Common Controlling Entity. The management and initialization of Hunt Groups is outside the scope of this standard (see clause 22).

3.1.70 Hypertext Transfer Protocol: A protocol for communicating various formats of text with embedded links and display controls (see [23], RFC 2616).

3.1.71 Idle: An Ordered Set of four Transmission Characters that are normally transmitted between frames (see 5.5).

3.1.72 Infinite buffer: A terminology to indicate that at FC-2 level, the amount of buffer available at the Sequence Recipient is unlimited.

3.1.73 Information Category: The category to which the frame Payload belongs (for example, Solicited Data, Unsolicited Data, Solicited Control and Unsolicited Control). Information category is indicated by the INFORMATION field in the frame header if the value of the ROUTING field in the frame header is '0000'b (Device_Data), '0010'b (Extended Link Services), '0011'b (FC-4 Link_Data), '0100'b (Video_Data), or '1111'b (Extended Routing) (see 9.3.3).

3.1.74 Information Unit: An organized collection of data specified by an upper level to be transferred as a single Sequence by FC-2.

3.1.75 initial relative offset: A relative offset value specified at the sending end by an upper level for a given data block and used by the sending FC-2 in the first frame of that data block (see data block, and relative offset). Initial relative offset value may be zero or non-zero (see clause 19).

3.1.76 Intermix: A service that interleaves Class 2 and Class 3 frames on an established Class 1 or Class 6 Connection (see 4.9 and 13.5).

3.1.77 Internet Protocol: A protocol for communicating data packets between identified endpoints on a multipoint network. It is in wide use in versions 4 and 6 (See [16], RFC 791 for version 4 or [21], RFC 2373 and [22] RFC 2460 for version 6).

3.1.78 IP Address: An identifier of an endpoint in Internet Protocol.

3.1.79 link: Two unidirectional fibres transmitting in opposite directions and their associated transmitters and

receivers.

3.1.80 Link Control Facility (LCF): A hardware facility that attaches to an end of a link and manages transmission and reception of data. It is contained within each FC_Port (see 4.4).

3.1.81 local Fx_Port: The Fx_Port to which an Nx_Port is directly attached by a link or an Arbitrated Loop (see remote Fx_Port).

3.1.82 L_Port: A port that contains Arbitrated Loop functions associated with Arbitrated Loop topology (see FC-AL-2).

3.1.83 Name_Identifier: A 64-bit identifier, with a 60-bit value preceded by a 4-bit Network_Address_Authority Identifier, used to identify entities in Fibre Channel (for example, Nx_Port, node, F_Port, or Fabric) (see clause 14).

3.1.84 Network_Address_Authority (NAA): An organization such as IEEE that administers network addresses (see clause 14).

3.1.85 Network_Address_Authority (NAA) identifier: A four-bit identifier defined to indicate a Network_Address_Authority (NAA) (see clause 14).

3.1.86 NL_Port: An N_Port that contains the Loop Port State Machine defined in FC-AL-2. It may be attached via a link to one or more NL_Ports and zero or more FL_Ports in an Arbitrated Loop topology. Without the qualifier "Public" or "Private," an NL_Port is assumed to be a Public NL_Port.

3.1.87 node: A collection of one or more Nx_Ports controlled by a level above FC-2 (see 4.1).

3.1.88 Node_Name: A Name_Identifier associated with a node (see Clause 14 and 15.6.4).

3.1.89 N_Port: A hardware entity that includes a LCF but not Arbitrated Loop functions associated with Arbitrated Loop topology, and has the ability to act as an Originator, a Responder, or both. Well-known addresses are considered to be N_Ports (See [2], FC-AL-2, and 9.4.2).

3.1.90 N_Port_ID: A Fabric unique address identifier of an Nx_Port. The identifier may be assigned by the fabric during the initialization procedure or by other procedures not defined in this standard. The identifier is used in the S_ID and D_ID fields of a frame (see 9.4).

3.1.91 N_Port_Name: A Name_Identifier associated with an Nx_Port (see clause 14 and 15.6.3).

3.1.92 Nx_Port: A port capable of operating as an N_Port or Public NL_Port, but not as a Private NL_Port. By use of the term Nx_Port, this standard neither specifies nor constrains the behavior of Private NL_Ports (see [2] FC-AL-2).

3.1.93 open: The period of time starting when a Sequence or an Exchange is initiated until that Sequence or Exchange is normally or abnormally terminated (see 17.6.1).

3.1.94 Ordered Set: A Transmission Word composed of a special character in its first (left-most) position and data characters in its remaining positions (see 5.5).

3.1.95 Originator: The logical function associated with an Nx_Port responsible for originating an Exchange.

3.1.96 Originator Exchange_ID (OX_ID): An identifier assigned by an Originator to identify an Exchange (see 4.12.4.2).

3.1.97 Payload: Contents of the Data Field of a frame, excluding Optional Headers and fill bytes, if present (see table 13, and clause 8, clause 10, and 11.1).

3.1.98 Policy: The rule used to determine how frames not received are handled during error recovery (see

21.6.3).

3.1.99 Primitive Sequence: An Ordered Set transmitted repeatedly and continuously until a specified response is received (see 5.5.4 and 17.6.1).

3.1.100 Primitive Signal: An Ordered Set designated to have a special meaning. (for example, an Idle or R_RDY) (see 5.5.3).

3.1.101 Private NL_Port: An NL_Port that does not attempt a Fabric Login and does not transmit OPN(00,x) (see [2], FC-AL-2).

3.1.102 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 a process uses to address another communicating process. Process_Associator is a generic reference to Originator Process_Associator and Responder Process_Associator (see 10.4).

3.1.103 Public NL_Port: An NL_Port that attempts a Fabric Login (see [2], FC-AL-2).

3.1.104 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, initial 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) (see 15.6.2.4.3).

3.1.105 receiver: The portion of a LCF dedicated to receiving an encoded bit stream from a fibre, converting this bit stream into Transmission Characters, and decoding these Transmission Characters using the rules specified by this standard (see 4.1).

3.1.106 relative offset: The displacement, expressed in bytes, of the first byte of a Payload related to an upper level defined origin for a given Information Category (see continuously increasing relative offset, random relative offset and 4.13.3).

3.1.107 relative offset space: A virtual address space defined by the sending upper level for a set of information carried in one or more information units.

3.1.108 remote Fx_Port: Relative to an Nx_Port that is communicating through a fabric to a remote Nx_Port, the Fx_Port to which the remote Nx_Port is directly attached (see local Fx_Port).

3.1.109 Responder: The logical function in an Nx_Port responsible for supporting the Exchange initiated by the Originator in another Nx_Port.

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

3.1.111 run length: Number of consecutive identical bits in the transmitted signal for example, the pattern '0011111010' has a maximum run length of five and a minimum run length of one (see 5.3.3.1).

3.1.112 running disparity: A binary value indicating the cumulative encoded signal unbalance between the one and zero signal state of all Transmission Characters since Transmission Word synchronization was been achieved (see 5.3.3.1).

3.1.113 Secured Hypertext Transfer Protocol: A protocol for communicating various formats of text with embedded links and display controls used in combination with a subordinate protocol that provides security features (see [24], RFC 2818).

3.1.114 Sequence: A set of one or more Data frames with a common Sequence_ID (SEQ_ID), transmitted unidirectionally from one Nx_Port to another Nx_Port with a corresponding response, if applicable, transmitted in response to each Data frame (see clause 17).

3.1.115 Sequence_ID (SEQ_ID): An identifier used to identify a Sequence (see clause 17).

3.1.116 Sequence Initiator: The Nx_Port that initiates a Sequence and transmits Data frames to the destination Nx_Port (see clause 17).

3.1.117 Sequence Recipient: The Nx_Port that receives Data frames from the Sequence Initiator and, if applicable, transmits responses (i.e., Link_Control frames) to the Sequence Initiator (see clause 17).

3.1.118 Sequence Status Block: A logical construct that tracks the status of a Sequence. Both the Sequence Initiator and the Sequence Recipient have a Sequence Status Block for each concurrently active Sequence (see 4.12.3.3 and 17.8.2).

3.1.119 Signal Failure: A condition in which an FC_Port capable of the Speed Negotiation procedure shall initiate that procedure (see 28.1).

3.1.120 Simple Network Management Protocol: A protocol for communicating simply structured management information. It is in wide use in versions 1 and 2 (See [19], RFC 1157, for version 1 or [20], RFC 1901, for version 2).

3.1.121 Source_Identifier (S_ID): The address identifier used to indicate the source Nx_Port of the transmitted frame (see 9.4.4).

3.1.122 source Nx_Port: The Nx_Port where a frame is originated.

3.1.123 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 (see 5.1).

3.1.124 special code: A code that, 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 (for example, K28.5) (see 5.2).

3.1.125 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 clause 17).

3.1.126 synchronization: Receiver identification of a Transmission Word boundary (see 6.1.4 and 6.1.5).

3.1.127 TCP Port Number: An identifier of a destination in Transmission Control Protocol.

3.1.128 Telnet: A protocol for communicating control of a character-oriented terminal over Transmission Control Protocol (see [18], RFC 854).

3.1.129 Transmission Character: Any valid or invalid encoded character transmitted across a physical interface specified by FC-0. Valid Transmission Characters are specified by the transmission code and include Data and special characters (see 5.1).

3.1.130 transmission code: A means of encoding data to enhance its transmission characteristics. The transmission code specified by this standard is byte-oriented, with valid data bytes and special codes encoded into 10-bit Transmission Characters (see clause 5).

3.1.131 Transmission Control Protocol: A protocol communicating reliable flow-controlled byte streams over Internet Protocol allowing independent concurrent streams to multiple destinations at any IP Address (see [17] RFC 793).

3.1.132 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 (see 5.4).

3.1.133 transmitter: The portion of a LCF 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) (see 6.2).

3.1.134 UDP Port Number: An identifier of a destination in User Datagram Protocol.

3.1.135 Unrecognized Ordered Set: A Transmission Word containing a K28.5 in its first (left-most) position that is not defined to have meaning by this standard, but that may be defined by other standards (for example see [2], FC-AL-2. (see 5.5.1).

3.1.136 Upper Level: A level above FC-2.

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

3.1.138 User Datagram Protocol: A protocol communicating a packet stream with no incremental reliability over Internet Protocol allowing multiple independent concurrent destinations at any IP Address (see [15] RFC 768).

3.1.139 valid data byte: A string of eight contiguous bits within FC-1 that represents a value with 0 to 255, inclusive (see 5.2).

3.1.140 valid frame: A frame received with a valid SOF, a valid EOF, valid data characters, and proper CRC of the Frame_Header and Data Field (See clause 8).

3.1.141 Virtual Circuit (VC): A unidirectional path between two communicating Nx_Ports through a fabric that permits Class 4 service to be used. Two Virtual Circuits are required to form a Class 4 Circuit (see 4.8.5 and 25.3.5).

3.1.142 Virtual Circuit Credit (VC_Credit): The number of receiver buffers allocated to a Virtual Circuit by an F_Port. It represents the maximum number of frames that an Nx_Port may transmit without causing a buffer overrun condition at the F_Port receiver (see 13.6 and clause 25).

3.1.143 Virtual Circuit Identifier (VC_ID): An identifier associated with either the Originator or Responder for a Virtual Circuit (see 9.5.2).

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

3.1.145 Well-known addresses: A set of address identifiers defined in this standard to access Generic Services functions. (for example, a name server) (see 9.4).

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

3.1.147 Worldwide Name: A Name_Identifier that is worldwide unique, and represented by a 64-bit value (see clause 14).

3.2 Editorial conventions

In this standard, 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 (for example, Exchange, Sequence). Any lower case uses of these words have the normal technical English meanings.

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

An alphanumeric list (for example, a, b, c) of items indicate the items in the list are unordered. A numeric list (for example, 1, 2, 3) of items indicate the items in the list are ordered (i.e., item 1 shall occur or complete before item 2).

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.

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 in table 1.

Table 1 - Comparison of ISO and American numbering conventions

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

Numbers in quotes and immediately followed by a lower-case b (for example, '0101' b) are binary values. Numbers or upper case letters in quotes immediately preceded by the word hex (for example, hex 'FA23') are hexadecimal values. Numbers not in this notation are decimal values. When X or Y is used in a hexadecimal notation, it represents a hexadecimal value of the range hex '0' to hex 'F'

3.3 Abbreviations, acronyms and symbols

3.3.1 Data rate abbreviations

The data rate abbreviations are shown in table 2. The exact data rate is used in the tables and the abbreviated form is used in text. Note that 1 062,5 Gbits/s is the preferred ISO method and is used instead of 1 062,5 Mbits/s where it makes sense to do so.

Table 2 - Data rate abbreviations

Abbreviation	Data Rate
1 063 Mbits/s	1 062,5 Mbits/s
2 125 Mbits/s	2 125 Mbits/s
4 250 Mbits/s	4 250 Mbits/s

3.3.2 Acronyms and other abbreviations

ABTS	Abort Sequence
ABTX	Abort Exchange
ACK	Acknowledgement
ADVC	Advise Credit

AL_PA	Arbitrated Loop Physical Address
BA_ACC	Basic Accept
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_SC_N	buffer-to-buffer State Change Number
BSY	busy
Credit_CNT	Credit_Count
CR_TOV	Connection Request_Timeout value
CTI	Circuit Initiator
CTR	Circuit Recipient
DF_CTL	Data_Field Control
D_ID	Destination_Identifier
DSCP	Differentiated Services Code Point
E_D_TOV	Error_Detect_Timeout value
EE_Credit	End-to-end Credit
EE_Credit_CNT	End-to-end Credit_Count
ELS	Extended Link Service
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
FC-2	Fibre Channel level 2
FC-4	Fibre Channel level 4
FCS	Frame Check Sequence
F_CTL	Frame Control
FDACT	Fabric Deactivate Alias
FLOGI	Fabric Login
F_RJT	Fabric Reject
FT_0	Frame Type 0 (see 8.5)
FT_1	Frame Type 1 (see 8.5)
GAID	Get Alias_ID

HBA	Host Bus Adapter
hex	hexadecimal notation
HG_ID	Hunt Group Identifier
HTTP	Hypertext Transfer Protocol
HTTPS	Secured Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
LCF	Link Control Facility
LCR	Link Credit Reset
LESB	Link Error Status Block (see 21.8)
LF1	NOS Receive State
LF2	NOS Transmit State
LILP	Loop Initialization Loop Position
LISA	Loop Initialization Soft Assigned
LOGO	Logout
LR	Link Reset Primitive Sequence
LR1	LR Transmit State
LR2	LR Receive State
LR3	LRR Receive State
LRR	Link Reset Response Primitive Sequence
LS_ACC	Link Service Accept
LS_Command	Link Service Command
m	Metre
MB	MegaByte
ms	millisecond
µs	microsecond
N/A	not applicable
NAA	Network_Address_Authority
NACT	N_Port Activate Alias
NAS	Network Attached Storage
NDACT	N_Port Deactivate Alias
NOP	No Operation
NOS	Not_Operational Primitive Sequence
ns	nanosecond
OL1	OLS Transmit State
OL2	OLS Receive State
OL3	Wait for OLS State
OLS	Offline Primitive Sequence

OX_ID	Originator Exchange_ID
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
R_C	Remove Circuit
RCS	Read Connection Status
R_CTL	Routing Control
RES	Read Exchange Status Block
RFC	Request for Comments
RJT	reject
RMC	Remove Connection
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
Rx	receiver
RX_ID	Responder Exchange_ID
s	second
SBCCS	Single Byte Command Code Sets
SCSI	Small Computer System Interface
SCR	State Change Registration
SEQ_CNT	Sequence Count
SEQ_ID	Sequence_ID
Serdes	Serializer/Deserializer
S_ID	Source_Identifier
SNMP	Simple Network Management Protocol
SOF	Start-of-Frame
SSB	Sequence Status Block
TCP	Transmission Control Protocol

TPLS	Test Process Login Status
Tx	transmitter
TYPE	Data structure type
UDP	User Datagram Protocol
ULP	Upper Level Protocol
VC	Virtual Circuit
VC_Credit	Virtual Circuit Credit
VC_ID	Virtual Circuit Identifier
VC_RDY	Virtual Circuit Ready
VI	Virtual Interface
WWN	Worldwide_Name
X_ID	Exchange_Identifier

3.3.3 Symbols

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

Table 3 - Symbols

	concatenation
μ	micro (for example, μm = micrometer)
L >>	Received from Link
x	multiplied by
*	multiplied by
**	exponentiation

3.4 Keywords

3.4.1 expected: Used to describe the behavior of the hardware or software in the design models assumed by this standard. Other hardware and software design models may also be implemented.

3.4.2 invalid: Used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field or code value shall be reported as error.

3.4.3 ignored: Used to describe a bit, byte, word, field or code value that shall not be examined by the receiving port. The bit, byte, word, field or code value has no meaning in the specified context.

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

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

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

3.4.7 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.

3.4.8 obsolete: A keyword indicating that an item was defined in a prior Fibre Channel standard but has been removed from this standard.

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

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

3.4.11 shall: A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard. This standard prescribes no specific response by a component if it receives information that violates a mandatory behavior.

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

3.4.13 should not: A keyword indicating flexibility of choice with a strongly preferred alternative; equivalent to the phrase "it is strongly recommended not to".

3.4.14 vendor specific: Functions, code values, and bits not defined by this standard and set aside for private usage between parties using this standard.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

4 Structure and Concepts

4.1 Introduction

This clause provides an overview of the structure, concepts and mechanisms used in FC-FS.

Fibre Channel (FC) is logically a bi-directional, point-to-point, serial data channel, structured for high performance capability. Fibre Channel may be implemented using any combination of the following three topologies:

- A point-to-point link between two N_Ports;
- A set of N_Ports interconnected by a switching network called a Fabric; and
- A set of L_Ports interconnected with a loop topology as defined in FC-AL-2 (See [2]).

This standard provides a general transport vehicle for Upper Level Protocols (ULPs) (e.g., Small Computer System Interface (SCSI) command sets, Internet Protocol (IP), and others). Proprietary and other command sets may also use and share Fibre Channel, but such use is not defined as part of this standard.

Fibre Channel is structured as a set of hierarchical functions as illustrated in figure 1. Fibre Channel Framing and Signalling interface (FC-FS) consists of related functions FC-1, FC-2, and FC-3. Each of these functions is described as a level. FC-FS does not restrict implementations to specific interfaces between these levels.

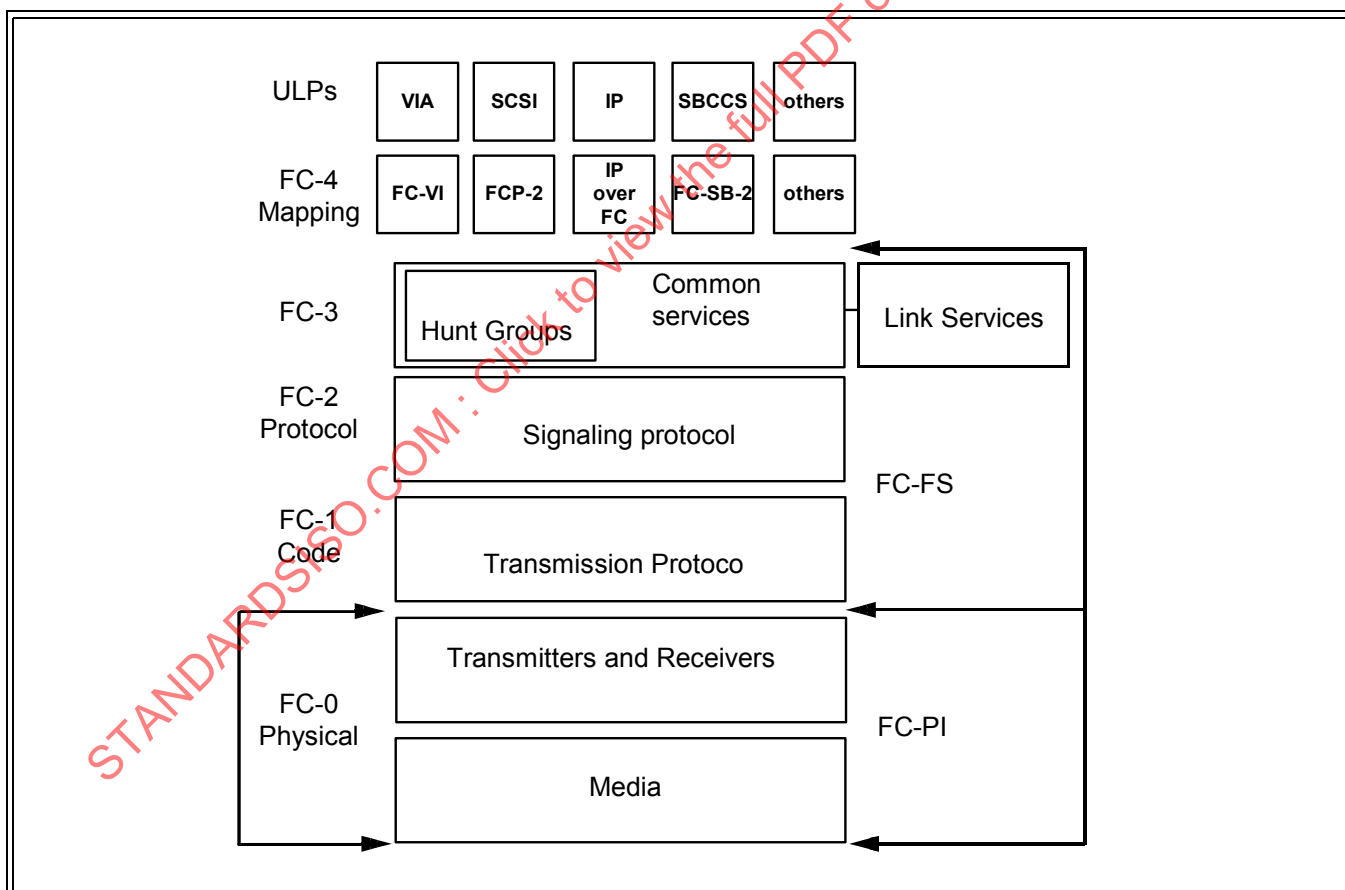


Figure 1 - Fibre Channel Structure

The physical interface (FC-0) consists of transmission media, transmitters, and receivers and their interfaces. Fibre, coax and twisted pair versions are defined. A variety of physical media, and associated drivers and receivers capable of operating at various speeds are specified to address variations in cable plants. See [8], FC-PI-2 and [1], 10GFC for the details.

FC-1 (see clause 5, clause 6, and clause 28) defines the transmission protocol that includes the serial encoding, decoding, and error control.

FC-2, the signalling protocol (see clause 7, clause 8, clause 9, clause 10, clause 11, clause 13, clause 14, clause 15, clause 16, clause 17, clause 18, clause 19, clause 20, clause 21, clause 23, clause 24, clause 25, clause 26, and clause 27), specifies the rules and provides mechanisms needed to transfer data blocks of data end-to-end. FC-2 defines a suite of functions and facilities available for use by an upper level. This may be more than what a given upper level requires. FC-2 functions include several classes of service, frame format definition, sequence disassembly and reassembly, exchange management, address assignment, alias address definition, multicast management, and Stacked Connect-requests.

FC-3 provides a set of services that are common across multiple Nx_Ports of a node. FC-3 includes protocols for Link Services (see clause 12) and Hunt Groups (see clause 22). The Link Services represent a mandatory function required by FC-2.

FC-4 is the highest level in the Fibre Channel standards set. An FC-4 defines the mapping between the lower levels of the Fibre Channel and an Upper Level Protocol (e.g., the SCSI and SBCCS command sets, IP, and other Upper Level Protocols (ULPs)). Fibre Channel provides a method for supporting a number of Upper Level Protocols (ULPs).

Of these levels, FC-1, FC-2, FC-3 and Link Services 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.

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

4.2 FC-1 general description

The Fibre Channel transmits information using an adaptive 8B/10B code. The encoding process results in the generation of Transmission Characters. The two types of Transmission Characters defined are data and special.

Certain combinations of Transmission Characters, referred to as Ordered Sets, are designated by this standard 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.

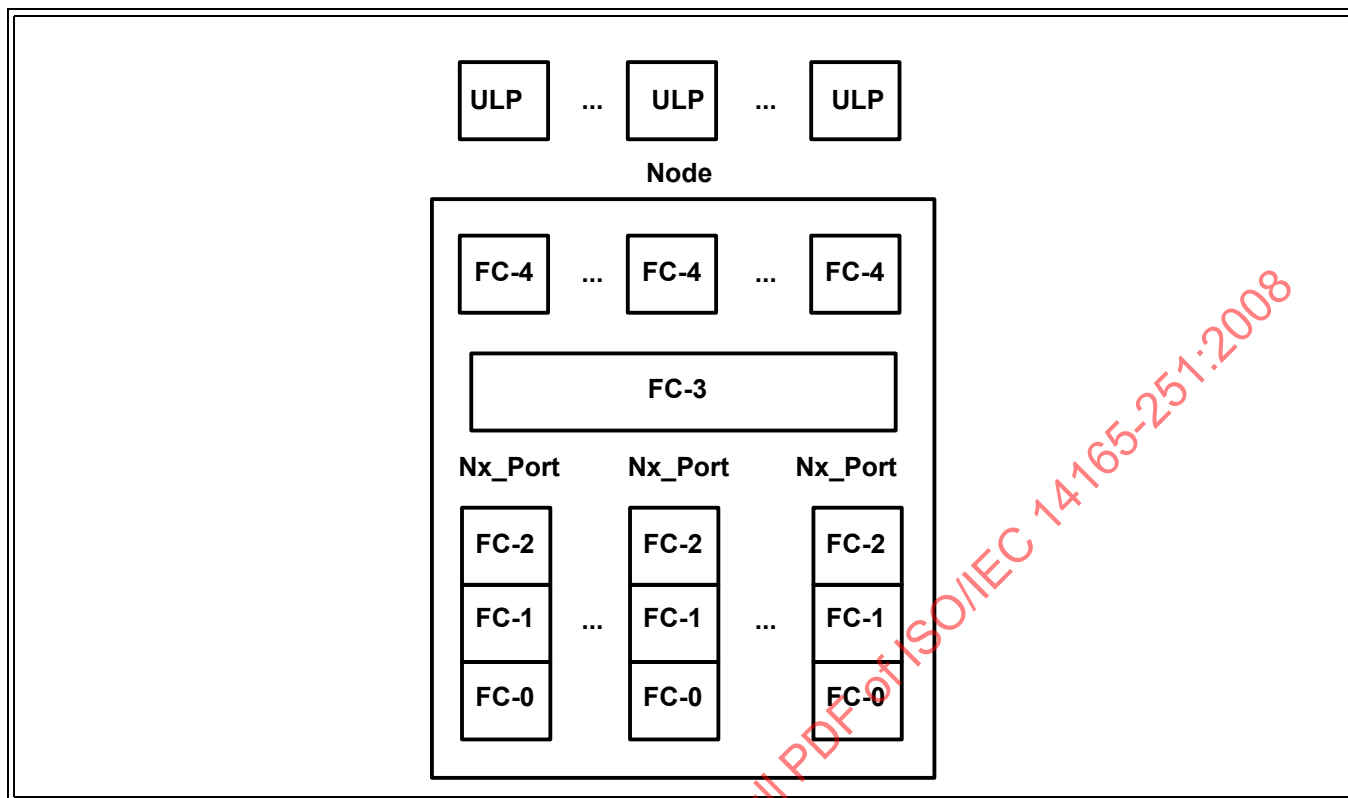


Figure 2 - Node functional configuration

4.3 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:

- a) node and Nx_Port and their identifiers,
- b) Communication models,
- c) Topologies based on the presence or absence of a Fabric,
- d) Classes of service provided by the Fabric and Nx_Ports,
- e) General Fabric model,
- f) Building Blocks and their hierarchy,
- g) Sequence and Sequence_Identifier,
- h) Exchange and Exchange_Identifier,
- i) Sequence Status Block,
- j) Exchange Status Block,
- k) Segmentation and reassembly, and
- l) Error detection and recovery.

In Class 1, an Nx_Port logically performs a point-to-point communication with another Nx_Port at any given time. This statement is true regardless of the presence of a Fabric. However, multiple Nx_Ports in a node are able to simultaneously perform data transfers in parallel with single or multiple Nx_Ports contained in one or more nodes (see 4.8.2 and 13.2).

In Class 2 and Class 3 service, an Nx_Port may multiplex frames to or demultiplex frames from multiple Nx_Ports (see 4.8.3, 4.8.4, 13.3, and 13.4).

In Class 4 service, an Nx_Port may multiplex frames to or demultiplex frames from multiple Nx_Ports through a fabric using a fractional bandwidth allocation protocol and Virtual Circuits (see 4.5.3, 4.8.5 and 13.6).

In Class 6, when a Switched Fabric is present that supports multicast, an Nx_Port may send a single frame to multiple destinations in an established multicast group of communicating Nx_Ports. It may also receive frames from other members of the same multicast group (see 4.8.6 and 13.7).

4.4 FC-FS physical model

Figure 3 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 Nx_Ports, each associated with a node, interconnected by a pair of fibres - one outbound and the other inbound at each Nx_Port. This pair of unidirectional fibres transmitting in opposite directions with their associated transmitters and receivers is referred to as a link. The link is used by the interconnected Nx_Ports to perform data communication.

Physical equipment (e.g., a processor, controller, or terminal) should be interconnected to other physical equipment through these links. Attached physical equipment supports one or more nodes and each node contains one or more Nx_Ports, with each Nx_Port containing a transmitter and a receiver.

The physical model shown in figure 3 is inherently capable of simultaneous, symmetrical bi-directional flow. A Fabric may be present between the Nx_Ports and some Fabrics may not support this type of flow. From the perspective of a given Nx_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.

This structure provides flexible mechanisms for attached equipment to perform simultaneous data transfers in parallel.

The Link Control facility (LCF) is a hardware facility that attaches to each end of a link and manages transmission and reception of data. It is contained within each Nx_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.

4.5 Communication models

4.5.1 Introduction

An Nx_Port transmits Data frames as a result of transfer requests made by an upper level at its end and receives the Link_Control responses for those Data frames. An Nx_Port receives Data frames from other Nx_Ports and transmits the appropriate Link_Control responses for those frames to the proper Nx_Ports.

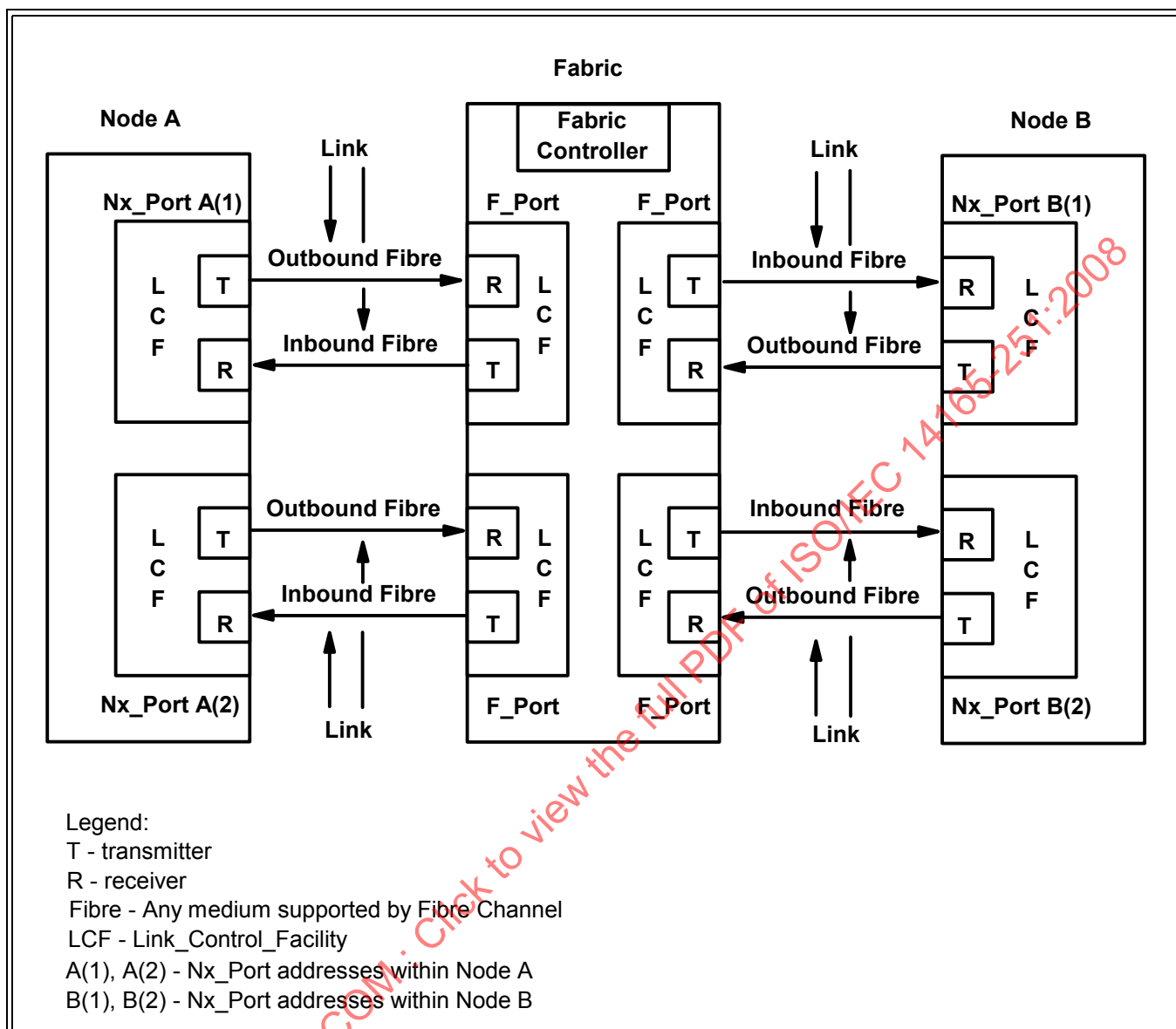


Figure 3 - FC-FS physical model

An Nx_Port may operate according to these communication models:

- Simplex operation is defined as an Nx_Port transferring Data frames in one direction only, with Link_Control frames flowing in the opposite direction;
- Full-duplex operation is defined as an Nx_Port simultaneously transmitting and receiving Data frames, with Link_Control frames flowing in both directions as well; or
- Half-duplex operation is defined as an Nx_Port both transmitting and receiving data, but not simultaneously. Data frames and Link_Control frames flow in both directions, but the flow is limited, to a single direction at a time.

4.5.2 Hunt Group

A Hunt Group is an enhancement to the FC-FS structure of Nx_Ports. Nx_Ports are addressed using N_Port_ID. A Hunt Group is a collection of Nx_Ports that may be addressed by either a group identifier (HG_ID) or N_Port_ID. Grouping Nx_Ports and addressing them as a group makes it possible to increase bandwidth and decrease latency by eliminating the need for sending individual frames to multiple destinations.

See clause 22 for details on Hunt Groups.

4.5.3 Fractional bandwidth

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

See clause 25 for details on Class 4.

4.6 Bandwidth

Effective transfer rate is a function of physical variants, the communication model, Payload size, fibre speed, class of service and overhead specified by this standard.

4.7 Topology

4.7.1 Types

Topologies are defined, based on the capability and the presence or absence of Fabric between the Nx_Ports. There are three basic types:

- a) Point-to-point topology
- b) Fabric topology
- c) Arbitrated Loop topology

The protocols specified herein are topology independent. However, attributes of the topology may restrict operation to certain communication models.

4.7.2 Point-to-point topology

The Point-to-point topology is shown in figure 4, in which communication between N_Ports occurs without the use of a Fabric

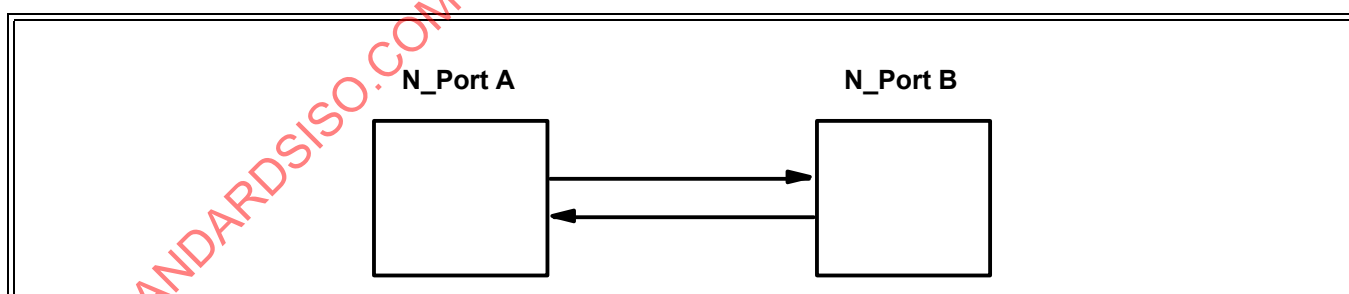


Figure 4 - Point-to-point topology

4.7.3 Fabric topology

The Fabric topology uses the D_ID embedded in the Frame_Header to route frames through a Fabric to the desired destination Nx_Port. Figure 5 illustrates multiple N_Ports interconnected by a Fabric.

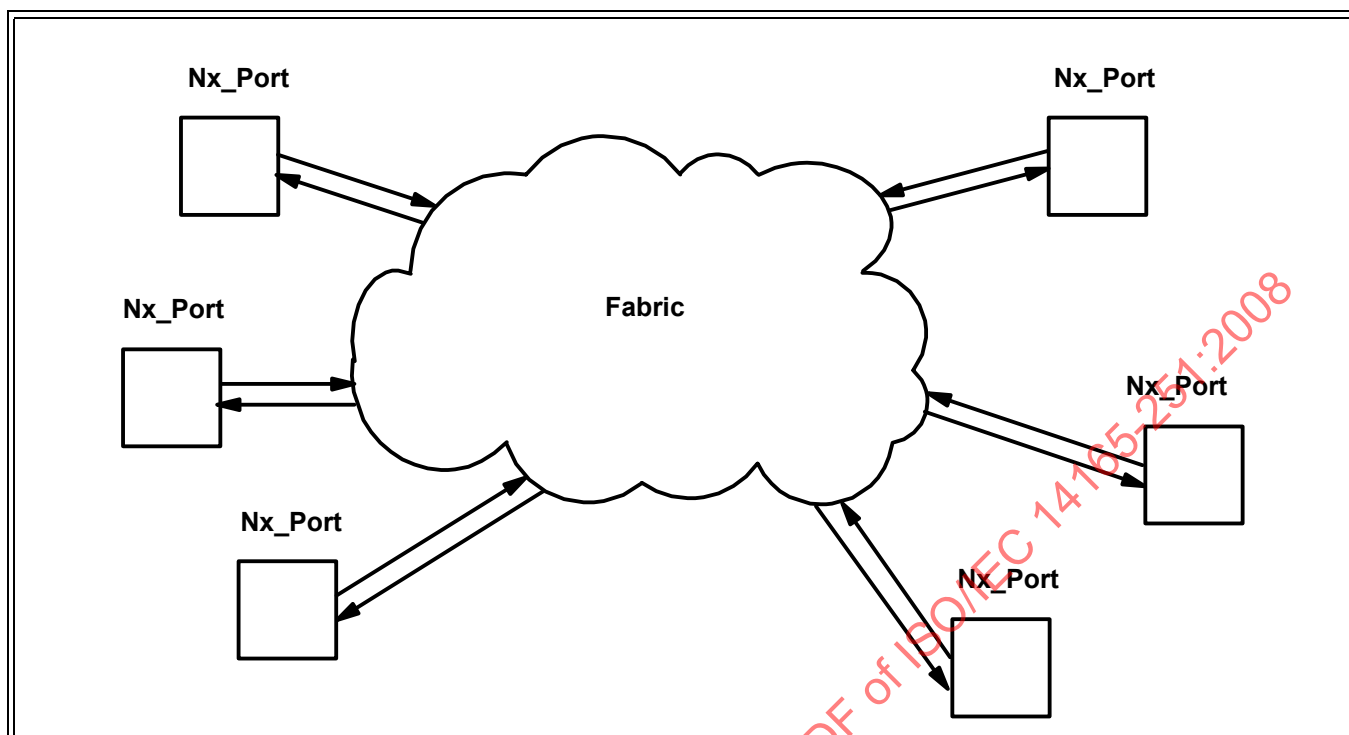


Figure 5 - Fabric topology

4.7.4 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 6 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 L_Ports. The second configuration shows an Arbitrated Loop composed of one FL_Port and three L_Ports. In this topology, additional FC_Ports may be attached to the fabric element.

4.8 Classes of service

4.8.1 General

Classes of service are distinguished primarily by the methodology used in allocating and retaining the communication circuit between the communicating Nx_Ports and the level of delivery integrity required for an application. Classes 1, 2, and 3 are topology independent. However, Classes 4 and 6 require a Fabric. If the Fabric is not present, the service is provided as a special case of point-to-point. FC_Ports are not required to support all classes of service.

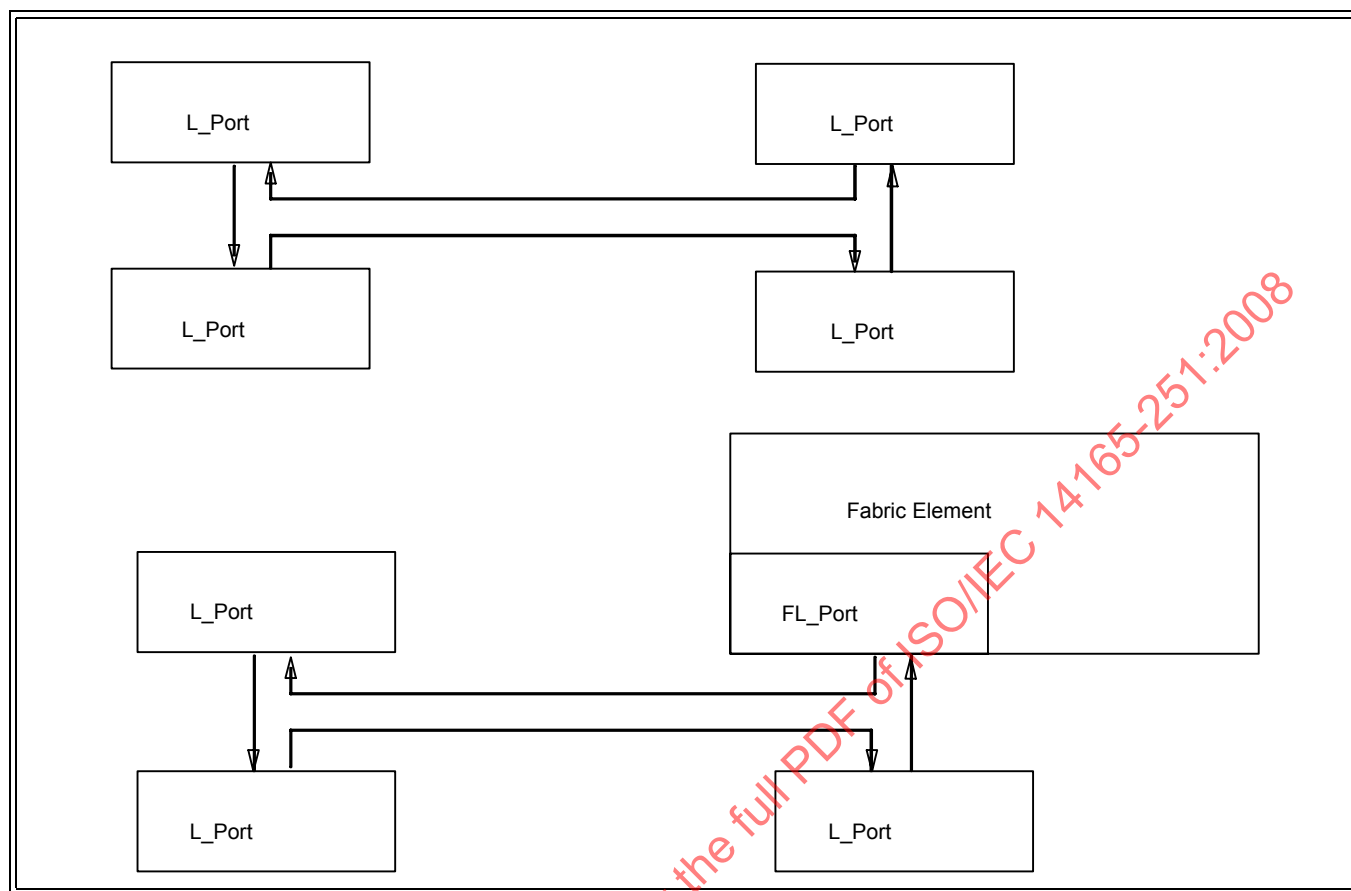


Figure 6 - Examples of the Arbitrated Loop Topology

4.8.2 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 the maximum bandwidth available between two Nx_Ports across the established connection.

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

4.8.3 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 or the destination Nx_Port guarantees notification of delivery in the absence of link errors. In case of link errors, notification is not guaranteed since the S_ID may not be error free.

4.8.4 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 Nx_Port does not send any confirmation of

Link_Control frames on receipt of valid Data frames. Any acknowledgement of Class 3 service is beyond the scope of this standard.

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.

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 Nx_Port if unable to deliver the frame.

4.8.5 Class 4 service – fractional bandwidth

Class 4 is a service that uses a virtual circuit established within a Fabric and between two communicating Nx_Ports to transmit frames to each other using a fabric-managed fractional bandwidth allocation protocol. This service requires a Fabric.

The transmitter transmits Class 4 Data frames in a sequential order within a given Sequence. In Class 4, frames are delivered to the destination Nx_Port by the Fabric in the same order as they are transmitted by the source Nx_Port.

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

4.8.6 Class 6 – multicast connection

Class 6 allows an Nx_Port to establish simultaneous dedicated connections with multiple Nx_Ports. Once established, these dedicated connections are retained and guaranteed by the Fabric. This service guarantees maximum bandwidth available from the source Nx_Port to each destination Nx_Port across the established connections.

Class 6 is inherently uni-directional. Data flows only from the source Nx_Port to the destination Nx_Ports. All destination Nx_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 Nx_Port. Frames are delivered to the destination Nx_Ports by the Fabric in the same order as they are transmitted by the source Nx_Port. This service requires a Fabric.

4.9 Intermixing other classes with Class 1 or Class 6

Intermix is an option of Class 1 and Class 6 service that allows interleaving of Class 2 and Class 3 frames during an established Connection between Nx_Ports. During a Class 1 or Class 6 Connection, an Nx_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 two connected Nx_Ports or between either of the connected Nx_Ports and other Nx_Ports. Support for the Intermix option of Class 1 and Class 6 service is indicated during Login (see 15.6.5.4.2). An Nx_Port that supports Intermix is capable of both transmitting and receiving intermixed frames. Class 4 frames are not permitted with the Intermix option.

In a point-to-point topology, both interconnected Nx_Ports are required to support Intermix if Intermix is to be used. In the presence of a Fabric, both the Nx_Port and the attached Fx_Port are required to support Intermix if Intermix is to be used. Fabric support for Intermix requires that the full bandwidth during a dedicated connection be maintained. Intermix permits the potentially unused Class 1 and Class 6 bandwidth to be used for transmission of Class 2 and Class 3 frames.

4.10 General Fabric model

4.10.1 General

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

A Fabric specifies the classes of service it supports in its Service Parameters (see 15.6.9.2). Fabrics are allowed to provide the classes of service through equivalent mechanisms and/or functions. See [7], (FC-SW-3) for the details.

Figure 7 illustrates the general Fabric model. The model is conceptual and may provide the following major functions:

- a) bi-directional Fabric Ports (Fx_Ports),
- b) receive buffer,
- c) connection Service,
- d) connectionless Service, and
- e) receive buffer queue management.

4.10.2 Fabric Ports (F_Ports)

The Fabric model contains two or more Fx_Ports. Each Fx_Port is attached to an Nx_Port through a link. Each Fx_Port is bi-directional and supports one or more communication models. Frames are routed to the Fx_Port attached to the destination Nx_Port.

The receiving F_Port responds to the sending Nx_Port according to the FC-2 protocol. The Fabric may verify the validity of the frame as it passes through the Fabric (See 8.7.3 and 8.9.2).

An F_Port may contain receive buffers for the incoming frames. The maximum frame size that the Fabric is able to handle for Class 2, Class 3, Class 4 and for the frame that requests a Class 1 connection is determined during Login. The frame sizes for Class 6 are the same as those for Class 1. One of the Fabric Service Parameters indicates the maximum frame size for the entire Fabric (see 15.6.2.6).

The Fabric routes the frame to the Fx_Port attached to the destination Nx_Port based on the value in the D_ID field embedded in the frame header of the frame. The routing mechanisms within the Fabric are transparent to Nx_Ports and are not specified in this standard.

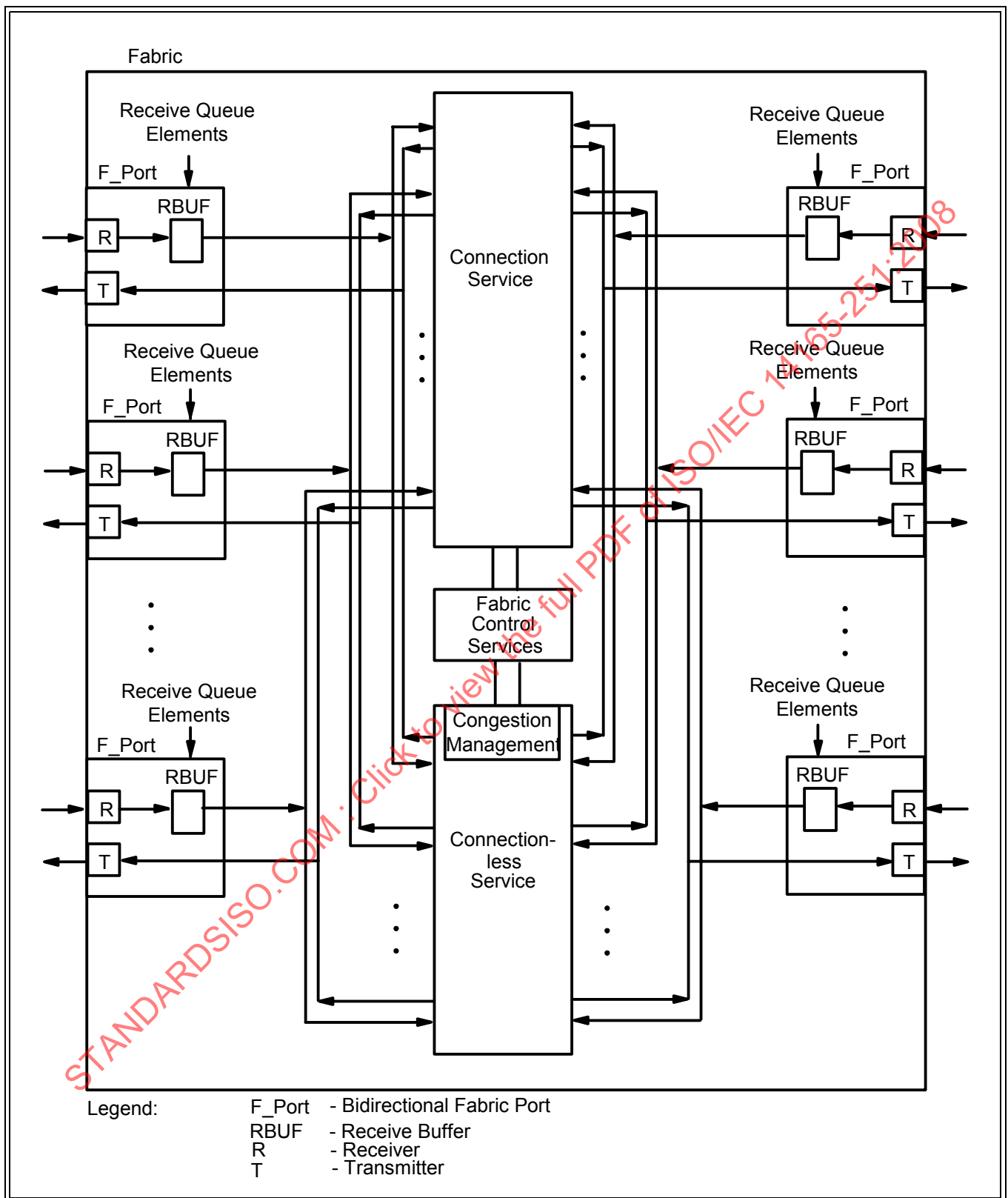


Figure 7 - Informative general Fabric model

4.10.3 Connection Service

The Connection Service provides dedicated connections between Fx_Ports and the Nx_Ports attached to these Fx_Ports. Such dedicated connections are bi-directional and support full transmission rate (except as specified for Class 6) concurrently in both directions. A dedicated connection is retained until a removal request is received from one of the communicating Nx_Ports or an exception condition occurs that causes the Fabric to remove the Connection.

On receiving a request from an Nx_Port, the Fabric establishes a dedicated connection to the destination Nx_Port through the Connection Service. The mechanisms used by the Fabric to establish the connection are transparent to FC-2.

The Fabric is not involved in flow control. Flow control is managed end-to-end by the Nx_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 Nx_Ports are routed through the same circuit and all responses are issued by Nx_Ports.

A Class 1 or Class 6 Nx_Port and the Fabric may support stacked connect-requests. This stacked connect-requests support allows an Nx_Port to request multiple dedicated connections to multiple destinations and allows the Fabric to service them in any order. While the Nx_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 Nx_Port and the Fabric support stacked connect-requests, when two Nx_Ports are in dedicated connection, a Class 1 connect-request from another source for one of these Nx_Ports is busied.

If Intermix is not supported and a Class 2 frame destined to one of the Nx_Ports established in a dedicated connection is received, the Class 2 frame shall be busied and the transmitting Nx_Port is notified. In the case of a Class 3 frame, the frame is discarded and no notification is sent.

If Intermix is supported and the Fabric receives a Class 2 or Class 3 frame destined to one of the Nx_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 frames (see 13.2).

4.10.4 Connectionless Service

The connectionless Service multiplexes frames at frame boundaries between an Fx_Port and any other Fx_Port and between the Nx_Ports attached to them. Connectionless Service does not guarantee full link bandwidth between communicating Nx_Ports.

The Fabric notifies the transmitting Nx_Port with a reason code embedded in a Link_Response frame, if it is unable to deliver a Class 2 or Class 4 frame. In the case of a Class 3 frame, the Fabric does not notify the transmitting Nx_Port.

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

For Classes 1, 2, 3 and 6, if any buffer-to-buffer flow control error occurs and as a result causes overflow (see 18.5), the Fabric logs the error and may discard the overflow frame without notification. Error logging is vendor specific.

For Class 4, if any VC_Credit flow control error occurs and as a result causes overflow (see 18.8), the Fabric logs the error and may discard the overflow frame without notification. Error logging is vendor specific.

4.11 Fibre Channel services

Fibre Channel services (e.g., Name Server) may be provided in a Fibre Channel configuration to meet the needs of the configuration. Each of these services is addressed with an N_Port_ID for the Nx_Port providing the service or with a well-known address (see 9.4.2). These well-known addresses are recognized and routed to by the Fabric. These services may be centralized or distributed (see [6], FC-GS-4).

4.12 Building Blocks

4.12.1 Building block hierarchy

The FC-2 building blocks are used in a hierarchical fashion, as illustrated in figure 8.

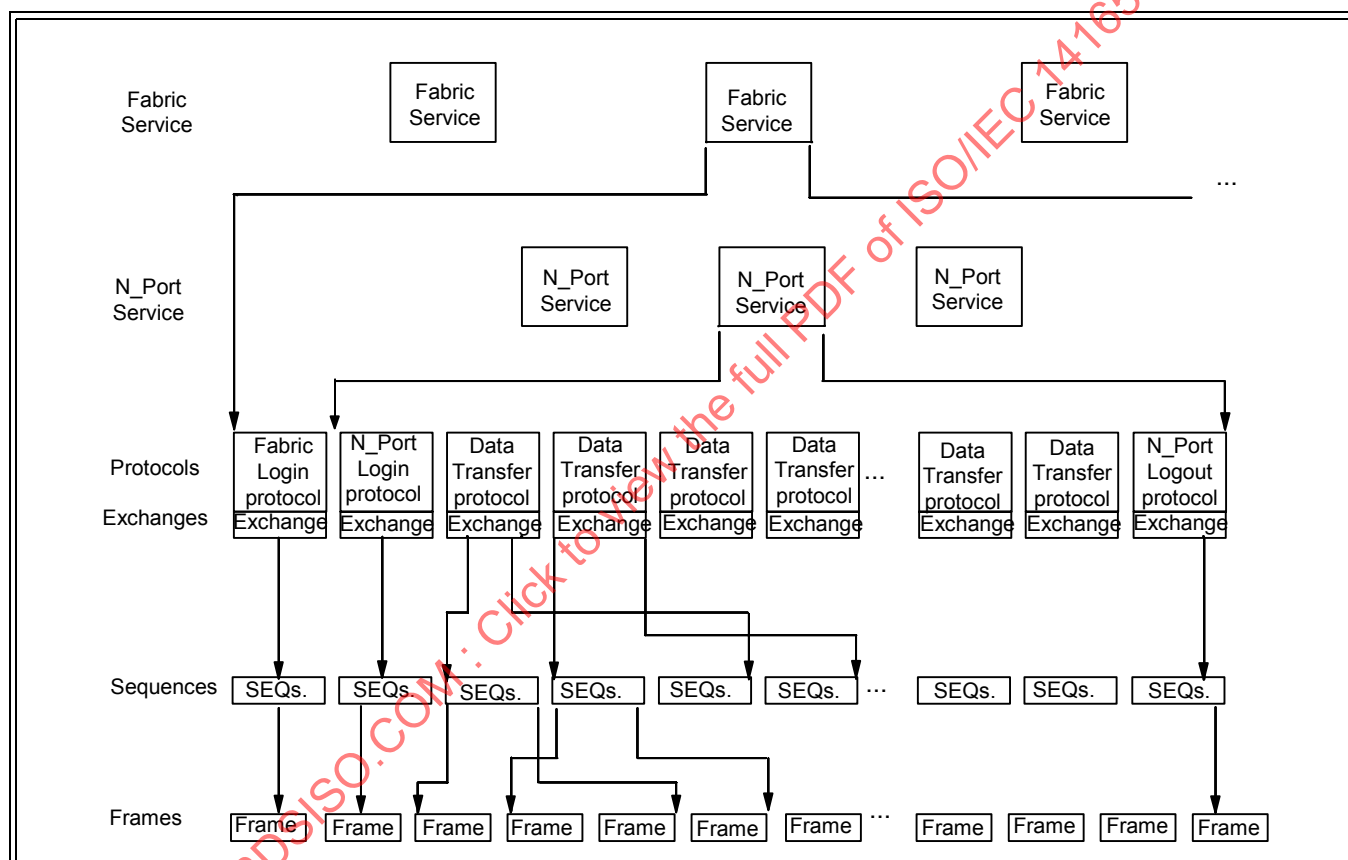


Figure 8 - FC-2 building block hierarchy

A Sequence is made up of one or more Data frames and if applicable, corresponding responses (see 17.6 and clause 11). 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 17).

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 Fabric Login and N_Port Login (see clause 15). Once these two Logins are performed, an FC-4 may start using the Fibre Channel until one or both of these Logins are invalidated.

Each Login uses an Exchange as the mechanism to accomplish the function. A data transfer is performed using an Exchange as the mechanism (see figure 8) with the related FC-4 translating the ULP protocol to FC-2 protocol.

4.12.2 Frame

Frames are based on a common frame format (see clause 8). Frames are categorized as Data frames and Link_Control command frames (see clause 11). Data frames (see clause 11) are classified as:

- a) Link_Data frames,
- b) Device_Data frames, and
- c) Video_Data frames.

Link_Control command frames (see 11.2) are classified as:

- a) Acknowledge (ACK) frames,
- b) Link_Response (Busy and Reject) frames, and
- c) Link_Control Command frames.

Selective retransmission of frames for error recovery is not supported in this standard (see clause 21). However, an individual frame may be busied in Class 2 and the sender may later retransmit the busied frame (see 11.2.3.2 and 4.12.3.3) up to the ability of the sender to retry (see 13.4). The number of times the sender may retry is not specified in this standard.

4.12.3 Sequence

4.12.3.1 Introduction

A Sequence is a set of one or more related Data frames transmitted unidirectionally from one Nx_Port to another Nx_Port with corresponding Link_Control frames, if specified, transmitted in response. An Nx_Port that transmits a Sequence is referred to as the Sequence Initiator and the Nx_Port that receives the Sequence is referred to as the Sequence Recipient. Sequence rules are specified in 17.6.

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 containing the frame may be retransmitted (see clause 21).

4.12.3.2 Sequence_Identifier (SEQ_ID)

The Sequence Initiator assigns to 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 Nx_Ports assigns SEQ_IDs independent of the other.

4.12.3.3 Sequence Status Blocks

A Sequence Status Block (SSB) is a logical construct representing the format of the Sequence status information (see 17.8.2). It is used to track the progress of a Sequence at an Nx_Port on a frame by frame basis. A Sequence Initiator SSB and a Sequence Recipient SSB are used by the respective Nx_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 that the Sequence Initiator has initiated. Both the Sequence Initiator and Sequence Recipient Nx_Ports track the status of the Sequence through the Sequence Initiator and the Sequence Recipient SSBs, respectively.

The maximum number of concurrent Sequences between two Nx_Ports is limited to the smaller of the number of SSBs available at these Nx_Ports. This value is established during N_Port Login through the Service Parameters (see 12.3.2.7).

4.12.4 Exchange

4.12.4.1 Introduction

An Exchange is composed of one or more non-concurrent Sequences (see clause 17). An Exchange may be unidirectional or bi-directional. A unidirectional Exchange results when the same Nx_Port initiates all the Sequences within the Exchange. A bi-directional Exchange results when the Sequences within the Exchange are initiated by both the Nx_Ports, but not concurrently.

An FC-4 may achieve full bandwidth utilization between two Nx_Ports by supporting two or more Exchanges concurrently with the two Nx_Ports using different Exchanges to transmit information. Coordination of the Exchanges is FC-4 specific. All frames and Sequences of a given Exchange shall be performed between the two Nx_Ports that first originated and received the Exchange.

Exchanges are used by upper levels to relate sequences.

4.12.4.2 Exchange_Identifiers (OX_ID and RX_ID)

Exchange_Identifiers shall 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 may assign a Responder Exchange_ID (RX_ID) that is 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 17.5). 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 it originates.

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

4.12.4.3 Association_Header

In some system models, there is an association between an Exchange and higher-level system constructs. A specific process or group of processes in such system models is identified using the Association_Header. The format of the Association_Header is specified in 10.4.

4.12.4.4 Exchange Status Blocks

An Exchange Status Block (ESB) is a logical construct representing the format of the Exchange status information (see 17.8.1). It is used to track the progress of an Exchange on a Sequence by Sequence basis. An Originator and a Responder use an Originator ESB and a Responder ESB, respectively, to track the status of an Exchange.

When an Originator initiates an Exchange, it assigns an Originator ESB associated with the Exchange (see 17.8.1).

The Responder assigns a Responder ESB to the Exchange. The Responder references the Responder ESB through the fully qualified X_ID (see 17.8.1 and [9] FCP-3).

Both the Originator and the Responder track the status of the Exchange at their respective Nx_Ports.

4.12.5 Exchange of service parameters

A Primitive Sequence is an Ordered Set that is transmitted repeatedly and continuously. Primitive Sequence protocols are based on Primitive Sequences and specified for Link Failure, Link Initialization, Link Reset, and

Online-to-Offline transition (see 7.6).

In order to manage the Fabric operating environment, an Nx_Port interchanges its Service Parameters with the Fabric, if present, by performing Fabric Login. This may be done explicitly or implicitly (see 15.1).

Before performing data transfer, the Nx_Port exchanges its Service Parameters with another Nx_Port by performing an implicit or explicit N_Port Login (see 15.1). An Nx_Port may request removal of its Service Parameters from the other Nx_Port by performing an N_Port Logout protocol (see 15.5). This request may be used to free up resources at the other Nx_Port.

The upper level data is transferred using Data transfer protocols.

4.13 Segmentation and reassembly

4.13.1 General

Segmentation and reassembly (see clause 19) are the FC-2 functions provided to subdivide Information Units to be transferred into Payloads, embed each Payload in an individual frame, transfer these frames over the link and reassemble the Information Units at the receiving end as sent by the sending end.

SEQ_CNT shall be used to perform the segmentation and reassembly.

4.13.2 Application data mapping

Mapping application data to FC-4 Information Units and other transformations within the ULP layers are outside the scope of this standard.

4.13.3 Relative offset

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

If relative offset is not supported (see clauses 15 and 19), then SEQ_CNT shall be used to perform the segmentation and reassembly.

4.13.4 Sending end mapping

Upper levels at the sending end specify to FC-2

- a) data blocks to be transferred within a Sequence,
- b) the Information Category for each data block (see 9.3.3),
- c) relative offset for each Information Category (see 9.3.3) if required, and
- d) an initial relative offset for each data block to be transferred (see 9.13) if required.

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

4.13.5 Capability

The Sequence Recipient indicates during N_Port Login its capability to support continuously increasing (see 15.6.2.4.1) or random relative offset (see 15.6.2.4.3). If only the former is supported, each Information Category transferred within a Sequence shall be treated as a data block by upper levels. If random relative offset is supported, an Information Category may be specified as data blocks by upper levels and portions of the data blocks may be transmitted in a random order.

4.13.6 FC-2 mapping

If Continuously Increasing Relative Offset for a Sequence is supported, a data block shall not be divided into portions. With continuously increasing relative offset, the FC-2 maps each data block as a single Information Category within the relative offset space specified. The FC-2 implementation may support mapping one or more Information Categories into a single Sequence as specified. If multiple information categories per sequence is not supported, each information category is transmitted as a separate Sequence.

If Random Relative Offset for a Sequence is supported, a data block may be divided into portions and the upper level specifies the initial relative offset of each portion; the initial relative offsets of successive data blocks are allowed to vary as required by the upper level. With random relative offset, the FC-2 maps portions of a data block corresponding to an Information Category into a single Sequence within the relative offset space as specified.

4.13.7 Segmentation

Segmentation is the process by which the FC-2 subdivides a data block into frames with Payloads of equal or varied sizes and transmits them over the link. During segmentation, the FC-2 determines the size of the 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 data 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 data block is computed as the sum of relative offset and the Payload length of the current frame.

4.13.8 Reassembly

Reassembly is the FC-2 process that reassembles each data block 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 19). Reassembly is performed on an Information Category basis.

4.14 Error detection and recovery

In general, detected errors fall into two broad categories: frame errors and link-level errors.

Frame errors result from missing frames or corrupted frames. Corrupted frames are discarded and for corrupted frames 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 (see 21.6.3) 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 upper level. 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 may be resolved after recovery at the link-level.

See clause 21 for detailed error detection and recovery requirements.

5 FC-1 8B/10B transmission code

5.1 Introduction

Information to be transmitted over a Fibre shall be encoded eight bits at a time into an 10-bit Transmission Character and then sent serially bit-by-bit across the link. Information received over the link 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 that are to be distinguishable from the contents of a frame.

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) that assists a receiver in achieving word alignment on the incoming bit stream.

5.2 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. An encoded bit contains either a binary zero or a binary one. 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 unencoded information bit labeled A corresponds to bit 0 in the bit numbering scheme of the FC-2 specification, B corresponds to bit 1, and so on, as shown in table 4. 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).

Table 4 - Bit designations

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

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. The value of Z is used to indicate whether the Transmission Character is a data character (Z = D) or a special character (Z = K);
- 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.

Table 5 shows an example of the conversion from FC-2 byte notation to the FC-1 Transmission Character naming convention described above.

Table 5 - Conversion Example

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

Most Kxx.y combinations do not result in valid Transmission Characters within the 8B/10B transmission code. Only those combinations that result in special characters as specified by table 7, are considered valid.

5.3 Character encoding and decoding

5.3.1 Introduction

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).

5.3.2 Transmission order

Within the definition of the 8B/10B transmission code, the bit positions of the 10 bit 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. 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 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 shall be transmitted sequentially beginning with the Ordered Set used to denote the start of frame (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 (EOF delimiter) is transmitted.

5.3.3 Valid and invalid Transmission Characters

5.3.3.1 Definitions

Tables 6 and 7 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 (+). The running disparity at the beginning of an Ordered Set is the beginning running disparity (beginning RD).

After powering on, the transmitter shall initialize the Current RD to negative. 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 and the Running Disparity at the beginning of the Transmission Character.

After powering on or exiting diagnostic mode (the definition of diagnostic mode is beyond the scope of this standard), 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 (see table 5.3.3 and table 6) and shall calculate a new value for its running disparity based on the contents of the received character and the Running Disparity at the beginning of the received Transmission Character.

The following rules for running disparity shall be used to calculate the new running disparity value for Transmission Characters that have been transmitted (i.e., transmitter's running disparity) and that have been received (i.e., 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' b) form one sub-block (six-bit sub-block) and the second four bits ('fghj' b) 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 sub-block 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:

- a) Running disparity at the end of any sub-block is positive
 - 1) if the sub-block contains more ones than zeros,
 - 2) if at the end of the six-bit sub-block, the six-bit sub-block is '000111' b, or
 - 3) if at the end of the four-bit sub-block, the four-bit sub-block is '0011' b.
- d) Running disparity at the end of any sub-block is negative
 - 1) if the sub-block contains more zeros than ones,
 - 2) if at the end of the six-bit sub-block, the six-bit sub-block is '111000' b, or
 - 3) if at the end of the four-bit sub-block, the four-bit sub-block is '1100' b.
- d) Otherwise, running disparity at the end of the sub-block is the same as at the beginning of the sub-block.

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

Table 6 - Valid Data Characters

Data Byte Name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD+ abcdei fghj (binary)	Data Byte Name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD+ abcdei fghj (binary)
D00.0	000 00000	100111 0100	011000 1011	D00.1	001 00000	100111 1001	011000 1001
D01.0	000 00001	011101 0100	100010 1011	D01.1	001 00001	011101 1001	100010 1001
D02.0	000 00010	101101 0100	010010 1011	D02.1	001 00010	101101 1001	010010 1001
D03.0	000 00011	110001 1011	110001 0100	D03.1	001 00011	110001 1001	110001 1001
D04.0	000 00100	110101 0100	001010 1011	D04.1	001 00100	110101 1001	001010 1001
D05.0	000 00101	101001 1011	101001 0100	D05.1	001 00101	101001 1001	101001 1001
D06.0	000 00110	011001 1011	011001 0100	D06.1	001 00110	011001 1001	011001 1001
D07.0	000 00111	111000 1011	000111 0100	D07.1	001 00111	111000 1001	000111 1001
D08.0	000 01000	111001 0100	000110 1011	D08.1	001 01000	111001 1001	000110 1001
D09.0	000 01001	100101 1011	100101 0100	D09.1	001 01001	100101 1001	100101 1001
D10.0	000 01010	010101 1011	010101 0100	D10.1	001 01010	010101 1001	010101 1001
D11.0	000 01011	110100 1011	110100 0100	D11.1	001 01011	110100 1001	110100 1001
D12.0	000 01100	001101 1011	001101 0100	D12.1	001 01100	001101 1001	001101 1001
D13.0	000 01101	101100 1011	101100 0100	D13.1	001 01101	101100 1001	101100 1001
D14.0	000 01110	011100 1011	011100 0100	D14.1	001 01110	011100 1001	011100 1001
D15.0	000 01111	010111 0100	101000 1011	D15.1	001 01111	010111 1001	101000 1001
D16.0	000 10000	011011 0100	100100 1011	D16.1	001 10000	011011 1001	100100 1001
D17.0	000 10001	100011 1011	100011 0100	D17.1	001 10001	100011 1001	100011 1001
D18.0	000 10010	010011 1011	010011 0100	D18.1	001 10010	010011 1001	010011 1001
D19.0	000 10011	110010 1011	110010 0100	D19.1	001 10011	110010 1001	110010 1001
D20.0	000 10100	001011 1011	001011 0100	D20.1	001 10100	001011 1001	001011 1001
D21.0	000 10101	101010 1011	101010 0100	D21.1	001 10101	101010 1001	101010 1001
D22.0	000 10110	011010 1011	011010 0100	D22.1	001 10110	011010 1001	011010 1001
D23.0	000 10111	111010 0100	000101 1011	D23.1	001 10111	111010 1001	000101 1001
D24.0	000 11000	110011 0100	001100 1011	D24.1	001 11000	110011 1001	001100 1001
D25.0	000 11001	100110 1011	100110 0100	D25.1	001 11001	100110 1001	100110 1001
D26.0	000 11010	010110 1011	010110 0100	D26.1	001 11010	010110 1001	010110 1001
D27.0	000 11011	110110 0100	001001 1011	D27.1	001 11011	110110 1001	001001 1001
D28.0	000 11100	001110 1011	001110 0100	D28.1	001 11100	001110 1001	001110 1001
D29.0	000 11101	101110 0100	010001 1011	D29.1	001 11101	101110 1001	010001 1001
D30.0	000 11110	011110 0100	100001 1011	D30.1	001 11110	011110 1001	100001 1001
D31.0	000 11111	101011 0100	010100 1011	D31.1	001 11111	101011 1001	010100 1001
D00.2	010 00000	100111 0101	011000 0101	D00.3	011 00000	100111 0011	011000 1100
D01.2	010 00001	011101 0101	100010 0101	D01.3	011 00001	011101 0011	100010 1100

Table 6 - Valid Data Characters (Continued)

Data Byte Name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD+ abcdei fghj (binary)	Data Byte Name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD+ abcdei fghj (binary)
D02.2	010 00010	101101 0101	010010 0101	D02.3	011 00010	101101 0011	010010 1100
D03.2	010 00011	110001 0101	110001 0101	D03.3	011 00011	110001 1100	110001 0011
D04.2	010 00100	110101 0101	001010 0101	D04.3	011 00100	110101 0011	001010 1100
D05.2	010 00101	101001 0101	101001 0101	D05.3	011 00101	101001 1100	101001 0011
D06.2	010 00110	011001 0101	011001 0101	D06.3	011 00110	011001 1100	011001 0011
D07.2	010 00111	111000 0101	000111 0101	D07.3	011 00111	111000 1100	000111 0011
D08.2	010 01000	111001 0101	000110 0101	D08.3	011 01000	111001 0011	000110 1100
D09.2	010 01001	100101 0101	100101 0101	D09.3	011 01001	100101 1100	100101 0011
D10.2	010 01010	010101 0101	010101 0101	D10.3	011 01010	010101 1100	010101 0011
D11.2	010 01011	110100 0101	110100 0101	D11.3	011 01011	110100 1100	110100 0011
D12.2	010 01100	001101 0101	001101 0101	D12.3	011 01100	001101 1100	001101 0011
D13.2	010 01101	101100 0101	101100 0101	D13.3	011 01101	101100 1100	101100 0011
D14.2	010 01110	011100 0101	011100 0101	D14.3	011 01110	011100 1100	011100 0011
D15.2	010 01111	010111 0101	101000 0101	D15.3	011 01111	010111 0011	101000 1100
D16.2	010 10000	011011 0101	100100 0101	D16.3	011 10000	011011 0011	100100 1100
D17.2	010 10001	100011 0101	100011 0101	D17.3	011 10001	100011 1100	100011 0011
D18.2	010 10010	010011 0101	010011 0101	D18.3	011 10010	010011 1100	010011 0011
D19.2	010 10011	110010 0101	110010 0101	D19.3	011 10011	110010 1100	110010 0011
D20.2	010 10100	001011 0101	001011 0101	D20.3	011 10100	001011 1100	001011 0011
D21.2	010 10101	101010 0101	101010 0101	D21.3	011 10101	101010 1100	101010 0011
D22.2	010 10110	011010 0101	011010 0101	D22.3	011 10110	011010 1100	011010 0011
D23.2	010 10111	111010 0101	000101 0101	D23.3	011 10111	111010 0011	000101 1100
D24.2	010 11000	110011 0101	001100 0101	D24.3	011 11000	110011 0011	001100 1100
D25.2	010 11001	100110 0101	100110 0101	D25.3	011 11001	100110 1100	100110 0011
D26.2	010 11010	010110 0101	010110 0101	D26.3	011 11010	010110 1100	010110 0011
D27.2	010 11011	110110 0101	001001 0101	D27.3	011 11011	110110 0011	001001 1100
D28.2	010 11100	001110 0101	001110 0101	D28.3	011 11100	001110 1100	001110 0011
D29.2	010 11101	101110 0101	010001 0101	D29.3	011 11101	101110 0011	010001 1100
D30.2	010 11110	011110 0101	100001 0101	D30.3	011 11110	011110 0011	100001 1100
D31.2	010 11111	101011 0101	010100 0101	D31.3	011 11111	101011 0011	010100 1100
D00.4	100 00000	100111 0010	011000 1101	D00.5	101 00000	100111 1010	011000 1010
D01.4	100 00001	011101 0010	100010 1101	D01.5	101 00001	011101 1010	100010 1010
D02.4	100 00010	101101 0010	010010 1101	D02.5	101 00010	101101 1010	010010 1010
D03.4	100 00011	110001 1101	110001 0010	D03.5	101 00011	110001 1010	110001 1010

Table 6 - Valid Data Characters (Continued)

Data Byte Name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD+ abcdei fghj (binary)	Data Byte Name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD+ abcdei fghj (binary)
D04.4	100 00100	110101 0010	001010 1101	D04.5	101 00100	110101 1010	001010 1010
D05.4	100 00101	101001 1101	101001 0010	D05.5	101 00101	101001 1010	101001 1010
D06.4	100 00110	011001 1101	011001 0010	D06.5	101 00110	011001 1010	011001 1010
D07.4	100 00111	111000 1101	000111 0010	D07.5	101 00111	111000 1010	000111 1010
D08.4	100 01000	111001 0010	000110 1101	D08.5	101 01000	111001 1010	000110 1010
D09.4	100 01001	100101 1101	100101 0010	D09.5	101 01001	100101 1010	100101 1010
D10.4	100 01010	010101 1101	010101 0010	D10.5	101 01010	010101 1010	010101 1010
D11.4	100 01011	110100 1101	110100 0010	D11.5	101 01011	110100 1010	110100 1010
D12.4	100 01100	001101 1101	001101 0010	D12.5	101 01100	001101 1010	001101 1010
D13.4	100 01101	101100 1101	101100 0010	D13.5	101 01101	101100 1010	101100 1010
D14.4	100 01110	011100 1101	011100 0010	D14.5	101 01110	011100 1010	011100 1010
D15.4	100 01111	010111 0010	101000 1101	D15.5	101 01111	010111 1010	101000 1010
D16.4	100 10000	011011 0010	100100 1101	D16.5	101 10000	011011 1010	100100 1010
D17.4	100 10001	100011 1101	100011 0010	D17.5	101 10001	100011 1010	100011 1010
D18.4	100 10010	010011 1101	010011 0010	D18.5	101 10010	010011 1010	010011 1010
D19.4	100 10011	110010 1101	110010 0010	D19.5	101 10011	110010 1010	110010 1010
D20.4	100 10100	001011 1101	001011 0010	D20.5	101 10100	001011 1010	001011 1010
D21.4	100 10101	101010 1101	101010 0010	D21.5	101 10101	101010 1010	101010 1010
D22.4	100 10110	011010 1101	011010 0010	D22.5	101 10110	011010 1010	011010 1010
D23.4	100 10111	111010 0010	000101 1101	D23.5	101 10111	111010 1010	000101 1010
D24.4	100 11000	110011 0010	001100 1101	D24.5	101 11000	110011 1010	001100 1010
D25.4	100 11001	100110 1101	100110 0010	D25.5	101 11001	100110 1010	100110 1010
D26.4	100 11010	010110 1101	010110 0010	D26.5	101 11010	010110 1010	010110 1010
D27.4	100 11011	110110 0010	001001 1101	D27.5	101 11011	110110 1010	001001 1010
D28.4	100 11100	001110 1101	001110 0010	D28.5	101 11100	001110 1010	001110 1010
D29.4	100 11101	101110 0010	010001 1101	D29.5	101 11101	101110 1010	010001 1010
D30.4	100 11110	011110 0010	100001 1101	D30.5	101 11110	011110 1010	100001 1010
D31.4	100 11111	101011 0010	010100 1101	D31.5	101 11111	101011 1010	010100 1010
D00.6	110 00000	100111 0110	011000 0110	D00.7	111 00000	100111 0001	011000 1110
D01.6	110 00001	011101 0110	100010 0110	D01.7	111 00001	011101 0001	100010 1110
D02.6	110 00010	101101 0110	010010 0110	D02.7	111 00010	101101 0001	010010 1110
D03.6	110 00011	110001 0110	110001 0110	D03.7	111 00011	110001 1110	110001 0001
D04.6	110 00100	110101 0110	001010 0110	D04.7	111 00100	110101 0001	001010 1110
D05.6	110 00101	101001 0110	101001 0110	D05.7	111 00101	101001 1110	101001 0001

Table 6 - Valid Data Characters (Continued)

Data Byte Name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD+ abcdei fghj (binary)	Data Byte Name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD+ abcdei fghj (binary)
D06.6	110 00110	011001 0110	011001 0110	D06.7	111 00110	011001 1110	011001 0001
D07.6	110 00111	111000 0110	000111 0110	D07.7	111 00111	111000 1110	000111 0001
D08.6	110 01000	111001 0110	000110 0110	D08.7	111 01000	111001 0001	000110 1110
D09.6	110 01001	100101 0110	100101 0110	D09.7	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	010111 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
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 7 - Valid Special Characters

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

5.3.3.2 Generating Transmission Characters

The appropriate entry in the table shall be found for the valid data byte or special code used in generating (encoding) a Transmission Character. 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.

5.3.3.3 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.

Detection of a code violation does not necessarily indicate that the Transmission Character where the code violation was detected is in error. Code violations may result from a prior error that altered the running disparity of the bit stream but that did not result in a detectable error at the Transmission Character where the error occurred. The example shown in table 8 exhibits this behavior.

Table 8 - Delayed Code Violation example

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

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.

5.4 Word encoding and decoding

A Transmission Word is composed of four contiguous valid or invalid Transmission Characters treated as a unit. Four valid data bytes and special codes shall be encoded according to the rules specified by 5.3 to create a Transmission Word. Likewise, the Transmission Characters of a Transmission Word shall be decoded according to the rules specified by 5.3 to create valid data bytes and special codes.

5.5 Ordered Sets

5.5.1 General

An Ordered Set is a four-character combination of data and special Transmission Characters. Ordered Sets provide the ability to obtain bit and word synchronization that also establishes word boundary alignment. See 6.1.4.2 for the synchronization rules.

If an unrecognized Ordered Set is detected, it shall be treated as an Idle Primitive Signal. Treating unrecognized Ordered Sets as Idles allows future introduction of Ordered Sets for additional features and functions beyond the scope of this standard.

Each EOF-delimiter Ordered Set is defined such that negative current running disparity shall result after processing of the final (right-most) character of the Ordered Set. This, in combination with the running disparity initialization rules, ensures that the first Ordered Set following an EOF delimiter, transmitter power on, or transmitter exit from diagnostic mode (the definition of diagnostic mode is beyond the scope of this standard) shall 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 are always 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 Idles 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.

The K28.5 special character is used as the first character of all Ordered Sets defined in this standard for the following reasons:

- Bits abcdef make up a comma; this is a singular bit pattern that in the absence of transmission errors shall not appear in any other location of a Transmission Character and shall not be generated across the boundaries of any two adjacent Transmission Characters. The comma should be used to easily find and verify character and word boundaries of the received bit stream.

- b) 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.

The third and fourth characters of the Delimiter functions, Receiver_Ready, and the Idle are repeated to ensure that an error affecting a single character shall not result in the recognition of an Ordered Set other than the one transmitted.

For some Primitive Signals and Primitive Sequences, the second byte of the Ordered Set specifies the function of the Ordered Set. Bytes 3 and 4 of the Ordered Set are used to carry parameter information. The receiving FC_Ports analyze the parameter information before taking any action. Other Primitive Signals and Sequences are defined in FC-AL-2.

5.5.2 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 and shall be recognized when a single Ordered Set is detected. The frame delimiters are listed in table 9. See 8.3 and 8.7 for the usage of each.

5.5.3 Primitive Signals

5.5.3.1 Introduction

A Primitive Signal is an Ordered Set designated by this standard to have special meaning. All FC_Ports shall at a minimum recognize R_RDY and IDLE Primitive Signals. All Primitive Signals not recognized by the FC_Port shall be treated as an IDLE. When a single Ordered Set is detected possible Primitive Signals detected are listed in table 10.

To assure a sufficient number of Idles between frames, the originator of any Primitive Signal (except ARByx, ARB(val), MRK, SYNx, SYNy, and SYNz) shall precede and follow the Primitive Signal by a minimum of two Idles. Because Idles may be removed by intermediate transmitters, the number of Idles preceding or following a Primitive Signal at a receiver may be reduced to zero.

5.5.3.2 Idle

Idle is a Primitive Signal transmitted on the link to indicate that link initialization is complete and to maintain link synchronization. Idles shall be transmitted on the link during periods of time when frames, other Primitive Signals or Primitive Sequences are not required to be transmitted. See 8.2 for the requirements for the insertion of Idle between frames.

5.5.3.3 Receiver_Ready (R_RDY)

The R_RDY Primitive Signal is used in the buffer-to-buffer credit management mechanisms described in 18.5.

Validity of the frame is not implied by R_RDY. See 8.2 for frame spacing requirements.

Table 9 - Frame Delimiters

Abbr.	Delimiter Function	Beginning RD	Ordered Set
SOF _{c1}	SOF Connect Class 1	Negative	K28.5 - D21.5 - D23.0 - D23.0
SOF _{i1}	SOF Initiate Class 1	Negative	K28.5 - D21.5 - D23.2 - D23.2
SOF _{n1}	SOF Normal Class 1	Negative	K28.5 - D21.5 - D23.1 - D23.1
SOF _{i2}	SOF Initiate Class 2	Negative	K28.5 - D21.5 - D21.2 - D21.2
SOF _{n2}	SOF Normal Class 2	Negative	K28.5 - D21.5 - D21.1 - D21.1
SOF _{i3}	SOF Initiate Class 3	Negative	K28.5 - D21.5 - D22.2 - D22.2
SOF _{n3}	SOF Normal Class 3	Negative	K28.5 - D21.5 - D22.1 - D22.1
SOF _{c4}	SOF Activate Class 4	Negative	K28.5 - D21.5 - D25.0 - D25.0
SOF _{i4}	SOF Initiate Class 4	Negative	K28.5 - D21.5 - D25.2 - D25.2
SOF _{n4}	SOF Normal Class 4	Negative	K28.5 - D21.5 - D25.1 - D25.1
SOF _f	SOF Fabric	Negative	K28.5 - D21.5 - D24.2 - D24.2
EOF _t	EOF Terminate	Negative	K28.5 - D21.4 - D21.3 - D21.3
		Positive	K28.5 - D21.5 - D21.3 - D21.3
EOF _{dt}	EOF Disconnect-Terminate-Class 1 or EOF Deactivate-Terminate-Class 4	Negative	K28.5 - D21.4 - D21.4 - D21.4
		Positive	K28.5 - D21.5 - D21.4 - D21.4
EOF _a	EOF Abort	Negative	K28.5 - D21.4 - D21.7 - D21.7
		Positive	K28.5 - D21.5 - D21.7 - D21.7
EOF _n	EOF Normal	Negative	K28.5 - D21.4 - D21.6 - D21.6
		Positive	K28.5 - D21.5 - D21.6 - D21.6
EOF _{ni}	EOF Normal-Invalid	Negative	K28.5 - D10.4 - D21.6 - D21.6
		Positive	K28.5 - D10.5 - D21.6 - D21.6
EOF _{dti}	EOF Disconnect-Terminate-Invalid Class 1 or EOF Disconnect-Deactivate-Invalid Class 4	Negative	K28.5 - D10.4 - D21.4 - D21.4
		Positive	K28.5 - D10.5 - D21.4 - D21.4
EOF _r	EOF Remove-Terminate Class 4	Negative	K28.5 - D21.4 - D25.4 - D25.4
		Positive	K28.5 - D21.5 - D25.4 - D25.4
EOF _{rti}	EOF Remove-Terminate Invalid Class 4	Negative	K28.5 - D10.4 - D25.4 - D25.4
		Positive	K28.5 - D10.5 - D25.4 - D25.4

Table 10 - 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
SYNx	Clock Synchronization Word X	Negative	K28.5 – D31.3 – CS_X – CS_X
SYNy	Clock Synchronization Word Y	Negative	K28.5 – D31.3 – CS_Y – CS_Y
SYNz	Clock Synchronization Word Z	Negative	K28.5 – D31.3 – CS_Z – CS_Z
ARByx	Arbitrate	Negative	K28.5 – D20.4 – y – x
ARB(val)	Arbitrate	Negative	K28.5 – D20.4 – val – val
CLS	Close	Negative	K28.5 – D5.4 – D21.5 – D21.5
DHD	Dynamic Half-Duplex	Negative	K28.5 – D10.4 – D21.5 – D21.5
MRKtx	Mark	Negative	K28.5 – D31.2 – MK_TP – AL_PS
OPNyx	Open full-duplex	Negative	K28.5 – D17.4 – AL_PD – AL_PS
OPNy	Open half-duplex	Negative	K28.5 – D17.4 – AL_PD – AL_PD
OPNyr	Open selective replicate	Negative	K28.5 – D17.4 – AL_PD – D31.7
OPNfr	Open broadcast replicate	Negative	K28.5 – D17.4 – D31.7 – D31.7

5.5.3.4 Virtual Circuit Ready (VC_RDY)

The VC_RDY Primitive Signal shall only be transmitted from an F_Port that supports class 4. The VC_RDY shall indicate that the interface buffer that is to receive the frame is available for frame reception and a single Class 4 frame is needed from the N_Port if it wishes to maintain the requested bandwidth. Validity of the frame is not implied by VC_RDY. The VC_RDY shall be recognized by an N_Port that supports class 4.

An N_Port shall not transmit VC_RDYs. An F_Port shall discard any VC_RDYs received.

5.5.3.5 BB_SCs

The BB_SCs Primitive Signal shall indicate that a predetermined number ($2^{BB\_SC\_N}$) of Class 1 connect-request (SOF_{c1} or SOF_{c4}), Class 2, or Class 3 frames were sent since the previous BB_SCs was sent. See 15.6.2.5 for requirements for determining BB_SC_N. See 18.5.11 for BB_SCs usage requirements. If the BB_SCs Primitive Signal is received by an Fx_Port, it shall be processed but shall not be passed through the Fabric. See 8.2 for frame spacing requirements.

5.5.3.6 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 15.6.2.5 for requirements for determining BB_SC_N. See 18.5 for BB_SCr usage requirements. If the BB_SCr Primitive Signal is received by an Fx_Port, it shall be processed but shall not be passed through the Fabric. See 9.1 for frame spacing requirements.

5.5.3.7 SYNx, SYNy, SYNz

See 27.3.3.1.

5.5.3.8 ARByx, ARB(val)See [2]

For details on the Arbitrate Primitive Signals (ARByz, ARB(val)) refer to the Arbitrated Loop topology standard (see [2], FC-AL-2).

5.5.3.9 CLS

For details on the Close Primitive Signal (CLS) refer to the Arbitrated Loop topology standard (see [2], FC-AL-2).

5.5.3.10 DHD

For details on the Dynamic half-duplex Primitive Signal (DHD) refer to the Arbitrated Loop topology standard (see [2], FC-AL-2).

5.5.3.11 MRKtx

For details on the Mark Primitive Signal (MRKtx) refer to the Arbitrated Loop topology standard (see [2], FC-AL-2).

5.5.3.12 OPNyx

For details on the Open full-duplex Primitive Signal (OPNyx) refer to the Arbitrated Loop topology standard (see [2], FC-AL-2).

5.5.3.13 OPNyy

For details on the Open half-duplex Primitive Signal (OPNyy) refer to the Arbitrated Loop topology standard (see [2], FC-AL-2).

5.5.3.14 OPNyr

For details on the Open selective replicate Primitive Signal (OPNyr) refer to the Arbitrated Loop topology standard (see [2], FC-AL-2).

5.5.3.15 DHD

For details on the Open broadcast replicate Primitive Signal (OPNfr) refer to the Arbitrated Loop topology standard (see [2], FC-AL-2).

5.5.4 Primitive Sequences

5.5.4.1 Introduction

A Primitive Sequence is an Ordered Set that is transmitted repeatedly and continuously. Primitive Sequences are transmitted to indicate specific conditions within or conditions encountered by the receiver logic of an FC_Port. See table 11 for bit encodings of Primitive Sequences. The NOS, OLS, LR, and LRR Primitive Sequences shall be

supported. If the port supports AL-2 the Arbitrated Loop topology standard (see [2], FC-AL-2), it shall support the various LIP, LPB, and LPE Primitives Sequences shown in table 11.

Table 11 - Primitive Sequences

Abbr	Primitive Sequence	Beginning RD	Ordered Set
NOS	Not_Operational	Negative	K28.5 – D21.2 – D31.5 – D5.2
OLS	Offline	Negative	K28.5 – D21.1 – D10.4 – D21.2
LR	Link_Reset	Negative	K28.5 – D9.2 – D31.5 – D9.2
LRR	Link_Reset_Response	Negative	K28.5 – D21.1 – D31.5 – D9.2
LIP(F7,F7)	Loop Initialization - F7,F7	Negative	K28.5 – D21.0 – D23.7 – D23.7
LIP(F8,F7)	Loop Initialization - F8,F7	Negative	K28.5 – D21.0 – D24.7 – D23.7
LIP(F7,x)	Loop Initialization - F7,x	Negative	K28.5 – D21.0 – D23.7 – AL_PS
LIP(F8,x)	Loop Initialization - F8,x	Negative	K28.5 – D21.0 – D24.7 – AL_PS
LIPyx	Loop Initialization - reset	Negative	K28.5 – D21.0 – AL_PD – AL_PS
LIPfx	Loop Initialization - reset all	Negative	K28.5 – D21.0 – D31.7 – AL_PS
LIPba	Loop Initialization - reserved LIPba	Negative	K28.5 – D21.0 – b – a
LPByx	Loop Port Bypass	Negative	K28.5 – D9.0 – AL_PD – AL_PS
LPBfx	Loop Port Bypass all	Negative	K28.5 – D9.0 – D31.7 – AL_PS
LPEyx	Loop Port Enable	Negative	K28.5 – D5.0 – AL_PD – AL_PS
LPEfx	Loop Port Enable all	Negative	K28.5 – D5.0 – D31.7 – AL_PS

Primitive Sequences shall be transmitted continuously while the condition exists. A detailed description of FC_Port state changes relative to Primitive Sequence reception and transmission is given in clause 7. When a Primitive Sequence is received and recognized, depending on the state of the FC_Port, a corresponding Primitive Sequence or Idles shall be transmitted in response as defined in clause 7.

Primitive Sequences transmitted from an Nx_Port to the locally attached Fx_Port shall remove a pending or existing dedicated connection. The locally attached Fx_Port shall respond appropriately to the Primitive Sequence received and shall notify the remote Fx_Port attached to the Nx_Port associated with this connection, which shall transmit the Link Reset Primitive Sequence if the connection is established or pending.

If a dedicated connection does not exist, a Primitive Sequence transmitted by an Nx_Port shall be received and recognized by the locally attached Fx_Port, but not transmitted through the Fabric.

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).

5.5.4.2 Not_Operational (NOS)

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

6 FC-1 Receiver and Transmitter State Diagrams

6.1 Receiver

6.1.1 Introduction

Whenever a signal (see [8], FC-PI-2 and [1], 10GFC) is detected on a fibre, the receiver attached to that fibre shall attempt to achieve synchronization on both bit and Transmission Word boundaries of the received encoded bit stream. Bit Synchronization is defined in 6.1.4.1. Transmission Word synchronization is defined in 6.1.4.2. Synchronization failures on either bit or Transmission Word boundaries are not separately identifiable; both cause loss-of-synchronization errors.

6.1.2 State Diagram Overview

The Receiver State Diagram is shown in figure 9.

The Receiver states are as follows:

- a) State A (Loss of Synchronization),
- b) State B.1 (No invalid Transmission Word Detected),
- c) State B.2 (First invalid Transmission Word Detected),
- d) State B.3 (Second invalid Transmission Word Detected),
- e) State B.4 (Third invalid Transmission Word Detected), and
- f) State C (Reset).

Being in one of the Word Synchronization Acquired states refers to being in any of the states B.1, B.2, B.3, or B.4.

The receiver state transitions are defined as follows:

- 1) Power-on,
- 2) Acquisition of Word Synchronization (see 6.1.6.2),
- 3) An invalid Transmission Word is detected (see 6.1.5.3),
- 4) A detection of a Loss-of-Signal condition (see 6.1.5.2),
- 5) Two consecutive valid Transmission Words are detected (see 6.1.5.3),
- 6) Reset condition imposed on the receiver (see 6.1.6.4), and
- 7) Exiting of receiver reset condition (see 6.1.6.4).

6.1.3 Operational and Not Operational conditions

When the receiver is Operational, it shall be in either State A (Loss-of-Synchronization State) or in one of the Word Synchronization Acquired states.

When the receiver is Not Operational, it shall be in State C (Reset State).

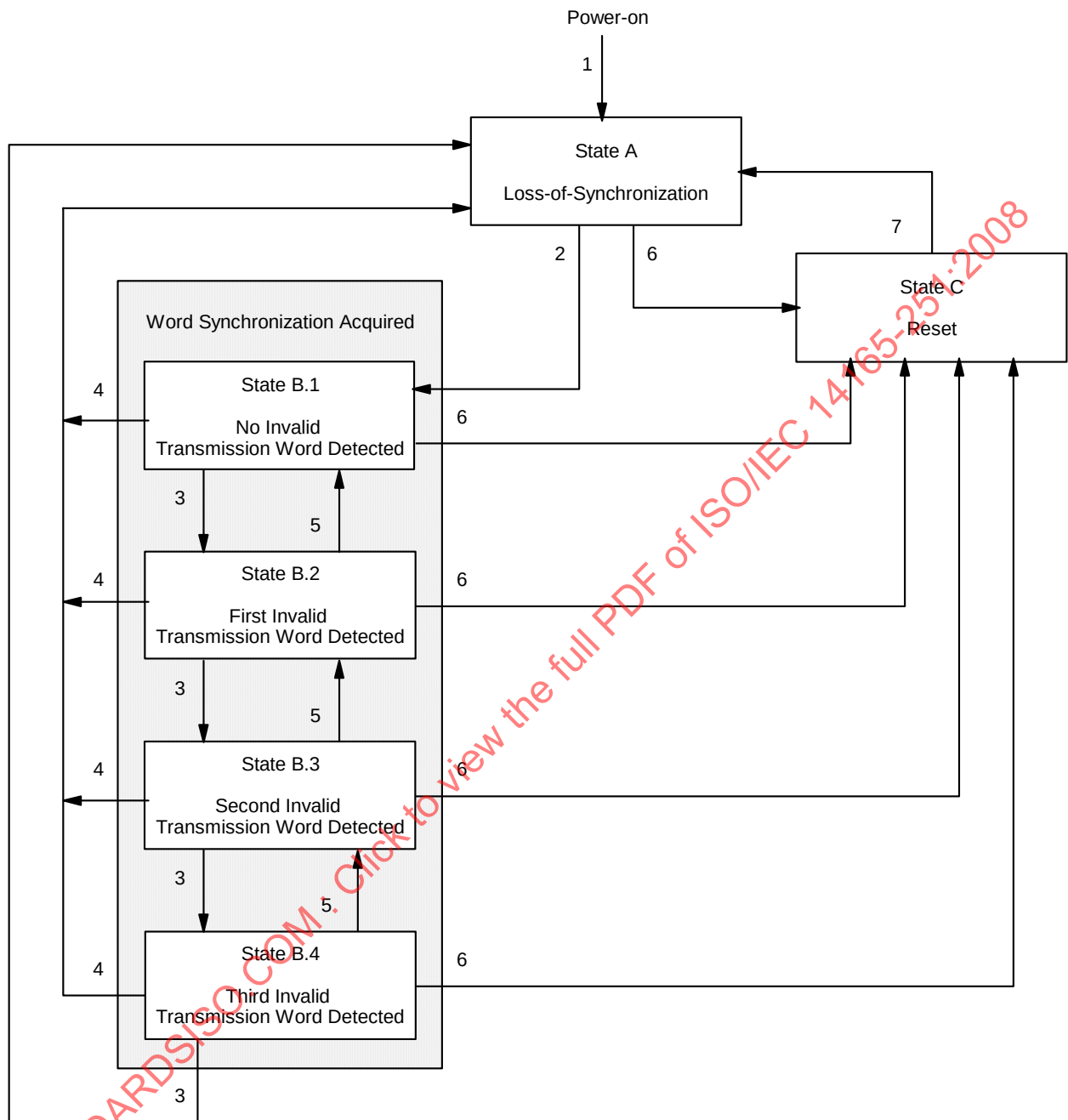


Figure 9 - Receiver State Diagram

6.1.4 Word Synchronization Procedure

The Word Synchronization procedure consists of first achieving Bit Synchronization (see 6.1.4.1), followed by achieving Word Synchronization (see 6.1.4.2).

6.1.4.1 Bit Synchronization

An Operational receiver that is in State A (Loss-of-Synchronization State) shall first acquire Bit Synchronization before attempting to acquire Transmission Word Synchronization. Bit Synchronization is defined FC-PI-2 (see [8],

FC-PI-2 and [1], 10GFC). After achieving Bit Synchronization, the receiver shall remain in State A (Loss-of-Synchronization State) until it achieves Transmission Word Synchronization.

6.1.4.2 Transmission Word synchronization

6.1.4.2.1 Introduction

The comma contained within the K28.1, K28.5, and K28.7 special characters is a singular bit pattern that in the absence of transmission errors shall not appear in any other location of a Transmission Character and shall not 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 that choose to implement the word alignment function in Continuous-mode alignment) 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.

Placement of a detected comma or K28.5 Transmission Character at the left-most position of a received Transmission Word ensures proper alignment of that word and of subsequently received Transmission Words. Ordered Set detection shall include both detection of the individual Transmission 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 (left-most) character position of the received Transmission Word).

6.1.4.2.2 Achieving Word Synchronization

An receiver that is in State A (Loss-of-Synchronization State) and 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 left-most bit positions without an intervening invalid Transmission Word, as specified in (see clause 6.1.5.3). The third detected Ordered Set shall change the state from State A (Loss-of-Synchronization State) to State B.1 (No invalid Transmission Word detected State) using transition 2. The third detected Ordered Set shall be considered valid information and shall be decoded and provided by the receiver to its FC_Port. A receiver in any of the Word Synchronization Acquired states shall provide information that has been received from its attached fibre and decoded.

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

6.1.4.2.3 Word alignment methods

6.1.4.2.3.1 Continuous-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).

6.1.4.2.3.2 Explicit-mode alignment

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.

6.1.5 Loss of Word Synchronization

6.1.5.1 Introduction

Loss of Word Synchronization shall occur in the following conditions:

- a) a Loss of Signal is detected when in any of the Word Synchronization Acquired states, or

- b) an invalid Transmission Word is detected in State B.4.

6.1.5.2 Detection of loss of Signal

A receiver may provide a Loss-of-Signal check.

6.1.5.3 Detection of an invalid Transmission Word

When in any of the Word Synchronization Acquired states each received Transmission Word is tested to determine the validity of the Transmission Word.

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

- a) A code violation on a character basis, as specified by tables 6 and 7, is detected within a Transmission Word. This is referred to as a code violation condition.
- b) 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.
- c) A defined Ordered Set (see 5.5) 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.

6.1.6 State transitions

6.1.6.1 Default State

A receiver shall enter state A (Loss-of-Synchronization) on power-on (i.e., default).

6.1.6.2 State A (Loss of Synchronization)

The Loss-of-Synchronization State shall be entered upon the following conditions:

- a) completion of the Loss-of-Synchronization procedure while in state B.4 (Third Invalid Transmission Word Detected) using transition 3;
- b) detection of Loss-of-Signal while in states B.1, B.2, B.3, or B.4 using transition 4; or
- c) completion of the reset while in State C (Reset) using transition 7.

While in the Loss-of-Synchronization State, the receiver may attempt to reacquire bit re synchronization. In some instances, this may allow the receiver to regain Word Synchronization when it otherwise would not be possible. However, initiation of bit re synchronization may also delay the re synchronization process by forcing the receiver to reestablish a clock reference when such reestablishment is otherwise unnecessary (see [8], FC-PI-2 and [1], 10GFC for a detailed discussion of Bit Synchronization).

When Word Synchronization is acquired the receiver enters state B.1 (No Invalid Transmission Word Detected) using transition 2. Imposing a reset condition upon the receiver shall cause any state to transition to the C state using transition 6.

6.1.6.3 State B (Word Synchronization Acquired)

The following four states are defined as Word Synchronization Acquired states:

- 1) No Invalid Transmission Word Detected State (state B.1),
- 2) The First Invalid Transmission Word Detected State (state B.2),
- 3) The Second Invalid Transmission Word Detected State (state B.3), and
- 4) The Third Invalid Transmission Word Detection State (state B.4).

NOTE The rationale for the loss-of-Synchronization procedure is to reduce the likelihood that a single error results in a

Loss-of-Synchronization. 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 that was affected by a disparity change resulting from the error. The procedure described above would maintain Synchronization in such a case.

6.1.6.3.1 State B.1 (No Invalid Transmission Word Detected State)

When the procedure is in state B.1, checking for an invalid Transmission Word shall be performed. Any invalid Transmission Word shall cause the B.1 state to transition to the B.2 state (transition 3). A Loss of Signal condition shall cause the B.1 state to transition to the A state (transition 4). A reset condition is imposed upon the receiver causes the B.1 state to transition to the C state (transition 6).

6.1.6.3.2 State B.2 (First Invalid Transmission Word Detected State)

When the procedure is in state B.2, checking for an invalid Transmission Word shall be performed. Any invalid Transmission Word causes the B.2 state to transition to the B.3 state (transition 3). If two consecutive valid Transmission Words are received, the B.2 state transitions to the B.1 state (transition 5). A Loss of Signal condition shall cause the B.2 state to transition to the A state (transition 4). A reset condition is imposed upon the receiver causes the B.2 state to transition to the Reset State C (transition 6).

6.1.6.3.3 State B.3 (Second Invalid Transmission Word Detected State)

When the procedure is in state B.3, checking for an invalid Transmission Word shall be performed. Any invalid Transmission Word causes the B.3 state to transition to the B.4 state (transition 3). If two consecutive valid Transmission Words are received, the B.3 state transitions to the B.2 state (transition 5). A Loss of Signal condition shall cause the B.3 state to transition to the A state (transition 4). A reset condition is imposed upon the receiver causes the B.3 state to transition to the Reset State C (transition 6).

6.1.6.3.4 State B.4 (Third Invalid Transmission Word Detection State)

When the procedure is in state B.4, checking for an invalid Transmission Word shall be performed. Any invalid Transmission Word causes the B.4 state to transition to the A state (transition 3). If two consecutive valid Transmission Words are received, the B.4 state transitions to the B.3 state (transition 5). A Loss of Signal condition shall cause the B.4 state to transition to the A state (transition 4). A reset condition is imposed upon the receiver causes the B.2 state to transition to the Reset State C (transition 6).

6.1.6.4 State C (Reset)

When a receiver reset condition is imposed on a receiver, either internally or externally, the receiver shall enter the Reset State (transition 6). 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.

NOTE 1 A typical use of receiver reset is to force a receiver in the Loss-of-Synchronization State to attempt reacquisition of Bit Synchronization. Entry into this state does not necessarily indicate loss of Bit Synchronization.

NOTE 2 The conditions required for a receiver in the Reset State to exit that state are not defined by this standard. Such conditions may be based on explicit indications. They may also be time-dependent in nature.

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 (State A).

NOTE 3 The conditions required for a receiver in the Reset State to exit that state are not defined by this standard. Such conditions may be based on explicit indications. They may also be time-dependent in nature.

6.2 Transmitter

6.2.1 State Diagram

The Transmitter State Diagram is shown in figure 10.

The transmitter states are as follows:

- a) Not enabled,
- b) Working, and
- c) Failure.

The transmitter state transition conditions are as follows:

- 1) Power on,
- 2) Receipt of transmitter enable request (see 6.2.3.2),
- 3) Receipt of a transmitter disable request (see 6.2.3.3), and
- 4) Detection of a transmitter failure condition (see 6.2.3.3).

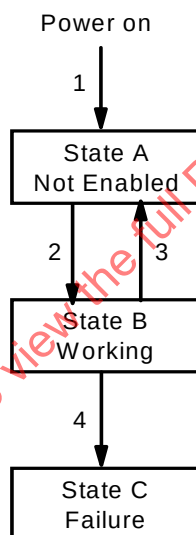


Figure 10 - Transmitter State Diagram

6.2.2 Operational condition

A transmitter actively attempting to transmit an encoded bit stream onto its attached fibre is defined to be in State B (Working).

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 FC_Port associated with a transmitter or as the result of an external event, a transmitter shall transition to State A (Not Enabled) using transition 3.

6.2.3 State Transitions

6.2.3.1 Not Enabled State

Entry into State A (Not Enabled) may result from external error conditions detected by the LCF, or it may result from internally generated signals. The transmitter shall enter State A (Not Enabled) under the following conditions:

- a) completion of transmitter initialization (transition 1), and

- b) processing of an Transmitter Disable Request (transition 3).

The transmitter shall remain in State A (Not Enabled) until it completes a Transmitter Enable Request (transition 2).

6.2.3.2 Working State

An Operational transmitter shall enter State B (Working) when its transmitter becomes enabled. A transmitter shall become enabled after completing a Transmitter Enable Request (transition 2).

While in State B (Working), a transmitter attached to a Fibre shall constantly attempt to transmit an encoded bit stream onto that Fibre.

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

- a) requests for Ordered Set transmission are aligned with the word alignment established by the transmitter upon entry into the Working State, and
- b) 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 shall not occur:

- a) transmission requests improperly aligned with transmission (word and byte) boundaries,
- b) transmission requests not consistent with the current transmission boundary,
- c) requests for Ordered Set transfers on a non-word boundary,
- d) requests for data transfers overlapping an Ordered Set transfer currently in progress, and
- e) 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.

A transmitter completing a Transmitter Disable Request shall transition from State B (Working) to State A (Not Enabled). Some transmitters are capable of monitoring this transmitted signal and verifying its validity. Such transmitters may transition from State B (Working) to State C (Failure) as a result of this monitoring.

6.2.3.3 Failure State

A transmitter shall enter State C (Failure) upon recognition (via signal monitoring) of a transmitter-failure condition detected while in State B (Working). Once State C (Failure) is entered, the transmitter shall become Not Operational and shall remain in State C (Failure) 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 how 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 State B (Working) to State C (Failure).

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 [8], FC-PI-2 and [1], 10GFC for a discussion of signal monitoring).

7 FC_Port state machine

7.1 State diagram

An FC_Port shall conform to table 12. For conditions not explicitly listed to cause state changes to occur, the FC_Port shall remain in the current state. See [2], FC-AL-2 for initialization of devices that support a loop port state machine.

7.2 Active State (AC)

An FC_Port shall enter the Active State when it completes the Link Initialization Protocol (see 7.6.2) or the Link Reset Protocol (see 7.6.3). When an FC_Port is in the Active State, it is able to transmit and receive frames and Primitive Signals. When a Primitive Sequence (see 5.5.4) is received, the FC_Port shall exit the Active State as defined in table 12. If any frame or Primitive Signal (see 5.5.3) is received and recognized, the FC_Port shall remain in the Active State.

If conditions as indicated in the referenced clauses are detected within the FC_Port, the Active State shall be exited and the FC_Port shall perform the indicated Primitive Sequence Protocol:

- a) Link Initialization (see 7.6.2) (The next state is OL1 State),
- b) Link Reset (see 7.6.3) (The next state is the LR1 State),
- c) Link Failure (see 7.6.4) (The next state is the LF2 State), and
- d) Online-to-Offline (see 7.6.5) (The next state is the OL1 State)

7.3 Link Recovery

7.3.1 Link recovery hierarchy

The Link Recovery hierarchy is shown in figure 36.

7.3.2 LR Transmit State (LR1)

7.3.2.1 General (while in the LR1 state)

An FC_Port shall enter the LR1 State to initiate the Link Reset Protocol. While in the LR1 State, the FC_Port shall transmit the LR Primitive Sequence. When a Primitive Sequence is received, the FC_Port shall respond as defined in table 12.

7.3.2.2 Class 1 (while in the LR1 state)

A pending or existing dedicated connection shall be removed and the end-to-end Credit associated with the connected Nx_Ports shall be reset to the Login value. An Fx_Port shall process or discard any Class 1 connect-request or Class 6 connect-request frames currently held in the receive buffer associated with the outbound fibre of the attached FC_Port. The locally attached Fx_Port shall enter the LR2 State and shall transmit the Link Reset Primitive Sequence to the remote Fx_Port (i.e., initiate Link Reset Protocol).

When LR is received and recognized, or transmitted; all Class 1 frame Sequences that are open in an Nx_Port shall be abnormally terminated. When an FC_Port is transmitting the Link Reset Primitive, all Class 1 frames received shall be discarded. See 20.8 for more discussion on Connection Recovery.

Table 12 - FC_Port states

	Current State								
	Active	Link Recovery			Link Failure		Offline		
	AC (see 7.2)	LR1 (see 7.3.2)	LR2 (see 7.3.3)	LR3 (see 7.3.4)	LF1 (see 7.4.1)	LF2 (see 7.4.2)	OL1 (see 7.5.2)	OL2 (see 7.5.3)	OL3 (see 7.5.4)
	Primitive Sequence Transmitted while in state:								
	IDLE	LR	LRR	IDLE	OLS	NOS	OLS	LR	NOS
Input Event:	Next State:								
L >> LR	LR2	LR2	LR2	LR2	LR2	LF2	LR2 ^b	LR2	LF2
L >> LRR	LR3 ^c	LR3	LR3	LR3	LF1	LF2	OL1	LR3	LF2
L >> IDLES	AC	LR1	AC	AC	LF1	LF2	OL1	OL2	OL3
L >> OLS	OL2	OL2	OL2	OL2	OL2	OL2	OL2 ^b	OL2	OL2
L >> NOS	LF1	LF1	LF1	LF1	LF1	LF1	LF1 ^b	LF1	LF1
Loss-of-Signal	LF2	LF2	LF2	LF2	LF2	LF2 ^a	OL3 ^b	OL3 ^a	OL3
Loss of Sync > Limit	LF2	LF2	LF2	LF2	LF2	LF2	OL3 ^b	OL3	OL3
Event time-out (R_T_TOV)	N/A	LF2	LF2	LF2	LF2 ^d	N/A	OL3 ^{b,f}	OL3 ^e	N/A
Link time-out (E_D_TOV)	LR1	LR1	LR1	LR1	LR1	LR1	LR1	LR1	LR1
LEGEND L >> means receiving from the Link N/A means not applicable									
^a Depending on Laser safety requirements, the transmitter may enter a "pulse" transmission mode of operation when Loss-of-Signal is detected. ^b All events are ignored until the FC_Port determines it is time to leave the OL1. ^c 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. ^d The time-out period starts timing when NOS is no longer recognized and none of the other events occur that cause a transition out of the state. ^e The time-out period starts timing when OLS is no longer recognized and none of the other events occur that cause a transition out of the state. ^f The time-out period starts timing when the FC_Port is attempting to go online transmits OLS, and none of the other events occur that cause a transition out of state.									

7.3.2.3 Class 2 and Class 3 (while in the LR1 state)

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

7.3.2.4 Class 4 (while in the LR1 state)

An N_Port in the LR1 State:

- a) shall retain Class 4 circuit states,
- b) shall set VC_Credit equal to zero for all Class 4 circuits, and
- c) may discard Class 4 frames received on Live Class 4 circuits.

The locally attached F_Port in the LR1 State:

- a) shall retain Class 4 circuit states,
- b) shall discard Class 4 frames destined for the N_Port, and
- c) may discard Class 4 frames received from the N_Port.

The QoS with the locally attached F_Port in the LR1 State:

- a) shall suspend VC_RDY pacing at the appropriate N_Ports for each Live Class 4 circuit to or from the N_Port being sent the LR Primitive Sequence in the LR1 State, and
- b) for each Live Class 4 circuit to or from the affected N_Port, the QoS shall send the END Link_Control command to each affected remote N_Port and the local N_Port to remove each Class 4 circuit when the Active State is re-entered for the affected N_Port (see 11.2.4.3 and 25.3.5.5).

7.3.2.5 Class 6 (while in the LR1 state)

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 LR2 State and shall notify all other F_Ports attached to the N_Ports participating in the Class 6 connection. The notified F_Ports shall transmit the Link Reset Primitive to their connected N_Ports (i.e., initiate Link Reset Protocol).

All Class 6 frame Sequences that are open in an N_Port when LR is received and recognized, or transmitted shall be abnormally terminated. When an N_Port is transmitting the Link Reset Primitive, all Class 6 frames received shall be discarded. See 20.8 for more discussion on Connection Recovery.

7.3.3 LR Receive State (LR2)

7.3.3.1 General (while in the LR2 state)

An FC_Port shall enter the LR2 State when it receives and recognizes the LR Primitive Sequence while it is not in the OL3 or LF2 State. While in the LR2 State, the FC_Port shall transmit the LRR Primitive Sequence. When a Primitive Sequence is received, the FC_Port shall respond as defined in table 12.

An FC_Port that 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 FC_Port shall be reset to its Login value.

7.3.3.2 Class 1 (while in the LR2 state)

A pending or existing dedicated connection shall be removed and the end-to-end Credit associated with the connected Nx_Ports shall be reset to the Login value. All Class 1 frame Sequences that are open in an Nx_Port when LR is received and recognized shall be abnormally terminated.

7.3.3.3 Class 4 (while in the LR2 state)

An N_Port in the LR2 State shall:

- a) retain Class 4 circuit states, and
- b) set VC_Credit equal to zero for all Class 4 circuits.

The locally attached F_Port in the LR2 State shall:

- a) retain Class 4 circuit states,
- b) discard Class 4 frames destined for the N_Port, and
- c) may discard Class 4 frames received from the N_Port.

The QoS with the locally attached F_Port in the LR1 State:

- a) shall suspend VC_RDY pacing for each appropriate N_Port for each Live Class 4 circuit to or from the N_Port sending the LR Primitive Sequence in the LR2 State, and
- b) for each Live Class 4 circuit to or from the affected N_Port, the QoS shall send the END Link Control command to each affected remote N_Port and the local N_Port to remove each Class 4 circuit when the Active State is re-entered for the affected N_Port (see 11.2.4.3 and 25.3.5.5).

7.3.3.4 Class 6 (while in the LR2 state)

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 that are open in an N_Port when LR is received and recognized shall be abnormally terminated.

7.3.4 LRR Receive State (LR3)

7.3.4.1 General (while in the LR3 state)

An FC_Port shall enter the LR3 State when it receives and recognizes the LRR Primitive Sequence while it is in the Active State, LR1 State, LR2 State, or OL2 State. While in the LR3 State, the FC_Port shall transmit Idles. When a Primitive Sequence is received, the FC_Port shall respond as defined in table 12.

7.3.4.2 Class 1 and Class 6 behavior (while in the LR3 state)

An Nx_Port in the LR3 State shall retain Class 1 and Class 6 dedicated connections.

The locally attached Fx_Port in the LR3 State:

- a) shall retain Class 1 and Class 6 dedicated connections,
- b) shall discard Class 1 and Class 6 frames destined for the Nx_Port.

7.3.4.3 Class 4 behavior (while in the LR2 state)

An N_Port in the LR3 State shall retain Class 4 circuit states.

The locally attached F_Port in the LR3 State:

- a) shall retain Class 4 circuit states, and
- b) shall discard Class 4 frames destined for the N_Port.

The QoS with the locally attached F_Port in the LR3 State:

- a) shall retain the suspension of VC_RDY transmission to the N_Port, and
- b) shall retain suspension of VC_RDY pacing to remote N_Ports for all Live Class 4 circuits for the affected N_Port.

7.4 Link Failure

7.4.1 NOS Receive State (LF1)

7.4.1.1 General

An FC_Port shall enter the LF1 State when it receives and recognizes the NOS Primitive Sequence. Upon entry into the LF1 State, the FC_Port shall update the appropriate error counter in the Link Error Status Block (see 21.8). Only one error per Link Failure event shall be recorded. When a Primitive Sequence is received, the FC_Port shall respond as defined in table 12.

7.4.1.2 Class 4 behavior

An N_Port in the LF1 State shall set all Class 4 circuits to the Nonexisting State.

The locally attached F_Port in the LF1 State shall set all Class 4 circuits to the Nonexisting State.

The Fabric with the locally attached F_Port in the LF1 State shall remove Dormant or Live Class 4 circuits when the LF1 State is entered (see 25.3.5.5).

7.4.2 NOS Transmit State (LF2)

7.4.2.1 General

An FC_Port shall enter the LF2 State when a Link Failure condition is detected. Upon entry into the LF2 State, the FC_Port shall update the appropriate error counter in the Link Error Status Block (see 21.8). Only one error per Link Failure event shall be recorded. The FC_Port shall remain in the LF2 State while the condition that caused the Link Failure, exists. While in the LF2 State, the FC_Port shall transmit the NOS Primitive Sequence.

When the Link Failure condition is no longer detected, the FC_Port shall respond to Primitive Sequences received as defined in table 12.

Transmission of NOS by an Nx_Port to its locally attached Fx_Port of a Fabric shall remove a pending or existing dedicated connection. The locally attached F_Port shall enter the LR2 State and shall notify all other F_Ports attached to the N_Ports participating in the Class 6 connection. The notified F_Ports shall transmit the Link Reset Primitive to their connected N_Ports (i.e., initiate Link Reset Protocol).

If a dedicated connection does not exist, NOS transmission by an Nx_Port shall be received and recognized by the locally attached Fx_Port, but not transmitted through the Fabric. The Fx_Port shall respond by entering the LF1 State.

7.4.2.2 Class 4 behavior

An N_Port in the LF2 State shall set all Class 4 circuits to the Nonexisting State.

The locally attached F_Port in the LF2 State shall set all Class 4 circuits to the Nonexisting State.

The QoS F with the locally attached F_Port in the LF2 State shall remove Dormant or Live Class 4 circuits for the affected N_Port when the LF2 State is entered (see 25.3.5.5).

7.5 Offline

7.5.1 General

While Offline, an FC_Port shall not record receiver errors (e.g., Loss-of-Synchronization). NOS Reception or Link Failure conditions that are detected shall not be recorded as Link Failure events in the Link Error Status Block (see

21.8).

7.5.2 OLS Transmit State (OL1)

7.5.2.1 Actions applicable to all classes

An FC_Port shall enter the OL1 State in order to:

- a) perform the Link Initialization Protocol (see 7.6.2) in order to exit the Offline State, or
- b) transition from Online-to-Offline using the Online-to-Offline Protocol (see 7.6.5).

When the FC_Port enters the OL1 State, it shall transmit OLS for a minimum time of 5 ms. After that period of time has elapsed, the FC_Port shall respond as defined table 12 when a Primitive Sequence is received.

While an FC_Port is attempting to go Online, if no Primitive Sequence is received or event detected that causes the FC_Port to exit the OL1 State after R_T_TOV, the FC_Port shall enter the OL3 State.

Transmission of OLS by an Nx_Port to its locally attached Fx_Port of a Fabric shall remove a pending or existing Class 1 or Class 6 connection. The locally attached F_Port shall enter the LR2 State and shall notify all other F_Ports attached to the N_Ports participating in the Class 6 connection. The notified F_Ports shall transmit the Link Reset Primitive to their connected N_Ports (i.e., initiate Link Reset Protocol).

If a dedicated connection does not exist, OLS transmission by an Nx_Port shall be received and recognized by the locally attached Fx_Port, but not transmitted through the Fabric. The Fx_Port shall respond by entering the OL2 State.

7.5.2.2 Class 4 behavior

An N_Port in the OL1 State shall set all Class 4 circuits to the Nonexisting State.

The locally attached F_Port in the OL1 State shall set all Class 4 circuits to the Nonexisting State.

The QoS F with the locally attached F_Port in the OLS Transmit states shall remove Dormant or Live Class 4 circuits for the affected remote N_Ports when the LF1 State is entered (see 25.3.5.5).

7.5.3 OLS Receive State (OL2)

7.5.3.1 General

An FC_Port shall enter the OL2 State when it receives and recognizes the OLS Primitive Sequence. When a Primitive Sequence is received, the FC_Port shall respond as defined in table 12. Detection of Loss-of-Signal or Loss-of-Synchronization shall not be counted as a Link Failure event in the Link Error Status Block.

7.5.3.2 Class 4 behavior

An N_Port in the OL2 State shall set all Class 4 circuits to the Nonexisting State.

The locally attached F_Port in the OL2 State shall set all Class 4 circuits to the Nonexisting State.

The QoS F with the locally attached F_Port in the OL2 State shall remove Dormant or Live Class 4 circuits for the affected remote N_Ports when the LF1 State is entered (see 25.3.5.5).

7.5.4 Wait for OLS State (OL3)

7.5.4.1 Actions applicable to all classes

An FC_Port shall enter the OL3 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 or Transmit State at an appropriate time during the Link Initialization Protocol (see 7.6.2).

Upon entry into the OL3 State, the FC_Port shall transmit the NOS Primitive Sequence. When a Primitive Sequence is received, the FC_Port shall respond as defined in table 12.

7.5.4.2 Class 4 behavior

An N_Port in the OL3 State shall retain Class 4 circuits in the Nonexisting State.

The locally attached F_Port in the OL3 State shall retain Class 4 circuits in the Nonexisting State.

7.6 Primitive Sequence protocols

7.6.1 Functions

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.

7.6.2 Link Initialization protocol

The Link Initialization protocol shall be performed by the FC_Port after one of the following events has occurred:

- a) powered-on
- b) internal reset (the definition of internally reset is beyond the scope of this standard), or
- c) has been offline and desires to come back online.

The FC_Ports involved may be an N_Port and F_Port or two N_Ports. The Link Initialization protocol begins when the FC_Port enters the OL1 State after one of the above events has been detected and is complete when the FC_Port enters the Active State.

7.6.3 Link Reset protocol

The Link Reset Protocol shall be performed when any of the following conditions are detected:

- a) Connection Recovery (see 20.8),
- b) Link timeout (see 21.2.3),
- c) buffer-to-buffer overrun (i.e., an FC_Port receives a frame subject to buffer-to-buffer flow control without a buffer available).

The Link Reset protocol begins when the FC_Port enters the LR1 State after one of the above events has been detected and is complete when the FC_Port enters the Active State.

If an FC_Port, performing the Link Reset Protocol, enters the LR3 State, without receiving an appropriate response within R_T_TOV, a Link Failure shall be detected and the Link Failure Protocol shall be performed.

7.6.4 Link Failure protocol

The Link Failure Protocol shall be performed after an FC_Port has detected one of the following conditions:

- a) a Loss-of-Synchronization for a period of time greater than R_T_TOV,
- b) Loss-of-Signal while not in the Offline State, or
- c) Link Reset Protocol timeout error is detected (see 7.6.3).

The Link Failure protocol begins when the FC_Port enters the LF2 State after one of the above events has been detected and is complete when the Active State is entered.

7.6.5 Online-to-offline protocol

The FC_Port shall perform the Online-to-offline Protocol to enter the Offline State from the Active State. This protocol shall be performed in order to power-down or to perform diagnostics (diagnostic requirements are beyond the scope of this standard). This Protocol provides an FC_Port with a graceful indication prior to Loss-of-Signal. This avoids logging an error event for a normal system function. The Online-to-offline protocol shall start when the FC_Port enters the OL1 State.

After transmitting OLS for the time specified in 7.5, the FC_Port shall be Offline and may do any of the following:

- a) perform diagnostic procedures,
- b) turn off its transmitter,
- c) transmit any signal (excluding Primitive Sequences) without errors being detected by the attached FC_Port
- d) power-down or
- e) start the Link Initialization protocol.

NOTE After entering the OL1 State and transmitting OLS for a minimum of 5 ms, the FC_Port may then transmit any Transmission Word other than LR, LRR, or NOS without causing the remote FC_Port to leave the OL2 State.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

8 Frame formats

8.1 General frame format

All FC-2 frames shall follow the frame format as shown in figure 11. An FC-2 frame is composed of a SOF delimiter, frame content, and an EOF 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 in this standard.

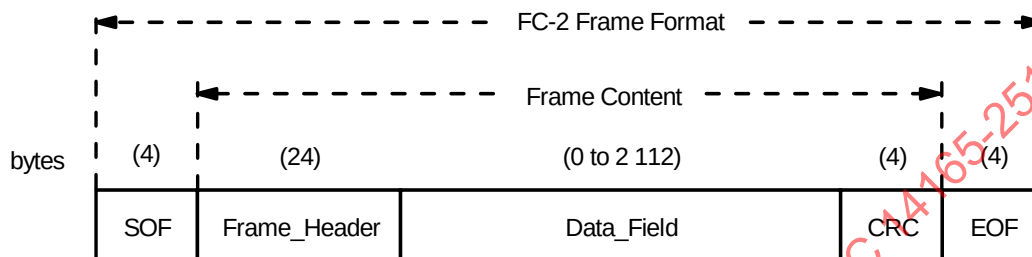


Figure 11 - FC-2 frame format

8.2 Frame transmission

Frame transmission shall be performed by inserting a frame immediately following a series of Idles. Idles shall be transmitted immediately upon completion of the frame. A minimum of two Idle words shall be transmitted consecutively following the EOF word of each frame transmitted by any FC_Port transmitter.

If an FC_Port transmitter is repeating the Transmission Word stream of a receiver that is not capable of exercising buffer-to-buffer flow control (e.g., L_Ports with REPEAT true, switch ports with a Class 1 connection open), the transmitter may insert or remove Idle words from the Transmission Word stream in order to adjust for timing skew between the receiver and transmitter; however, it shall retain at least two Idle words consecutively following each EOF.

If an FC_Port transmitter is repeating the Transmission Word stream of a receiver that is capable of exercising buffer-to-buffer flow control, the FC_Port shall transmit at least six primitive signals between each EOF and the next SOF. Of the six or more primitive signals transmitted, at least four shall be Idle words.

If an FC_Port transmitter is not repeating the Transmission Word stream of a receiver, the FC_Port shall transmit at least six primitive signals between each EOF and the next SOF. Of the six or more primitive signals transmitted, at least four shall be Idle words.

See [2], FC-AL-2 for Arbitrated Loop specific frame transmission requirements. See 8.9 for frame reception requirements.

8.3 Start-of-Frame (SOF) delimiter

8.3.1 Introduction

The Start-of-Frame (SOF) delimiter is an Ordered Set that immediately precedes the frame content. There are multiple SOF delimiters defined for Sequence control. The SOF delimiter shall be transmitted on a word boundary. Tables 42 and 46, respectively, specify allowable delimiters by class for Data and Link_Control frames. The bit encodings for the SOF delimiters are defined in table 9. SOF_x is used to represent any SOF.

8.3.2 SOF 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, during Login, by the destination Nx_Port or by the Fabric, if present, whichever size is smaller. This delimiter also identifies the first frame of a Sequence (i.e., implicit SOF_{i1}).

8.3.3 SOF Circuit Activate Class 4 (SOF_{c4})

The SOF_{c4} shall be used to activate a Class 4 circuit. This delimiter shall also identify the first frame of a Sequence (i.e., implicit SOF_{i4}).

8.3.4 SOF Initiate (SOF_{ix})

8.3.4.1 Applicability

A Sequence shall be initiated and identified by using SOF_{i1}, SOF_{i2}, SOF_{i3}, or SOF_{i4} in the first frame. SOF_{ix} is used to represent these four SOF delimiters.

8.3.4.2 SOF 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 Dedicated Connection is established, all subsequent Sequences within that Dedicated Connection shall be initiated with SOF_{i1}.

8.3.4.3 SOF Initiate Class 2 (SOF_{i2})

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

8.3.4.4 SOF Initiate Class 3 (SOF_{i3})

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

8.3.4.5 SOF Initiate Class 4 (SOF_{i4})

A Class 4 circuit shall be activated with SOF_{c4}. After a Class 4 circuit is established, all subsequent Sequences within that Virtual Connection shall be activated with SOF_{i4}. The SOF_{i4} shall be used in the first frame of a Sequence for Class 4 Service.

8.3.5 SOF Normal (SOF_{nx})

8.3.5.1 Applicability

The following 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}, SOF_{n3} and SOF_{n4}.

8.3.5.2 SOF 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.

8.3.5.3 SOF 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.

8.3.5.4 SOF 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.

8.3.5.5 SOF Normal Class 4 (SOF_{N4})

The SOF_{N4} shall be used for all frames except the first frame of a Sequence for Class 4 Service.

8.3.6 SOF Fabric (SOF_f)

If an Nx_Port or Fx_Port receives a Class F frame, indicated by an SOF_f delimiter, it shall be discarded by the Nx_Port or Fx_Port. The receiving Nx_Port or Fx_Port may send an R_RDY.

NOTE Sending an R_RDY for a Class F frame is optional for a port not internal to the Fabric (i.e., an Nx_Port or non-switch internal port). This allows backwards compatibility with existing implementations. New Implementations should send an R_RDY for class F frames.

8.4 Frame_Header

The Frame_Header shall be the first field of the frame content and shall immediately follow the SOF delimiter for all frames. The Frame_Header shall be transmitted on a word boundary. The Frame_Header is used by the LCF to control link operations, control device protocol transfers, and detect missing or out of order frames. The Frame_Header is defined in clause 9.

8.5 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 to 28 in the R_CTL field of the Frame_Header.

The two frame types are:

- a) FT_0 (Data Field = 0 bytes)

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

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

When the R_CTL field bits 31 to 28 are set to values other than '1100' b, 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 that is not divisible by 4, the remaining bytes (up to 3) shall contain fill bytes as specified in the F_CTL field in table 31. 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 10.

A discussion of FT_0 (Link_Control) and FT_1 (Data) frames is found in 11.2 and 11.1, respectively.

8.6 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 A.1). For transmission order see 5.3.2.

The CRC specified in this standard follows the CRC specified in Fibre Distributed Data Interface (FDDI) – Media Access Control (MAC) (See [3]). See Annex A for further discussions of the CRC. The CRC shall use the following 32-bit polynomial:

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

8.7 End-of-Frame (EOF) delimiter

8.7.1 Introduction

The End-of-Frame (EOF) delimiter is an Ordered Set that immediately follows 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. Tables 42 and 46, respectively, specify allowable delimiters by class for Data and Link_Control frames. There are three categories of EOF delimiters:

- The first category shall indicate that the frame is valid from the sender's perspective and potentially valid from the receiver's perspective.
- The second category (EOF_{dti}, EOF_{rti}, and EOF_{ni}) shall indicate that the frame content is invalid. This category shall only be used by an Fx_Port that receives a complete frame and decodes it before forwarding it.
- The third category (EOF_a) shall indicate the frame content is corrupted and the frame was truncated during transmission. The third category shall be used by FC_Ports to indicate an internal malfunction (e.g., a transmitter failure that does not allow the entire frame to be transmitted normally).

The bit encodings for the EOF delimiters are defined in table 9.

All frames other than the last frame of a Sequence shall be terminated with an EOF_n delimiter.

Each Sequence other than a Sequence that deactivates or removes a Class 1, Class 4 or Class 6 circuit shall terminate with an EOF_t delimiter. Use of EOFs that deactivate or remove a Class 1, Class 4, or Class 6 circuit is described in clause 13.

If an Fx_Port detects a frame error, the Fx_Port shall replace either an EOF_n or an EOF_t of the frame in error with the EOF_{ni} delimiter.

EOF_x is used to represent any EOF.

8.7.2 Valid frame content

8.7.2.1 EOF Normal (EOF_n)

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

8.7.2.2 EOF Terminate (EOF_t)

The EOF_t shall indicate that the Sequence associated with this SEQ_ID is complete. EOF_t or EOF_{dt} shall be used to properly close a Sequence without error.

8.7.2.3 EOF Disconnect Terminate (EOF_{dt}) (Class 1 or Class 6)

The EOF_{dt} shall remove a dedicated connection through a Fabric, if present. The EOF_{dt} 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 EOF_{dt}, shall be

abnormally terminated and may require Sequence recovery on a ULP protocol-dependent basis.

8.7.2.4 EOF Deactivate Terminate (EOFdt) (Class 4)

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 EOFdt, shall be abnormally terminated and may require Sequence recovery on a ULP protocol-dependent basis. The receipt of an EOFdt delimiter deactivates a class 4 connection.

8.7.2.5 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.

8.7.3 Invalid frame content

8.7.3.1 General

There are five EOF delimiters that 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 forwarded 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 (e.g., if the Fabric decodes a frame into bytes and is unable to validly decoded one or more Transmission Characters 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 that indicates the frame content is invalid, the invalid frame condition shall be reported by the entity that replaces the EOF delimiter that indicates invalid frame content. The destination Nx_Port, at its discretion, may report the event of receiving a frame with one of these delimiters.

Errors are counted at the point where they are detected. Events may also be reported at the point where they are recognized.

NOTE Due to the importance of the EOFdt delimiter in Class 1 and Class 6, it is recommended that the Fabric forward a frame that has either EOFdt or EOFdti.

8.7.3.2 End of Frame Abort (EOFa)

The EOFa 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. If the transmitter retransmits the aborted frame, it shall transmit the frame with the same SEQ_CNT.

An invalid EOF (EOFdti, EOFrti or EOFni) delimiter may be changed to an EOFa delimiter under the conditions specified for EOFa.

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

It is also used by the Fabric to replace missing EOF delimiters or to truncate over length frames.

8.7.3.3 EOF Disconnect Terminate Invalid (EOF_{dti}) (Class 1 and Class 6)

The EOF_{dti} shall replace a recognized EOF_{dt} delimiter on a frame with invalid frame content. The EOF_{dti} shall remove a dedicated connection through a Fabric, if present. It shall also indicate that all Class 1 and Class 6 Open Sequences associated with the Connected Nx_Port are abnormally terminated and may require Sequence recovery on a ULP protocol-dependent basis.

The frame containing the EOF_{dti} shall be processed by the receiver in the following manner:

- a) no response frame shall be transmitted
- b) the Data Field may be used at the receiver's discretion (see 8.9.3)
- c) the dedicated connection to the receiver is removed.

8.7.3.4 EOF Deactivate Terminate Invalid (EOF_{dti}) (Class 4)

The EOF_{dti} delimiter shall replace a recognized EOF_{dt} delimiter on a frame with invalid frame content.

All other actions for an EOF_{dti} shall be the same as for an EOF_{dt} delimiter (see 8.7.2.4).

8.7.3.5 EOF Remove Terminate Invalid (EOF_{rtdi})

The EOF_{rtdi} delimiter shall replace a recognized EOF_{rt} delimiter on a frame with invalid frame content.

All other actions for an EOF_{rtdi} shall be the same as for an EOF_{rt} delimiter (see 8.7.2.5).

8.7.3.6 EOF 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:

- a) no response frame shall be transmitted
- b) the Data Field may be used at the receiver's discretion (see 8.9.3)

8.8 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 11, table 13, and figure 13 until the EOF delimiter is transmitted.

Table 13 relates the frame format to the ordered byte stream transferred from the Upper Level Protocol (or FC-4) and transmitted onto a link.

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.

Table 13 - Frame byte order

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
SOF	K28.5 A0	Dxx.x A1	Dxx.x A2	Dxx.x A3
0	R_CNTL	D_ID		
	B0	B1	B2	B3
1	CS_CTL	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	Data Field			
	B24	B25	B26	B27
7	Data Field			
	B28	B29	B30	B31
n to 1	CRC			
	B32	B33	B34	B35
EOF	K28.5 C0	Dxx.x C1	Dxx.x C2	Dxx.x C3

If there is one-byte of fill in the Data Field of this frame, the fill byte is B31. With no optional header present, the relative offset (Parameter Field) shall be specified as follows:

- a) relative offset + 0 specifies B24;
- b) relative offset + 3 specifies B27; and
- c) 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 figures A.1 and A.2).

8.9 Frame reception

8.9.1 Rules

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 a SOF delimiter.
- c) 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.
- d) Frame reception shall continue until an Ordered Set, or a Link Failure is detected.
- e) If the frame content between the SOF and EOF delimiters exceeds the maximum allowable Data Field size as specified by Login, an Nx_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. An Nx_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.
- f) In either process or discard policy, if an EOF_a terminates frame reception, the entire frame shall be discarded, including the Frame_Header and Data Field.
- g) If an EOF_{dt} or EOF_{dti} terminates frame reception, a pending or existing Class 1 dedicated connection shall be recognized to be removed without regard to frame validity. If the frame is in Class 4 and the Class 4 circuit is in the Live State, the Class 4 circuit shall make the transition to the Dormant State.
- h) If an EOF_{rt} or EOF_{rti} terminates frame reception, the Class 4 circuit shall be removed.

8.9.2 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 EOF_{ni}, EOF_{dti}, EOF_{rti} or EOF_a, 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 EOF_n, EOF_t, EOF_{dt} or EOF_{rt} 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 normal 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 that are rejectable in Class 1, 2, 4 and 6 using the P_RJT Link_Response frame (see 11.2.3.4). 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 (e.g. invalid Transmission Word and invalid CRC) are detected, the event count in the Link Error Status Block shall be updated (see 21.8). If delimiter usage does not follow allowable delimiters by class as specified in tables 42 and 46, a valid frame shall be considered rejectable.

8.9.3 Invalid frame processing

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

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

9 Frame_Header

9.1 Introduction

The Frame_Header shall be subdivided into fields as shown in table 14.

Table 14 - Frame_Header

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

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

9.2 Identification

9.2.1 Frame identification

D_ID, S_ID, SEQ_ID, SEQ_CNT, and Sequence Context (see 9.7) uniquely identify a single frame. The OX_ID and RX_ID fields (collectively defined as X_ID) may be used by a Sequence Initiator or Recipient Nx_Port to provide a locally assigned value that 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 Sequence Initiator and Recipient Nx_Ports 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 21.7.2.2), a Recovery_Qualifier range is identified by the Sequence Recipient that consists of S_ID, D_ID, OX_ID and 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 12.3.2.14).

9.2.2 Sequence identification

The set of IDs defined previously (S_ID, D_ID, OX_ID, RX_ID, and SEQ_ID) is referred to as the Sequence_Qualifier throughout the remainder of this standard. An Nx_Port implementation makes use of these IDs in an implementation-dependent manner to uniquely identify open Sequences (see 17.6.1).

NOTE An Nx_Port's freedom to assign a SEQ_ID is based on Sequence Context (Initiator or Recipient). This may affect how an Nx_Port implementation chooses to uniquely identify Sequences.

A given Sequence Initiator Nx_Port may choose to provide frame tracking outside of the signalling protocol of this standard (FC-2). Setting the OX_ID to hex 'FF FF' indicates this. This implies that the Exchange Originator shall

only have one Exchange active with a given destination Nx_Port. If an Nx_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 Nx_Port Responder may choose to provide frame tracking outside of the signalling protocol of this standard (FC-2). Setting the RX_ID to hex 'FF FF' indicates this. If an Nx_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.

9.3 Routing Control (R_CTL)

9.3.1 Introduction

The R_CTL field is a one-byte field in Word 0 Bits 31 to 24 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 to 24), it provides an FC_Port with assistance in frame routing, data routing, or addressing.

The R_CTL field is further subdivided into the ROUTING field (bits 31 to 28) and the INFORMATION field (bits 27 to 24).

9.3.2 ROUTING Field

Table 15 shows the frame types associated with the ROUTING field.

Table 15 - R_CTL - Type Code Summary

R_CTL		Frame type
ROUTING	INFORMATION	
'0000'b	(see table 16)	Device_Data frames (see clause 11)
'0010'b	(see table 61)	Extended Link Services (see 12.3)
'0011'b	(see table 18)	FC-4 Link_Data (see 9.3.3)
'0100'b	(see table 19)	Video_Data (see 9.3.3)
'1000'b	(see table 55)	Basic Link Services (see 12.2)
'1100'b	(see table 45)	Link_Control Frame (see 11.2)
'1111'b	(see table 20)	Extended Routing (see 9.3.3)
Others	Reserved	Reserved

9.3.3 INFORMATION Field

The INFORMATION field is included in R_CTL to assist the receiver of a Data frame in directing the Data Field content to the appropriate buffer pool.

The R_CTL field for Device_Data frames shall be set according to table 16.

Table 16 - Device_Data Information Categories

R-CTL		Description
ROUTING	INFORMATION	
'0000' b	'0000' b	Uncategorized information
	'0001' b	Solicited Data
	'0010' b	Unsolicited Control
	'0011' b	Solicited Control
	'0100' b	Unsolicited Data
	'0101' b	Data Descriptor
	'0110' b	Unsolicited Command
	'0111' b	Command Status
	Others	Reserved

The INFORMATION field value of "Uncategorized information", does not offer assistance in routing.

When the ROUTING field is '0000'b and the INFORMATION field is '0101'b, the Data Descriptor is formatted as shown in table 17.

Table 17 - 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

The R_CTL field for FC-4 Link_Data frames shall be set according to table 18.

Table 18 - FC-4 Link_Data Information Categories

R-CTL		Description
ROUTING	INFORMATION	
'0011' b	'0000' b	Uncategorized information
	'0001' b	Solicited Data
	'0010' b	Unsolicited Control
	'0011' b	Solicited Control
	'0100' b	Unsolicited Data
	'0101' b	Data Descriptor
	'0110' b	Unsolicited Command
	'0111' b	Command Status
	Others	Reserved

The R_CTL field for Video_Data frames shall be set according to table 19.

Table 19 - Video_Data Information Categories

R_CTL		Description
ROUTING	INFORMATION	
'0100' b	'0100' b	Unsolicited Data
	Others	Reserved

The R_CTL field for Extended Routing frames shall be set according to table 20.

Table 20 - Extended Routing Information Categories

R_CTL		Description
ROUTING	INFORMATION	
'1111' b	'0000' b	Vendor Unique
	Others	Reserved

9.4 Address identifiers (D_ID, S_ID)

9.4.1 General

Each Nx_Port shall have a native N_Port_ID that is unique within the address domain of a Fabric. In addition, an Nx_Port may optionally have one or more Alias_IDs that may be shared across multiple Nx_Ports (e.g., Alias addressing may be used to implement a Hunt Group).

An N_Port_ID of binary zeros indicates that an Nx_Port is unidentified. When an Nx_Port completes the Link Initialization (see 7.6.2) Primitive Sequence protocol, its N_Port_ID shall be unidentified. While an Nx_Port is unidentified, it shall

- a) accept all frames with any D_ID value,
- b) not Reject (P_RJT) any frames with a reason code of "Invalid D_ID", and
- c) Reject (P_RJT) frames other than Basic and Extended Link Service with a reason code of "Login required".

An Nx_Port determines its N_Port_ID by performing the Login Procedure as specified in clause 15. During the Login Procedure, an Nx_Port may be assigned an N_Port_ID or it may determine its own N_Port_ID.

9.4.2 Reserved address identifiers

Address identifiers in the range of hex 'FF FC 01' to hex 'FF FC FE' are reserved for Domain Controllers. Address identifiers in the range of hex 'FF FF F0' to hex 'FF FF FE' are reserved for well-known addresses. The address identifier of hex 'FF FF FF' is reserved as a broadcast address. See table 21 for the complete list.

9.4.3 Destination_ID (D_ID)

The D_ID is a three-byte field (Word 0, Bits 23 to 0) that shall contain the address identifier of the destination Nx_Port.

9.4.4 Source_ID (S_ID)

The S_ID is a three-byte field (Word 1, Bits 23 to 0) that shall contain the address identifier of the source Nx_Port.

Table 21 - Domain Controller and Well-known address identifiers

Address Value	Description
hex 'FF FC 01 to hex 'FF FC FE'	Reserved for Domain Controllers
hex 'FF FF F0' to hex 'FF FF F4'	Reserved
hex 'FF FF F5'	Multicast Server (see 23.3.4)
hex 'FF FF F6'	Clock Synchronization Server (see clause 27)
hex 'FF FF F7'	Security Key Distribution Server (See [6], FC-GS-4)
hex 'FF FF F8'	Alias Server (see 24.2)
hex 'FF FF F9'	Quality of Service Facilitator - Class 4 (QoSF) (see 25.3.5.2)
hex 'FF FF FA'	Management Server
hex 'FF FF FB'	Time Server
hex 'FF FF FC'	Directory Server
hex 'FF FF FD'	Fabric Controller
hex 'FF FF FE'	F_Port
hex 'FF FF FF'	Broadcast Alias_ID (see 23.4)

9.5 Class Specific Control (CS_CTL)/Priority

9.5.1 Introduction

The meaning of the CS_CTL field is controlled by the CS_CTL/Priority Enable bit (F_CTL, bit 17). When the CS_CTL/Priority Enable bit is set to zero, word 1, bits 31 to 24 shall be interpreted to be CS_CTL information as defined in 9.5.2. When the CS_CTL/Priority Enable bit is set to one, word 1, bits 31 to 24 shall be interpreted to be Priority information as described in 9.5.3.

9.5.2 CS_CTL

9.5.2.1 General

When bit 17 of F_CTL is set to zero, Word 1, bits 31 to 24 of the Frame_Header is defined as the CS_CTL field. It contains management information for the class of service identified by the SOF. The meaning of the CS_CTL field is dependent on the class of service.

9.5.2.2 Class 1 and Class 6

The CS_CTL field is defined in table 22 for Class 1. Stacked Connect-request shall be meaningful only in the connect-request frame.

Table 22 - CS_CTL field - Class 1

Bit	Abbr.	Meaning
31	Simplex - obsolete	Reserved
30	Stacked Connect-request	0 = Stacked Connect-request not requested 1 = Stacked Connect-request requested
29	COR - obsolete	Reserved
28	BCR - obsolete	Reserved
27 to 24		Reserved

9.5.2.3 Class 2

The CS_CTL field is defined in table 23 for Class 2.

Table 23 - CS_CTL field - Class 2

Bits	Abbr.	Meaning
31	PREF	0 = Frame is delivered with no Preference 1 = Frame may be delivered with Preference
30		Reserved for additional Preference function
29 to 24	DSCP	Differentiated Services Code Point

PREF shall be meaningful in all Class 2 frames.

Bits 29 to 24 shall be define policies to differentiate traffic flows. The default value shall be '000000' b, that is defined as the best effort QoS. Other values are defined in [13], RFC 2597, Assured Forwarding PHB Group, and [14], RFC 2598, An Expedited Forwarding PHB. Values other than '000000' b and those defined in the referenced RFCs are reserved.

9.5.2.4 Class 3

The CS_CTL field is defined in table 24 for Class 3.

Table 24 - 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		Reserved for additional Preference function
29 to 24	DSCP	Differentiated Services Code Point

PREF shall be meaningful in all Class 3 frames.

Bits 29 to 24 shall be define policies to differentiate traffic flows. The default value shall be '000000' b, that is

defined as the best effort QoS. Other values are defined in [13], RFC 2597, Assured Forwarding PHB Group, and [14], RFC 2598, An Expedited Forwarding PHB. Values other than '000000' b and those defined in the referenced RFCs are reserved.

9.5.2.5 Class 4

The CS_CTL field is defined in table 25 for Class 4.

VC_ID shall be meaningful in all Class 4 frames.

Table 25 - CS_CTL field - Class 4

Bit	Abbr.	Meaning
31 to 24	VC_ID	Virtual Circuit Identifier

The Class 4 VC_ID field is a Class of Service specific use of the CS_CTL field. The VC_ID identifies from which VC the frame is associated, 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_ID value that is also identified in the QoS. 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 QoS is recognized by the QoSF and for the CTR when it's LS_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 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_ID.

The Fabric uses the VC_ID field in the frame header to route frames to a destination N_Port. The N_Port during the Class 4 circuit setup process picks the VC_ID field value. Each N_Port picks its own VC_ID values.

The value in the field is either a CTI VC_ID (CIVC_ID) or a CTR VC_ID (CRVC_ID), depending on which N_Port is used to transmit the frame.

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 12).

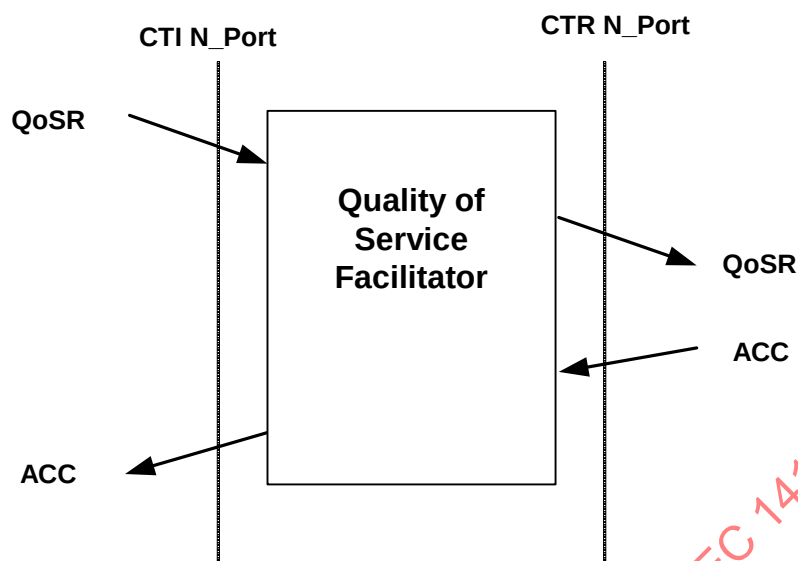


Figure 12 - CIVC_ID and CRVC_ID management

As shown in figure 12, 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).

9.5.3 Priority

9.5.3.1 Introduction

When supported by FC_Ports (see 15.6.5.4.5), the Priority field shall be used to resolve resource contention or to determine the order to deliver frames. The definition and use of the Priority field is class dependent.

Word 1, bits 31 to 24 of the Frame_Header shall be defined as the Priority field when the CS_CTL/Priority Enable bit (F_CTL, bit 17) is set to one. The Priority field contains priority information for the class of service identified by the SOF. A value of '0000000' b in word 1, bits 31 to 25 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 (e.g., a Priority of hex '23' shall be considered to have a lower priority than a Priority of hex '57'). The meaning of Word 1, bit 24 is dependent on the class of service.

9.5.3.2 Class 1 and Class 6

The Priority field is defined in table 26 for Class 1 and Class 6.

Table 26 - Priority Field - Class 1 and Class 6

Word 1, bit(s)	Meaning
31 to 25	Priority
24	0 = No Preemption 1 = Preemption

In Class 1 and Class 6, word 1, bits 31 to 25 shall be interpreted to be the priority value, and word 1, bit 24 shall be interpreted to be the preemption request flag.

The Priority for a dedicated connection shall be established by the Priority value provided by the SOF_{c1} frame. The Priority and Preemption flag shall have no meaning for frames other than SOF_{c1} frames.

When supported by an FC_Port, the Preemption flag shall be used to interrupt and remove a Class 1 or Class 6 Dedicated Connection between Nx_Ports and establish a new Class 1 or Class 6 Dedicated Connection. An Nx_Port sourcing a SOF_{c1} frame with the Preemption flag set initiates preemption. The Preemption flag in a SOF_{c1} frame shall not be meaningful when establishing a new Class 1 or Class 6 Dedicated Connection between unconnected Nx_Ports. Also, the preemption request flag has no meaning for frames other than a SOF_{c1} frame. See 20.1.3 for more information on Preempting a Dedicated Connection.

9.5.3.3 Class 2 and Class 3

The Priority field is defined in table 27 for Class 2 and 3.

Table 27 - Priority Field

Word 1, bit(s)	Meaning
31 to 25	Priority
24	0 = No Preemption 1 = Preemption

In Class 2 and Class 3, word 1, bits 31 to 25 shall be the priority value, and word 1, bit 24 shall be the preemption request flag. The priority for a sequence shall be established by the priority provided by the Sequence Initiator SOFi2 or SOFi3 frame. The Sequence Initiator should set the Priority to the same value for all frames in a given Sequence. Priority should only be set on the SOFi2 or SOFi3 frames. Changing priority in subsequent frames in a sequence may result in out of order delivery of data frames. However, priority does not in itself guarantee in order deliver. Both the Fabric and the Nx_Ports shall not be required to validate the consistency of the Priority Field throughout a Sequence.

When supported by an FC_Port, the Preemption flag shall be used to interrupt and remove a Class 1 or Class 6 Dedicated Connection between Nx_Ports and allow the forwarding of the Class 2 or Class 3 frame. An Nx_Port sourcing an SOF_{x2} or SOF_{x3} frame with the Preemption flag set initiates preemption. The Preemption flag in an SOF_{x2} or SOF_{x3} frame shall not be meaningful if the destination Nx_port(s) are unconnected. Also the Preemption flag has no meaning for frames other than a SOF_{x2} or SOF_{x3} frame. See 20.1.3 for more information on Preempting a Dedicated Connection.

The Sequence Initiator shall set the Priority to the same value for all frames in a given Sequence. Priority shall only be set on a SOFi2 or SOFi3 frame. Changing priority in subsequent frames in a sequence may result in out of order delivery of data frames. However, priority does not in itself guarantee in order delivery.

Both the Fabric and the Nx_Ports shall not be required to validate the constancy of the Priority Field throughout a Sequence.

When supported by an FC_Port, the Preemption flag shall be used to interrupt and remove a Class 1 or Class 6 Dedicated Connection between Nx_Ports and forward a Class 2 or Class 3 frame. An Nx_Port sourcing a SOFi2 or SOFi3 frame with the Preemption flag set initiates preemption. The Preemption flag in a SOFi2 or SOFi3 frame shall not be meaningful when forwarding a Class 2 or Class 3 frame to an unconnected Nx_Port. Also, the preemption request flag has no meaning for frames other than a SOFi2 or SOFi3 frame. See 20.1.3 for more information on Preempting a Dedicated Connection.

9.5.3.4 Class 4

For Class 4, word 1, bits 31 to 24 of the Frame_Header shall be interpreted as the VC_ID as defined in 9.5.2.5.

9.6 Data structure type (TYPE)

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

When the Routing field (word 0, bits 31 to 28) indicates a Link_Control frame other than F_BSY, the TYPE field (word 0, bits 31 to 24) is reserved. F_BSY frames use the TYPE field to indicate a reason code for the F_BSY. When the F_BSY is in response to a Link_Control frame, the Information category field (word 0, bits 27 to 24) of the busied frame is copied by the Fabric into the TYPE field (word 2, bits 27 to 24). The bit encodings are shown in table 45.

NOTE Copying the Link_Control command code allows a source Nx_Port to easily retransmit the frame if it is busied by the Fabric (see 11.2.3.2).

When the Routing bits in R_CTL indicate Basic or Extended Link_Data, TYPE codes are decoded as shown in table 28.

Table 28 - TYPE codes - Link Service

Encoded Value Word 2, bits 31 to 24	Description
hex '00'	Basic Link Service
hex '01'	Extended Link Service
hex '02' to hex 'CF'	Reserved
hex 'D0' to hex 'FF'	Vendor specific

When the Routing bits in R_CTL indicate Video_Data, the TYPE codes are decoded as shown in table 29.

Table 29 - TYPE codes - Video_Data

Encoded Value Word 2, bits 31 to 24	Description
hex '02' to hex 'CF'	Reserved
hex 'D0' to hex 'FF'	Vendor specific

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 30

Table 30 - TYPE codes - FC-4 (Device_Data and Link_Data)

Encoded Value Word 2, bits 31 to 24	Description
hex '00' to hex '03'	Reserved
hex '04'	ISO/IEC 8802-2 LLC/SNAP (in order)
hex '05'	IP over FC, RFC 2625 (See [12])
hex '06' to hex '07'	Reserved
hex '08'	FCP-3 (See [9])
hex '09'	SCSI - GPP
hex '0A' to hex '0F'	Reserved - SCSI
hex '10'	Reserved - IPI-3
hex '11'	IPI-3 Master
hex '12'	IPI-3 Slave
hex '13'	IPI-3 Peer
hex '14' to hex '17'	Reserved
hex '18'	Reserved - SBCCS
hex '19'	SBCCS - Channel
hex '1A'	SBCCS - Control Unit
hex '1B'	FC-SB-3 Channel to Control Unit
hex '1C'	FC-SB-3 Control Unit to Channel
hex '1D' to hex '1F'	Reserved – FC-SB-3
hex '20'	Fibre Channel Common Transport (FC-GS-4)
hex '21'	Reserved
hex '22'	Switch Fabric Internal Link Services (FC-SW-3)
hex '23'	FC-AL
hex '24'	SNMP
hex '25' to hex '27'	Reserved - Fabric Services
hex '28' to hex '2F'	Reserved
hex '30' to hex '33'	Reserved - Scalable Coherent Interface
hex '34' to hex '3F'	Reserved

Table 30 - TYPE codes - FC-4 (Device_Data and Link_Data) (Continued)

Encoded Value Word 2, bits 31 to 24	Description
hex '40'	HIPPI-FP
hex '41' to hex '47'	Reserved - HIPPI
hex '48'	MIL-STD-1553 (FC-AE)
hex '49'	ASM (FC-AE)
hex '4A' to hex '4F'	Reserved for FC-AE
hex '50' to hex '57'	Reserved for future use by FC-BB
hex '58'	Virtual Interface (FC-VI)
hex '59' to hex '5F'	Reserved
hex '60'	FC-AV Container
hex '61' to hex '63'	Reserved - FC-AV
hex '64' to hex 'DE'	Reserved
hex 'DF'	Reserved for RNID uses
hex 'E0' to hex 'FF'	Vendor specific

9.7 Frame Control (F_CTL)

9.7.1 Introduction

The Frame Control (F_CTL) field (Word 2, Bits 23 to 0) is a three-byte field that contains control information relating to the frame content. The following subclause describe the valid uses of the 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 11.2.3.4) for Classes 1, 2, 4 and 6. The format of the F_CTL bits are defined in table 31.

When a bit is designated as meaningful under a set of conditions, that bit shall be ignored if those conditions are not present (e.g., Bit 18 is only meaningful when bit 19 is set to one; this means that bit 18 shall be ignored unless bit 19 is set to one).

9.7.2 Exchange Context

An Exchange shall be started by the Originator LCF within an Nx_Port. The other Nx_Port of the Exchange shall be known as the Responder. If the Exchange Context bit (bit 23) is set to zero, the S_ID is associated with the

Exchange Originator. If the bit is set to one, the S_ID is associated with the Exchange Responder.

Table 31 - Exchange/Sequence Control (F_CTL)

Control Field	Word 2, Bits	Description	Reference
Exchange Context	23	0 = Originator of Exchange 1 = Responder of Exchange	See 17.4.
Sequence Context	22	0 = Sequence Initiator 1 = Sequence Recipient	See 17.4
First_Sequence	21	0 = Sequence other than first of Exchange 1 = first Sequence of Exchange	See 17.4
Last_Sequence	20	0 = Sequence other than last of Exchange 1 = last Sequence of Exchange	See 17.7
End_Sequence	19	0 = Data frame other than last of Sequence 1 = last Data frame of Sequence	See 9.7.6
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 4 circuit	See 20.7.3 See clause 25
CS_CTL/Priority Enable	17	0 = Word 1, Bits 31 to 24 = CS_CTL 1 = Word 1, Bits 31 to 24 = Priority	See 9.5.2 See 9.5.3
Sequence Initiative	16	0 = hold Sequence initiative 1 = transfer Sequence initiative	See 17.6.3 See 17.6.4
X_ID reassigned - Obsolete	15		
Invalidate X_ID - Obsolete	14		
ACK_Form	13 to 12	'00' b = No assistance provided '01' b = Ack_1 Required '10' b = reserved '11' b = Ack_0 Required	See 9.7
Data Compression – Obsolete	11		

Table 31 - Exchange/Sequence Control (F_CTL) (Continued)

Control Field	Word 2, Bits	Description	Reference
Data Encryption - Obsolete	10		
Retransmitted Sequence	9	0 = Original Sequence transmission 1 = Sequence retransmission	See 21.7.2.3
Unidirectional Transmit (Class 1) or Remove Connection (Class 4 only)	8	0 = Bi-directional transmission (Class 1) 1 = Unidirectional transmission (Class 1)	See 20.5.4
		0 = Retain or deactivate circuit (Class 4) 1 = Remove circuit (Class 4)	See clause 25
Continue Sequence Condition	7 to 6	Last Data frame - Sequence Initiator '00' b = No information '01' b = Sequence to follow-immediately '10' b = Sequence to follow-soon '11' b = Sequence to follow-delayed	See 17.6.5.6
Abort Sequence Condition	5 to 4	ACK frame - Sequence Recipient '00' b = Continue sequence '01' b = Abort Sequence, Perform ABTS '10' b = Stop Sequence '11' b = Immediate Sequence retransmission requested	See 12.2.2
		Data frame (1 st of Exchange) - Sequence Initiator '00' b = Abort, Discard multiple Sequences '01' b = Abort, Discard a single Sequence '10' b = Process policy with infinite buffers '11' b = Discard multiple Sequences with immediate retransmission	See 12.2.2
Relative offset present	3	0 = Parameter field defined for some frames 1 = Parameter Field = relative offset	See 9.13
Exchange reassembly	2	Reserved for Exchange reassembly	See 9.7.16
Fill Data Bytes	1 to 0	End of Data Field - bytes of fill '00' b = 0 Bytes of fill '01' b = 1 Byte of fill (last byte of Data Field) '10' b = 2 Bytes of fill (last 2 bytes of Data Field) '11' b = 3 Bytes of fill (last 3 bytes of Data Field)	See 9.7.17

9.7.3 Sequence Context

A Sequence shall be started by a Sequence Initiator facility within an Nx_Port. The destination FC_Port of the Sequence shall be known as the Sequence Recipient. If the Sequence Context (bit 22) bit is set to zero, it indicates that the S_ID is associated with the Sequence Initiator. If the bit is set to one, it indicates that the S_ID is associated with the Sequence Responder. This indicates the Sequence Context.

Knowledge of Sequence context 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.

9.7.4 First_Sequence

The First_Sequence bit (bit 21) 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.

9.7.5 Last_Sequence

The Last_Sequence bit (bit 20) shall be set to one on the last Data frame in the Last Sequence of an Exchange. However, it may 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.

9.7.6 End_Sequence

The End_Sequence bit (bit 19) 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 that may not be the last Data frame transmitted. This indication is used for Sequence termination by the two Nx_Ports involved in addition to EOF_{ti} or EOF_{dt} (see 17.3.8). This bit shall be set to zero for all other frames within a Sequence.

9.7.7 End_Connection (E_C) (Class 1 or 6) or Deactivate Class 4 circuit

In Class 1 or Class 6, the E_C bit (bit 18) shall be set to one in the last Data frame of a Sequence to indicate that the Nx_Port transmitting E_C is beginning the disconnect procedure. The Nx_Port transmitting E_C set to one on the last Data frame of a Sequence is requesting the receiving Nx_Port transmit an ACK frame terminated by EOF_{dt} if the receiving Nx_Port has complete the currently active Sequence. If the receiving Nx_Port is not able to transmit EOF_{dt}, E_C set to one requests that the receiving Nx_Port complete the currently active Sequence 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 Class 1 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 20.7.3 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 Nx_Port transmitting E_C is beginning the deactivation or removal procedure. The Nx_Port transmitting E_C set to one on the last Data frame of a Sequence is requesting the receiving Nx_Port to transmit an ACK frame terminated by EOF_{dt}, in the case of deactivation or EOF_{rt} in the case of removal if the receiving Nx_Port has completed the currently active Sequence. If bit 8, Remove_Connection (R_C) is set to one in addition to E_C set to one, then a Class 4 circuit removal is requested, otherwise a deactivation is requested. If the receiving Nx_Port is not able to deactivate or remove the Class 4 circuit, E_C set to one requests that the receiving Nx_Port complete the currently active Sequence and not initiate any new Sequences during the current activation cycle.

9.7.8 CS_CTL/Priority Enable

When the CS_CTL/Priority Enable bit (bit 17) is set to zero, word 1, bits 31 to 24 of the Frame_Header shall be interpreted to be the CS_CTL field as described in 9.5.2. When CS_CTL/Priority Enable is set to one, word 1, bits 31 to 24 of the Frame_Header shall be interpreted to be the Priority field as described in 9.5.3.

In Class 1 and Class 6, CS_CTL/Priority Enable is meaningful only for the SOF_{c1} frame. CS_CTL/Priority Enable

shall be ignored for all subsequent frames in a Dedicated Connection.

In Class 2 and Class 3, the Sequence Initiator shall set CS_CTL/Priority Enable to the same value for all frames in a given Sequence. CS_CTL/Priority Enable shall only be modified on the initial Sequence SOFi2 or SOFi3 frame. Changing CS_CTL/Priority Enable in subsequent frames may result in out of order delivery of frames.

Both the Fabric and the Nx_Ports shall not be required to validate the constancy of CS_CTL/Priority Enable throughout a Sequence.

In Class 4, CS_CTL/Priority Enable shall be ignored. In Class 4 frames, bits 31 to 24 of word 1 are always VC_ID.

9.7.9 Sequence Initiative

The Originator of an Exchange shall initiate the first Sequence as the Sequence Initiator. If the Sequence Initiative bit (bit 16) 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 set to one). In Class 1, 2, 4 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 set to one. Setting bit 16 to one is only meaningful when End_Sequence is set to one.

9.7.10 ACK_Form

The ACK_Form bits (bits 13 to 12) provide an optional assistance to the Sequence Recipient by translating the ACK capability bits in the Nx_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, 4 or 6 Data frames of a Sequence and on all connect-request frames. ACK_Form is not meaningful on Class 1, 2, 4 or 6 Link_Control frames, or any Class 3 frames. The meaning of the ACK_Form bits is given in table 31.

9.7.11 Retransmitted Sequence

The Retransmitted Sequence bit (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 '11' b (Discard multiple Sequences with immediate retransmission). Retransmitted Sequence shall be set to one if the Sequence Initiator has received an ACK with the Abort Sequence Condition bits (F_CTL bits 5 to 4) set to one. Retransmitted Sequence shall be set to one in all frames of the retransmitted Sequence. If the Sequence Initiator is not able to determine that all Sequences are complete prior to the Sequence identified in the ACK with the Abort Sequence Condition bits set to one (i.e., missing final ACK), 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) ELS request (see 21.7.2.3).

9.7.12 Unidirectional Transmit or Remove_Connection

For Class 1, Unidirectional Transmit bit (bit 8) is meaningful on a connect-request (SOFc1) in Class 1. If Unidirectional Transmit is set to zero, 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 Unidirectional Transmit is set to one, the dedicated connection is unidirectional and only the Nx_Port that transmitted the connect-request that establishes the Connection shall transmit Data frames.

For Class 1 other than the connect-request, Unidirectional Transmit is meaningful on the first and last Data frames of a Sequence. After the connect-request with Unidirectional Transmit set to one, the Connection Initiator may set Unidirectional Transmit to zero 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 Unidirectional Transmit to zero, in an ACK frame. Once set to zero in an ACK, all subsequent ACKs shall

be transmitted with Unidirectional Transmit set to zero. The Connection Initiator is not required to honor the request to become bi-directional. See 20.5.4 for additional requirements.

For Class 4, the Remove_Connection (R_C) bit is only meaningful on the last Data frame of a Sequence and only if the E_C bit is set to one. See the requirements for bit 18, End_Connection (E_C) (Class 1 or 6) or Deactivate Class 4 circuit (see 9.7.7).

For Class 6, Unidirectional Transmit Bit (bit 8) is meaningful on a connect-request (SOF_{c1}). Unidirectional Transmit shall be set to one, the dedicated connection is unidirectional and only the Nx_Port that transmitted the connect-request that establishes the Connection shall transmit Data frames.

9.7.13 Continue Sequence Condition

The Continue Sequence Condition bits (bits 7-6) 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 set to one and the Sequence Initiative bit set to one.

The Continue Sequence Condition bits are also meaningful on the ACK to the last Data frame (End_Sequence set to one) when Sequence Initiative is also transferred (Sequence Initiative bit set to one). In this case, the Continue Sequence Condition bits shall indicate the expected delay until the Nx_Port that received Sequence Initiative transmits the first frame of its next Sequence.

The Continue Sequence Condition bits may be used to manage link resources within an Nx_Port (e.g., 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 an Nx_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 being used.

The meaning of the Continue Sequence Condition bits is given in table 32.

Table 32 - Continue Sequence Condition Bits Definition

Encoding	Meaning
'00' b	No information is being offered regarding when the next Sequence is being transmitted.
'01' b	The next consecutive Sequence for this Exchange shall be transmitted immediately.
'10' b	The next consecutive Sequence for this Exchange shall be transmitted in a time period that is less than the time to remove and reestablish a Class 1 Connection (soon).
'11' b	The next consecutive Sequence transmission for this Exchange shall be delayed for a period of time that 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 in the Exchange.

9.7.14 Abort Sequence Condition

The Abort Sequence Condition bits (bits 5 to 4) 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 the 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 '10' b on each Data frame within 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 first Sequence of an Exchange shall be the error policy supported by both Nx_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 Nx_Ports participating in the Exchange (see 21.6.3).

The definition of the Abort Sequence Condition Bits by the Sequence Initiator is given in table 33.

An Nx_Port, in the PLOGI sequence shall indicate process policy support. Discard policy shall be supported.

If the delivery order of Sequences, without gaps, is required by an 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.

The definition of the Abort Sequence Condition Bits by the Sequence Recipient is given in table 34.

Table 33 - Abort Sequence Condition Bits Definition by Sequence Initiator

Encoding	Meaning
'00' b	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 Classes 1, 2, 3, 4 and 6.
'01' b	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.
'10' b	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 11.2.2.3.3).
'11' b	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 in F_CTL set to one 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.

9.7.15 Relative offset present

When relative offset present 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. Relative offset present is only meaningful on Data frames of

a Sequence and shall be ignored on all other frames. Relative offset present is not meaningful on Link_Control or Basic Link Service Link Data frames. When relative offset present is set to zero on a Data frame, the value in the Parameter Field shall be passed to the upper level.

9.7.16 Exchange reassembly

The Sequence Initiator shall set the Exchange reassembly bit (bit 2) to zero to indicate that the Payload in this Data frame is associated with an Exchange between a single pair of Nx_Ports. Therefore, reassembly is confined to a single destination Nx_Port.

Exchange reassembly being set to one 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 Nx_Ports at either the source, destination, or both.

Table 34 - Abort Sequence Condition Bits Definition by Sequence Recipient

Encoding	Meaning
'00' b	Continue Sequence
'01' b	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 12.2.2 and 21.7.2.2 for a description of the Abort Sequence protocol.
'10' b	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 21.7.3 for a description of the Stop Sequence protocol.
'11' b	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 21.7.2.3). 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.

9.7.17 Fill Data Bytes

When the bits associated with the Fill Data Bytes (bits 1 to 0) 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 this standard but shall be a valid data byte.

Fill Data Bytes 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 (e.g., if a Sequence contains Data frames of a single Information Category, non-zero values Fill Data Bytes shall only be meaningful on the last Data frame of the Sequence). The fill Data bytes shall not be included in the Payload.

9.7.18 F_CTL bits on Data frames

Table 35 shows the interactions between specific bits within the F_CTL field. The top part of table 35 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 control function may become effective when an F_CTL bit is set to one. The locations where the function is meaningful are indicated in the top part of the table 35.

The bottom part of table 35 describes those bits that are conditionally meaningful (e.g., Bit 19 set to one (column) is only meaningful on the last Data frame of a Sequence. Bit 18 set to one (column) is only meaningful on the last Data frame when bit 19 set to one).

Table 35 - 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 to 6	5 to 4	3 = 1	1 to 0
1 st frame of Seq	M	M	M				M		M	M			M	
last frame of Seq	M	M	M		M		M		M	M		MF	M	M
any frame of Seq	M	M	M				M		M	M			M	
Class 1	M	M	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														
Legend: M = Meaningful MF = Only meaningful on first Data frame of a Sequence ML = Only meaningful on last Data frame of a Sequence														

9.7.19 F_CTL bits on Link_Control frames

Table 36 shows the interactions with F_CTL bits on ACK, BSY, and RJT frames and should be reviewed together with table 35. F_CTL bits 19, 18, and 16 in an ACK frame are transmitted to reflect confirmation (1) or denial (0) of those indications by the Sequence Recipient (e.g., if bits 5 to 4 are set to '01' b in response to a Data frame in which bit 19 is set one and bit 16 is set to one, 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 17.3.8, 17.3.10 and 21.7.2.2 for additional information on setting the Abort Sequence Condition bits.

A control function may become effective when a F_CTL bit is set to one. The locations where the function is meaningful are indicated in the top part of the table 36.

Table 36 - 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 to 6	5 to 4	3	1 to 0
ACK to 1 st frame	V	V	E										Ma		
ACK to last frame	V	V	E							M	M		Ma		
ACK to any frame	V	V	E							M	M	ML	Ma		
ACK to Connect-request	V	V	E	E	ML		ML			M	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								

Legend:

M = Meaningful

Ma = Meaningful only on ACK frames

ML = Meaningful only on last ACK, BSY and RJT frames of a Sequence

E = Echo (meaningful) - contains the same value as the received frame

V = Inverse or invert (meaningful) - contains the inverse of the received frame

9.8 Sequence_ID (SEQ_ID)

The SEQ_ID is a one-byte field (Word 3, Bits 31 to 24) assigned by the Sequence Initiator that shall be unique for a specific D_ID and S_ID pair while the Sequence is open. Both the Sequence 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 (e.g., S_ID, D_ID, or the other Nx_Port's X_ID) for tracking (see 9.2.1, 9.11 and 9.12).

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 15.6.5.10) (e.g., 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 (e.g., consecutive SEQ_IDs of 21-74-21-74 is acceptable. The examples show when a SEQ_ID is be allowed to be repeated). A series of SEQ_IDs for the same Exchange may also be random and never repeat (see 17.3.4). See 17.6.2 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 17.8.2 for a description of the Sequence Status Block maintained by the Sequence Recipient.

9.9 Data Field Control (DF_CTL)

Data Field Control (DF_CTL) is a one-byte field (Word 3, Bits 23 to 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. The ESP_Header bit is also meaningful on Extended Link Service and FC-4 Link Service frames. Control bit usage is shown in table 37.

Table 37 - DF_CTL bit definition

Word 3, Bit(s)	Optional Header
23	Reserved
22	0 b = Neither ESP_Header nor ESP_Trailer 1 b = Both ESP_Header and ESP_Trailer
21	0 = No Network_Header 1 = Network_Header
20	0 = No Association_Header 1 = Association_Header
19 to 18	Reserved
17 to 16	'00' b = No Device_Header '01' b = 16 Byte Device_Header '10' b = 32 Byte Device_Header '11' b = 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 13 and figure 14.

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 (e.g., 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 10 for Optional Headers requirements.

9.10 Sequence count (SEQ_CNT)

The sequence count (SEQ_CNT) is a two-byte field (Word 3, Bits 15 to 0) that shall indicate the sequential order of Data frame transmission within a single Sequence or multiple consecutive Sequences for the same Exchange. The SEQ_CNT of the first Data frame of the first Sequence of the Exchange transmitted by either the Originator or Responder shall be binary zero. The SEQ_CNT of each subsequent Data frame in the Sequence shall be incremented by one.

If a Sequence is streamed, the SEQ_CNT of the first Data frame of the Sequence shall be incremented by one from the SEQ_CNT of the last Data frame of the previously sent Sequence (this is termed continuously increasing SEQ_CNT). If a Sequence is non-streamed, the starting SEQ_CNT 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 SEQ_CNT shall wrap to zero after reaching a value of 65 535. The SEQ_CNT shall then only be incremented to (but not including) the SEQ_CNT of an unacknowledged frame of the same Sequence. Otherwise, data integrity is not ensured. Sequences of Data frames and SEQ_CNT values are discussed in clause 17. See 15.6.5.9 regarding data integrity, Credit, and SEQ_CNT.

9.11 Originator Exchange_ID (OX_ID)

The Originator Exchange_ID is a two-byte field (Word 4, Bits 31 to 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 set 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 set to 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 that comprises an Exchange. See 17.8.1 for a description of the Exchange Status Block.

NOTE If hex 'FF FF' is used as the OX_ID throughout the Exchange, the Originator uses an alternate Sequence tracking mechanism. If the OX_ID is unique, it may be used as an index into a control structure that may be used in conjunction with other constructs to track frames.

9.12 Responder Exchange_ID (RX_ID)

The Responder Exchange_ID is a two byte field (Word 4, Bits 15 to 0) assigned by the Responder that 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 set a unique value for RX_ID other than hex 'FF FF', if RX_ID is being used, by one of two methods:

- a) in an ACK to a Data frame in the first Sequence of an Exchange in Class 1 and 2 or
- b) in the first Sequence transmitted as a Sequence Initiator, if any, in Class 3.

An RX_ID of hex 'FF FF' shall indicate that the RX_ID is unassigned. RX_ID is not used in Class 6 and shall be set to hex 'FF FF' for all Class 6 frames. 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 17.3.2 and 17.5.3).

A Responder Exchange Status Block associated with the RX_ID is used to track the progress of a series of Sequences that compose an Exchange. See 17.8.1 for a description of the Exchange Status Block.

NOTE If hex 'FF FF' is used as the RX_ID throughout the Exchange, the Responder uses an alternate Sequence tracking mechanism. If the RX_ID is unique, it may be used as an index into a control structure that may be used in conjunction with other constructs to track frames.

9.13 Parameter

The Parameter field (Word 5, Bits 31 to 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 offset, 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 offset is expressed in terms of bytes (see 8.8). 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 zero, the Parameter field shall be set and interpreted in a protocol specific manner that may depend on the type of Information Unit carried by the frame.

Continuously increasing relative offset is 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) is the initial relative offset for the Information Category I.

See clause 19 for relative offset requirements. See clause 11 for requirements using the Parameter field in Link_Control frames. See clause 11 for requirements using the Parameter field in Basic Link Data frames.

10 Optional headers

10.1 Introduction

Optional headers defined within the Data Field of a frame are

- a) ESP_Header and ESP_Trailer,
- b) Network_Header,
- c) Association_Header, and
- d) 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 13 and figure 14.

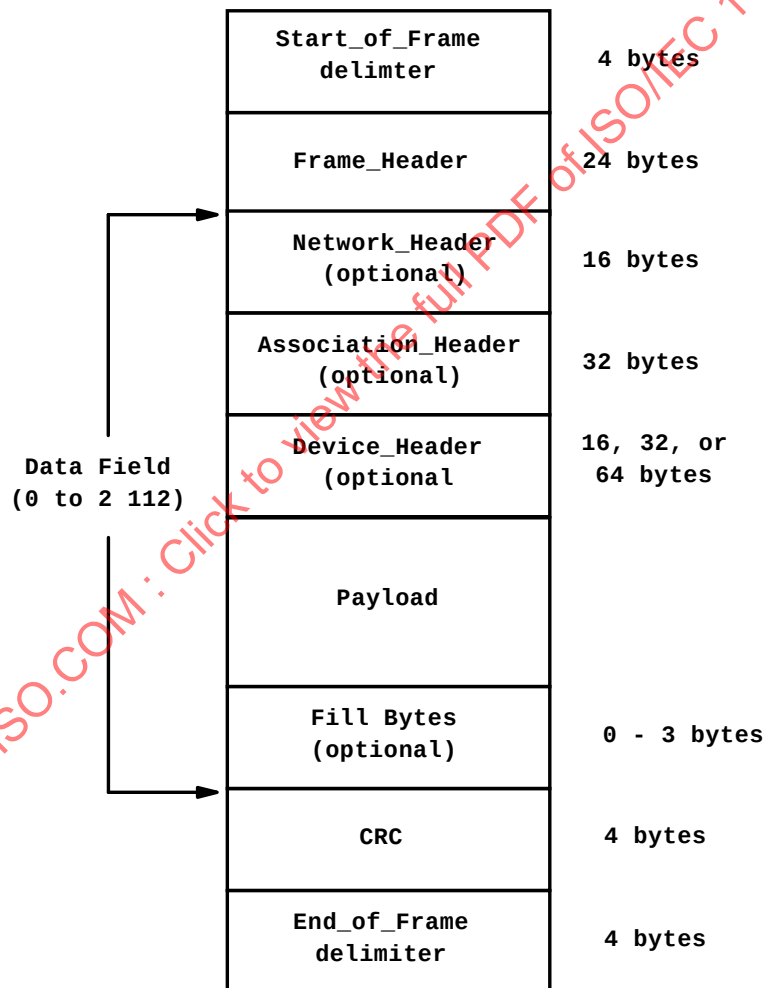


Figure 13 - Frame structure when ESP_Header is not used

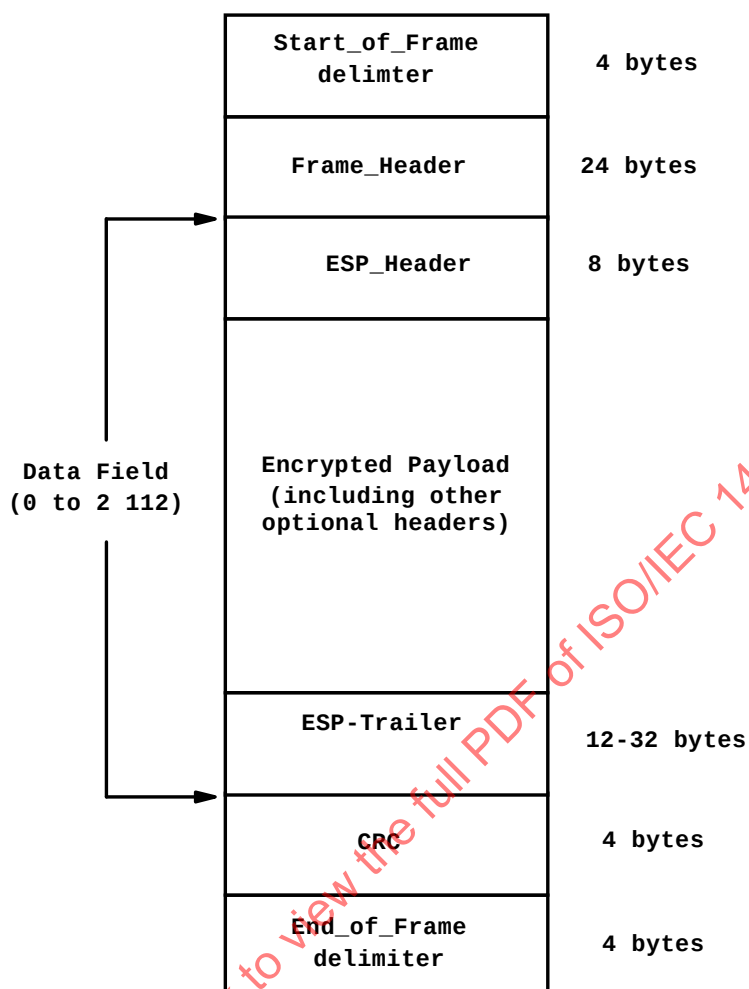


Figure 14 - Frame structure with ESP_Header and ESP_Trailer

If DF_CTL bit 22 is set to 1, an ESP_Header shall be the first 8 bytes of the Data Field, and the ESP_Trailer shall end the Data Field. No Fill Bytes are required when the ESP_Trailer is present. 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.

Optional headers are provided for use of the FC-4 layer. If an optional header different from the ESP_Header is being used within a Sequence, it shall be present in the first data frame of that Sequence, and may be present in all data frames of that Sequence. If the ESP_Header is being used within a Sequence, it shall be present in all frames of that Sequence. The use of the optional headers is not defined by this standard.

10.2 ESP_Header

The Encapsulating Security Payload (ESP) is defined in the IETF document RFC 2406 (see [25]). It is a generic mechanism to provide confidentiality, data origin authentication, and anti-replay protection to IP packets. FC-SP defines how to use ESP in Fibre Channel, including any negotiation procedure, additional encryption/authentication algorithm and processing requirements. This clause defines the structure of a Fibre Channel frame conveying an ESP_Header.

ESP shall be applied to FC frames in transport mode (see [25], RFC 2406). The Authentication option shall be

used, Confidentiality may be negotiated by the two communicating FC_Ports.

Figure 14 shows the format of an FC frame to which ESP is applied. A sender shall apply ESP processing to the Data Field of an FC frame as follows:

- a) Add a fixed length ESP_Header (8 bytes), specifying a Security Parameter Index (SPI) and an ESP Sequence Number.
- b) Pad the Payload Data received from the FC-4, including any possible other optional header, to the block size required by the negotiated encryption/authentication algorithms, that is a multiple of 32 bits. This ensure that the FC fill bytes are not needed. The Pad Length field shall contain the length of this ESP padding.
- c) Encrypt, if negotiated, the data resulting from item b).
- d) Compute an Integrity Check Value (ICV), using the negotiated authentication algorithm and parameters, covering:
 - A) the FC_Header, with the S_ID, D_ID, and CS_CTL/Priority fields set to zero for the purpose of the ICV computation;
 - B) the ESP_Header;
 - C) the data resulting from item c).
- e) Add an ESP_Trailer containing the ICV computed in item d). The length of the ESP_Trailer shall be negotiated and shall be a multiple of 32 bits.

A receiver shall process an FC frames carrying an ESP_Header and ESP_Trailer as follows:

- a) Check the ESP_Header, using the SPI to retrieve the negotiated parameters required to interpret the received FC frame, and the ESP Sequence Number to avoid replay attacks (see [25], RFC 2406). The length of the ESP_Trailer is one of the retrieved parameters.
- b) Compute an ICV, using the retrieved parameters, covering:
 - A) the FC_Header, with the S_ID, D_ID, and CS_CTL/Priority fields set to zero for the purpose of the ICV computation;
 - B) the ESP_Header;
 - C) the encrypted payload.
- c) Check the computed ICV with the content of the ESP_Trailer. If they are equal the authentication is successful, otherwise not.
- d) Decrypt, if negotiated, the encrypted payload.
- e) Remove the ESP padding and pass the resulting Payload Data, including any other optional header, to the upper level.

This specification adheres to [25], RFC2406, except for the ICV coverage, that shall include the FC header, except the S_ID, D_ID, and CS_CTL/Priority fields, because the interpretation of a Fibre Channel Payload is determined by the fields contained in the FC header. The CS_CTL/Priority field has to be excluded because is a mutable field, while excluding S_ID and D_ID permits address translation, if needed.

The ESP processing shall be transparent to the FC-4. On the sending side the ESP processing shall be applied to every frame of a sequence to be protected. On the receiving side, the ESP processing shall be applied to every frame that carries an ESP_Header, and only after that the sequence shall be reassembled and sent to the FC-4.

The ESP_Header and ESP_Trailer, if used, shall be present in every frame of a Sequence. If the receiving FC_Port does not support the ESP_Header function, it shall discard the FC frame.

The ESP_Header and ESP_Trailer format in a frame is specified in table 38 as follows:

- a) the ESP_Header consists of words 6 and 7;
- b) the ESP_Trailer consists of words P+1 through Q;
- c) confidentiality covers words 8 through P; and
- d) authentication covers words 0 through P.

Table 38 - ESP_Header and ESP_Trailer in a frame

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	R-CTL	D_ID		
1	CS_CTL / Priority	S_ID		
2	TYPE	F_CTL		
3	SEQ_ID	DF_CTL	SEQ_CNT	
4	OX_ID		RX_ID	
5	Parameter			
6	Security Parameter Index (SPI)			
7	ESP Sequence Number			
8 .. N	Payload Data (variable length)			
N+1 .. P	Padding (0 to 255 bytes)			
		Pad Length		Not meaningful
P+1 .. Q	Integrity Check Value			

10.3 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 table 39, is an optional header within the Data_Field content. Its presence shall be indicated by bit 21 in the DF_CTL field 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. See clause 14 for the definition of these fields.

Table 39 - Network_Header

Bits Word	31 .. 28	23 .. 00
0	D_NAA	Network_Destination_Address (high order bits)
1		Network_Destination_Address (low order bits)
2	S_NAA	Network_Source_Address (high order bits)
3		Network_Source_Address (low order bits)

The Network_Header, if used, shall be present only in the first Data frame of a Sequence. If the receiving Nx_Port does not support the header function, it shall ignore the header and skip the Data Field by the header length (16 bytes). Destination Network_Address_Authority (D_NAA) or Source Network_Address_Authority (S_NAA) field

indicates the format of the Name_Identifier used for the network address. See clause 14 for a description of the Name_Identifier formats.

10.4 Association_Header

10.4.1 Introduction

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 an Nx_Port has indicated during Login that an Initial Process_Associator is required to communicate with it, the Association_Header shall be used by that Nx_Port to identify a specific Process or group of Processes within a node associated with an Exchange.

The Association_Header shall be subdivided into fields as illustrated in table 40. 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 41. The contents of each Associator of the Association_Header are meaningful to the node that generates that particular field. They may not be meaningful to the receiving node.

Table 40 - Association_Header

Bits Word	31 .. 24	23 .. 00
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	

The Association_Header is applicable to Class 1, 2, 3 or 6.

The Association_Header is provided to support system architectures that require more than two levels of identifiers, (i.e., X_ID and SEQ_ID).

The Association_Header, if present, shall be present only in the first Data frame of a Sequence.

Table 41 specifies the meaning of the Validity bits (V) in table 41.

Table 41 - Association_Header Validity bits (Word 0, Bits 31 to 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 to 25	Reserved
24	Multicast Process_Associator 0 = Unicast 1 = Multicast

10.4.2 Process_Associators

10.4.2.1 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 where 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 Nx_Port.
- The contents of Process_Associator fields are implementation dependent.
- A Process_Associator shall span, within the node, all Nx_Ports having access to the related process.

10.4.2.2 Multicast Process_Associator

The Multicast Process_Associator bit set to one shall indicate that the recipient Nx_Port shall process the passed Process_Associator as a Multicast Process_Associator (i.e., the Nx_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 Recipient Nx_Port shall process the passed Process_Associator as a unicast Process_Associator (i.e., the Nx_Port shall pass the Payload of the received frame to the image specified by the Process_Associator). See clause 23 for a description of Multicast support.

10.4.2.3 Operation_Associators

Operation_Associators are obsolete in this standard. See FC-PH, FC-PH-2, and FC-PH-3 for information about this obsolete field and its functions.

10.5 Device_Header

The Device_Header, if present, shall be 16, 32, or 64 bytes in size. The contents of the Device_Header are controlled at a level above FC-2 based on the TYPE field (see 9.6).

The Device_Header, if present, shall be present either in the first Data frame or in all Data frames of a Sequence. ULP types may use a Device_Header, requiring the Device_Header 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".

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

11 Data frames and responses

11.1 Data frames

11.1.1 Introduction

When the term Data frame is used in this standard, it refers to any of the types of Data frames that may be transmitted.

Data frames may be used to transfer information (e.g., data, control, and status information from a source Nx_Port to a destination Nx_Port). In Class 1, 2, 4 and 6, each Data frame successfully transmitted shall be acknowledged to indicate successful delivery to the destination Nx_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, 4 and 6.

Data frames may be streamed, (i.e., a single Nx_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 15 for the specification of Login and Service Parameters and clause 18 for the specification of flow control rules.

A set of one or more Data frames, related by the same SEQ_ID transmitted unidirectionally from one Nx_Port to another Nx_Port, is called a Sequence.

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 21.6.4, 21.6.5 and 21.7, respectively, for Sequence integrity, Sequence error detection, and Sequence recovery requirements.

Each Data frame within a Sequence shall be transmitted within an E_D_TOV timeout period to avoid timeout errors at the destination Nx_Port.

11.1.2 Frame Delimiters

Table 42 specifies, by class, the allowable frame delimiters for Data frames (see 8.3 and 8.7).

Table 42 - Allowable Data frame delimiters

Data frame	Delimiters
Class 1	SOFc1, SOFi1, SOFn1, EOFn
Class 2	SOFi2, SOFn2, EOFn
Class 3	SOFi3, SOFn3, EOFn, EOFt
Class 4	SOFc4, SOFi4, SOFn4, EOFn
Class 6	SOFc1, SOFi1, SOFn1, EOFn

11.1.3 Addressing

The S_ID field designates the source Nx_Port of the Data frame. The D_ID field designates the destination Nx_Port of the Data frame.

11.1.4 Data Field

The Data_Field 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 10).

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 to 0 (see 9.7) and shall only be meaningful on the last frame of an instance of an Information Category. The fill byte value is not specified by this standard.

11.1.5 Payload size

The Payload size is determined by the number of bytes between the SOF and EOF minus the 24-byte Frame_Header, any Optional Headers, the fill bytes (0, 1, 2, or 3) and the CRC.

11.1.6 Responses

11.1.6.1 R_RDY response

R_RDY transmission shall indicate that the interface buffer, that received the frame, is available for further frame reception. By transmitting an R_RDY, the port has agreed to receive a frame up to the buffer-to-buffer Receive Data Field Size if logged in, and 128 if not logged in. The use of R_RDY is defined for each class as follows:

- a) 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.
- b) In Class 2 and Class 3, Data and Link_Control frames received shall be responded to by transmitting the R_RDY Primitive Signal.
- c) The R_RDY response is not applicable to Class 4.

11.1.6.2 Data frame responses

11.1.6.2.1 Introduction

Responses to Data frames are called Link_Control response frames (see 11.2). There are two types:

- a) ACK frames - ACK_0 and ACK_1
- b) Link_Response frames - P_BSY, P_RJT, F_BSY, and F_RJT.

All Link_Control response frames shall be transmitted in the same class as the frame that it is responding to. However, ACK_0 or ACK_1 to a connect-request frame (SOF_{c1}) shall be sent as a Class 1 frame and not subject to buffer-to-buffer flow control.

11.1.6.2.2 ACK frames - successful Data frame delivery

Table 43 defines what ACK frames shall be used for each class for successful Data frame delivery.

Table 43 - ACK Frames by Class

Data frame	ACK
Class 1	ACK_0, ACK_1
Class 2	ACK_0, ACK_1
Class 3	No Response
Class 4	ACK_0, ACK_1
Class 6	ACK_0, ACK_1

11.1.6.3 Link_Response frames - Unsuccessful Data frame delivery

Table 44 defines what RJT or BSY frames shall be used for each class for unsuccessful Data frame delivery.

Table 44 - Link_Response Frames by Class

Data frame	ACK
Class 1	F_BSY (Fabric Busy) P_BSY (Nx_Port Busy) F_RJT (Fabric Reject) P_RJT (Nx_Port Reject)
Class 2	F_BSY (Fabric Busy) P_BSY (Nx_Port Busy) F_RJT (Fabric Reject) P_RJT (Nx_Port Reject)
Class 3	No Response
Class 4	F_RJT (Fabric Reject) P_RJT (N_Port Reject)
Class 6	F_BSY (Fabric Busy) P_BSY (N_Port Busy) F_RJT (Fabric Reject) P_RJT (N_Port Reject)

11.2 Link Control Frames

11.2.1 Introduction

Link_Control frames (ACK and Link_Response frames) shall be used by the Nx_Port to control Class 1, 2, 4 and 6 frame transfers. Busy responses are not valid for Class 4.

ACK and Link_Response frames indicate successful or unsuccessful frame delivery of a valid frame to the FC-2 level in Nx_Ports. The ACK and Link_Response frames also participate in end-to-end flow control. ACK frames shall indicate successful delivery to the destination Nx_Port, while Link_Response frames shall indicate unsuccessful delivery to the Fabric and Nx_Port.

If preemption is not enabled, the Fabric shall reply to a Class 1, 2 or 6 preemption request frame, with an F_RJT

frame to the requesting Nx_Port with a Preemption not enabled" reason code. If preemption is rejected, the Fabric shall reply to a Class 1, 2 or 6 preemption request frame, with an F_RJT frame to the requesting Nx_Port with a Preemption request rejected" reason code.

The Fabric, in reply to a Class 1, 2 or 6 preemption request frame, shall send an F_BSY frame to the requesting Nx_Port, if the Fabric is unable to complete the preemption.

Link_Control frames are identified by the ROUTING field being set to '1100' b and the INFORMATION field as shown in table 45.

Table 45 - Link_Control Information Categories

ROUTING	INFORMATION	Description	Abbr.
'1100' b	'0000' b	Acknowledge_1	ACK_1
	'0001' b	Acknowledge_0	ACK_0
	'0010' b	Nx_Port Reject	P_RJT
	'0011' b	Fabric Reject	F_RJT
	'0100' b	Nx_Port Busy	P_BSY
	'0101' b	Fabric Busy to Data frame	F_BSY
	'0110' b	Fabric Busy to Link_Control frame	F_BSY
	'0111' b	Link Credit Reset	LCR
	'1000' b	Notify - obsolete	NTY
	'1001' b	End	END
	others	reserved	

The Parameter field is reserved except for ACK_1 (see 11.2.2.3.2) and ACK_0 (see 11.2.2.3.3).

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.

An Nx_Port shall provide sufficient resources to receive Link_Control frames in response to Data frames it originated. An Nx_Port shall not transmit P_BSY 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 (see 11.2.4.2) may always be retransmitted in response to an F_BSY. For ACK 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 46 indicates allowable delimiters for Link_Control frames by class.

Table 46 - Link_Control frame delimiters

ACK, BSY, RJT	Delimiters
Class 1	SOF _{n1} , EOF _n , EOF _t , EOF _{dt}
Class 2	SOF _{n2} , EOF _n , EOF _t
Class 4	SOF _{c4} , SOF _{n4} , EOF _n , EOF _t , EOF _{dt} , EOF _{rt}
Class 6	SOF _{n1} , EOF _n , EOF _t , EOF _{dt}
LCR	Delimiters
Class 2	SOF _{n2} , EOF _n
Class 4	SOF _{n4} , EOF _n , EOF _t , EOF _{dt} , EOF _{drt} , EOF _{rti}

11.2.2 Link_Continue function

11.2.2.1 Introduction

The Link_Continue function provides a positive feedback mechanism to control the flow of Data frames on the link. A Data frame shall only be transmitted when the applicable FC_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 FC_Ports. The R_RDY Primitive is transmitted when the corresponding interface buffer is available for receipt of any Class 1 or 6/SOF_{c1}, Class 2, Class 3 frame.
- ACK_0 - successful or unsuccessful delivery of a Sequence (see 11.2.2.3) between Initiator and Recipient Nx_Ports, with or without a Fabric present. ACK_0 is only applicable to Class 1, Class 2, Class 4 and Class 6 frames.
- ACK_1 - end-to-end flow control for a single Data frame transfer between Initiator and Recipient Nx_Ports with or without a Fabric present. The ACK_1 frame is transmitted on receipt of a Class 1, Class 2, Class 4 and Class 6 frame. An FC_Port should transmit R_RDY and Link_Control frames before Data frames in order to avoid buffer-to-buffer and end-to-end Credit problems.

11.2.2.2 Receiver Ready (R_RDY)

The R_RDY Primitive shall indicate that a single frame (valid or invalid) was received and that the interface buffer, that received the frame, is available for further frame reception. An FC_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_RDYs shall be transmitted between FC_Ports with or without a Fabric present for all Class 1 and 6/SOF_{c1}, Class 2 and Class 3 frames. The R_RDY Primitive is not forwarded to any higher levels within an FC_Port or passed beyond the entry or exit point of the Fabric.

See 8.2 for specification of the number of Idles before and after the R_RDY Primitive during transmission.

There is no response to an R_RDY Primitive.

11.2.2.3 Acknowledge (ACK)

11.2.2.3.1 General

ACK_0 or ACK_1 may be used for acknowledgment of Data frames between Initiator and Recipient Nx_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.

The ACK frame shall indicate that one or more valid Data frames were received by the destination Nx_Port for the corresponding Sequence_Qualifier and SEQ_CNT of a valid Exchange as specified in the Sequence_Qualifier, and that the interface buffers that received the frame or frames are available for further frame reception. ACK frames shall be used in Class 1, 2, 4 and 6 and transmitted in the same class as the Data frame or frames that are being acknowledged.

NOTE 1 In Class 1 and 6, it is recommended that Nx_Ports transmit ACK_1 in the same order that the corresponding Data frames are received.

When multiple ACK forms are supported by both the Sequence Initiator N_Port Login parameters and the 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 table 47.

Table 47 - ACK precedence

Sequence Recipient word 1, bit 31 (ACK_0 Capable)	Sequence Initiator word 0, bit 10 (ACK_0 Capable)	ACK form required	Sequence Recipient word 1, bit 31 (ACK_0 Capable)
0	0	ACK_1	0
0	1	ACK_1	0
1	0	ACK_1	1
1	1	ACK_0	1

For all forms of ACK, when the History bit (bit 16) of the Parameter Field is set to zero, it shall indicate that the Sequence Recipient 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 to one, it shall indicate that at least one previous ACK has not been transmitted (e.g., Data frame not processed, or Data frame not received) by the Sequence Recipient. Using this historical information allows an Nx_Port to reclaim end-to-end Credit for a missing ACK frame.

Being able to reclaim end-to-end Credit does not relieve the Nx_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 Nx_Port as a Sequence Initiator and Sequence Recipient (see 15.6.5.5.3 and 15.6.5.6.2).

NOTE 2 Throughout this standard, 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 to 0) are used in a consistent manner.

The ACK frame provides end-to-end flow control for one or more Data frames between Initiator and Recipient Nx_Ports as defined in ACK_0 or ACK_1. See 18.4.3.3 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.

11.2.2.3.2 ACK_1

All Nx_Ports, as the default, prior to Login shall support ACK_1. The SEQ_CNT of the ACK_1 shall match the single Data frame being acknowledged. If an Nx_Port only supports ACK_0, it shall Logout any Nx_Port that attempts to Login that does not support ACK_0. The Parameter Field contains a value of hex '0001' in ACK_CNT (bits 15 to 0) to indicate that a single Data frame is being acknowledged. The INFORMATION field (Word 0, bits 27 to 24) shall be set to '0000' b.

11.2.2.3.3 ACK_0

ACK_0 is the designation used when the ACK_CNT (bits 15 to 0) of the Parameter Field of the ACK_0 frame contains a value hex '0000' to indicate that all Data frames of a Sequence are being acknowledged. The SEQ_CNT of the ACK_0 shall match the SEQ_CNT of the last Data frame received within the Sequence. The INFORMATION field (Word 0, bits 27 to 24) shall be set to '0001' b.

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 Nx_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:

- a) perform X_ID interlock or
- b) 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.

Although infinite buffers is indicated at the FC-FS level within an Nx_Port, individual FC-4s (e.g., FCP-3) may agree on a maximum Sequence size that is specified at the upper level through Mode Select in SCSI. In each protocol, a burst size is specified that indicates the largest single Sequence that shall be transferred. By further controlling the maximum number of concurrent Sequences, each Nx_Port may limit the amount of buffering that is actually required.

11.2.2.3.4 Header definition for all ACK forms

11.2.2.3.4.1 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).

11.2.2.3.4.2 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 36.

11.2.2.3.4.3 SEQ_ID

Equal to the SEQ_ID of the frame being replied to by ACK.

11.2.2.3.4.4 SEQ_CNT

Shall be equal to the SEQ_CNT of the highest Data frame being replied to by the ACK.

11.2.2.3.4.5 Parameter field

The Parameter Field is defined as follows:

- a) History Bit (bit 16)
0 = all previous ACKs transmitted
1 = at least one previous ACK not transmitted
- b) ACK_CNT (bits 15 to 0)
N = 0 All Data frames (ACK_0)
N = 1 Data frames (ACK_1)
N > 1 Reserved

11.2.2.3.5 Responses

The responses to ACK are F_RJT, P_RJT or F_BSY (except Class 4, when Class 4 circuit in Live State).

11.2.3 Link_Response

11.2.3.1 Introduction

Link_Response frames shall be sent for Class 1, 2, 4 and 6. The F_BSY and P_BSY responses are not applicable to a Class 4 circuit. An FC_Port shall only send Link_Response frames in reply to valid frames (see 8.9.2).

A Link_Response frame indicates that the frame identified by the Sequence_Qualifier and SEQ_CNT was not delivered to or processed by the destination FC_Port. When an FC_Port generates a Link_Response frame, it is routed to the FC_Port indicated by the D_ID in the frame. Link_Response frames may be:

- a) Busy - indicates a busy condition was encountered by the FC_Port.
- b) Reject - indicates that delivery of the frame is being denied.

11.2.3.2 Fabric Busy (F_BSY)

11.2.3.2.1 Description

The F_BSY frame shall indicate that the FC_Port generating the F_BSY is temporarily occupied with other link activity and is unable to deliver the frame. A reason code is identified in the TYPE field (word 2, bits 31 to 28). 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 45. When word 0, bits 27 to 24 has a value of '0101' b, the F_BSY is in response to a Data frame. When word 0, bits 27 to 24 has a value of '0110' b, 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). If the Fabric is unable to deliver the F_BSY frame, it shall be discarded.

When an Nx_Port receives an F_BSY frame in response to a Data frame, the Nx_Port shall retransmit the busied frame if it has not exhausted its ability to retry. Therefore, an Nx_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 an Nx_Port has exhausted its ability to retry Data frames in response to an F_BSY, it shall notify the FC-4 or an upper level. The Nx_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 in response to Link_Control frames. In Class 2, if an Nx_Port receives an F_BSY in response to an ACK frame, it shall discard the F_BSY frame.

If a Fabric determines it needs to send an F_BSY in response to a frame, it shall set fields in the header as follows:

- Copy the S_ID and D_ID fields from the busied frame into the D_ID and S_ID fields, respectively (i.e., interchange them). Thus, 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.
- Invert the Exchange and Sequence Context bits in the F_CTL field. Other F_CTL bits may also be set in accordance with table 36.
- 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 SEQ_ID, SEQ_CNT and Parameter fields shall be copied unchanged from the frame being busied.
- The Data Field (if any) shall be discarded
- Select the most appropriate reason code (see table 48) and place it in the TYPE field (Word 2, bits 31 to 28).
- If the frame being busied is a Link_Control frame, the Link_Control command code (see table 45) of the busied frame in the INFORMATION field (Word 0, bits 27 to 24) shall be copied to the TYPE field (Word 2, bits 27 to 24) of the F_BSY frame.

The Fabric shall use EOF_{dt} for Class 1 and 6 F_BSY frames (connect-terminate) and EOF_n for Class 2 F_BSY frames.

The fabric shall not F_BSY any Class 4 frames.

Table 48 - F_BSY Reason Codes

Encoded Value Word 2, bits 31 to 28	Name	Description
'0001' b	Fabric busy	The Fabric is unable to deliver the frame to the destination Nx_Port due to conditions internal to the Fabric.
'0011' b	N_Port busy	The destination Nx_Port is currently involved in a Class 1 dedicated connection and the Fabric is unable to deliver the frame.
Others	Reserved	

11.2.3.2.2 Responses

The following are the possible responses:

- R_RDY Primitive (SOF_{x2})
- Link_Response: none

11.2.3.3 N_Port Busy (P_BSY)

11.2.3.3.1 Description

The P_BSY shall indicate that the destination Nx_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). If the Nx_Port is unable to accept the P_BSY frame, it shall be discarded.

When an Nx_Port receives P_BSY in response to a frame transmission, the Nx_Port shall retransmit the busied frame if it has not exhausted its ability to retry. Therefore, an Nx_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 an Nx_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 Nx_Port may perform the Abort Sequence protocol based on the Exchange Error Policy.

P_BSY indicates that the Busy was issued by the destination Nx_Port. P_BSY shall not be issued in response to a Link_Control frame. An Nx_Port shall process a Link_Control frame for each unacknowledged Data frame transmitted.

If an Nx_Port determines it needs to send a P_BSY in response to a frame, it shall set fields in the header as follows:

- Copy the S_ID and D_ID fields from the busied frame into the D_ID and S_ID fields, respectively (i.e., interchange them). Thus, 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.
- Invert the Exchange and Sequence Context bits in the F_CTL field. Other F_CTL bits may also be set in accordance with table 36.
- The SEQ_ID and SEQ_CNT fields shall be copied unchanged from the frame being busied.
- The four bytes of the Parameter field shall indicate the action and reason code for the P_BSY response as defined in Table 49. Tables 50 and 51 specify the P_BSY action and reason codes, respectively.
- The Data Field (if any) shall be discarded

The N_Port shall not P_BSY any Class 4 frames.

Table 49 - P_BSY code format

Parameter field	
Bits	Value
31 to 24	Action Code (see table 50)
23 to 16	Reason Code (see table 51)
15 to 8	Reserved
7 to 0	Vendor Unique Code

Table 50 - P_BSY action codes

Encoded Value Word 5, bits 31 to 24	Description
'00000001' b	Action 1: indicates 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 (SOFc1) or in response to an interlocked Data frame associated with X_ID assignment (SOFi2). The frame and Sequence are retryable at a later time.
'00000010' b	Action 2: 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.
Others	Reserved

Table 51 - P_BSY Reason Codes

Encoded Value Word 5, bits 23 to 16	Definition	Description
'00000001' b	Physical N_Port busy (P_BSY)	The destination Nx_Port LCF is currently busy and is unable to accept of the frame.
'00000011' b	N_Port Resource busy	The destination Nx_Port is unable to process the Data frame at the present time.
'00000111' b	Partial Multicast Busy	See clause 23
'11111111' b	Vendor specific Busy (See Bits 7 to 0)	May be used to specify vendor unique reason codes.
Others	Reserved	

11.2.3.3.2 Responses

The responses to ACK are F_RJT, P_RJT or F_BSY (except Class 4, when Class 4 circuit in Live State).

11.2.3.4 Reject (P_RJT, F_RJT)

11.2.3.4.1 Introduction

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 may be transmitted for a variety of conditions. For certain conditions retry is possible, whereas for other conditions it is not and intervention beyond the scope of this standard may be required.

In Class 1, 2, 4 and 6, if an FC_Port detects an error in a Data frame, it shall transmit a Reject frame with one of the reason codes specified in table 54. If an error is detected in 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
- FC_Port not available-temporary
- FC_Port not available-permanent
- Login required (Fabric)
- Class 6 non-uniform Link_Control frame

An Nx_Port shall only reject a Link_Control frame because of an unexpected ACK with Action code 2 as specified in table 53.

If an Nx_Port detects an error in a Link_Control frame for a valid Exchange for a reason not listed above, 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 FC_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.

If an Nx_Port determines it needs to send a Reject (either F_RJT or P_RJT) in response to a frame, it shall set fields in the header as follows:

- a) Copy the S_ID and D_ID fields from the rejected frame into the D_ID and S_ID fields, respectively (i.e., interchange them). Thus, the D_ID field designates the source of the frame encountering the reject condition while the S_ID field designates the destination of the frame encountering the reject condition.
- b) Invert the Exchange and Sequence Context bits in the F_CTL field. Other F_CTL bits may also be set in accordance with table 36.
- c) The SEQ_ID and SEQ_CNT shall be copied unchanged from the frame being rejected.
- d) The four bytes of the Parameter field shall indicate the action and reason for the Reject response as defined in table 52. Table 53 and Table 54 specify the Reject Action codes and Reject Reason Codes respectively.
- e) The Data Field (if any) shall be discarded.

11.2.3.4.2 Class 4

Class 4 shall use the F_RJT and P_RJT.

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 11.2.3.4).

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 rejected.

The Fabric shall not reject any SOF_n4 or SOF_i4 delimited frames.

The VC_ID field shall be set to zero in an F_RJT frame with the reason code "Invalid VC_ID".

FC_Ports shall not reject an END Link_Control command. FC_Ports shall discard END Link_Control commands with invalid D_ID || S_ID || VC_ID frame header content.

11.2.3.4.3 Parameter field

11.2.3.4.3.1 Reject Code format

The four bytes of this field shall indicate the action code and reason for rejecting the request (see tables 52, 53 and 54).

The first error detected shall be the error reported; the order of checking is not specified.

Table 52 - Reject Code format

Parameter field	
Bits	Value
31 to 24	Action Code (see table 53)
23 to 16	Reason Code (table 54)
15 to 8	Reserved
7 to 0	Vendor Unique Code

Table 53 - Reject Action Codes

Encoded Value Word 5, bits 31 to 24	Description	Action
'00000001' b	Retryable error	Action 1: indicates that if the condition indicated in the reject Reason code is changed or corrected, the sequence may be retryable. Applicability: by Fabric when D_ID = Fabric by Fabric when D_ID = Nx_Port by Nx_Port when D_ID = Nx_Port
'00000010' b	Non-retryable error	Action 2: indicates that the Sequence is non-retryable and further recovery (e.g., Abort Exchange) may be required Applicability: by Fabric when D_ID = Fabric by Nx_Port when D_ID = Nx_Port
Other codes	Reserved	

Table 54 - Reject Reason Codes

Encoded Value Word 5, bits 23 to 16	Description	By	Action Code
'00000001' b	Invalid D_ID	B	R
'00000010' b	Invalid S_ID	B	R
'00000011' b	Nx_Port not available, temporary	F	R
'00000100' b	Nx_Port not available, permanent	F	R
'00000101' b	Class not supported	B	R
'00000110' b	Delimiter usage error	B	N
'00000111' b	TYPE not supported	B	N
'00001000' b	Invalid Link_Control	P	N
'00001001' b	Invalid R_CTL field	P	N
'00001010' b	Invalid F_CTL field	P	N
'00001011' b	Invalid OX_ID	P	N
Legend: F = F_RJT (Fx_Port) P = P_RJT (Nx_Port) B = Both F_RJT and P_RJT R = Retryable N = Non-retryable			

Table 54 - Reject Reason Codes (Continued)

Encoded Value Word 5, bits 23 to 16	Description	By	Action Code
'00001100' b	Invalid RX_ID	P	N
'00001101' b	Invalid SEQ_ID	P	N
'00001110' b	Invalid DF_CTL	F	N
'00001111' b	Invalid SEQ_CNT	P	N
'00010000' b	Invalid Parameter field	P	N
'00010001' b	Exchange error	P	N
'00010010' b	Protocol error	P	N
'00010011' b	Incorrect length	B	N
'00010100' b	Unexpected ACK	P	N
'00010101' b	Class of service not supported by entity at hex 'FF FF FE'	F	R
'00010110' b	Login Required	B	R
'00010111' b	Excessive Sequences attempted	P	R
'00011000' b	Unable to Establish Exchange	P	R
'00011001' b	Reserved	P	N/A
'00011010' b	Fabric path not available	F	R
'00011011' b	Invalid VC_ID (Class 4)	F	N
'00011100' b	Invalid CS_CTL field	B	N
'00011101' b	Insufficient resources for VC (Class 4)	B	N
'00011111' b	Invalid class of service	B	N
'00100000' b	Preemption request rejected	F	N
'00100001' b	Preemption not enabled	F	N
'00100010' b	Multicast error	F	N
'00100011' b	Multicast error terminate	F	N
'00100100' b	Process login required	P	R
Legend: F = F_RJT (Fx_Port) P = P_RJT (Nx_Port) B = Both F_RJT and P_RJT R = Retryable N = Non-retryable			

Table 54 - Reject Reason Codes (Continued)

Encoded Value Word 5, bits 23 to 16	Description	By	Action Code
'11111111' b	Vendor specific reject (See bits 7 to 0)	P	R
Others	Reserved	P	N/A
Legend: F = F_RJT (Fx_Port) P = P_RJT (Nx_Port) B = Both F_RJT and P_RJT R = Retryable N = Non-retryable			

If a frame within a Sequence is rejected, the Sequence shall be abnormally terminated or aborted. If an EOF_t or EOF_d ends the RJT frame, the FC_Port transmitting the RJT frame has terminated the Sequence. An FC_Port shall only terminate the Sequence using a Reject with an EOF_d in response to a connect-request (SOF_{c1}). In Class 2 an FC_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 FC_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.

11.2.3.4.3.2 Invalid D_ID

F_RJT - the Fabric is unable to locate the destination Nx_Port address.

P_RJT - the Nx_Port that received this frame does not recognize the D_ID as its own Identifier.

11.2.3.4.3.3 Invalid S_ID

F_RJT - the S_ID does not match the N_Port_ID assigned by the Fabric.

P_RJT - the destination Nx_Port does not recognize the S_ID as valid.

11.2.3.4.3.4 Nx_Port not available, temporary

F_RJT - The Nx_Port specified by the D_ID is a valid destination address but the Nx_Port is not functionally available (e.g., the Nx_Port is online and may be performing a Link Recovery Protocol).

11.2.3.4.3.5 Nx_Port not available, permanent

F_RJT - The Nx_Port specified by the D_ID is a valid destination address but the Nx_Port is not functionally available. The Nx_Port is Offline or Powered Down.

11.2.3.4.3.6 Class not supported

F_RJT or P_RJT - The class specified by the SOF delimiter of the frame being rejected is not supported.

11.2.3.4.3.7 Delimiter usage error

F_RJT or P_RJT - The SOF or EOF is not appropriate for the current conditions (e.g., a frame started by SOF_{c1} is received while a Class 1 dedicated connection already exists with the same Nx_Port). See tables 42 and 46 for allowable delimiters by class.

11.2.3.4.3.8 TYPE not supported

F_RJT or P_RJT - The TYPE field of the frame being rejected is not supported by the FC_Port replying with the Reject frame.

11.2.3.4.3.9 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.

11.2.3.4.3.10 Invalid R_CTL field

P_RJT - The R_CTL field is invalid or inconsistent with the other Frame_Header fields or conditions present.

11.2.3.4.3.11 Invalid F_CTL field

P_RJT - The F_CTL field is invalid or inconsistent with the other Frame_Header fields or conditions present.

11.2.3.4.3.12 Invalid OX_ID

P_RJT - The OX_ID specified is invalid or inconsistent with the other Frame_Header fields or conditions present.

11.2.3.4.3.13 Invalid RX_ID

P_RJT - The RX_ID specified is invalid or inconsistent with the other Frame_Header fields or conditions present.

11.2.3.4.3.14 Invalid SEQ_ID

P_RJT - The SEQ_ID specified is invalid or inconsistent with the other Frame_Header fields or conditions present.

11.2.3.4.3.15 Invalid DF_CTL

P_RJT - The DF_CTL field is invalid.

11.2.3.4.3.16 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 9.7 bits 5 to 4 (F_CTL Abort Sequence Condition)).

11.2.3.4.3.17 Invalid Parameter field

P_RJT - The Parameter field is incorrectly specified or invalid.

11.2.3.4.3.18 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.

11.2.3.4.3.19 Protocol Error

P_RJT - This indicates that an error has been detected that violates the rules of FC-2 signalling protocol that are not specified by other error codes.

11.2.3.4.3.20 Incorrect length

F_RJT or P_RJT - The frame being rejected is an incorrect length for the conditions present.

11.2.3.4.3.21 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 Nx_Port (i.e., the EOF delimiter shall be EOF_n).

11.2.3.4.3.22 Class of service not supported by entity at hex 'FF FF FE'

F_RJT - The class specified by the SOF delimiter of the frame being rejected is not supported by the Fx_Port (hex 'FF FF FE')

11.2.3.4.3.23 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 an Nx_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 Nx_Port.

11.2.3.4.3.24 Excessive Sequences attempted

P_RJT - A new Sequence was initiated by an Nx_Port that exceeded the capability of the Sequence Recipient as specified in the Service Parameters during Login.

11.2.3.4.3.25 Unable to Establish Exchange

P_RJT - A new Exchange was initiated by an Nx_Port that exceeded the capability of the Responder facilities.

11.2.3.4.3.26 Fabric path not available

F_RJT - The speed of the source and destination N_Ports do not match. Other Fabric characteristics related to multiple Fabric domains may also use this reason code.

11.2.3.4.3.27 Invalid VC_ID (Class 4)

F_RJT - The VC_ID field is invalid.

11.2.3.4.3.28 Invalid CS_CTL Field

F_RJT or P_RJT - The CS_CTL field is invalid.

11.2.3.4.3.29 Insufficient resources for VC (Class 4)

F_RJT or P_RJT - The port sending the reject has insufficient resources for the VC.

11.2.3.4.3.30 Invalid class of service

F_RJT or P_RJT - The class of service indicated by the SOF is invalid for the conditions present

11.2.3.4.3.31 Preemption request rejected

F_RJT - The Preemption request is rejected (see 20.1.3)

11.2.3.4.3.32 Preemption not enabled

F_RJT - Preemption Enable bit was not set to one during login (see 15.6.5.4.5)

11.2.3.4.3.33 Multicast error

F_RJT - All multicast destination Nx_Ports responded with identical valid reject Link_Control Frames (see 23.3.4).

11.2.3.4.3.34 Multicast error terminate

F_RJT - One or more destination Nx_Ports responded with an ACK, BSY, or RJT frame with an EOF_{dt} delimiter (see 23.3.4).

11.2.3.4.3.35 Vendor Specific Reject

F_RJT or P_RJT - The Vendor specific Reject bits (bits 7 to 0) may be used to specify vendor specific reason codes.

11.2.3.4.3.36 Responses

F_RJT, P_RJT or F_BSY (except Class 4, when Class 4 circuit in Live State).

11.2.4 Link_Control commands

11.2.4.1 Introduction

Link_Control commands are Link_Control frames that initiate a low-level action at the destination Nx_Port. 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.

11.2.4.2 Link Credit Reset (LCR)

11.2.4.2.1 Description

The LCR frame shall indicate that the Nx_Port specified by the S_ID requests that the Nx_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. Both Nx_Ports abnormally terminate all active Sequences with the S_ID as Sequence Initiator and the D_ID as Sequence Recipient for all classes of service.

The Nx_Port specified by the S_ID shall perform Exchange and Sequence recovery at the discretion of the appropriate Upper Level Protocol. After transmitting the LCR frame, the Nx_Port that requested the Credit Reset shall wait R_A_TOV before initiating Sequences with the destination Nx_Port. The LCR 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 21.2.4.4 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 Nx_Port buffers are also reset, although a dedicated connection, if present, is unaffected. Therefore, an Nx_Port should remove a dedicated connection prior to transmitting LCR in Class 2 to the same destination Nx_Port. LCR shall be transmitted with SOF_{n2} and EOF_n.

If an Nx_Port is out of EE_Credit in Class 1 or 6 and times out all Sequences, it shall perform Connection Recovery (see 20.8). It shall not use LCR.

Class 4 behavior shall not be affected by LCR (see clause 25).

11.2.4.2.2 Protocol

- a) LCR
- b) No reply frame

11.2.4.2.3 Request Sequence

Format: FT_0

Addressing: The S_ID field designates the Nx_Port that is requesting a buffer reset by the destination Nx_Port or D_ID.

11.2.4.2.4 Responses

The possible responses are:

- a) R_RDY Primitive (SOF_{n2})
- b) F_RJT, P_RJT
- c) F_BSY (except Class 4)

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 an Nx_Port to avoid special casing LCR in Class 2. However, the Nx_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.

11.2.4.3 End (END)

11.2.4.3.1 Description

The END Link_Control command is used by the QOSF in Class 4 to deactivate or remove a Class 4 circuit initiated by the fabric.

The Class 4 circuit is removed if the EOF delimiter is EOF_{rt} or EOF_{rti}. The Class 4 circuit is deactivated if the EOF delimiter is EOF_{dt} or EOF_{dti}. The END Link_Control frame shall be discarded if the EOF delimiter is EOF_n, EOF_t, EOF_{ni}, EOF_a 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 Nx_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:

- a) Fabric Relogin is attempted,
- b) N_Port or the F_Port enters the Offline State,
- c) Link failure is detected at the N_Port,
- d) error recovery for the setup process (see 25.3.5.2), or
- e) 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 (e.g., by Link failure, Offline).

The END Link_Control frame shall have no Payload and shall be sent only in Class 1, 2 or 4.

11.2.4.3.2 Protocol

- a) END
- b) No reply frame

11.2.4.3.3 Request Sequence

Format: FT_0

Addressing: The D_ID field shall designate the Nx_Port receiving the END Link_Control frame. In Class 4, the S_ID and VC_ID field shall designate the N_Port on whose behalf the QoS is transmitting the END command.

11.2.4.3.4 Reply Sequence

None

11.3 ACK generation assistance

11.3.1 Introduction

If a Sequence Recipient supports multiple ACK forms, an indication about the required ACK form by the Sequence Initiator as indicated during Login may be of assistance to the Sequence Recipient in generating it. This shall be done in accordance with table 47. See 15.6.5.5.3 and 15.6.5.6.2 for definition of the login bits referenced in table 47.

11.3.2 N_Port Login

11.3.2.1 Capability Indicator

The ACK generation assistance capability is indicated during N_Port Login in word 0, bit 9 of Nx_Port Class Service Parameters.

The Initiator Control Flags are specified in 15.6.5.5.

11.3.3 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, 4 or 6 Data frames of a Sequence and on all connect-request frames. ACK_Form is not meaningful on Class 1, 2, 4 or 6 Link_Control frames or any Class 3 frames.

11.3.4 F_CTL bits

F_CTL Bits 13 to 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 31) its ACK capability determined during N_Port Login.

11.3.5 Login rules

Only ACK_1 shall be used during or before the establishment of Login parameters. Additional rules are specified for ACK_Form field usage during these conditions:

- a) In Classes 1 and 2, ACK_1 shall be used to acknowledge PLOGI and FLOGI and the corresponding LS_ACC. This rule shall be followed whether FLOGI or PLOGI and the corresponding LS_ACC flow in the same Class 1 connection or two separate connections.

- b) If Class 1 is used for FLOGI or PLOGI and the corresponding LS_ACC, ACK_1 shall be used including the SOF_{c1} frame. This rule shall be followed whether FLOGI or PLOGI and the corresponding LS_ACC flow in the same Class 1 connection or two separate connections.
- c) If ACK generation assistance is not provided, the ACK_Form field shall be set to '00' b on the FLOGI or PLOGI frame and the corresponding LC_ACC frame.
- d) If ACK generation assistance is provided, the ACK_Form field shall be set to '01' b on the FLOGI or PLOGI frame and the corresponding LC_ACC frame.
- e) Once established, the ability or inability to provide ACK generation assistance shall not change until logout or Relogin occurs.

11.3.6 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".

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

12 Link Services

12.1 Sequence and Exchange management

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 11.

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 (e.g., 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 ELSs are performed within a single Exchange, intended exclusively for the purpose of ELS processing. Most of ELS protocols are performed as a two Sequence Exchange. Each of these two Sequence Exchanges consists of a Request Sequence by the Originator, transfer of Sequence Initiative, and a reply Sequence from the Responder that 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 17. The following rules regarding Sequence and Exchange management apply to ELSs in addition to the rules specified in clause 17.

- a) Reply frames and Sequences shall be transmitted in the same class as the request.
- b) Request and Reply Sequences shall be sent in Class 1, Class 2, Class 3 and Class 4.

NOTE Not all of the above classes make sense for all ELSs, but there is no restriction. Class 6 makes no sense for most ELSs, but there is no restriction.

- c) If Login has not been completed successfully, the default Login values shall be used. Since Concurrent Sequences is limited to one, originating a second Exchange to perform an ELS (e.g., Read Exchange Status) is not possible. Therefore, ABTS is the preferred recovery action.
- d) If Login has completed successfully, the Originator of the Exchange shall use the Discard multiple Sequences Error Policy for all ELS Exchanges.
- e) If the Exchange Originator of an ELS Exchange detects an Exchange error, it shall abort the Exchange using ABTX and retry the protocol of the aborted Exchange with a different Exchange.
- f) If the Sequence Initiator aborts a Sequence using ABTS due to receiving an ACK with the Abort Sequence bits (5 to 4) set to '01' b, the Sequence Initiator shall retry the Sequence after the BA_ACC is received for the aborted Sequence one time only. If the retry fails, the ELS Exchange shall be aborted using ABTX.
- g) If the Sequence Initiator attempts to abort a Sequence using ABTS and it detects an E_D_TOV waiting for the ACK frame in response to the ABTS, it shall abort the Exchange using ABTX, if conditions permit, and retry with a different Exchange.

12.2 Basic Link Service commands

12.2.1 Introduction

Nx_Ports shall support all Basic Link Service commands.

Since DF_CTL bits are not meaningful on Basic Link_Data frames, Basic Link Service frames shall not contain optional headers.

The R_CTL field shall be set as defined in table 55 to indicate Basic Link Service commands.

The TYPE field (Word 1 bits 31 to 24) shall be set to zero.

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.

The timeout for a Basic Link Service shall be 2 times R_A_TOV.

Table 55 - Basic Link Service Information Categories

R_CTL		Description	Abbreviation
ROUTING	INFORMATION		
'1000' b	'0000' b	No Operation	NOP (see 12.2.5)
	'0001' b	Abort Sequence	ABTS (see 12.2.2)
	'0010' b	Remove Connection	RMC (see 12.2.6)
	'0100' b	Basic_Accept	BA_ACC (see 12.2.3)
	'0101' b	Basic_Reject	BA_RJT (see 12.2.4)
	'0110' b	Dedicated Connection Preempted	PRMT (see 12.2.7)
	Others	Reserved	

12.2.2 Abort Sequence (ABTS)

12.2.2.1 Overview

The ABTS frame shall be used by the Sequence Initiator

- a) to request that the Sequence Recipient abort one or more Sequences (see 12.2.2.2 and 21.7.2.2).
- b) or Sequence Recipient to request that the ABTS Recipient abort the entire Exchange (see 12.2.2.3).

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 to 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 one. If the parameter field bit 0 is set to zero, 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 '01' b on an ACK frame (see 9.7.14). The ABTS frame may be transmitted without regard to which Nx_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. = Bit 0 of the parameter field shall be as specified in table 56.

Table 56 - ABTS Parameter Field Definitions

Parameter Field, bit 0 Value	Meaning
0	Abort Exchange
1	Abort Sequence

12.2.2.2 Aborting Sequences using ABTS

12.2.2.2.1 Introduction

When aborting sequences using ABTS:

- a) none, one or multiple Sequences are aborted;
- b) ABTS is transmitted by the Sequence Initiator of the last Sequence; and
- c) 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. 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, (e.g., 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.

12.2.2.2.2 ABTS Initiator

Since ABTS is used for error recovery, the following relaxed behaviors are allowed. An ABTS Initiator may transmit ABTS, even if

- a) there is no end-to-end Credit available,
- b) it does not hold the Sequence Initiative,
- c) there is no Sequence open,
- d) maximum number of Concurrent Sequences supported are in use, and
- e) it is a Connection Recipient in Unidirectional Class 1 Connection.

After transmitting the ABTS frame, an Nx_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 BA_ACC 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 BA_ACC.

NOTE The ABTS frame may 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.

12.2.2.2.3 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 21.6.3). 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 (see 4.12.4) may be processed after the ABTS frame is received based on the policy for the Exchange, but before the BA_ACC is transmitted.

Transmission of the BA_ACC to the ABTS frame is an atomic function in that any Data frames identified in the Recovery_Qualifier range (identified in the BA_ACC Payload) shall be discarded after the BA_ACC is transmitted to the Sequence Initiator. The BA_ACC 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 BA_ACC. The ABTS Sequence Recipient shall set F_CTL bit 16 to zero in the BA_ACC to indicate that it holds the Sequence Initiative for the Exchange or set it to one to indicate that the ABTS Sequence

Initiator holds the Sequence Initiative.

The format of the BA_ACC Payload is shown in table 57. 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 BA_ACC Payload, if the low SEQ_CNT equals high SEQ_CNT and the last valid SEQ_ID in the BA_ACC matches the last Sequence that was transmitted, then no Sequences have been aborted (i.e., 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.

12.2.2.2.4 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, 3 or 4 and a Sequence error has been indicated, a Recovery_Qualifier range shall be established for both Nx_Ports. If a Recovery_Qualifier exists, the Sequence Initiator of the ABTS frame shall discard ACK and Link_Response frames received that correspond to the Recovery_Qualifier between the low and high SEQ_CNT values. After transmission of the BA_ACC to the ABTS frame the Sequence Recipient of the ABTS frame shall discard Data frames received that 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 BA_ACC Payload, the Sequence Initiator of the ABTS shall issue a Reinstate Recovery Qualifier (RRQ) ELS Request Sequence after waiting an R_A_TOV timeout period after reception of the BA_ACC (see 12.3.2.14).

After the BA_ACC 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 21.7.2 and 21.2.4 for more discussion on abnormal termination of Sequences and Sequence timeout. See 21.2.7.2 for examples of the ABTS protocol that include several special cases (e.g., 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 BA_ACC to the Abort Sequence frame, the Sequence Initiator may perform Sequence recovery under guidance from the appropriate FC-4.

12.2.2.2.5 Protocol

- a) Abort Sequence Request frame
- b) BA_ACC or BA_RJT Reply frame

12.2.2.2.6 Request Sequence

Format: FT_1

Addressing: The D_ID field designates the Sequence Recipient Nx_Port. The S_ID field designates the source Sequence Initiator Nx_Port that 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 and SEQ_CNT: The SEQ_ID shall be the same as the last Sequence transmitted for this Exchange by the Nx_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.

12.2.2.2.7 Reply Sequence

BA_RJT: BA_RJT signifies rejection of the ABTS command, however, the Sequence may have been aborted without Sequence information (see 12.2.4).

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.

BA_ACC: BA_ACC signifies that the destination Nx_Port has aborted and discarded no Sequences, one Sequence, or multiple Sequences.

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:

- a) same as SEQ_CNT of the ABTS frame
- b) equal to the SEQ_CNT of the last Data frame of the last deliverable Sequence
- c) set to hex '00 00'.

The Payload is specified for each of the permitted cases:

- a) 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:
 - A) SEQ_ID Validity equal valid (hex '80')
 - B) SEQ_ID equal the SEQ_ID of the Sequence in which the ABTS has been received from the Sequence Initiator
 - C) Low SEQ_CNT equal High SEQ_CNT equal SEQ_CNT of the ABTS frame
- b) 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:
 - A) SEQ_ID Validity equal valid (hex '80')
 - B) SEQ_ID equal 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,
 - C) Low SEQ_CNT equal the SEQ_CNT of the last Data frame of the last deliverable Sequence
 - D) High SEQ_CNT equal the SEQ_CNT of the ABTS frame
- c) 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:
 - A) SEQ_ID Validity equal invalid (hex '00')
 - B) SEQ_ID equal invalid in this case
 - C) Low SEQ_CNT equal hex '00 00'
 - D) High SEQ_CNT equal the SEQ_CNT of the ABTS frame.

12.2.2.3 Aborting Exchanges using ABTS

12.2.2.3.1 Introduction

Using ABTS to abort an Exchange is specified in this section. In this method,

- a) an entire Exchange is aborted;
- b) ABTS is transmitted by the Sequence Initiator or the Sequence Recipient of the last Sequence; and

- c) ABTS is transmitted as part of the open Sequence or in a new Sequence.

12.2.2.3.2 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.

12.2.2.3.3 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.

12.2.2.3.4 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. 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 (e.g., 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.

12.2.2.3.5 ABTS by the last Sequence Recipient

If the last Sequence Recipient transmits an ABTS frame, it shall transmit ABTS in a new Sequence with a SEQ_ID available for use from its Nx_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

- a) the Last_Sequence bit to one in BA_ACC or
- b) 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.

12.2.2.3.6 Request Sequence

Format: FT_1

Addressing: The D_ID field designates the ABTS Recipient Nx_Port. The S_ID field designates the ABTS Initiator Nx_Port that 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 and 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 Nx_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 Nx_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.

12.2.2.3.7 Reply Sequence

BA_RJT: BA_RJT signifies rejection of the ABTS command, however, the Exchange may have been aborted without Sequence information (see 12.2.4).

BA_ACC: BA_ACC signifies that the destination Nx_Port has aborted and discarded no Sequences, one Sequence, multiple Sequences, or the entire Exchange.

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'.

The BA_ACC Payload is shown in table 57. The Payload is specified for each of the permitted cases:

- a) 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:
 - A) SEQ_ID Validity = valid (hex '80')
 - B) SEQ_ID = the SEQ_ID of the last deliverable Sequence received from the ABTS Initiator, and
 - C) Low SEQ_CNT = High SEQ_CNT = SEQ_CNT of ABTS frame

Table 57 - BA_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	SEQ_ID Validity hex '80' = valid hex '00' = invalid	SEQ_ID of last Sequence deliverable to ULP (if valid indicated)	Reserved	
1	OX_ID		RX_ID	
2	Low SEQ_CNT		High SEQ_CNT	

- b) 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 one and shall set, in the BA_ACC Payload:
 - A) SEQ_ID Validity = invalid (hex '00'),
 - B) SEQ_ID = invalid in this case
 - C) Low SEQ_CNT = hex '00 00' and
 - D) High SEQ_CNT = hex 'FF FF'

- c) To indicate that it has information on the last deliverable Sequence, but it is aborting the entire Exchange due to uncertainty (e.g., 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:
- A) SEQ_ID Validity = valid (hex '80')
 - B) SEQ_ID = the SEQ_ID of the last deliverable Sequence received from the ABTS Initiator,
 - C) Low SEQ_CNT = hex '00 00' and
 - D) High SEQ_CNT = hex 'FF FF'

12.2.3 Basic Accept (BA_ACC)

12.2.3.1 Description

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 BA_ACC Link Service Reply Sequence shall transfer the Sequence Initiative by setting the Sequence Initiative bit (Bit 16) to one 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 zero to indicate that the transmitter of the BA_ACC 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.

12.2.3.2 Protocol

BA_ACC is the Reply Sequence to Abort Sequence Basic Link Service command.

12.2.3.3 Request Sequence

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).

12.2.3.4 Reply Sequence

none

12.2.4 Basic Reject (BA_RJT)

12.2.4.1 Description

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 that 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 Link Service Request frame was transmitted. The SEQ_ID shall be assigned following the normal rules for SEQ_ID assignment.

The first error condition detected shall be the error reported.

12.2.4.2 Protocol

BA_RJT may be a Reply Sequence to ABTS.

12.2.4.3 Request Sequence

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 tables 58, 59 and 60).

12.2.4.4 Reply Sequence

none

Table 58 - BA_RJT Payload Format

Bits	Value
31 to 24	Reserved
23 to 16	Reason Code (see table 59)
15 to 8	Reason Explanation (see table 60)
7 to 0	Vendor Unique Code

Table 59 - BA_RJT reason codes

Encoded Value (Bits 23 to 16)	Name	Description
'00000001' b	Invalid command code	The Command code in the Sequence being rejected is invalid.
'00000011' b	Logical error	The request identified by the Command code is invalid or logically inconsistent for the conditions present.
'00000101' b	Logical busy	The Basic Link Service is logically busy and unable to process the request at this time.
'00000111' b	Protocol error	This indicates that an error has been detected that violates the rules of FC-2 protocol that are not specified by other error codes.
'00001001' b	Unable to perform command request	The Recipient of a Link Service command is unable to perform the request at this time.
Others	Reserved	The Vendor specific error bits may be used to specify vendor unique reason codes.
'11111111' b	Vendor specific error (see bits 7 to 0)	

Table 60 - BA_RJT Reason Code Explanation

Encoded Value (Bits 15 to 8)	Description	Applicable commands
'00000000' b	No additional explanation	ABTS
'00000011' b	Invalid OX_ID-RX_ID combination	ABTS
'00000101' b	Sequence aborted, no sequence information provided	ABTS
Others	Reserved	

12.2.5 No Operation (NOP)

12.2.5.1 Description

The NOP 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 Nx_Port and Fabric, if present. A NOP frame may be used to initiate a Class 1 or 6 Connection using SOF_{c1}, 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 20.4.3).

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.

12.2.5.2 Protocol

- a) No Operation Request
- b) No Reply frame

12.2.5.3 Request Sequence

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.

12.2.5.4 Reply Sequence

none

12.2.6 Remove Connection (RMC)

12.2.6.1 Description

The RMC Basic Link Service frame shall be used to request immediate removal of a Class 1 or 6 Connection. An ACK frame ended by EOF_{at} 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 or Class 6 Connections since all open Sequences shall be abnormally terminated. See 20.4.3 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.

12.2.6.2 Protocol:

- a) Remove Connection Request
- b) No Reply frame

12.2.6.3 Request Sequence

Format: FT_1

Addressing: The D_ID field designates the Nx_Port being requested to terminate the Class 1 or 6 Connection. The S_ID field designates the Nx_Port requesting removal.

Payload: The RMC Basic Link Service command has no Payload.

12.2.6.4 Reply Sequence

none

12.2.7 Dedicated connection preempted (PRMT)

12.2.7.1 Description

The PRMT basic link service command shall indicate that the connection for which this Nx_Port is participating has been preempted and no longer exists. All sequences associated with this connection have ended abnormally.

12.2.7.2 Protocol

- a) Connection Preempted Notification
- b) No Reply frame

12.2.7.3 Request Sequence

Format: FT_1

Addressing: The D_ID field shall designate the Nx_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 SOF_x preemption request frame that triggered this preemption notification.

Priority: Bit 17 of the E_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 SOF_x preemption request frame that triggered this preemption.

Payload: None

12.2.7.4 Reply Sequence

None

12.3 Extended Link Services

12.3.1 Introduction

An Extended Link Service (ELS) request solicits a destination FC_Port to perform a function or service. An ELS reply shall be transmitted in response to an ELS request, unless otherwise specified. 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 ELS frames, Sequences, and Exchanges. An Accept (LS_ACC) to an ELS shall terminate the Exchange by setting the Last Sequence bit (Bit 20) to one on the last frame of the reply. An ELS request and the corresponding reply shall be performed within a single Exchange.

The TYPE field for ELS frames shall be set to hex '01'.

The R_CTL shall be set as specified in table 61.

Table 61 - Extended Link Routing Bits and Information Categories

R_CTL		Description
ROUTING	INFORMATION	
'0010'b	'0001'b	Solicited Data ^a
	'0010'b	Request
	'0011'b	Reply
	Others	Reserved
^a This value is only used by the Clock Synchronization Update (CSU) ELS.		

The first byte of the Payload (LS_Command code) of the request or reply Sequence shall be as shown in table 62. 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).

12.3.2 Extended Link Service requests

12.3.2.1 Introduction

A Sequence Initiator shall transmit an ELS Sequence in order to solicit the destination Nx_Port to perform a link-level function or service. Unless otherwise noted, Extended Link Service requests shall not be issued prior to completion of N_Port Login. Table 62 applies to ELSs sent to or received by all valid addresses, including well known addresses. FLOGI to a fabric is required before all ELSs if a fabric is present.

Table 62 - ELS_Command codes

Value (Bits 31 to 24)	Description	Abbr.	Reference	PLOGI Required
hex '01'	Link Service Reject	LS_RJT	12.3.3.4	N/A
hex '02'	Link Service Accept	LS_ACC	12.3.3.2	N/A
hex '03'	N_Port Login	PLOGI	12.3.2.7	No
hex '04'	F_Port Login	FLOGI	12.3.2.7	No
hex '05'	Logout	LOGO	12.3.2.8	No
hex '06'	Abort Exchange	ABTX	12.3.2.2	Yes
hex '07'	Read Connection Status	RCS	12.3.2.9	Yes
hex '08'	Read Exchange Status Block	RES	12.3.2.10	Yes
hex '09'	Read Sequence Status Block	RSS	12.3.2.12	Yes
hex '0A'	Request Sequence Initiative	RSI	12.3.2.15	Yes
hex '0B'	Establish Streaming	ESTS	12.3.2.6	Yes
hex '0C'	Estimate Credit	ESTC	12.3.2.5	Yes
hex '0D'	Advise Credit	ADVC	12.3.2.3	Yes
hex '0E'	Read Timeout Value	RTV	12.3.2.13	Yes
hex '0F'	Read Link Error Status Block	RLS	12.3.2.11	Yes
hex '10'	Echo	ECHO	12.3.2.4	Yes
hex '11'	Test	TEST	12.3.2.16	Yes
hex '12'	Reinstate Recovery Qualifier	RRQ	12.3.2.14	Yes
hex '13'	Read Exchange Concise	REC	12.3.2.54	Yes
hex '14'	Reserved for legacy implementations ^a			
hex '20'	Process Login	PRLI	12.3.2.24	Yes
hex '21'	Process Logout	PRLO	12.3.2.25	Yes
hex '22'	State Change Notification	SCN	12.3.2.26	N/A
hex '23'	Test Process Login State	TPLS	12.3.2.27	Yes
hex '24'	Third Party Process Logout	TPRLO	12.3.2.43	Yes

^a Some implementations of early revisions of FCP-3 (see [9]) may have used the value hex '14' for SRR (Sequence Retransmission Request). This code is permanently reserved in this standard to avoid conflicts with such implementations. See FCP-2 for the standard implementation of SRR as an FC-4 Link Service

Table 62 - ELS_Command codes (Continued)

Value (Bits 31 to 24)	Description	Abbr.	Reference	PLOGI Required
hex '25'	Login Control List Management	LCLM	12.3.2.46	No
hex '30'	Get Alias_ID	GAID	12.3.2.33	No
hex '31'	Fabric Activate Alias_ID	FACT	12.3.2.34	No
hex '32'	Fabric Deactivate Alias_ID	FDACT	12.3.2.35	No
hex '33'	N_Port Activate Alias_ID	NACT	12.3.2.36	No
hex '34'	N_Port Deactivate Alias_ID	NDACT	12.3.2.37	No
hex '40'	Quality of Service Request	QoSR	12.3.2.38	Yes
hex '41'	Read Virtual Circuit Status	RVCS	12.3.2.39	Yes
hex '50'	Discover N_Port Service Parameters	PDISC	12.3.2.40	Yes
hex '51'	Discover F_Port Service Parameters	FDISC	12.3.2.40	Yes
hex '52'	Discover Address	ADISC	12.3.2.42	Yes
hex '53'	Report node Capability	RNC	12.3.2.17	N/A
hex '54'	Fibre Channel Address Resolution Protocol Request	FARP_REQ	12.3.2.28	No
hex '55'	Fibre Channel Address Resolution Protocol Reply	FARP_REPLY	12.3.2.29	Yes
hex '56'	Read Port Status Block	RPS	12.3.2.47	Yes
hex '57'	Read Port List	RPL	12.3.2.48	Yes
hex '58'	Report Port Buffer Condition	RPBC	12.3.2.49	Yes
hex '60'	Fabric Address Notification	FAN	12.3.2.18	No
hex '61'	Registered State Change Notification	RSCN	12.3.2.22	No
hex '62'	State Change Registration	SCR	12.3.2.23	No
hex '63'	Report node FC-4 Types	RNFT	12.3.2.50	Yes
hex '68'	Clock Synchronization Request	CSR	12.3.2.44	No
hex '69'	Clock Synchronization Update	CSU	12.3.2.45	No
hex '70'	Loop Initialize	LINIT	12.3.2.19	No
hex '71'	Loop Port Control (obsolete)	LPC	12.3.2.20	No
^a Some implementations of early revisions of FCP-3 (see [9]) may have used the value hex '14' for SRR (Sequence Retransmission Request). This code is permanently reserved in this standard to avoid conflicts with such implementations. See FCP-2 for the standard implementation of SRR as an FC-4 Link Service				

Table 62 - ELS_Command codes (Continued)

Value (Bits 31 to 24)	Description	Abbr.	Reference	PLOGI Required
hex '72'	Loop Status	LSTS	12.3.2.21	No
hex '77'	Vendor Specific			N/A
hex '78'	Request node Identification Data	RNID	12.3.2.30	Yes
hex '79'	Registered Link Incident Report	RLIR	12.3.2.31	Yes
hex '7A'	Link Incident Record Registration	LIRR	12.3.2.32	Yes
hex '7B'	Scan Remote Loop	SRL	12.3.2.51	Yes
hex '7C'	Set Bit-error Reporting Parameters	SBRP	12.3.2.52	Yes
hex '7D'	Report Port Speed Capabilities	RPSC	12.3.2.53	Yes
hex '7E'	Reserved for security			
Others	Reserved			
^a Some implementations of early revisions of FCP-3 (see [9]) may have used the value hex '14' for SRR (Sequence Retransmission Request). This code is permanently reserved in this standard to avoid conflicts with such implementations. See FCP-2 for the standard implementation of SRR as an FC-4 Link Service				

If an ELS 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 ELS Protocol is composed of an ELS request Sequence and, for some requests, an ELS Reply Sequence. The last Data frame of an ELS request Sequence with a Reply Sequence shall transfer the Sequence Initiative to the Recipient in order to allow the reply to be transmitted (see clause 5).

The following ELS requests and their replies shall be supported by an Nx_Port (all others are optional):

- a) FLOGI
- b) PLOGI
- c) LOGO

Table 63 - Responses to Received ELSs

Login Required? (see table 62)	Logged in with Source N_Port?	
	Yes	No
Yes	Respond as appropriate for the ELS and the current state of the Nx_Port	If a reply sequence is defined for the ELS, reply with a LOGO ELS Sequence or an LS_RJT ELS Sequence with a reason code of "unable to perform command request" and a reason code explanation of "N_Port Login required". If a reply sequence is not defined for the ELS, it shall be discarded
No	Respond as appropriate for the ELS and the current state of the Nx_Port.	Respond as appropriate for the ELS and the current state of the Nx_Port.

An Nx_Port is not required to generate and send the PLOGI ELS request. However, if an Nx_Port receives a PLOGI ELS request, the Nx_Port shall respond with a LS_ACC Link Service Reply, or if the PLOGI frame has invalid information, the Nx_Port shall respond with LS_RJT. LS_RJT shall not be issued for the reason "PLOGI command not supported".

NOTE If an Nx_Port that does not generate PLOGI, is in a point-to-point topology, and has an N_Port_Name greater than the other Nx_Port's, the other Nx_Port may timeout, waiting to receive PLOGI.

All ELS requests, excluding ESTS, ESTC, and ADVC, and the corresponding replies shall be performed within a single Exchange (see 18.4.11 for the procedure using ESTS, ESTC, and ADVC). The Advise Credit request may also be performed in a separate Exchange.

The Originator of the ELS request shall assign an OX_ID when the request Sequence and Exchange are originated. If the only Exchange that the Responder has open with the Originator Nx_Port is the one for the ELS 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 9 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.

ELS request and reply frames shall use FT_1. More than one frame may be used to form a request or reply Sequence.

The Originator of an ELS Exchange shall detect an Exchange error following Sequence Initiative transfer if the Reply Sequence is not received within a timeout interval of $2 \times R\_A\_TOV$.

There are two ELS request Sequences in which a collision is possible with the other Nx_Port involved in the same target Exchange. Those two requests are Abort Exchange (ABTX) and Request Sequence Initiative (RSI) (e.g., Nx_Port (A) may transmit an ABTX request to Nx_Port (B) at the same time that Nx_Port (B) transmits an ABTX request to Nx_Port (A) for the same target Exchange).

If such an instance occurs, the Originator Nx_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 Nx_Port of the target Exchange shall honor and process the ABTX or RSI request Sequence normally.

12.3.2.2 Abort Exchange (ABTX)

12.3.2.2.1 Description

The ABTX ELS 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 Originator Nx_Port or the Responder Nx_Port of the target Exchange makes the request.

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, and shall contain the S_ID of the Exchange Originator of the Exchange being aborted.

Transmission of an ABTX frame is allowed while the identified Exchange is open (see 17.3.14). Both the Originator and Responder shall ensure that the OX_ID and RX_ID pair being terminated is 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 21.7.2.2). 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 Nx_Ports for reuse after the Exchange has

been successfully aborted (i.e., LS_ACC transmitted and received). If all frames are not accounted for in Class 2, 3 or 4, a Recovery_Qualifier shall be established with a low SEQ_CNT of '0000' b and a high SEQ_CNT of hex 'FF FF'.

The Nx_Port that 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 LS_ACC 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 Nx_Port, then the Nx_Port that requested the ABTX shall transmit a Reinstate Recovery Qualifier (RRQ) ELS request following an R_A_TOV timeout in order to free those Exchange resources.

Each Nx_Port abnormally terminates any open Sequences associated with the Exchange. The LS_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.

N_Port Login shall be required to send this ELS.

12.3.2.2.2 Protocol

- a) Abort Exchange Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.2.3 Request Sequence

Format: FT_1

Addressing: The D_ID field designates the other Nx_Port of the Exchange being aborted. The S_ID field designates the Nx_Port that is requesting that the Exchange be aborted.

Exchange: A separate Exchange shall be used for the ABTX other than the Exchange being aborted in order to properly track status.

SEQ_ID and SEQ_CNT: The SEQ_ID and the SEQ_CNT shall be appropriate for an open Sequence.

Payload: The format of the Payload is shown in table 64.

Table 64 - ABTX Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	ABTX (hex '06')	hex '00'	hex '00'	hex '00'
1	Recovery_Qualifier status	Originator S_ID		
2	OX_ID		RX_ID	
3	MSB			
..	Association_Header (optional)			
10	LSB			

Recovery Qualifier: The Recovery Qualifier values are given in table 65.

Table 65 - Recovery Qualifier Values

Value	Meaning
hex '00'	no Recovery Qualifier
hex '80'	Recovery Qualifier Required

12.3.2.2.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the ABTX command.

LS_ACC: LS_ACC signifies that the ABTX recipient Nx_Port has terminated the Exchange. The format of the LS_ACC Payload is shown in table 66.

Table 66 - ABTX LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'

12.3.2.3 Advise Credit (ADVC)

12.3.2.3.1 Description

The ADVC ELS shall be used to advise the destination Nx_Port of the estimated end-to-end Credit that the source Nx_Port requests to be allocated. The ADVC ELS request shall be a separate Sequence. It may also be requested in a separate Exchange. See 18.4.11.2.4 for the usage of this frame. The ADVC request may also be used independently from the Estimate Credit procedure (see 18.4.11). If the recipient Nx_Port is not logged in with the requesting Nx_Port it shall reply with an LS_RJT ELS Sequence with a reason code of "Unable to perform command request" and a reason code explanation of "N_Port Login Required".

12.3.2.3.2 Protocol

- Advise Credit Request Sequence
- LS_ACC or LS_RJT Reply Sequence

12.3.2.3.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting Credit revision. The D_ID field designates the destination Nx_Port.

Payload: The format of the Payload is shown in table 67. The Payload shall contain the requested End-to-end Credit in the End-to-end Credit field of the appropriate Class Service Parameters (see 15.6.5.9) as indicated by the Class Validity bit. For each class in which a revised end-to-end Credit is requested, the Class Validity bit shall be

set to one. The receiver shall ignore the other Service Parameter fields.

Table 67 - ADVC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	ADVC (hex '0D')	hex '00'	hex '00'	hex '00'
1	MSB	Common Service Parameters (16 bytes)		
..				
4				LSB
5	MSB	N_Port_Name (8 bytes)		
6				LSB
7	MSB	Node_Name (8 bytes)		
8				LSB
9	MSB	Class 1 Service Parameters (16 bytes)		
..				
12				LSB
13	MSB	Class 2 Service Parameters (16 bytes)		
..				
16				LSB
17	MSB	Class 3 Service Parameters (16 bytes)		
..				
20				LSB
21	MSB	Reserved (16 bytes)		
..				
24				LSB
25	MSB	Vendor Version Level (16 bytes)		
..				
28				LSB

12.3.2.3.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the ADVC command.

LS_ACC: LS_ACC signifies successful completion of the ADVC function and permanently replaces the end-to-end Credit in effect for the current N_Port Login.

The format of the LS_ACC Payload is shown in table 68. 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 Nx_Port transmitting the LS_ACC Sequence (see 15.6.5.9). For each class in which a revised end-to-end Credit is updated, the Class Validity bit shall be set to one. The receiver shall ignore the other Service Parameter fields. This revised end to-end Credit value is determined by the destination Nx_Port based on its buffering scheme, buffer management, buffer

availability, and Nx_Port processing time. See 18.4.11.2.4 for the determination of this value.

Table 68 - ADVC LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	MSB	Common Service Parameters (16 bytes)		
..				
4				LSB
5	MSB	N_Port_Name (8 bytes)		
6				LSB
7	MSB	Node_Name (8 bytes)		
8				LSB
9	MSB	Class 1 Service Parameters (16 bytes)		
..				
12				LSB
13	MSB	Class 2 Service Parameters (16 bytes)		
..				
16				LSB
17	MSB	Class 3 Service Parameters (16 bytes)		
..				
20				LSB
21	MSB	Reserved (16 bytes)		
..				
24				LSB
25	MSB	Vendor Version Level (16 bytes)		
..				
28				LSB

12.3.2.4 Echo (ECHO)

12.3.2.4.1 Description

The Echo ELS 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 LS_ACC Reply Sequence. The Echo frame shall indicate End_Sequence and Sequence Initiative transfer as well as other appropriate F_CTL bits. The Echo ELS 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 LS_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.

12.3.2.4.2 Protocol

- Echo Request Sequence
- LS_ACC or LS_RJT Reply Sequence

12.3.2.4.3 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 69. The Payload size is limited by the smallest Data Field size supported by the destination Nx_Port, the Fabric, and the source destination Nx_Port for the class being used since the Echo Data Field LS_ACC frame shall be equal in size to the Echo Data Field size.

Table 69 - ECHO Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	ECHO (hex '10')	hex '00'	hex '00'	hex '00'
1	MSB			
..	ECHO data			
n	(up to max. frame length, any byte boundary)			
				LSB

12.3.2.4.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the ECHO command.

LS_ACC: LS_ACC signifies successful completion of the ECHO function. The format of the LS_ACC Payload is shown in table 70. The Payload shall contain the ECHO data contained in the Payload of the ECHO Request frame.

Table 70 - ECHO LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	MSB			
..	ECHO data			
n	(up to max. frame length - 4, any byte boundary)			
	Excludes word 0 of ECHO Payload.			
				LSB

12.3.2.5 Estimate Credit (ESTC)

12.3.2.5.1 Description

The ESTC ELS shall be used to estimate the minimum Credit required to achieve the maximum bandwidth for a given distance between an Nx_Port pair. The ESTC ELS Request shall have the frame size as determined by Login with the destination Nx_Port.

The class of the SOF delimiter of the ESTC Request identifies the class for which Credit is being estimated. The destination Nx_Port shall acknowledge Data frames as specified by its Login parameters. See 18.4.11.2.3 for the usage of this frame.

12.3.2.5.2 Protocol

- Estimate Credit Request Sequence
- No Reply Sequence

12.3.2.5.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting the Credit estimate. The D_ID field designates the destination Nx_Port specified in the Establish Streaming frame.

Payload: The format of the Payload is shown in table 71. 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 71 - ESTC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	ESTC (hex '0C')	hex '00'	hex '00'	hex '00'
1	MSB	Any data (up to max. frame length - 4, any byte boundary)		
..				
n				LSB

12.3.2.5.4 Reply Sequence

None.

12.3.2.6 Establish Streaming (ESTS)

12.3.2.6.1 Description

The ESTS ELS shall request a temporary allocation of Credit known as Streaming Credit large enough to perform continuous streaming of Data frames. The SOF delimiter of the ESTS Request identifies the class for which Credit is being estimated. See 18.4.11.2.2 for the usage of this frame.

If the recipient Nx_Port is not logged in with the requesting Nx_Port it shall reply with an LS_RJT ELS Sequence with a reason code of "Unable to perform command request" and a reason code explanation of "N_Port Login required".

12.3.2.6.2 Protocol

- Establish Streaming Request Sequence
- LS_ACC or LS_RJT Reply Sequence

12.3.2.6.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting Streaming. The D_ID field designates the destination Nx_Port addressed.

Payload: The format of the Payload is shown in table 72.

Table 72 - ESTS Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	ESTS (hex '0B')	hex '00'	hex '00'	hex '00'

12.3.2.6.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the ESTS command

LS_ACC: LS_ACC signifies successful completion of the ESTS function. The format of the LS_ACC Payload is shown in table 73. The Payload shall contain Streaming Credit (L) allocated in the Credit field of the appropriate Class Service Parameters. The Class Validity bit, when set to one, identifies the class that contains the Streaming Credit. The receiver shall ignore the other Service Parameter fields.

Table 73 - ESTS LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	MSB	Common Service Parameters (16 bytes)		
4				LSB
5	MSB	Port Name (8 bytes)		
6				LSB
7	MSB	node Name (8 bytes)		
8				LSB
9	MSB	Class 1 Service Parameters (16 bytes)		
12				LSB
13	MSB	Class 2 Service Parameters (16 bytes)		
16				LSB
17	MSB	Class 3 Service Parameters (16 bytes)		
20				LSB
21	MSB	Reserved (16 bytes)		
24				LSB
25	MSB	Vendor Version Level (16 bytes)		
38				LSB

12.3.2.7 Login (FLOGI/PLOGI)

12.3.2.7.1 Description

The FLOGI/PLOGI ELS shall transfer Service Parameters from the initiating Nx_Port to the FC_Port associated with the D_ID. The FLOGI frame provides the means by which an Nx_Port may request Login with the Fabric (see 15.3). The PLOGI frame provides the means by which an Nx_Port may request Login with another Nx_Port prior to other Data frame transfers (see 15.4).

In order to Login with the Fabric and determine the Fabric operating characteristics, an Nx_Port shall specify the D_ID as the well-known F_Port_ID (i.e., hex 'FF FF FE').

In order to direct the Login ELS frame to a Fibre Channel Service, an Nx_Port shall specify the N_Port_ID for the Nx_Port providing the service or the appropriate well-known Address Identifier (see table 21).

When an Nx_Port receives a Login from an Nx_Port, all open Sequences with the Nx_Port performing Login shall be abnormally terminated.

12.3.2.7.2 Protocol

- a) FLOGI or PLOGI Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.7.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting Login. If unidentified, as in Fabric Login, binary zeros are used. The D_ID field designates the destination Nx_Port or Fx_Port of the Login.

Payload: The format of the Payload is shown in table 216. The Service Parameters are defined in 15.6.

12.3.2.7.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the FLOGI or PLOGI Request Sequence

LS_ACC: LS_ACC signifies successful completion of the FLOGI or PLOGI Request Sequence. The format of the LS_ACC Payload is shown in table 216. The Service Parameters are defined in 15.6.

12.3.2.8 Logout (LOGO)

12.3.2.8.1 Description

The LOGO ELS, when sent to an N_Port, shall request invalidation of the Service Parameters and N_Port_Name that have been saved by an Nx_Port, freeing those resources. This provides a means by which an Nx_Port may request Logout or remove service between two Nx_Ports. (see 15.5)

The LOGO ELS, when sent to the well-known (hex 'FF FF FE'), shall request the removal and logout of the previously-assigned N_Port_ID specified in the S_ID field.

The N_Port_ID and N_Port_Name of the Nx_Port requesting Logout are identified in the Payload. For the case of N_Port logout with another N_Port, this allows an Nx_Port to Logout its old Identifier using a new Identifier after its native N_Port_ID has changed. Both the source Nx_Port and the destination Nx_Port of the Logout Request Sequence shall abnormally terminate all open Exchanges (see 17.7.2) that used the N_Port_ID indicated in the Payload of the Logout Request Sequence.

12.3.2.8.2 Protocol

- a) Logout Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.8.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting Logout. The D_ID field designates the destination Nx_Port of the Logout Request, or the well-known F_Port ID, hex 'FF FF FE'.

Payload: The format of the Payload is shown in table 74.

Table 74 - LOGO Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	LOGO (hex '05')	hex '00'	hex '00'	hex '00'
1	Reserved	N_Port_ID		
2	MSB	N_Port_Name		
3		(8 bytes) LSB		

12.3.2.8.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the LOGO command.

LS_ACC: When the LOGO ELS was sent to an N_Port, LS_ACC signifies that Service has been removed; when the LOGO ELS was sent to the well-known F_Port ID (hex 'FF FF FE'), LS_ACC signifies that N_Port_ID specified in the D_ID field of the LS_ACC has been removed and logged out. The format of the LS_ACC Payload is shown in table 75.

Table 75 - LOGO LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'

12.3.2.9 Read Connection Status (RCS)

12.3.2.9.1 Description

The RCS ELS shall request the Fabric Controller to return the current dedicated connection status for the Nx_Port specified in the Payload of the RCS frame. The RCS Request provides the means by which an Nx_Port may interrogate the Fabric for the Connection status of other Nx_Ports within the Fabric.

In order to direct the RCS ELS Request frame to the Fabric, an Nx_Port specifies the D_ID as the Fabric Controller (i.e., hex 'FF FF FD').

12.3.2.9.2 Protocol

- a) Read Connection Status Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.9.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_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 76. The first word of the Payload shall contain the LS_Command code. The second word shall contain the N_Port_ID for which Connection status is being requested.

Table 76 - RCS Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RCS (hex '07')	hex '00'	hex '00'	hex '00'
1	Reserved	N_Port_ID		

12.3.2.9.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the RCS command.

LS_ACC: LS_ACC signifies that the Fabric has completed the request. The format of the LS_ACC Payload is shown in table 77.

Table 77 - RCS LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	Connection Status	N_Port_ID		

The Connection Status Codes are defined in table 78.

Bits 23 through 0 specify the address identifier of the Nx_Port involved in a dedicated connection with the other

Nx_Port specified by the RCS ELS Request Sequence frame (i.e., the Connection established bit is set to one).

Table 78 - Connection Status Codes

Name	Bit Number	Value	Definition
Connect-Request delivered	31	0	The specified Nx_Port is either not Connected, or is involved in an Established Connection based on the setting of the Connection established bit.
		1	A connect-Request has been delivered to the specified Nx_Port, but the Nx_Port has not yet responded with a proper response frame and a dedicated connection does not yet exist.
Connect-Request stacked	30	0	No connect-Request is stacked for the specified Nx_Port on behalf of the requesting Nx_Port.
		1	One or more connect-Requests are stacked, but have not been delivered to the specified Nx_Port on behalf of the requesting Nx_Port
Connection established	29	0	The specified Nx_Port in the RCS Request is not in a dedicated connection
		1	The specified Nx_Port is involved in a dedicated connection. The address identifier in bits 23 to 0 identifies the other Nx_Port involved in the dedicated connection.
Intermix mode	28	0	The N_Port specified in the RCS frame is not functioning in Intermix mode.
		1	The N_Port specified in the request is functioning in intermix mode. An 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.

12.3.2.10 Read Exchange Status Block (RES)

12.3.2.10.1 Description

The RES ELS Request Sequence requests an Nx_Port to return the Exchange Status Block as defined in 17.8.1 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 provides information for the destination Nx_Port to locate the status information requested. If the RX_ID is hex 'FF FF', then a Responder destination Nx_Port shall only use the OX_ID, otherwise only the RX_ID shall be used. Similarly, the Originator would use the OX_ID and ignore the RX_ID. This provides the Nx_Port transmitting the request with information regarding the current status of the Exchange specified.

If the destination Nx_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 of “Unable to perform the command request”.

N_Port Login shall be required to send this ELS.

12.3.2.10.2 Protocol

- a) Read Exchange Status Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.10.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting the Exchange Status Block. The D_ID field designates the destination Nx_Port to which the request is being made.

Payload: The format of the Payload is shown in 79.

Table 79 - RES Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '08'	hex '00'	hex '00'	hex '00'
1	Reserved	Originator S_ID		
2	OX_ID		RX_ID	
2 + N	MSB			
..	Association_Header (optionally required) (32 bytes)			
9 + N	LSB			

12.3.2.10.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the RES command.

LS_ACC: LS_ACC signifies that the Nx_Port has transmitted the requested data. The format of the LS_ACC Payload is shown in table 80. The format of the Exchange Status Block is specified in table 246.

Table 80 - RES LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	Reserved	Originator S_ID		
2	MSB	Exchange Status Block		
..		(see 17.8.1)		
1 + N		(N words)		
		LSB		
2 + N	MSB	Association_Header (optionally required)		
..		(32 bytes)		
9 + N		LSB		

12.3.2.11 Read Link Error Status Block (RLS)

12.3.2.11.1 Description

The RLS ELS shall request an FC_Port to return the Link Error Status Block associated with the Port_ID specified in the Payload. This provides the Nx_Port transmitting the request with information regarding Link Errors detected within the destination FC_Port. If an Fx_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”.

12.3.2.11.2 Protocol

- a) Read Link Error Status Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.11.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting the Link Error Status Block. The D_ID field designates the destination FC_Port to which the request is being made.

Payload: The format of the Payload is shown in table 81.

Table 81 - RLS Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RLS (hex '0F')	hex '00'	hex '00'	hex '00'
1	Reserved			
	N_Port_ID			

12.3.2.11.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the RLS command.

LS_ACC: LS_ACC signifies that the FC_Port has transmitted the requested data. The format of the LS_ACC Payload is shown in table 82. The format of the Link Error Status Block is specified in table 256.

Table 82 - RLS LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	MSB			
..	Link Error Status Block (see table 256) (24 bytes)			
6				
	LSB			

12.3.2.12 Read Sequence Status Block (RSS)

12.3.2.12.1 Description

The RSS ELS shall request an Nx_Port to return the Sequence Status Block for the SEQ_ID specified in the Payload. The Payload also specifies the S_ID of the Exchange Originator in addition to the associated OX_ID and RX_ID. The specification of OX_ID and RX_ID provide information for the destination Nx_Port to locate the status

information requested. A Responder destination Nx_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 Nx_Port transmitting the request with information regarding the current status of the Sequence it identified.

If the destination Nx_Port of the RSS Request determines that the SEQ_ID, Originator S_ID, OX_ID, or RX_ID are inconsistent, it shall reply with an LS_RJT Sequence with a reason code of "Unable to perform the command request".

N_Port Login shall be required to send this ELS.

12.3.2.12.2 Protocol

- a) Read Sequence Status Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.12.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting the Sequence Status Block. The D_ID field designates the destination Nx_Port to which the request is being made.

Payload: The format of the Payload is shown in table 83.

Table 83 - RSS Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RSS (hex '09')	hex '00'	hex '00'	hex '00'
1	SEQ_ID	Originator S_ID		
2	OX_ID		RX_ID	

12.3.2.12.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the RSS command.

LS_ACC: LS_ACC signifies that the Nx_Port has transmitted the requested data. The format of the LS_ACC Payload is shown in table 84. The format of the Sequence Status Block is specified in 17.8.2.

Table 84 - RSS LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	MSB	Sequence Status Block (see 17.8.2) (N words)		
..				
N				LSB

12.3.2.13 Read Timeout Value (RTV)

12.3.2.13.1 Description

The RTV ELS shall request an FC_Port to return the R_A_TOV and the E_D_TOV in the LS_ACC. The LS_ACC also indicates that the FC_Port uses the default value or the short value for R_T_TOV. This provides the FC_Port transmitting the RTV ELS with information regarding these values from another FC_Port. Usage of R_A_TOV, E_D_TOV and R_T_TOV requirements are given in 21.2.1.

12.3.2.13.2 Protocol

- a) Read Timeout Value (RTV) Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.13.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting the timeout interval values. The D_ID field designates the destination FC_Port to which the request is being made.

Payload: The format of the Payload is shown in table 85.

Table 85 - RTV Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RTV (hex '0E')	hex '00'	hex '00'	hex '00'

12.3.2.13.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the RTV command.

LS_ACC: LS_ACC signifies that the FC_Port has transmitted the requested data. The format of the LS_ACC Payload is shown in table 86. E_D_TOV Timeout values are specified as a count of either 1 ms or 1 ns increments, depending on the setting of the E_D_TOV Resolution (see 15.6.2.4.8).

Table 86 - RTV LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	Resource_Allocation_Timeout Value (R_A_TOV) (see 21.2.1.4)			
2	Error_Detect_Timeout Value (E_D_TOV) (see 21.2.1.4)			
3	Timeout Qualifier (see 15.6.2.10)			

The Timeout Qualifier word is defined as follows:

- a) Bits 31 to 27: Reserved
- b) Bit 26: E_D_TOV Resolution
- c) If the E_D_TOV Resolution bit is zero, the value specified in the E_D_TOV field shall indicate a count of 1 ms increments. If the E_D_TOV Resolution bit is one, the value specified in the E_D_TOV field shall indicate a count of 1 ns increments.
- d) Bits 25 to 20: Reserved

- e) Bit 19: R_T_TOV Value
- f) If this bit is set to zero, the value of R_T_TOV shall be the default value of 100 ms. If it is set to one, the value of R_T_TOV shall be 100 μ s.
- g) Bits 18 to 0: Reserved

12.3.2.14 Reinstate Recovery Qualifier (RRQ)

12.3.2.14.1 Description

The RRQ ELS shall be used to notify the destination Nx_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 minus 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 Nx_Port or the Responder Nx_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, in addition to the S_ID of the Nx_Port that originated the Exchange being aborted. 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 LS_ACC 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 is 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 LS_ACC for ABTX has been received) by the Nx_Port that transmitted and successfully completed either the ABTX or the ABTS frame.

N_Port Login shall be required to send this ELS.

12.3.2.14.2 Protocol

- a) Reinstate Recovery Qualifier Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.14.3 Request Sequence

Format: FT_1

Addressing: The D_ID field designates the destination Nx_Port of the RRQ Request Sequence while the S_ID field designates the source Nx_Port that is requesting that the Recovery_Qualifier be reinstated.

Exchange: A separate Exchange is required.

SEQ_ID and SEQ_CNT: The SEQ_ID and the SEQ_CNT shall be appropriate for an open Sequence.

Payload: The format of the Payload is shown in table 87.

Table 87 - RRQ Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RRQ (hex '12')	hex '00'	hex '00'	hex '00'
1	Reserved	Originator S_ID		
2	OX_ID		RX_ID	
3	MSB			
..				
10	Association_Header (optional) (32 bytes)			LSB

12.3.2.14.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the RRQ command.

LS_ACC: LS_ACC signifies that the destination Nx_Port reinstated the Recovery_Qualifier. The format of the LS_ACC Payload is shown in table 88.

Table 88 - RRQ LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'

12.3.2.15 Request Sequence Initiative (RSI)

12.3.2.15.1 Description

The RSI ELS 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 Originator Nx_Port or the Responder Nx_Port of the target Exchange makes the request. 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, in addition to the S_ID of the Nx_Port that originated the Exchange. The LS_ACC 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 is a Sequence 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 one in F_CTL.
- 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 one in F_CTL.

If there is no Sequence 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 LS_ACC to the Exchange requesting Sequence Initiative shall be transmitted after Sequence Initiative has been passed (see 9.7.9) on the target Exchange.

12.3.2.15.2 Protocol

- a) Request Sequence Initiative Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.15.3 Request Sequence

Format: FT_1

Addressing: The D_ID field designates the destination Nx_Port of the Exchange for which Sequence Initiative is being requested and the S_ID field designates the source Nx_Port that is requesting Sequence Initiative.

Exchange: A separate Exchange is required other than the Exchange for which Sequence Initiative is being requested in order to properly track status.

SEQ_ID and SEQ_CNT: The SEQ_ID and the SEQ_CNT shall be appropriate for an open Sequence.

Payload: The format of the Payload is shown in table 89.

Table 89 - RSI Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RSI (hex '0A')	hex '00'	hex '00'	hex '00'
1	Reserved	Originator S_ID		
2	OX_ID		RX_ID	
3	MSB			
..				
10	Association_Header (optional) (32 bytes)			LSB

12.3.2.15.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the RSI command.

LS_ACC: LS_ACC signifies that the destination Nx_Port has transferred the Sequence Initiative for the target Exchange. The format of the LS_ACC Payload is shown in table 90.

Table 90 - RSI LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'

12.3.2.16 Test (TEST)

12.3.2.16.1 Description

The TEST ELS 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.

12.3.2.16.2 Protocol

- a) Test Request Sequence
- b) No Reply Sequence

12.3.2.16.3 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 91.

Table 91 - TEST Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	TEST (hex '11')	hex '00'	hex '00'	hex '00'
1	MSB	TEST data		
..		(up to max. Data Field size, any byte boundary)		
n				LSB

The Payload size is limited by the smallest Data Field size supported by the destination Nx_Port and the Fabric for the class being used.

12.3.2.16.4 Reply Sequence

none

12.3.2.17 Report node Capability Information (RNC)

Obsolete. For an ELS that reports FC-4 types, see 12.3.2.50.

12.3.2.18 Fabric Address Notification (FAN)

12.3.2.18.1 Description

The FAN ELS shall be sent by an FL_Port (hex 'FF FF FE') to all known previously logged in (via FLOGI) attached NL_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 FL_Port shall report the F_Port_Name and Fabric_Name as they were reported in the prior FLOGI, and shall report the current Loop Fabric Address. The FL_Port shall send this ELS 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 FL_Port shall report identical information to all attached NL_Ports. If the information changes in any way before the FL_Port is able to send the service to all attached ports, the FL_Port shall begin a new initialization event.

The attached NL_Ports shall not initiate a Reply Sequence to this ELS.

12.3.2.18.2 Protocol

- a) Fabric Address Notification Request Sequence
- b) No Reply Sequence

12.3.2.18.3 Request Sequence

Format: FT_1

Addressing: The S_ID is the FL_Port (hex 'FF FF FE') sending the FAN. The D_ID is the NL_Port receiving the FAN.

Payload: The format of the FAN Request Payload is shown in table 92.

Table 92 - FAN Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	FAN (hex '60')	hex '00'	hex '00'	hex '00'
1	Reserved	Loop Fabric Address		
2	MSB	F_Port_Name (8 bytes)		LSB
3				
4	MSB	Fabric_Name (8 bytes)		LSB
5				

12.3.2.18.4 Reply Sequence

none

12.3.2.19 Loop Initialize (LINIT)

12.3.2.19.1 Description

The LINIT ELS shall request the start 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 LS_ACC 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. See [2], FC-AL-2). If the LINIT ELS is received by a non-FL_Port, the port shall originate an LS_RJT with a reason code of "Command not supported".

12.3.2.19.2 Protocol

- a) Loop Initialize Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.19.3 Request Sequence

Format: FT_1

Addressing: The S_ID designates the Nx_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 93.

Table 93 - LINIT Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	LINIT (hex '70')	hex '00'	hex '00'	hex '00'
1	Reserved	Initialization Function	LIP byte 3	LIP byte 4

The Initialization Function field defines modifications of the initialization to be performed. The format of the Initialization Function is shown in table 94.

Table 94 - 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 Loop Initialization Soft Assigned (LISA) Sequence to force all affected L_Ports to perform FLOGI.	1
Reserved	2 to 255

The LIP Byte 3 field is the 3rd character of the LIP Primitive Sequence to be originated. The LIP Byte 4 field is the 4th character of the LIP Primitive Sequence to be originated. LIP Byte 3 and LIP Byte 4 should only be set to values permitted by FC-AL-2.

12.3.2.19.4 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the LINIT command

LS_ACC: LS_ACC signifies acceptance of the LINIT Request and initiation of Loop Initialization. The format of the LS_ACC Payload is shown in table 95:

Table 95 - LINIT LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	Reserved			Status

The format of the Status field is shown in table 96.

Table 96 - LINIT Status

State	Value
Reserved	0
Success - The requested function was completed.	1
Failure - The requested function could not be completed.	2
Reserved	3 to 255

12.3.2.20 Loop Port Control (LPC) (Obsolete)

The LPC ELS is obsolete.

12.3.2.21 Loop Status (LSTS)

12.3.2.21.1 Description

The LSTS ELS shall be used to request the Fabric Controller to report on the state of the specified loop.

12.3.2.21.2 Protocol

- a) Loop Status Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.21.3 Request Sequence

Format: FT_1

Addressing: The S_ID is the Nx_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 97.

Table 97 - LSTS Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	LSTS (hex '72')	hex '00'	hex '00'	hex '00'

12.3.2.21.4 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the LSTS command.

LS_ACC: LS_ACC supplies the requested status for the identified Loop. The format of the LS_ACC Payload is shown in table 98.

Table 98 - LSTS LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	Reserved	Failed receiver	FC-FLA Compliance Level	Loop State

Table 98 - LSTS LS_ACC Payload

2	MSB		
..		Current Public Loop Devices bit map	
5		(16 bytes)	LSB
6	MSB		
..		Current Private Loop Devices bit map	
9		(16 bytes)	LSB
10	MSB		
..		AL_PA Position Map	
41		(128 bytes)	LSB

The Payload fields are defined as follows:

a) Failed receiver field: The Failed receiver field shall contain 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 Fabric Controller is unable to determine the port that detected the failure.

NOTE The Failed Receiver field is valid only if the Loop State indicates a Loop Failure. The Failed Receiver field shall contain the AL_PA of the L_Port that detected the loop failure.

b) Compliance Level field: The FC-FLA Compliance Level field shall contain 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 99. See [9], FC-FLA for the meaning of the various compliance levels.

Table 99 - FC-FLA Compliance Level

Level	Value
Reserved	0
FC-FLA Level A	1
FC-FLA Level B	2
Reserved	3 to 255

c) **Loop State field:** The format of the Loop State field is shown in table 100.

Table 100 - 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 to 255

d) **Current Public Loop Devices:** The format of the Current Public Loop Devices field follows the AL_PA bit mapped format defined in FC-AL-2. 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.

e) **Current Private Loop Devices:** The format of the Current Private Loop Devices field follows the AL_PA bit mapped format defined in FC-AL-2. 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.

f) **AL_PA Position Map:** The format of the AL_PA Position Map field follows the AL_PA position map format defined in FC-AL-2 for the Loop Initialization Loop Position (LILP) Sequence. If Byte 0 of the first word of the AL_PA position map is set to zero, then no AL_PA position map is available. This field is valid only when the value in the Loop State field is Online.

12.3.2.22 Registered State Change Notification (RSCN)

12.3.2.22.1 Introduction

A RSCN ELS shall be sent to registered Nx_Ports when an event occurs that may have affected the state of one or more Nx_Ports, or the ULP state within the Nx_Port. The term state is used here to refer to any condition of an Nx_Port that is considered important enough to notify other Nx_Ports of a change in that state. The definition of important is specific to an Nx_Port implementation, but should include the Login state or Link state. The RSCN additionally provides an indication of the change of state that is being reported.

RSCN is intended to provide a timely indication of changes in nodes to avoid the considerable traffic that polling may generate. RSCN may be used to indicate a failed node, allowing the release of 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.

A RSCN Request may be sent by the fabric to notify registered Nx_Ports of changes detected by the fabric. The Fabric Controller (hex 'FF FF FD') issues the RSCN Request to the registered Nx_Ports. An Nx_Port may also issue a RSCN Request to the Fabric Controller or another Nx_Port to indicate changes of state within the Nx_Port that are not otherwise detectable by the fabric.

The Payload of a RSCN Request includes a list containing the addresses of the affected Nx_Ports. The RSCN includes a summary indication of the type of state change being reported to assist in analyzing the change. The sending of RSCN between Nx_Ports, neither of which are the Fabric Controller, is permitted.

12.3.2.22.2 RSCNs issued by the Fabric Controller

The Fabric Controller shall issue an RSCN Request to all registered Nx_Ports for an affected Nx_Port when the fabric detects an event. The Fabric Controller shall ensure that any Fabric-provided resources (e.g., 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:

- a) an implicit fabric Logout of the affected Nx_Port, including Loss-of-Signal, NOS, and OLS, or when the fabric receives a FLOGI that contains new or different information from a port that had already completed FLOGI;
- b) a loop initialization of the affected L_Port, and the L_bit was set in the LISA Sequence;
- c) a fabric Login from an affected Nx_Port not previously logged in;
- d) the fabric path between the affected Nx_Port and any other Nx_Port has changed (e.g., 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);
- e) any other fabric-detected state change of the affected Nx_Port;
- f) the affected Nx_Port issues an RSCN Request to the Fabric Controller.

A registered Nx_Port that receives an RSCN Request may perform any operation 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 Nx_Port addresses for subsequent delivery to reduce the volume of RSCN traffic.

12.3.2.22.3 RSCN issued by the affected Nx_Port

An Nx_Port shall issue an RSCN Request to the Fabric Controller or the Nx_Port that has registered for receipt of RSCN Requests or both when an event is detected by an Nx_Port. Fabric Controllers are implicitly registered to receive RSCNs after a successful implicit or explicit FLOGI. Nx_Ports, neither of which are the Fabric Controller, that request RSCNs are hereafter called registered Nx_Ports. Unless it is providing a service associated with a well known address an Nx_Port shall only issue RSCNs with an event qualifier of 02h or 00h. An event may include any of the following:

- a) a failure within the affected Nx_Port;
- b) any other important state change of the affected Nx_Port (e.g., an event defined by a FC-4).

12.3.2.22.4 RSCN initiative

An affected Nx_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.

12.3.2.22.5 RSCN registration

A node (Fabric Controller, service, Nx_Port) shall only issue RSCN requests to Nx_Ports that have registered with the node to be notified of state changes. These registrations shall be performed using the State Change Registration (SCR) ELS.

12.3.2.22.6 Protocol

- a) RSCN Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.22.7 Request Sequence

Format: FT 1

Addressing: If the fabric is using RSCN to notify a Registered Nx_Port of a state change (see 12.3.2.22.2), the S_ID is the Fabric Controller, (hex 'FF FF FD') and the D_ID is the address of the Registered Nx_Port destination. If an affected Nx_Port is using RSCN to notify the fabric or a Registered Nx_Port of a state change (see 12.3.2.22.3), the S_ID designates either the Nx_Port indicating a state change to the Fabric Controller or the Registered Nx_Port and the D_ID is either the Fabric Controller, (hex 'FF FF FD') or the address of the Registered Nx_Port destination.

Payload: The format of the RSCN Request Payload is shown in table 101.

Table 101 - RSCN Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RSCN (hex '61')	Page Length (hex '04')	Payload Length	
1				
..	affected Port_ID pages			
N	(1 to 255 pages, 4 bytes each)			

The RSCN Payload fields are defined as follows:

- a) The Page Length field is the length in bytes of an affected Port_ID page. This value is fixed at hex '04'.
- b) The Payload Length field is the length in bytes of the entire Payload, inclusive of the word 0. This value shall be a multiple of 4 bytes. The minimum value of this field is 8 bytes. The maximum value of this field is 1024 bytes.
- c) Affected Port_ID Pages: Each affected Port_ID page contains the ID of the Nx_Port, Fabric Controller, E_Port, domain, or area for which the event was detected. The RSCN Payload shall contain one or more of these pages. The generic format of the affected Port_ID page is shown in table 102.

Table 102 - Generic affected Port ID page

Bit in Byte	7	6	5	4	3	2	1	0
Byte								
0 (Bits 31 to 24)	reserved		RSCN Event Qualifier				Address Format	
1 (Bits 23 to 16)	affected Port_ID byte 1 (Domain)							
2 (Bits 15 to 08)	affected Port_ID byte 2 (Area)							
3 (Bits 07 to 00)	affected Port_ID byte 3 (Port) Sequence							

RSCN Event Qualifier: The RSCN Event Qualifier values are shown in table 103.

Table 103 - RSCN Event Qualifier values

RSCN event Qualifier	Value			
	Bit 5	Bit 4	Bit 3	Bit 2
Event is not specified	0	0	0	0
CHANGED NAME SERVER OBJECT - An object maintained by the Name Server has changed state for the port, area or domain indicated by the affected Port_ID.	0	0	0	1
CHANGED PORT ATTRIBUTE - An internal state of the port specified by the affected Port_ID has changed. The change of state is identified in a protocol specific manner.	0	0	1	0
CHANGED SERVICE OBJECT - An object maintained by the service identified by the well-known address contained in affected Port_ID has changed state. This Event Qualifier value shall not be used by services accessed through N_Port_ID that are not well-known addresses.	0	0	1	1
CHANGED SWITCH CONFIGURATION - Switch configuration has changed for the area or domain specified by the affected Port_ID.	0	1	0	0
Reserved	All Other Values			

Address Format field: The format of the Address Format field is shown in table 104.

Table 104 - Address Format

Format	Value
Port Address - Bytes 1, 2, and 3 of the affected Port_ID are valid and indicate a single Nx_Port or service with a well-known address.	0
Area Address Group - Bytes 1 and 2 of the affected Port_ID are valid and indicate a group of addresses that encompass an Area of E_Port or Nx_Port addresses. Byte 3 shall be zero. Any links and ports within the area may be affected.	1
Domain Address Group - Byte 1 of the affected Port_ID is valid and indicates a group of addresses that encompass a Domain. Bytes 2 and 3 shall be zero. Any links and ports within the domain may be affected.	2
Fabric Address Group - This format indicates a group of addresses that encompass the entire Fabric of Nx_Port addresses. Bytes 1, 2 and 3 shall be zero. Any links and ports within the area may be affected.	3
Reserved	4 to 255

12.3.2.22.8 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the RSCN command

LS_ACC: LS_ACC signifies acceptance of the RSCN Request. The format of the LS_ACC Payload is shown in

table 105.

Table 105 - RSCN LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'

12.3.2.23 State Change Registration (SCR)

12.3.2.23.1 Description

The SCR ELS shall request the Fabric Controller or Nx_Port to add the Nx_Port that is sending the SCR Request to the list of Nx_Ports registered to receive the RSCN ELS.

12.3.2.23.2 Protocol

- a) State Change Registration Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.23.3 Request Sequence

Format: FT_1

Addressing: The S_ID designates the Nx_Port requesting registration for State Change Notification. The D_ID designates either the Fabric Controller, hex 'FF FF FD', or the address of the Nx_Port from which RSCNs are desired.

Payload: The format of the SCR Request Payload is shown in table 106.

Table 106 - SCR Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	SCR (hex '62')	hex '00'	hex '00'	hex '00'
	Reserved			Registration Function

The format of the Registration Function field is shown in table 107.

Table 107 - 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
Nx_Port Detected registration - Register to receive all RSCN Requests issued for events detected by the affected Nx_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 to 254
Clear registration - Remove any current RSCN registrations.	255

12.3.2.23.4 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the SCR command.

LS_ACC: LS_ACC signifies acceptance of the SCR Request and registration for RSCN. The format of the LS_ACC Payload is shown in table 108.

Table 108 - SCR LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'

12.3.2.24 Process login (PRLI)

12.3.2.24.1 Introduction

The PRLI ELS shall be used to establish the operating environment between a group of related processes at the originating Nx_Port and a group of related processes at the responding Nx_Port. (see 16.1).

12.3.2.24.2 Protocol

- Process Login Request Sequence
- LS_ACC or LS_RJT Reply Sequence

12.3.2.24.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the Nx_Port requesting Process Login. The D_ID field designates the destination Nx_Port of the Process Login.

Payload: The format of the Payload is shown in table 109.

Table 109 - PRLI Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	PRLI (hex '20')	Page Length (hex '10')	Payload Length	
1	Service Parameter pages (4 words each page)			
..				
N				

The Payload Field are defined as follows:

- Page length: Byte 1 of word 0 contains an 8-bit value that specifies the length of each Service Parameter page. The right-most two bits shall be zeros.
- Payload length: Bytes 2 to 3 of word 0 contain a 16-bit value that specifies the length of the PRLI Payload. The right-most 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:N 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 110.

Table 110 - PRLI service parameter page format

Item	Word	Bit
TYPE Code or Common Service Parameters ^a	0	31 to 24
TYPE Code Extension	0	23 to 16
Originator Process_Associator Validity	0	15
Responder Process_Associator Validity	0	14
Establish Image Pair	0	13
Reserved	0	12 to 0
Originator Process_Associator	1	31 to 0
Responder Process_Associator	2	31 to 0
Service Parameters	3	31 to 0
^a 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.		

The PRLI service parameter page format fields are defined as follows:

- a) Word 0, Bits 31 to 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 hex '00', the Service Parameter page contains Service Parameters common to all FC-4 Types at that image pair or Nx_Port pair. If byte 0 of the first word of a Service Parameter page contains the value other than hex '00', the Service Parameter page contains Service Parameters for the FC-4 TYPE indicated.
- b) Word 0, Bits 23 to 16 - TYPE code extension
Reserved for future use.
- c) Word 0, Bit 15 - Originator Process_Associator validity
0 = not meaningful
1 = meaningful
- d) Word 0, Bit 14 - Responder Process_Associator validity
0 = not meaningful
1 = meaningful
- e) Word 0, Bit 13 - Establish Image Pair
0 = Exchange Service Parameters only
1 = Establish image pair and exchange Service Parameters
- f) Word 3, Bits 31 to 0 - Service parameters
No Common Service Parameters are currently specified.

12.3.2.24.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the PRLI Request

LS_ACC: LS_ACC signifies successful completion of the PRLI Request. The format of the LS_ACC Payload is shown in table 111.

Table 111 - PRLI LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	Page Length (hex '10')	Payload Length	
1	Service Parameter Response pages (4 words each page)			
..				
N				

The LS_ACC Payload fields are defined as follows:

- a) Page length:
Byte 1 of word 0 contains an 8-bit value that specifies the length of each Service Parameter Response page. The right-most two bits shall be zeros.
- b) Payload length:
Bytes 2 to 3 of word 0 contain a 16-bit value that specifies the length of the PRLI LS_ACC Payload. The right-most two bits shall be zeros. The value specified shall be greater than or equal to 20, and less than or equal to 65 532.
- c) Service parameter response page(s):
Words 1:N of the PRLI LS_ACC Payload contain one or more Service Parameter Response pages. Each Service Parameter Response page contains Service Parameter responses for a single image pair or

Nx_Port pair and is associated with a single FC-4 TYPE or common to all FC-4 Types at that image pair or Nx_Port pair.

The number of Service Parameter pages and the Payload Length field of the PRLI LS_ACC Reply Sequence shall match the number of Service Parameter pages and the Payload Length field of the PRLI Request Sequence.

The format of PRLI LS_ACC Service Parameter Response pages is described in table 112.

Table 112 - PRLI LS_ACC service parameter response page format

Item	Word	Bit
TYPE Code or Common Service Parameters ^a	0	31 to 24
TYPE Code Extension	0	23 to 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 113)	0	11 to 8
Reserved	0	7 to 0
Originator Process_Associator	1	31 to 0
Responder Process_Associator	2	31 to 0
Service Parameter Response	3	31 to 0
^a 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.		

The PRLI LS_ACC service parameter response page format fields are defined as follows:

- a) Word 0, Bits 31 to 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 hex '00', the Service Parameter page contains Service Parameters common to all FC-4 Types at that image pair or Nx_Port pair. If byte 0 of the first word of a Service Parameter Response page contains a value other than hex '00', the Service Parameter page contains Service Parameters for the FC-4 TYPE indicated.

- b) Word 0, Bits 23 to 16 - TYPE code extension

Reserved for future use.

- c) Word 0, Bit 15 - Originator Process_Associator validity

0 = not meaningful

1 = meaningful

- d) Word 0, Bit 14 - Responder Process_Associator validity

0 = not meaningful

1 = meaningful

e) 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

f) Word 0, Bits 11 to 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 113.

g) Word 3, Bits 31 to 0 - Service parameter response

Provides feedback to the Originator as to the resultant state of the Service Parameters as returned by the Responder.

Table 113 - PRLI accept response code

Encoded Value Word 0, Bits 11 to 8	Description
'0000' b	Reserved
'0001' b	Request executed
'0010' b	The Exchange recipient has no resources available for establishing image pairs between the specified source and destination Nx_Ports. The PRLI Request may be retried.
'0011' b	Initialization is not complete for the Exchange recipient. The PRLI Request may be retried.
'0100' b	The Exchange recipient corresponding to the Responder Process_Associator specified in the PRLI Request and PRLI LS_ACC response does not exist. The PRLI Request shall not be retried.
'0101' b	The Exchange recipient has a predefined configuration that precludes establishing this image pair. The PRLI Request shall not be retried.
'0110' b	Request executed conditionally. Some Service Parameters were not able to be set to their requested state (see table 110)
'0111' b	The destination Nx_Port is unable to process multiple page PRLI Requests. The PRLI request may be retried as a single page request.
'1000' b	Service parameters are invalid
'1001' b to '1111' b	Reserved

12.3.2.25 Process logout (PRLO)

12.3.2.25.1 Description

The PRLO ELS shall be used to request invalidation of the operating environment between an image at the initiating Nx_Port and an image at the recipient Nx_Port. PRLO frees resources committed by a previous PRLI function. (see 16.2)

12.3.2.25.2 Protocol

a) Process Logout Request Sequence

- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.25.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the Nx_Port requesting Process Logout. The D_ID field designates the destination Nx_Port of the Process Logout.

Payload: The format of the Payload is shown in table 114.

Table 114 - PRLO Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	PRLO (hex '21')	Page Length (hex '10')	Payload Length	
1	Logout Parameter pages (4 words each page)			
..				
N				

The PRLO Payload fields shall be defined as follows:

- a) Page length:

Byte 1 of word 0 contains an 8-bit value that specifies the length of each Logout Parameter page. The right-most two bits shall be zeros.

- b) Payload length:

Bytes 2 to 3 of word 0 contain a 16-bit value that specifies the length of the PRLO Payload. The right-most two bits shall be zeros. The value specified shall be greater than or equal to 20, and less than or equal to 65 532.

c) Logout parameter page(s):

Words 1:N 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 115.

Table 115 - PRLO logout parameter page format

Item	Word	Bit
TYPE Code or Common Logout Parameters ^a	0	31 to 24
TYPE Code Extension	0	23 to 16
Originator Process_Associator Validity	0	15
Responder Process_Associator Validity	0	14
Reserved	0	13 to 0
Originator Process_Associator	1	31 to 0
Responder Process_Associator	2	31 to 0
Reserved	3	31 to 0
^a If byte 0 of the first word of a Logout Parameter page is set to the value hex '00', the Logout Parameter page is common to all FC-4 Types at that image pair.		

The PRLO logout parameter page format fields shall be defined as follows:

a) Word 0, Bits 31 to 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 hex '00', the Logout Parameter page contains Logout Parameters common to all FC-4 Types at that image pair or Nx_Port pair. If byte 0 of the first word of a Logout Parameter page contains the value other than hex '00', the Logout Parameter page contains Logout Parameters for the FC-4 TYPE indicated.

b) Word 0, Bits 23 to 16 - TYPE code extension

Reserved for future use.

c) Word 0, Bit 15 - Originator Process_Associator validity

0 = not meaningful

1 = meaningful

d) Word 0, Bit 14 - Responder Process_Associator validity

0 = not meaningful

1 = meaningful

12.3.2.25.4 Reply sequence

LS_RJT: LS_RJT signifies rejection of the PRLO Request

LS_ACC: LS_ACC signifies successful completion of the PRLO Request

LS_ACC Payload: The format of the LS_ACC Payload is shown in table 116.

Table 116 - PRLO LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	Page Length (hex '10')	Payload Length	
1				
..	Logout Parameter Response pages (4 words each page)			
N				

The PRLO LS_ACC Payload fields shall be defined as follows:

a) Page length:

Byte 1 of word 0 contains an 8-bit value that specifies the length of each Logout Parameter Response page. The right-most two bits shall be zeros.

b) Payload length:

Bytes 2 to 3 of word 0 contain a 16-bit value that specifies the length of the PRLO LS_ACC Payload. The right-most two bits shall be zeros. The value specified shall be greater than or equal to 20, and less than or equal to 65 532.

c) Logout parameter response page(s):

Words 1:N of the PRLO LS_ACC 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 Nx_Port pair.

The number of Logout Parameter pages and the Payload Length field of the PRLO LS_ACC Reply Sequence shall match the number of Logout Parameter pages and the Payload Length field of the PRLO Request Sequence.

The format of PRLO LS_ACC Logout Parameter Response pages is described in table 117.

Table 117 - PRLO LS_ACC logout parameter response page format

Item	Word	Bit
TYPE Code or Common Logout Parameters ^a	0	31 to 24
TYPE Code Extension	0	23 to 16
Originator Process_Associator Validity	0	15
Responder Process_Associator Validity	0	14
Reserved	0	13 to 12
Response Code (see table 118)	0	11 to 8
Reserved	0	7 to 0
Originator Process_Associator	1	31 to 0
Responder Process_Associator	2	31 to 0
Reserved	3	31 to 0
^a If byte 0 of the first word of a Logout Parameter page is set to the value hex '00', the Logout Parameter page is common to all FC-4 Types at that image pair.		

The PRLO LS_ACC logout parameter response page format fields shall be defined as follows:

- a) Word 0, Bits 31 to 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 hex '00', the Logout Parameter page contains Logout Parameters common to all FC-4 Types at that image pair or Nx_Port pair. If byte 0 of the first word of a Logout Parameter Response page contains a value other than hex '00', the Logout Parameter page contains Logout Parameters for the FC-4 TYPE indicated.

- b) Word 0, Bits 23 to 16 - TYPE code extension

Reserved for future use.

- c) Word 0, Bit 15 - Originator Process_Associator validity

0 = not meaningful

1 = meaningful

- d) Word 0, Bit 14 - Responder Process_Associator validity

0 = not meaningful

1 = meaningful

- e) Word 0, Bits 11 to 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 118.

Table 118 - PRLO accept response code

Encoded Value Word 0, Bits 11 to 8	Description
'0000' b	Reserved
'0001' b	Request executed
'0010' b to '0011' b	Reserved
'0100' b	The Exchange recipient corresponding to the Responder Process_Associator specified in the PRLO Request and PRLO LS_ACC response does not exist. The PRLO Request shall not be retried.
'0101' b to '0110' b	Reserved
'0111' b	The destination Nx_Port is unable to process multiple page PRLO Requests. The PRLO Request may be retried as a single page request.
'1000' b	Service Parameters are invalid
'1001' b to '1111' b	Reserved

12.3.2.26 State change notification (SCN)

Obsolete

12.3.2.27 Test Process Login State (TPLS)**12.3.2.27.1 Description**

The TPLS ELS shall be used to determine whether 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 Nx_Port checks whether it has an image pair established for each specified image.

TPLS verifies the Login state for the source Nx_Port at the destination Nx_Port.

The LS_ACC 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 LS_ACC Reply indicates the state of the image pair.

If the recipient Nx_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" and a reason code explanation of "N_Port Login required".

12.3.2.27.2 Protocol

- a) Test Process Login State Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.27.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port associated with the image pair. The D_ID field

designates the destination Nx_Port associated with the image pair.

Payload: The format of the Payload is shown in table 119.

Table 119 - TPLS Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	TPLS (hex '23')	Page Length (hex '10')	Payload Length	
1				
..	Image Pair ID pages (4 words each page)			
N				

a) Page length:

Byte 1 of word 0 contains an 8-bit value that specifies the length of each Image Pair ID page. The right-most two bits shall be zeros.

b) Payload length:

Bytes 2 to 3 of word 0 contain a 16-bit value that specifies the length of the TPLS Payload. The right-most two bits shall be zeros. The value specified shall be greater than or equal to 20, and less than or equal to 65 532.

c) Image pair ID page(s):

Words 1:N 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 120.

Table 120 - TPLS image pair ID page format

Item	Word	Bit
Reserved	0	31 to 16
Originator Process_Associator Validity	0	15
Responder Process_Associator Validity	0	14
Reserved	0	13 to 0
Originator Process_Associator	1	31 to 0
Responder Process_Associator	2	31 to 0
Reserved	3	31 to 0

A) Word 0, Bit 15 - Originator Process_Associator validity

0 = not meaningful

1 = meaningful

B) Word 0, Bit 14 - Responder Process_Associator validity

0 = not meaningful

1 = meaningful

12.3.2.27.4 Reply sequence

LS_RJT: LS_RJT signifies rejection of the TPLS Request.

LS_ACC: LS_ACC signifies successful completion of the TPLS Request. The format of the LS_ACC Payload is shown in table 121.

Table 121 - TPLS LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	(hex '02')	Page Length (hex '10')	Payload Length	
1				
..	TPLS Response pages (4 words each page)			
N				

a) Page length:

Byte 1 of word 0 contains an 8-bit value that specifies the length of each Image Pair ID page. The right-most two bits shall be zeros.

b) Payload length:

Bytes 2 to 3 of word 0 contain a 16-bit value that specifies the length of the TPLS LS_ACC 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.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

c) TPLS response page(s):

Words 1:N 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 LS_ACC 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 122.

Table 122 - TPLS response page format

Item	Word	Bits
TPLS Response page	0 to 3	31 to 0
Reserved	0	31 to 16
Originator Process_Associator Validity	0	15
Responder Process_Associator Validity	0	14
Reserved	0	13 to 12
Response Code (see table 123)	0	11 to 8
Reserved	0	7 to 0
Originator Process_Associator	1	31 to 0
Responder Process_Associator	2	31 to 0
Reserved	3	31 to 1
Image Pair State	3	0

A) Word 0, Bit 15 - Originator Process_Associator validity

0 = not meaningful

1 = meaningful

B) Word 0, Bit 14 - Responder Process_Associator validity

0 = not meaningful

1 = meaningful

C) Word 0, Bits 11 to 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 123.

Table 123 - TPLS accept response code

Encoded Value Word 0, Bits 11 to 8	Description
'0000' b	Reserved
'0001' b	Request executed
'0010' b to 0110' b	Reserved
'0111' b	The destination Nx_Port is unable to process multiple page PRLI requests. The PRLI request may be retried as a single page request.
'1000' b to '1111' b	Reserved

D) Word 3, Bit 0 - Image pair state

1 = image pair established

0 = image pair not established

12.3.2.28 Fibre Channel Address Resolution Protocol Request (FARP_REQ)

12.3.2.28.1 Description

The FARP_REQ ELS shall be used to resolve Port_IDs of communicating Fibre Channel devices. A FARP_REQ may be used to retrieve a specific Nx_Port's current Port_ID given an N_Port_Name, Node_Name or IP address by requesting either a FARP_REPLY ELS (see 12.3.2.29) and/or by indicating that the responding Nx_Port shall perform Login with the requesting Nx_Port.

A FARP_REQ Match shall be considered successful at the Recipient when the FARP_REQ Request is successfully received and recognized by an Nx_Port, and the requested Match Address Code Point is successful.

NOTE The values for R_CTL, TYPE and DF_CTL for FARP_REQ are incorrect in RFC 2625 (See [12]). The values specified in this standard should be used.

12.3.2.28.2 Protocol

- a) FARP_REQ Request Sequence
- b) No Reply Sequence

12.3.2.28.3 Request Sequence

Format: FT_1

Addressing: The S_ID designates the Requestor Nx_Port requesting addressing information, The D_ID is the Broadcast Alias_ID (hex 'FF FF FF').

Payload: The format of the FARP_REQ Request Payload is shown in table 124.

Table 124 - FARP_REQ Request Payload Item

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	FARP_REQ (hex '54')	hex '00'	hex '00'	hex '00'
1	Match Address Code Points	Requesting N_Port_ID		
2	Responder Action	Responding N_Port_ID		
3	MSB	Requesting Nx_Port N_Port_Name (8 bytes)		
4				LSB
5	MSB	Requesting Nx_Port Node_Name (8 bytes)		
6				LSB
7	MSB	Responding Nx_Port N_Port_Name (8 bytes)		
8				LSB
9	MSB	Responding Nx_Port Node_Name (8 bytes)		
10				LSB
11	MSB	Requesting Nx_Port IP Address (16 bytes)		
..				
14				LSB
15	MSB	Responding Nx_Port IP Address (16 bytes)		
..				
18				LSB

a) 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. The values of these code points are defined in table 125. The Match Address Code Points indicate which of the fields shall be specified, as defined in table 126.

b) Requesting N_Port_ID:

The Requesting N_Port_ID contains the address identifier of the Nx_Port issuing the FARP_REQ.

c) 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 127.

d) Requesting Nx_Port N_Port_Name:

The Requesting Nx_Port N_Port_Name contains the N_Port_Name of the Nx_Port issuing the FARP_REQ.

e) Requesting Nx_Port Node_Name:

The Requesting Nx_Port Node_Name contains the Node_Name of the Nx_Port issuing the FARP_REQ.

f) Responding Nx_Port N_Port_Name:

The Responding Nx_Port N_Port_Name contains the N_Port_Name that the receiving Nx_Port compares with its own N_Port_Name to determine a Match.

g) Responding Nx_Port Node_Name:

The Responding Nx_Port Node_Name contains the Node_Name that the receiving Nx_Port compares with its own Node_Name to determine a Match.

h) Requesting Nx_Port IP Address:

The Requesting Nx_Port IP address contains the IP address of the Nx_Port issuing the FARP_REQ. The format of the IP address is IPv6 (See [21], IP Version 6 Addressing Architecture).

i) Responding Nx_Port IP Address:

The Responding Nx_Port IP address contains the IP address that the receiving Nx_Port compares with, its own IP address to determine a Match. The format of the IP address is IPv6 (See [21], IP Version 6 Addressing Architecture).

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

Table 125 - FARP_REQ Match Address Code Points

Value	Definition
'00000001' b	The Match is successful if all of the following are true: a) the N_Port_Name of the Nx_Port that receives the FARP_REQ equals the Responding N_Port_Name; b) ignore the Responding Node_Name; c) ignore the Responding IP address.
'00000010' b	The Match is successful if all of the following are true: a) ignore the Responding N_Port_Name; b) the Node_Name of the Nx_Port that receives the FARP_REQ equals the Responding Node_Name; c) ignore the Responding IP address.
'00000011' b	The Match is successful if all of the following are true: a) the N_Port_Name of the Nx_Port that receives the FARP_REQ equals the Responding N_Port_Name; b) the Node_Name of the Nx_Port that receives the FARP_REQ equals the Responding Node_Name; c) ignore the Responding IP address.
'00000100' b	The Match is successful if all of the following are true: a) ignore the Responding N_Port_Name; b) ignore the Responding Node_Name; c) the IP address of the Nx_Port that receives the FARP_REQ equals the Responding IP address.
'00000101' b	The Match is successful if all of the following are true: a) the N_Port_Name of the Nx_Port that receives the FARP_REQ equals the Responding N_Port_Name; b) ignore the Responding Node_Name; c) the IP address of the Nx_Port that receives the FARP_REQ equals the Responding IP address.
'00000110' b	The Match is successful if all of the following are true: a) ignore the Responding N_Port_Name; b) the Node_Name of the Nx_Port that receives the FARP_REQ equals the Responding Node_Name; c) the IP address of the Nx_Port that receives the FARP_REQ equals the Responding IP address.
'00000111' b	The Match is successful if all of the following are true: a) the N_Port_Name of the Nx_Port that receives the FARP_REQ equals the Responding N_Port_Name; b) the Node_Name of the Nx_Port that receives the FARP_REQ equals the Responding Node_Name; c) the IP address of the Nx_Port that receives the FARP_REQ equals the Responding IP address.
all others	Reserved

Table 126 - FARP_REQ Address Fields

Code Point Value	Requesting Nx_Port Port_Name	Requesting Nx_Port Node_Name	Requesting Nx_Port IP Address	Responding Nx_Port Port_Name	Responding Nx_Port Node_Name	Responding Nx_Port IP Address
'00000001' b	correct	ignored	ignored	correct	ignored	ignored
'00000010' b	ignored	correct	ignored	ignored	correct	ignored
'00000011' b	correct	correct	ignored	correct	correct	ignored
'00000100' b	ignored	ignored	correct	ignored	ignored	correct
'00000101' b	correct	ignored	correct	correct	ignored	correct
'00000110' b	ignored	correct	correct	ignored	correct	correct
'00000111' b	correct	correct	correct	correct	correct	correct
all others	Reserved					

Legend:
"correct" means the value of the field shall be correct for the Nx_Port;
"ignored" means that the value shall be either zero, or any valid value, since the field is not examined for comparison.

Table 127 - Responder Action

Value (hex)	Definition
hex '00'	Perform no action
hex '01'	If the Match is successful, the receiving Nx_Port shall perform Login with the Requesting N_Port_ID.
hex '02'	If the Match is successful, the receiving Nx_Port shall originate FARP_REPLY with the Requesting N_Port_ID. The receiving Nx_Port shall not perform Login with the Requesting N_Port_ID.
hex '03'	If the Match is successful, the receiving Nx_Port shall perform Login with the Requesting N_Port_ID; and, if the Login is successful, the receiving Nx_Port shall originate FARP_REPLY with the Requesting N_Port_ID.
all others	Reserved

12.3.2.28.4 Reply Sequence

None

12.3.2.29 Fibre Channel Address Resolution Protocol Reply (FARP_REPLY)

12.3.2.29.1 Description

A FARP_REPLY ELS communicates the information solicited by a previously received FARP_REQ ELS. A FARP_REPLY ELS shall be generated if and only if the conditions specified in Table 127 are met.

NOTE The values for R_CTL, TYPE and DF_CTL for FARP_REQ are incorrect in RFC 2625 (see [12]). The values specified in this standard should be used.

12.3.2.29.2 Protocol

- a) FARP_REPLY Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.29.3 Request Sequence

Format: FT_1

Addressing: The S_ID designates the Responding Nx_Port providing addressing information. The D_ID designates the Requesting Nx_Port requesting addressing information.

Payload: The format of the FARP_REPLY Request Payload is shown in table 128.

Table 128 - FARP_REPLY Request Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	FARP_REPLY (hex '55')	hex '00'	hex '00'	hex '00'
1	Match Address Code Points	Requesting N_Port_ID		
2	Responder Action	Responding N_Port_ID		
3	MSB	Requesting Nx_Port N_Port_Name (8 bytes)		LSB
4				
5	MSB	Requesting Nx_Port Node_Name (8 bytes)		LSB
6				
7	MSB	Responding Nx_Port N_Port_Name (8 bytes)		LSB
8				
9	MSB	Responding Nx_Port Node_Name (8 bytes)		LSB
10				
11	MSB	Requesting Nx_Port IP Address (16 bytes)		
12				
13				LSB
14				
15	MSB	Responding Nx_Port IP Address (16 bytes)		
16				
17				LSB
18				

- a) 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 125. The value of this field shall equal the value sent in the corresponding FARP_REQ Request. The Match Address Code Points indicate which of the fields shall be specified, as defined in table 129.

b) Requesting N_Port_ID:

The Requesting N_Port_ID contains the address identifier of the Nx_Port that issued the FARP_REQ.

c) Responding N_Port_ID:

The Responding N_Port_ID contains the address identifier of the Nx_Port that is responding to the FARP_REQ.

d) Responder Action:

The Responder Action field shall be set to the same value as in the corresponding FARP_REQ.

e) Requesting Nx_Port N_Port_Name:

The Requesting Nx_Port N_Port_Name contains the N_Port_Name of the Nx_Port that issued the FARP_REQ.

f) Requesting Nx_Port Node_Name:

The Requesting Nx_Port Node_Name contains the Node_Name of the Nx_Port that issued the FARP_REQ.

g) Responding Nx_Port N_Port_Name:

The Responding Nx_Port N_Port_Name contains the N_Port_Name of the Nx_Port that is responding to the FARP_REQ.

h) Responding Nx_Port Node_Name:

The Responding Nx_Port Node_Name contains the Node_Name of the Nx_Port that is responding to the FARP_REQ.

i) Requesting Nx_Port IP Address:

The Requesting Nx_Port IP address contains the IP address of the Nx_Port that issued the FARP_REQ. The format of the IP address is IPv6 (See [21], IP Version 6 Addressing Architecture).

j) Responding Nx_Port IP Address:

The Responding Nx_Port IP address contains the IP address of the Nx_Port that is responding to the FARP_REQ. The format of the IP address is IPv6 (See [21], IP Version 6 Addressing Architecture).

Table 129 - FARP_REPLY Address Fields

Code Point Value	Requesting Nx_Port Port_Name	Requesting Nx_Port Node_Name	Requesting Nx_Port IP Address	Responding Nx_Port Port_Name	Responding Nx_Port Node_Name	Responding Nx_Port IP Address
'00000001' b	REQ	REQ	REQ	correct	ignored	ignored
'00000010' b	REQ	REQ	REQ	ignored	correct	ignored
'00000011' b	REQ	REQ	REQ	correct	correct	ignored
'00000100' b	REQ	REQ	REQ	ignored	ignored	correct
'00000101' b	REQ	REQ	REQ	correct	ignored	correct
'00000110' b	REQ	REQ	REQ	ignored	correct	correct
'00000111' b	REQ	REQ	REQ	correct	correct	correct
all others	Reserved					
Legend: "correct" means the value of the field shall be correct for the Nx_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.						

12.3.2.29.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the FARP_REPLY command (see 12.3.2.29).

LS_ACC: LS_ACC signifies that the Nx_Port has transmitted the requested data. The format of the LS_ACC Payload is shown in table 130.

Table 130 - FARP_REPLY LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'

12.3.2.30 Request Node Identification Data (RNID)

12.3.2.30.1 Introduction

The RNID ELS is an ELS for acquiring Node Identification Data. The normal response shall be an Accept (ACC) ELS Sequence with Node Identification Data in its payload. If the recipient Nx_Port or FL_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 Nx_Port or FL_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 Payload containing only the Common Identification Data.

If an Nx_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 Nx_Port to retry the RNID ELS with a different Node Identification Data Format request if an Nx_Port sends an

RNID ELS and receives a reply with an Accept ELS containing only Common Identification Data (see Table 135) in its payload, the ULP usage of the Common Identification Data and any further discovery processing is beyond the scope of this standard.

N_Port Login shall not be required prior to sending the RNID ELS.

The RNID ELS may be sent to any Nx_Port or FL_Port in the Fabric. If the destination is known to be a FL_Port, then the RNID ELS shall be sent to the Fabric Controller (hex 'FF FF FD'). If an RNID ELS is sent to the Fabric Controller (hex 'FF FF FD'), the reply shall represent the fabric controller node that contains the FL_Port to which the requesting FC port is attached. If the receiving FC port has been determined to be an Nx_Port, then the RNID ELS shall be sent to that Nx_Port.

12.3.2.30.2 Protocol:

- a) Request Node-Identification (RNID) request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.30.3 Request Sequence

Addressing: The S_ID field designates the requesting source port or the Fabric Controller (hex 'FF FF FD'). The D_ID field designates the Nx_Port or FL_Port receiving the RNID request or the Fabric Controller (hex 'FF FF FD').

Format: FT_1

Payload: The format of the RNID Payload is shown in table 131.

Table 131 - RNID Payload

Bits Word	31 ... 24 Byte 0	23 ... 16 Byte 1	15 ... 08 Byte 2	07 ... 00 Byte 3
0	RNID (hex '78')	hex '00'	hex '00'	hex '00'
1	Node Identification Data Format	Reserved		

Node Identification Data Format: The format of the Node Identification Data field is shown in table 132.

Table 132 - Node Identification Data Format

Value	Description
hex '00'	Shall be set when the requesting Nx_Port does not want Specific Node Identification Data (see table 135). This allows the RNID Accept Payload to return any Specific-coded value in the Node Identification Data Format field.
hex '01' to hex 'DE'	Shall be set to the Specific-coded value when requesting data for a specific ULP (e.g., FCP-3 (See [9])). Shall use the TYPE codes defined in table 30.
hex 'DF'	Shall be used if the General Topology Discovery format (see 12.3.2.31) is to be returned in the RNID Accept Payload.
hex 'E0' to hex 'FF'	Shall be used to indicate that Specific Node Identification Data in a vendor specific format is to be returned.

12.3.2.30.4 Reply Sequence:

LS_RJT: Signifies the rejection of the RNID request

LS_ACC: Signifies acceptance of the RNID request and presents Node Identification Data Format. The format of the Accept Payload is shown in table 133.

Table 133 - RNID Accept Payload

Bits Word	31 ... 24 Byte 0	23 ... 16 Byte 1	15 ... 08 Byte 2	07 ... 00 Byte 3
0	hex '02'	hex '00'	hex '00'	hex '00'
1	Node Identification Data Format	Common Identification Data Length (0 or 16)	Reserved	Specific Identification Data Length
2	Common Identification Data (0 bytes or 16 bytes)			
...				
5				
6	Specific Identification Data (m) (0 - max. bytes)			
...				
m				

a) Node Identification Data Format:

The value of the Node Identification Data Format field is shown in table 134.

Table 134 - Node Identification Data Format

Value	Description
hex '00'	The RNID Accept Payload only contains the Common Identification Data (see Table 135).
hex '01' to hex 'DE'	The RNID Accept Payload may contain the Common Identification Data and shall contain the Specific Identification Data for the ULP that is assigned an FC-4 frame type (see table 30) equal to the value of the Node Identification Data Format from the RNID Payload (see Table 132).
hex 'DF'	The RNID Accept Payload shall contain the Common Identification Data and General Topology Discovery format Specific Identification Data.
hex 'E0' to hex 'FF'	The RNID Accept Payload may contain the Common Identification Data and shall contain vendor specific Specific Identification Data.

b) Common Identification Data Length:

When Common Node Identification Data is included in the RNID Accept payload, the Common Identification-Data length shall specify 16 bytes as the size of the Common Identification-Data field. When the ULP indicates in the Node Identification Data Format that no (see table 134) Common Node Identification Data is to be included in the RNID Accept payload, the Common Identification-Data length shall be set to zero.

c) Specific Identification Data Length:

The Specific Identification Data Length field indicates the number of bytes in the Specific Identification-Data field. The length is determined by the Node Identification Data Format requested for the respective ULP. This value shall be a multiple of 4. The minimum value of this field is zero bytes. The maximum value of this field is 252 bytes.

d) Common Identification-Data:

The format of the Common Identification Data field is shown in table 135.

Table 135 - Common Identification Data

Bits Word	31 ... 24 Byte 0	23 ... 16 Byte 1	15 ... 08 Byte 2	07 ... 00 Byte 3
0	MSB	N_Port_Name (8 bytes)		LSB
1				
2	MSB	Node_Name (8 bytes)		LSB
3				

A) N_Port_Name:

The N_Port_Name field provides the Name_Identifier (See 15.6.3 and 15.6.4) of the Nx_Port or FL_Port to which the RNID ELS was directed.

B) Node_Name:

The Node_Name field provides the Name_Identifier (See 15.6.3 and 15.6.4) of the node associated with the Nx_Port or FL_Port indicated in the N_Port_Name field.

e) Specific Identification Data:

The format of the Specific Identification Data field shall be dependent on the Node Identification Data Format field (see Table 132). If the value of the Node Identification Data Format field is set to hex 'DF' see 12.3.2.31. For all other values of the Node Identification Data Format field the standard for the related ULP defines the format, fields used, and size of the parameters.

NOTE The information in the Specific Identification Data should only be used to determine the product identity of a node. The information in the Specific Identification Data should not be used to determine the functional characteristics or the service capabilities of a node.

General Topology Discovery format: When the Node Identification Data Format field in the RNID LS_ACC Payload (see table 134) indicates the General Topology Discovery format (hex 'DF') the RNID LS_ACC payload shall contain the following:

- Node Identification Data Format field set to hex 'DF';
- Common Identification-Data length set to 16;
- Specific Identification-Data length set to 52;
- Common Node Identification Data (see table 135);

e) Specific Identification Data (see table 136).

Table 136 - General Topology Specific Identification Data

Bits Word	31 ... 24 Byte 0	23 ... 16 Byte 1	15 ... 08 Byte 2	07 ... 00 Byte 3
0	MSB			
...	Vendor Unique (16 bytes)			
3	LSB			
4	Associated Type			
5	Physical Port Number			
6	Number of Attached Nodes			
7	Node Management	IP Version	UDP/TCP Port Number	
8	MSB			
...	IP Address (16 bytes)			
11	LSB			
12	Reserved		Vendor Specific	

- A) Associated Type:
The Associated Type (see table 136) is the type of Fibre Channel functionality associated with the node of the Nx_Port or FL_Port receiving the RNID request (e.g., switch, hub, storage device).
- B) Physical Port Number:
A vendor unique value that identifies the physical port that has a Fibre Channel link attached.
- C) Number of Attached Nodes:
The number of nodes attached to the Nx_Port, Fx_Port, or L_Port receiving the RNID request. The minimum value is one. The maximum value is 126. For any Nx_Port receiving the RNID, the Number of Attached Nodes shall be set to one. For any FL_Port receiving the RNID, the Number of Attached Nodes shall be set to the number of nodes known to that FL_Port (i.e., the number of loop devices known to that FL_Port).
- D) Node Management:
The Node Management field contains the protocol by which a Node Management Entity for the responding node may be accessed (see table 139).

Table 137 - Associated Type

Value - hex	Type
hex '00 00 00 00'	Reserved
hex '00 00 00 01'	Unknown
hex '00 00 00 02'	Other (none of the following)
hex '00 00 00 03'	Hub
hex '00 00 00 04'	Switch
hex '00 00 00 05'	Gateway
hex '00 00 00 06'	Obsolete
hex '00 00 00 07'	Obsolete
hex '00 00 00 09'	Storage device (i.e., disk drive, CD-ROM drive, tape drive).
hex '00 00 00 0A'	Host
hex '00 00 00 0B'	Storage subsystem (e.g., raid, library)
hex '00 00 00 0E'	Storage Access Device (e.g., Media changer)
hex '00 00 00 11'	NAS server
hex '00 00 00 12'	Bridge
hex '00 00 00 13'	Virtualization device
hex 'xx xx xx FF'	Multi-function device (see Table 138 for values to fill in for xx xx xx)
All Others	Reserved

Table 138 - Multi-function device bit definitions

Bit position	Function
31	Hub
30	Switch
29	Gateway
28	Storage device
27	Host
26	Storage subsystem
25	Storage access device
24	Wavelength division multiplexer
23	NAS server
22	Bridge
21	Virtualization device
20 to 8	Reserved

Table 139 - Node Management

Value	Protocol	Reference
hex '00'	IP/UDP/SNMP	See [16], RFC 791; [22], RFC 2460; [15], RFC 768; [19], RFC 1157, and [20], RFC 1901
hex '01'	IP/TCP/Telnet	See [16], RFC 791; [22], RFC 2460; [17], RFC 793 and [18], RFC 854
hex '02'	IP/TCP/HTTP	See [16], RFC 791; [22], RFC 2460; [17], RFC 793, and [23], RFC 2616
hex '03'	IP/TCP/HTTPS	See [16], RFC 791; [22], RFC 2460; [17], RFC 793; and [24], RFC 2818
hex '04'	IP/TCP/HTTP/ XML	See [16], RFC 791; [22], RFC 2460; [17], RFC 793; [23], RFC 2616, CIM ^a , XML ^b CIM-XML ^c , and CIM-HTTP ^d
hex '05 to hex 'FF'	Reserved	

^a Distributed Management Task Force, *Common Information Model (CIM) in XML*, Version 2.2, June 14, 1999

^b World Wide Web consortium, *Extensible Markup Language (XML) 1.0*, Second Edition, October, 6, 2000

^c Distributed Management Task Force, *Specification for the Representation of CIM in XML (CIM-XML)*, Version 2.0, July 20, 1999

^d Distributed Management Task Force, *Specification for CIM Operations over HTTP (CIM-HTTP)*, Version 1.0, August 11, 1999

E) IP Version:

The IP versions are shown in table 140.

Table 140 - IP Version

Value	Version	Description
hex '00'	None	Indicates that this port is not providing IP support, and the Node Management, UDP Port, and IP Address fields shall be ignored.
hex '01'	IPv4 (IP version 4)	See RFC 791
hex '02'	IPv6 (IP version 6)	See RFC 2460
hex '03' to hex 'FF'	Reserved	

F) UDP/TCP Port Number:

The numerical value used in UDP (See [15], RFC 768) or TCP (See [17], RFC 793) to distinguish among multiple destinations at the same IP address.

G) IP Address:

The IP address by which a Node Management Entity for the responding node may be reached. If the IP version field is set to one, the IPv4 address shall be stored in the least significant word of the IP field and the remainder shall be set to zero.

12.3.2.31 Registered Link Incident Report (RLIR)

12.3.2.31.1 Description

The following terms are used to reference the ports, nodes, and records related to link incident reporting:

- a) incident port: The port on which the link incident occurred and that detected the link incident.
- b) connected port or connected node: The port or node connected directly to the incident port by a Fibre Channel link.
- c) reporting port: The port that sends the Link Incident Record.
- d) registered port: A port that is registered with the incident port to receive link incident reports.
- e) reporting node: The node that creates the Link Incident Record describing the incident on the incident port.

The RLIR ELS shall provide a method for a reporting Nx_Port to send a Link Incident Record to a registered Nx_Port. The normal response to an RLIR ELS Sequence shall be a LS_ACC ELS Sequence with no Payload. If the recipient Nx_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 Nx_Port is unable to 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 Nx_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" and a reason code explanation of "N_Port Login required", or a LOGO ELS.

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.

12.3.2.31.2 Link Incident reporting procedure

The incident port's reporting node shall generate one Link Incident Record for each link incident. The number of

Link Incident Records a reporting node may hold is vendor specific. 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 shall 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 a LS_ACC is received for the RLIR within 2xR_A_TOV.

If a reporting port is not able to successfully deliver a RLIR ELS to a registered port, the registered port's registration for the incident port may be discarded. If a reporting node is not able to 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 vendor specific as to whether the incident port's reporting node generates and retains or discards the Link Incident Record.

12.3.2.31.3 Protocol

- a) RLIR Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.31.4 Request 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 141.

Table 141 - RLIR Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RLIR (hex '79')	hex '00'	hex '00'	hex '00'
1	Link Incident Record Format	Common Link Incident Record Length	Common Link Incident- Descriptor Length	Specific Link Incident Record Length
2	MSB			
..	Common Link Incident Record			
m	(m) (m=4 or 16)			
	LSB			
m+1	Common Link Incident Descriptor			
	IQ	IC	EPAI (domain/area of ILS)	
m+2				
..	Specific Link Incident Record			
n	(0-max. bytes)			

The RLIR Payload fields shall be defined as follows:

- a) **Link Incident Record Format:** The format of the Link Incident Record-Format field is shown in table 142.

Table 142 - Link Incident Record Format

Description	Value
Common Link Incident Record	hex '00'
Specific-coded value	hex '01' to hex 'FF'

The Specific-coded value field of the Link Incident Record Format shall be set to the FC-4 Type code registered by the client that is being presented with the incident record (e.g., see [9], FCP-3). The Specific-coded value uses the TYPE codes FC-4 defined in Table 28. 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.

- b) **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 specify 64 bytes. The link identified by the N_Port_Name and the Node_Name shall be one end of the link for which the report is being made. The report may be passed through that link or by any other path to the node that has registered to receive RLIR ELSs.
- c) **Common Link Incident Descriptor Length:** The Common Link Incident Descriptor length shall be set to 4 bytes specifying the number of bytes in the Link Incident Descriptor field.
- d) **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.
- e) **Common Link Incident Record:** 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 143. The Common Link Incident Record may optionally contain the additional fields shown in table 143. Optional fields that have

unknown values shall be set to zero. The Common Link Incident Record Data Fields shall be defined as follows:

- A) **Incident port N_Port Name:** The Name_Identifier that is the incident port's N_Port_Name.
 B) **Incident port Node_Name:** The Name_Identifier that is the incident port's Node_Name.

Table 143 - Common Link Incident Record Data

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	MSB Incident port N_Port_Name (Mandatory) LSB			
1				
2	MSB Incident port Node_Name LSB			
3				
4	Incident Port Type (Optional)	Incident Port_ID		
5	MSB Connected port N_Port_Name LSB			
6				
7	MSB Connected port Node_Name LSB			
8				
9	MSB Fabric_Name LSB			
10				
11	Incident Physical Port number			
12	Transaction ID			
13	reserved			Time Stamp Format
14	MSB Time Stamp LSB			
15				

- C) **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.
- D) **Incident Port_ID:** The N_Port_ID of the incident port. All ports shall supply a valid Domain address (bits 23 to 16). Ports that do not have complete N_Port_ID may use zeros for the lower two bytes. A value of all zero in the Port_ID field indicates the Port_ID is unknown or unspecified.
- E) **Connected Port N_Port Name:** The Name_Identifier that is the connected port's N_Port_Name.
 a) If the incident port is an L_Port in a private loop, the connected port shall be the port associated with the link incident, if known.
 b) If the incident port is an L_Port in a public loop, the connected port shall be the FL_Port.
 c) If the incident port is an Nx_Port connected to a fabric, the connected port shall be the F_Port.
 d) If the incident port is an FL_Port, the connected port shall be the port associated with the link incident, if known.
 e) If the connected port's N_Port_Name is unknown or unspecified, the Connected Port N_Port_Name may be zero.
- F) **Connected Port Node_Name:** The Name_Identifier that is the Node_Name of the port described by the Connected Port N_Port_Name. If the connected port is an Fx_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.
- G) **Fabric_Name:** The Fabric_Name of the incident port's fabric. For Nx_Port devices this is the value obtained in the FLOGI LS_ACC Payload. Private loop devices do not have a Fabric_Name and shall report zeros.

- H) **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.
- I) **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.
- J) **Time Stamp Format:** This field specifies the format of the Time Stamp field as shown in table 144.
- K) **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.

Table 144 - Time Stamp Format values

Value	Meaning
hex '00'	The Time Stamp field is unknown or unspecified.
hex '01'	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 [6], FC-GS-4).
hex '02'	Clock synchronization format: The 64-bit time stamp is reported as defined for the Clock Synchronization Update (CSU) ELS.
hex '03' to hex 'FF'	reserved

f) **Common Link Incident Descriptor**

- A) **Incident Qualifier (IQ):** Word 8 bits 31 to 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 is defined in table 145.

Table 145 - Incident Qualifier

Bits	Meaning	
31	Reserved.	
30	Reserved.	
29	Switch: When set to one, indicates that the incident port is a port on a switch node. When set to zero, indicates that the incident port is not a port on a switch node.	
28	Expansion Port: When set to one, indicates that the switch port is an Inter-Switch-Link Expansion port (E_Port). When zero, bit 28 indicates that the switch port is not an Inter-Switch-Link Expansion port.	
27 to 26	Severity Indication: Bits 27 to 26 constitute a two-bit code that identifies the severity indication for the link incident. The codes and their meanings are as follows:	
	Code	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 that 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 that generated the Link Incident Record.
	3	Reserved.
25	Subassembly type: When set to one, specifies that the type of subassembly used for the port that is the subject of this Link Incident Record is a laser. When set to zero, specifies that the type of subassembly used for the port that is the subject of this Link Incident Record is not a laser.	
24	FRU identification: When set to one, specifies that the Specific-Link Incident Record Data is in a format that provides field-replaceable-unit (FRU) identification. When set to zero, specifies that the Specific-Link Incident Record Data is not in a format that provides field-replaceable-unit (FRU) identification.	

- B) **Incident Code (IC):** Word 8 bits 23 to 16 (byte 1 of the Common Link Incident Descriptor) contains the incident code that describes the incident that was observed by the incident node.
- a) Bit 23 is reserved.
 - b) Bits 22 to 16 is the value that specifies the type of incident that was observed. The values that shall be specified and their meanings are shown in table 146.

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.

Table 146 - Incident Code values

Value	Meaning
hex '00'	Reserved.
hex '01'	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.
hex '02'	Bit-error-rate threshold exceeded: The number of code-violation errors recognized by the incident port has exceeded a threshold (see 21.2.7).
hex '03'	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 6.1.5.2).
hex '04'	Link Failure - Not-Operational Primitive Sequence (NOS) recognized: The Not_Operational Primitive Sequence has been recognized by the incident port (see 5.5.4.2)
hex '05'	Link Failure - Primitive Sequence timeout: The incident port has recognized either a Link-Reset-Protocol timeout (see 7.6.3), or a timeout when timing for the appropriate response while in the LF1 State and after NOS is no longer recognized (see 7.4.2).
hex '06'	Link Failure - Invalid Primitive Sequence for port state: The incident port recognized either a LR or LRR Primitive Sequence while in the OL3 State (see 7.5.4).
hex '07'	Link Failure - Loop Initialization time out: The incident port failed to complete loop initialization within the normal loop time out period (See [2], FC-AL-2).
hex '08'	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.
hex '09' to hex 'FE'	Reserved.

- C) **Expansion Port address identifier (EPAI):** When word m+1 (see table 141) bits 29 and 28 (IQ field) of the Payload are both ones, indicating that the incident port is an Inter-Switch-Link Expansion port (word 0 bits 15 to 0 of the Common Link Incident Descriptor) contain the Domain and Area address identifiers (hex 'DD AA') of the connected Expansion port on this Inter-Switch-Link.
- g) **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, for example:

- A) **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.
- B) **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.
- C) **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.
- D) **Incident specific information:** When word 8 bit 24 (IQ field) of the Payload 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 vendor specific incident information.

12.3.2.31.5 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the RLIR Request

LS_ACC: LS_ACC signifies acceptance of the RLIR Request and its Link Incident Record. The format of the LS_ACC Payload is shown in table 147.

Table 147 - RLIR LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'

12.3.2.32 Link Incident Record Registration (LIRR)

12.3.2.32.1 Description

See 12.3.2.30.1 for the definition of ports, nodes, and records related to link incident reporting.

The LIRR ELS shall request the recipient to add or remove this source Nx_Port to or from the list of Nx_Port's registered to receive the Registered Link Incident Report (RLIR) ELS. The normal response to a LIRR ELS Sequence shall be a LS_ACC ELS Sequence with no Payload. If the recipient Nx_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 Nx_Port is unable to 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 an Nx_Port is registering or de-registering with the Fabric for receipt of Link Incident Records from the Fabric, the LIRR ELS shall be sent to the Management Server (hex 'FF FF FA').

12.3.2.32.2 Registration for Link Incident Records

To obtain and process Link Incident Records, a ULP shall register it's port with other ports of interest. The term 'ports of interest' refers to all ports that the registered port may address and for which the port requires their Link Incident Records.

An Nx_Port shall attempt to register as a valid recipient of subsequent Link Incident Records by sending a LIRR ELS as soon as the requesting Nx_Port has determined its address identifier and has determined the address identifiers for the ports of interest. The ULP shall indicate the format of Link Incident Records of interest that are being registered. The recipient Nx_Port shall respond with a LS_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 shall 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 a LS_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 a LS_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 vendor specific. 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.

12.3.2.32.3 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 12.3.2.31.2). 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”.

12.3.2.32.4 Protocol

- a) LIRR Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.32.5 Request 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

Bits	31 .. 24	23 .. 16	15 .. 08	07 .. 00
Word				
0	LIRR (hex '7A')	hex '00'	hex '00'	hex '00'
1	Registration Function	Link Incident Record-Registration Format	Reserved	

The LIRR Payload field shall be defined as follows:

- a) **Registration Function:** The Registration function shall specify the mode of registration (i.e., whether the recipient node adds or removes the sending Nx_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

value	Function
hex '00'	Reserved
hex '01'	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.
hex '02'	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.
hex '03' to hex 'FE'	Reserved
hex 'FF'	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).

- b) **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

Value	Description
hex '00'	Common Format
hex '01' to hex 'FF'	Specific-coded value

A Link Incident Record Registration Format of hex '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., see [9], FCP-3). The Specific-coded value uses the TYPE codes defined in FC-4 data structures of this standard. 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.

12.3.2.32.6 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the LIRR Request

LS_ACC: LS_ACC Signifies acceptance of the LIRR Request and the registration for RLIR ELSs. The format of the LS_ACC Payload is shown in table 151.

Table 151 - LIRR LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'

12.3.2.33 Get Alias_ID (GAID)

12.3.2.33.1 Description

The GAID ELS shall be sent to the Fabric Controller by the Alias Server (see clause 24) to request a unique Alias_ID to be associated with the Alias Group indicated in the passed Alias_Token.

12.3.2.33.2 Protocol

- a) GAID Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.33.3 Request Sequence

Format: FT_1

Addressing: The S_ID designates the Alias Server (hex 'FF FF F8'). The D_ID field designates the Fabric Controller (hex 'FF FF FD').

Payload: The format of the Payload is shown in table 152.

Table 152 - Get Alias_ID Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Get Alias_ID (hex '30')	hex '00'	hex '00'	hex '00'
1	MSB			
2	Alias_Token (12 bytes)			
3				
4	Alias_SP (80 bytes)			
..				
23				
24	NP_List_Length (Number of NP_List entries = n)			
25	NP_List (1)			
..	..			
24 + n	NP_List (n)			

- a) **Alias Group Token (Alias_Token):** This is the token identifying the Alias Group. See [6], FC-GS-4 for a description of the Alias_Token.
- b) **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 (see 15.6).

NOTE 1 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.

- c) **Nx_Port List Length (NP_List_Length):** The NP_List_Length specifies the number of entries in the following NP_List.

- d) **Nx_Port List (NP_List):** The NP_List contains one entry for each N_Port_ID to be included in the Alias Group. The N_Port_ID shall be right aligned within the NP_List entry (i.e., the high-order byte of the entry shall be ignored and the low-order 3 bytes shall contain the N_Port_ID. The N_Port_ID shall not be an Alias_ID).

NOTE 2 The Fabric Controller uses the NP_List to determine whether it supports an Alias Group composed of these Nx_Ports (e.g., if the Fabric does not support an Alias Group that spans Domains, and the NP_List contains Nx_Ports in multiple Domains, the request may be rejected). Alternatively, the Fabric may form an Alias Group from the Nx_Ports within a single Domain.

12.3.2.33.4 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the GAID Request. An LS_RJT shall be returned if an Alias_ID is not generated. A reason code of “Unable to perform command request” shall be generated, along with one of the reason code explanations given in table 206.

LS_ACC: 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 LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	Alias_ID			

The Alias Group Identifier field contains the Alias_ID that the Nx_Port shall recognize, in addition to its native address identifier.

12.3.2.34 Fabric Activate Alias_ID (FACT)

12.3.2.34.1 Description

The FACT ELS shall be sent to the Fabric Controller by the Alias Server (see clause 24) to cause it to assign the passed Alias_ID as an Alias_ID for the passed Nx_Ports.

12.3.2.34.2 Protocol

- FACT Request Sequence
- LS_ACC or LS_RJT Reply Sequence

12.3.2.34.3 Request Sequence

Format: FT_1

Addressing: The S_ID designates the Alias Server (hex 'FF FF F8'). The D_ID field designates the Fabric Controller (hex 'FF FF FD').

Payload: The Payload of the request contains, among other parameters a list of the Nx_Ports for which the Alias_ID is to be recognized and the Alias_ID to be recognized as an Alias_ID. The format of the Payload is shown in table 154.

Table 154 - Fabric Activate Alias_ID Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	FACT (hex '31')	hex '00'	hex '00'	hex '00'
1	Alias_ID			
2	NP_List_Length (Number of NP_List entries = n)			
3	NP_List (1)			
..	..			
2 + n	NP_List (n)			

- a) **Alias Group Identifier (Alias_ID):** This is the Alias_ID that the Fabric shall recognize for the Nx_Ports in the list, in addition to their native address identifiers.
- b) **Nx_Port List Length (NP_List_Length):** The NP_List_Length specifies the number of entries in the following NP_List.
- c) **Nx_Port List (NP_List):** The NP_List contains one entry for each Nx_Port for which the Alias_ID is to be assigned as an Alias_ID. The N_Port_ID shall be right aligned within the NP_List entry (i.e., the high-order byte of the entry shall be ignored and the low-order 3 bytes shall contain the N_Port_ID).

12.3.2.34.4 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the FACT Request. An LS_RJT shall be returned if an Alias_ID is not activated as an Alias_ID. A reason code of "Unable to perform command request".

LS_ACC: The format of the Payload for the LS_ACC indicating that this request has been successfully completed is shown in table 155.

Table 155 - Fabric Activate Alias_ID LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'

12.3.2.35 Fabric Deactivate Alias_ID (FDACT)

12.3.2.35.1 Description

The FDACT ELS shall be sent to the Fabric Controller by the Alias Server (see 24.2) to request that it deassign the indicated Alias_ID as an Alias Group identifier for the passed Nx_Ports. When there are no longer any Nx_Ports for which the Alias_ID is active, the Fabric Controller shall free up the Alias_ID for subsequent reassignment.

12.3.2.35.2 Protocol

- a) FDACT Request Sequence
b) LS_ACC or LS_RJT Reply Sequence

12.3.2.35.3 Request Sequence

Format: FT_1

Addressing: The S_ID designates the Alias Server (hex 'FF FF F8'). The D_ID field designates the Fabric

Controller (hex 'FF FF FD').

Payload: The Payload of the request contains, among other parameters, the Alias_ID to be deactivated, and the list of Nx_Ports for which this Alias_ID is to be deactivated. The format of the Payload is shown in table 156.

Table 156 - Fabric Deactivate Alias_ID Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	FACT (hex '32')	hex '00'	hex '00'	hex '00'
1	Alias_ID			
2	NP_List_Length (Number of NP_List entries = n)			
3	NP_List (1)			
..	..			
2 + n	NP_List (n)			

- a) **Alias Group ID (Alias_ID):** This is the Alias_ID to be deactivated.
- b) **Nx_Port List Length (NP_List_Length):** The NP_List_Length specifies the number of entries in the following NP_List.
- c) **Nx_Port List (NP_List):** The NP_List contains one entry for each Nx_Port for which the Alias_ID is to be deactivated as an Alias_ID. The N_Port_ID shall be right aligned within the NP_List entry (i.e., the high-order byte of the entry shall be ignored and the low-order 3 bytes shall contain the N_Port_ID).

12.3.2.35.4 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the FFACT Request. An LS_RJT shall be returned if the Alias_ID is not deactivated as an Alias_ID. A reason code of "Unable to perform command request".

LS_ACC: 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 LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'

12.3.2.36 N_Port Activate Alias_ID (NACT)

12.3.2.36.1 Description

The NACT ELS shall be sent to an Nx_Port by Alias Server (see 24.2) to cause it to assign the passed Alias_ID as an Alias_ID.

12.3.2.36.2 Protocol

- a) NACT Request Sequence
- b) LS_ACC or LS_RJT

12.3.2.36.3 Reply Sequence

Format: FT_1

Addressing: The S_ID designates the Alias Server (hex 'FF FF F8'). The D_ID field designates the Nx_Port receiving the NACT.

Payload: The Payload of the request contains, among other parameters, the Alias_Token of the Alias Group that the Nx_Port is joining, the Service Parameters associated with that Alias Group, and the Alias_ID to be assigned to the Nx_Port. The format of the Payload is shown in table 158.

Table 158 - N_Port Activate Alias_ID Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	FACT (hex '33')	hex '00'	hex '00'	hex '00'
1	MSB			
2	Alias_Token (12 bytes)			
3				
4	Alias_ID			
5				
..	Alias_SP (80 bytes)			
24				

- a) **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 Nx_Port is being requested to join. The Nx_Port validates that it supports that particular Alias Class. See [6], FC-GS-4 for a description of the Alias_Token.
- b) **Alias Group Identifier (Alias_ID):** The Alias Group Identifier field contains the Alias_ID that the Nx_Port shall recognize, in addition to its native address identifier.
- c) **Alias Group Service Parameters (Alias_SP):** The Alias_SP defines the Service Parameters to be used for all operations with this Alias Group (see 15.6).

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 used.
For Hunt Groups, all Service Parameters may be used.

The Nx_Port shall transmit a LS_ACC to notify the Alias Server if the Nx_Port:

- a) Supports the Alias_Class in the Alias_Token.
- b) Supports the Alias Group Service Parameters specified in Alias_SP.
- c) Has assigned the passed Alias_ID as an Alias_ID for this Nx_Port.

12.3.2.36.4 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the NACT Request. An LS_RJT shall be returned if the Alias_ID is unable to be activated as an Alias_ID. A reason code of "Unable to perform command request" shall be generated, along with one of the reason code explanations given in table 206.

LS_ACC: The format of the Payload for the LS_ACC indicating that this request has been successfully completed is shown in table 159. If the Nx_Port is unable to perform all of the above functions, it shall send an LS_RJT as a

reply.

Table 159 - N_Port Activate Alias_ID LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'

12.3.2.37 N_Port Deactivate Alias_ID (NDACT)

12.3.2.37.1 Description

The NDACT ELS shall be sent to an Nx_Port by Alias Server (see 24.2) to cause it to deactivate the passed Alias_ID as an Alias_ID.

12.3.2.37.2 Protocol

- NDACT Request Sequence
- LS_ACC or LS_RJT Reply Sequence

12.3.2.37.3 Request Sequence

Format: FT_1

Addressing: The S_ID designates the Alias Server (hex 'FF FF F8'). The D_ID field designates the Nx_Port receiving the NDACT.

Payload: The Payload of the request contains, among other parameters, the Alias_ID to be deactivated by the Nx_Port. The format of the Payload is shown in table 160. The Nx_Port shall transmit a LS_ACC to notify the Alias Server of a deactivation of an Alias_ID.

Table 160 - N_Port Deactivate Alias_ID Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	FACT (hex '34')	hex '00'	hex '00'	hex '00'
1	Alias_ID			

Alias Group Identifier (Alias_ID): The Alias Group Identifier field shall contain the Alias_ID that the Nx_Port no longer recognizes.

12.3.2.37.4 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the NDACT Request. An LS_RJT shall be returned if the Alias_ID is unable to be deactivated as an Alias_ID. A reason code of "Unable to perform command request" shall be generated, along with one of the reason code explanations in table 206.

LS_ACC: The format of the Payload for the LS_ACC indicating that this request has been successfully completed is shown in table 161. If the Nx_Port is unable to perform all of the above functions, it shall send an LS_RJT as a

reply.

Table 161 - N_Port Deactivate Alias_ID LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'

12.3.2.38 Quality of Service Request (QoSR)

12.3.2.38.1 Description

The QoSR ELS shall request setup of a Class 4 circuit and that the requested level of service be granted by the Quality of Service Facilitator (QoSF) for this Class 4 circuit. The N_Port requesting a Class 4 circuit is the circuit initiator (CTI), the Recipient N_Port is the circuit recipient (CTR).

A QoSR transaction is first sent from the CTI to the QoSF to request setup of a Class 4 circuit. A QoSR transaction shall be sent from the QoSF to the CTR to obtain acceptance from the CTR to setup a Class 4 circuit (see 25.3.5.2).

The CTR shall grant the setup of the Class 4 circuit to the QoSF using an LS_ACC before the Class 4 circuit setup is granted by the QoSF to the CTI using an LS_ACC (see figure 15).

12.3.2.38.2 Protocol

- QoSR Request Sequence
- LS_ACC or LS_RJT Reply Sequence

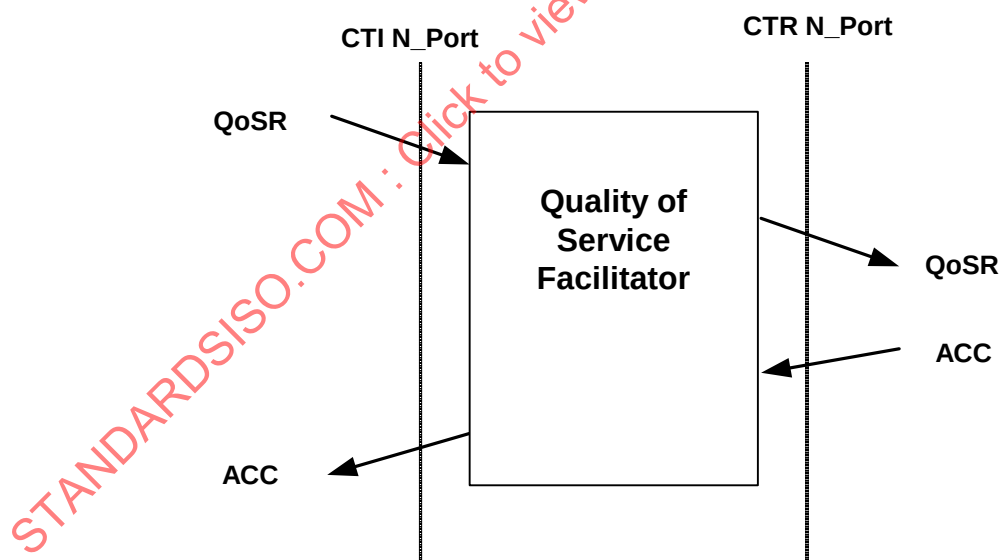


Figure 15 - QoSR Sequence flow

12.3.2.38.3 Request Sequence

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

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	QoS SR (hex '40')	hex '00'	hex '00'	hex '00'
1	Reserved			
2	CTI VC_ID	CTI address identifier		
3	CTI Maximum Bandwidth (Bytes/s)			
4	CTI Minimum Bandwidth (Bytes/s)			
5	CTI Maximum Delay (μs)			
6	CTI VC Data Field Size			
7	Reserved			
8	Reserved			
9	Reserved			
10	CTR VC_ID	CTR address identifier		
11	CTR Maximum Bandwidth (Bytes/s)			
12	CTR Minimum Bandwidth (Bytes/s)			
13	CTR Maximum Delay (μs)			
14	CTR VC Data Field Size			
15	Reserved			
16	Reserved			
17	Reserved			
18	Reserved	Live VC credit limit	Class 4 end-to-end credit	

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 CTI address identifier may be different from the S_ID in the QoS SR from the CTI. However, the CTI address identifier shall be a valid, native or Alias_ID for the CTI. The CTI address identifier shall be the address identifier used as the S_ID in all Class 4 frames sent by the CTI or the D_ID on all Class 4 frames sent by the CTR for the Class 4-circuit setup by the LS_ACC for this QoS SR. This same value shall be in the request from the QoSF to the CTR.

CTI Maximum Bandwidth (Bytes/s): The maximum bandwidth in bytes per second (Bytes/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 CTI Maximum Bandwidth limit. An N_Port may use end-to-end flow control to further limit the available bandwidth. The CTI Maximum Bandwidth field value divided by the CTI VC Data Field Size provides the maximum number of VC_RDYs that the QoSF shall send to the CTI per second.

CTI Minimum Bandwidth (Bytes/s): The minimum bandwidth in bytes per second (Bytes/s) is the minimum acceptable rate that the Fabric shall be capable of supporting from the CTI to the CTR. This same value shall be in the request from the QoSF to the CTR. The CTI Minimum Bandwidth field value divided by the CTI VC Data Field Size provides the minimum number of VC_RDYs that the QoSF shall send to the CTI per second.

CTI Maximum Delay (µs): The maximum acceptable delay in µs 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:

$$E\_D\_TOV > 2 (CIMD + CRMD)$$

where

- a) CIMD CTI maximum delay
- b) CRMD CTR maximum delay

CTR 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: The field is only meaningful in the LS_ACC for a QOSR transaction. When sent in a QOSR request from the CTI, this field shall be zero. This same value shall be in the QOSR request from the QoSF to the CTR.

CTR address identifier: The address identifier of the CTR for this Class 4 circuit. The CTR address identifier may be different from the D_ID in the QoSR from the CTI. However, the CTR address identifier shall be a valid, native or Alias_ID for the CTR. The CTR address identifier shall be the address identifier used as the D_ID in all Class 4 frames sent by the CTI or the S_ID on all Class 4 frames sent by the CTR for the Class 4-circuit setup by the LS_ACC for this QoSR. This same value shall be in the request from the QoSF to the CTR.

CTR Maximum Bandwidth (Bytes/s): The maximum bandwidth in bytes per second, 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 CTR Maximum Bandwidth limit. An N_Port may use end-to-end flow control to further limit the available bandwidth. The CTR Maximum Bandwidth field value divided by the CTR VC Data Field Size provides the maximum number of VC_RDYs that the QoSF shall send to the CTR per second.

CTR Minimum Bandwidth (Bytes/s): The minimum bandwidth in bytes per second (Bytes/s) is the minimum acceptable rate that the Fabric shall be capable of supporting from the CTR to the CTI. This same value shall be in the request from the QoSF to the CTR. The CTR Minimum Bandwidth field value divided by the CTR VC Data Field Size provides the minimum number of VC_RDYs that the QoSF shall send to the CTR per second.

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:

$$E\_D\_TOV > 2 (CIMD + CRMD)$$

where

- a) CIMD CTI maximum delay
- b) 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 VC in the Live State shall be set by the Fabric in the QoSR from the QoSF to the CTR. This field is not meaningful to the CTI and shall be set to zero.

Class 4 end-to-end Credit: For the CTR, this shall be set by the CTI and represents 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.

12.3.2.38.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the QoSR command

LS_ACC:

- a) From the CTR to the QoSF - signifies that the CTR has accepted the QoS parameters.
- b) From the QoSF - signifies that the Fabric has accepted the QoS parameters and has setup the Class 4 circuit. The QoSF shall await the LS_ACC from the CTR before it transmits the LS_ACC to the CTI.

The format of the LS_ACC Payload is shown in table 163.

Table 163 - QoS LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	Reserved			
2	CTI VC_ID	CTI address identifier		
3	CTI Maximum Bandwidth (B/s)			
4	CTI Minimum Bandwidth (B/s)			
5	CTI Maximum Delay (μs)			
6	CTI VC Data Field Size			
7	Reserved			
8	Reserved			
9	Reserved			
10	CTR VC_ID	CTR address identifier		
11	CTR Maximum Bandwidth (Bytes/s)			
12	CTR Minimum Bandwidth (Bytes/s)			
13	CTR Maximum Delay (μs)			
14	CTR VC Data Field Size			
15	Reserved			
16	Reserved			
17	Reserved			
18	Reserved	Live VC credit limit	Class 4 end-to-end credit	

CTI VC_ID: The value from the request shall be in the LS_ACC from the CTR to the QoSF and in the LS_ACC from the QoSF to the CTI.

CTI address identifier: The value from the request shall be in the LS_ACC from the CTR to the QoSF and in the LS_ACC from the QoSF to the CTI.

CTI Maximum Bandwidth (Bytes/sec): The value from the request shall be in the LS_ACC from the CTR to the QoSF and in the LS_ACC from the QoSF to the CTI.

CTI Minimum Bandwidth (Bytes/sec): The value from the request shall be in the LS_ACC from the CTR to the QoSF and in the LS_ACC from the QoSF to the CTI.

CTI Maximum Delay (μ s): The value from the request shall be in the LS_ACC from the CTR to the QoSF and in the LS_ACC from the QoSF to the CTI.

CTI VC Data Field Size: The value from the request shall be in the LS_ACC from the CTR to the QoSF and in the LS_ACC from the QoSF to the CTI.

CTR VC_ID: The field is only meaningful in the LS_ACC and shall contain the CRVC_ID for the CTR. The CTR N_Port shall choose an unassigned value in the range of hex '01' through hex 'FE'. This same value shall be in the LS_ACC from the QoSF to the CTI.

CTR address identifier: The value from the LS_ACC from the CTR shall be in the LS_ACC from the QoSF to the CTI. The value in the LS_ACC from the CTR may be different from the value in the QoSF from the QoSF. The CTR address identifier may also be different from the S_ID in the LS_ACC from the CTR. However, the CTR address identifier shall be a valid, native or Alias_ID for the CTR. The CTR address identifier shall be the address identifier used as the S_ID in all Class 4 frames sent by the CTR or the D_ID in all Class 4 frames sent by the CTI for the Class 4-circuit setup by this LS_ACC.

CTR Maximum Bandwidth (Bytes/s): The value from the request shall be in the LS_ACC from the CTR to the QoSF and in the LS_ACC from the QoSF to the CTI.

CTR Minimum Bandwidth (Bytes/s): The value from the request shall be in the LS_ACC from the CTR to the QoSF and in the LS_ACC from the QoSF to the CTI.

CTR Maximum Delay (µs): The value from the request shall be in the LS_ACC from the CTR to the QoSF and in the LS_ACC from the QoSF to the CTI.

CTR VC Data Field Size: The value from the request shall be in the LS_ACC from the CTR to the QoSF and in the LS_ACC from the QoSF to the CTI.

Live VC credit limit: The VC credit limit for the Fabric in the LS_ACC shall set the VC in the Live State in the LS_ACC from the QoSF to the CTI. This field is not meaningful to the CTR.

Class 4 end-to-end Credit: For the CTI, this shall be set by the CTR and represents 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 LS_ACC from the CTR shall be in the LS_ACC from the QoSF to the CTI.

12.3.2.39 Read Virtual Circuit Status (RVCS)

12.3.2.39.1 Description

The RVCS ELS shall request the Virtual Circuit Status in the QoS Facilitator.

12.3.2.39.2 Protocol

- a) RVCS Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.39.3 Request 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

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RVCS (hex '41')	hex '00'	hex '00'	hex '00'
1	Reserved	N_Port_ID		

12.3.2.39.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the RVCS command.

LS_ACC: LS_ACC signifies that the QoSF has completed the request. The format of the LS_ACC Payload is shown in Table 165.

Table 165 - RVCS LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	Reserved			
2	Class 4 VC Status Block (see table 166), N = number of Class 4 circuits for the designated N_Port (N x 8 bytes)			
..				
1+(N x 2)				

The LS_ACC 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

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	CTI VC_ID	CTI address identifier		
1	CTR VC_ID	CTR address identifier		

12.3.2.40 Discover N_Port/Service Parameters (PDISC)

12.3.2.40.1 Description

The PDISC ELS shall transfer Service Parameters from the initiating Nx_Port to the Nx_Port associated with the D_ID without affecting the operating environment between the two ports. This provides the means for exchange of Service Parameters without terminating open Sequences or open Exchanges.

12.3.2.40.2 Protocol

- PDISC Sequence
- LS_ACC or LS_RJT Reply Sequence

12.3.2.40.3 Request Sequence

The Payload in PDISC and the LS_ACC are, except for the LS_Command, identical to the Payloads in PLOGI (see 12.3.2.7) and the corresponding LS_ACC (see 12.3.3.2). The interchange of PDISC information shall not modify

the operating environment or Service Parameters between the two ports. Service Parameters contained in FDISC or PDISC shall be ignored and no error condition shall be reported.

12.3.2.40.4 Reply Sequence

A response shall not be sent to a PDISC if a LR or loop initialization occurs before the LS_ACC is sent. An LS_ACC for a PDISC shall be ignored if a LR or loop initialization occurred between the PDISC and the LS_ACC.

12.3.2.41 Discover F_Port Service Parameters (FDISC)

12.3.2.41.1 Description

The FDISC ELS shall transfer Service Parameters from the initiating Nx_Port to the Fx_Port at well-known F_Port_ID (i.e., hex 'FF FF FE'). This provides the means for the exchange of Service Parameters and the assignment of an additional N_Port_IDs without changing service parameters.

12.3.2.41.2 Protocol

- a) FDISC Sequence
- b) LS_ACC, LS_RJT Reply Sequence

12.3.2.41.3 Request Sequence

The Payload in FDISC and the corresponding LS_ACC are, except for the LS_Command, identical to the Payloads in FLOGI (see 12.3.2.7) and the LS_ACC (see 12.3.3.2). When the S_ID of the FDISC ELS is equal to zero, the FDISC ELS shall transfer an N_Port Name and node name, and request the assignment and login of an additional N_Port_ID. The FDISC ELS with an S_ID equal to zero shall only be sent by an N_Port with at least one N_Port_ID that is currently logged in with the Fabric, and the assignment of an additional N_Port_ID shall not modify the Service Parameters between the two ports.

When the S_ID of the FDISC ELS is set to a previously-assigned N_Port_ID, the FDISC ELS shall contain the N_Port_Name and node name corresponding to that N_Port_ID. The interchange of FDISC information shall not modify the operating environment or Service Parameters between the two ports. Service Parameters contained in FDISC shall be ignored and no error condition shall be reported.

12.3.2.41.4 Reply Sequence

When the S_ID of the FDISC ELS is zero, the D_ID field of the LS_ACC shall be set to the additional N_Port_ID being assigned. When the fabric sends the LS_ACC, it shall consider fabric login to have occurred for the additional N_Port_ID.

When the S_ID of the FDISC ELS is not zero and is currently logged in, the D_ID of the LS_ACC shall be set to the S_ID of the FDISC ELS.

When an FDISC ELS with an S_ID set to zero is sent by an N_Port when no N_Port_ID is logged in, or if the S_ID is set to a non-zero N_Port_ID that is not currently logged in, then an LS_RJT shall be returned to the S_ID of the FDISC ELS.

12.3.2.42 Discover Address (ADISC)

12.3.2.42.1 Description

The ADISC ELS shall exchange addresses and identifiers of communicating Nx_Ports.

12.3.2.42.2 Protocol

- a) ADISC Sequence

- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.42.3 Request Sequence

The ADISC Payload format is shown in table 167.

Table 167 - ADISC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	ADISC (hex '52')	hex '00'	hex '00'	hex '00'
1	Reserved	Hard Address of Originator		
2	MSB	N_Port_Name of Originator (8 bytes)		LSB
3				
4	MSB	Node_Name of Originator (8 bytes)		LSB
5				
6	Reserved	N_Port_ID of Originator		

12.3.2.42.4 Reply Sequence

The LS_ACC Payload format is shown in table 168.

A Hard Address is the 24-bit L_Port_ID that 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 that an L_Port attempts to acquire in the LIHA sequence during loop initialization in the least significant byte (See [2], FC-AL-2).

If an L_Port does not have a hard address, or if a port does not have FC-AL-2 capability, it shall report zeroes in this field.

Table 168 - ADISC LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	hex '00'	hex '00'	hex '00'
1	Reserved	Hard Address of Responder		
2	MSB	N_Port_Name of Responder (8 bytes)		LSB
3				
4	MSB	Node_Name of Responder (8 bytes)		LSB
5				
6	Reserved	N_Port_ID of Responder		

N_Port_Name in ADISC is the 8-byte N_Port_Name of the Originator and N_Port_Name in LS_ACC to ADISC is the 8-byte N_Port_Name of the Responder.

Node_Name in ADISC is the 8-byte Node_Name of the Originator and Node_Name in LS_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 LS_ACC frame (for the Responder).

A response shall not be sent to an ADISC if an LR or loop initialization occurs before the LS_ACC is sent. An LS_ACC for a ADISC shall be ignored if an LR or loop initialization occurred between the ADISC and the LS_ACC.

Table 169 summarizes the responses to FDISC, PDISC and ADISC.

Table 169 - Response summary to FDISC/PDISC

ELS command	Port Status	Nx_Port	Fx_Port	
		Class 1, 2, 3, 4	Class 1, 2, or 4	Class 3
FDISC	Logged in	LS_RJT ^a	LS_ACC	LS_ACC
	Not Logged in	LS_RJT ^a	F_RJT ^b	Discard
PDISC ^c	Logged in	LS_ACC	F_RJT ^b	Discard
	Not Logged in	LS_RJT ^a	F_RJT ^b	Discard
ADISC	Logged in	LS_ACC	F_RJT ^b	Discard
	Not Logged in	LS_RJT ^a	F_RJT ^b	Discard
^a LS_RJT may be issued with a reason code = "Unable to supply requested data". ^b F_RJT may be issued with a reason code = "Login required". ^c If an FC_Port receives a PDISC from an Nx_Port with which it has not logged in or receives an FDISC, it may choose to issue a Logout or OLS.				

12.3.2.43 Third Party Process Logout (TPRLO)

12.3.2.43.1 Description

The TPRLO ELS shall be used to invalidate the operating environments between the specified image(s) at the recipient Nx_Port and the specified image(s) in the specified Nx_Port(s) that have performed Process Login with the recipient Nx_Port for the specified TYPE.

If the recipient Nx_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 explanation of "N_Port Login required".

TPRLO has the same effect on the recipient Nx_Port as if all specified Third Party Originators performed PRLO with the recipient Nx_Port using the specified Process Associators.

12.3.2.43.2 Protocol

- a) TPRLO Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.43.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the Nx_Port initiating TPRLO. The D_ID field designates the destination Nx_Port of the TPRLO.

Payload: The TPRLO format is shown in table 170.

Table 170 - TPRLO Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	TPRLO (hex '24')	Page Length (hex'10')	Payload Length	
1				
..	Logout Parameter pages			
n	(16 - max. bytes)			

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 to 24
TYPE Code Extension	0	23 to 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 to 0
Third Party Originator Process_Associator	1	31 to 0
Responder Process_Associator	2	31 to 0
Reserved	3	31 to 24
Third Party Originator N_Port_ID	3	23 to 0

The TPRLO LS_ACC Payload format is shown table 172.

Table 172 - TPRLO LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	Page Length (hex '10')	Payload Length	
1				
..	Logout Parameter pages			
n	(16 - max. bytes)			

Third Party Originator Process Associator validity, if set to one, indicates that the Responder Process_Associator field is meaningful.

The Global bit, if set to one, specifies that all image pairs for all Nx_Ports with which Process Login has been performed be removed from the recipient Nx_Port for the specified TYPE. If the Global bit is set to one, 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 one, it frees all resources associated with the establishment of all image pairs of the specified TYPE at the recipient Nx_Port.

Third Party Originator N_Port_ID validity, if set to one, indicates that the Third Party Originator N_Port_ID field is 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 Nx_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.

12.3.2.44 Clock Synchronization Request (CSR)

12.3.2.44.1 Description

The CSR ELS shall be used to request the Clock Synchronization Server to either send or to quit sending periodic Clock Synchronization Update (CSU) ELS frames or clock synchronization primitives, depending on the method implemented. See 27.2 and 27.3.

12.3.2.44.2 Protocol

- a) Clock Synchronization Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.44.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting clock synchronization updates. The D_ID field designates either the Clock Synchronization Server (hex 'FF FF F6') or the Fabric Controller (hex 'FF FF FD').

Payload: The format of the Payload is shown in table 261.

12.3.2.44.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the CSR command.

LS_ACC: LS_ACC signifies that the Clock Synchronization Server agrees to perform the action requested in the CSR Payload. The format of the LS_ACC Payload is shown in table 263. The Clock Sync Mode item in the CSR LS_ACC Payload shall contain the value that was received in the Clock Sync Mode item of the CSR Payload.

12.3.2.45 Clock Synchronization Update (CSU)

12.3.2.45.1 Description

The CSU ELS shall be used by the Clock Synchronization Server to send its current clock value to its clients. (see 27.2.4.3)

12.3.2.45.2 Protocol

- a) Clock Synchronization Update Request Sequence
- b) No Reply Sequence

12.3.2.45.3 Request Sequence

Format: FT_1

Information Category: The Information Category value (Header word 0, bits 24 to 27) shall indicate Solicited Data ('0001' b).

Addressing: The S_ID field designates the Clock Synchronization Server well-known address (hex 'FF FF F6'). The D_ID field designates the Nx_Port(s) that is/are to receive the clock information.

Payload: The format of the Payload is shown in table 265

12.3.2.45.4 Reply Sequence

none.

12.3.2.46 Login Control List Management (LCLM)

12.3.2.46.1 Description

The LCLM ELS supports the management of a Login Control List (LCL) for Login Control. Login Control is the authentication of PLOGIs by an Nx_Port based on information stored in the Nx_Port about the originator of the PLOGI. The Nx_Port should use an LCL to determine which originators are allowed to perform a PLOGI.

This optional ELS is applicable for operation between an Originator Nx_Port and a Responder Nx_Port. Nx_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 Nx_Ports that are utilizing Login Control. It may be sent prior to PLOGI in order to perform the clear LCL command.

The LCLM ELS shall provide a method for managing the LCL in a Nx_Port. The normal response shall be a LS_ACC ELS Sequence. This LS_ACC shall indicate that the LCL modification has been received by the destination Nx_Port. If the recipient Nx_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 Nx_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 Nx_Port does not allow this Nx_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 may be sent by an Nx_Port to an Nx_Port for which it has the capability of managing the LCL. More than one Nx_Port may be allowed to manage the LCL on a given Nx_Port. This is determined by the capabilities that the first Nx_Port to send an LCL to the Nx_Port 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 LS_ACC Reply Sequence.

The mechanisms whereby the Nx_Port implements login control based on the LCLM ELS is beyond the scope of this standard.

LCLM operates by sending a command to a Nx_Port. The command shall be of the following:

- a) an addition of a Port/Node_Name to the LCL stored on the Nx_Port
- b) the deletion of a Port/Node_Name from the LCL stored on the Nx_Port

- c) a Read of the LCL stored on the Nx_Port for a given Port/Node_Name
- d) a Read of the list of Port/Node_Names in the LCL on the Nx_Port
- e) a clear of the entire LCL on the Nx_Port

The originator of any LCL Request shall be able to accept LS_RJT, reason code of “Logical busy, and a reason code explanation of “Command (request) already in progress” or “Unable to supply requested data”.

LCLM addition: An LCLM addition shall add a Port/Node_Name and the associated capabilities to the LCL. Responses may be one of the following:

- a) 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 to one, an LS_RJT shall be returned with a reason code of “Unable to perform command request” and reason code explanation of “Invalid Originator S_ID”.
- b) 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 to one and the LCL Management bit for this Port/Node_Name is set to zero, an LS_RJT shall be returned with a reason code of “Unable to perform command request” and reason code explanation of “Invalid Originator S_ID”.

LCLM deletion: An LCLM deletion shall delete any LCL record for the Port/Node_Name in the ELS. Responses may be one of the following:

- a) 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 to one, an LS_RJT shall be returned with a reason code of “Unable to perform command request” and reason code explanation of “Invalid Originator S_ID”.
- b) 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 to one and the LCL Management bit for this Port/Node_Name is set to zero, 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”.
- c) If the Port/Node_Name in the LCLM deletion is not currently in the LCL of the Nx_Port, the Nx_Port shall return the LS_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.

LCLM read: An LCLM Read shall return the global capabilities of the given Port/Node_Name for the Nx_Port that was queried. Responses may be one of the following:

- a) 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 to one, 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”.
- b) 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 to one and the LCL Management bit for this Port/Node_Name is set to zero, 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 LS_ACC shall be the same as the format for the LCLM ELS that added the Port/Node_Name to the LCL on the Nx_Port.
- c) If the Port/Node_Name is not in the LCL of the Nx_Port, the Nx_Port 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”.
- d) If the Nx_Port is unable to access the requested the data, a LS_RJT ELS Sequence with reason code of “Logical busy” and reason code explanation of “Unable to supply requested data” shall be returned.

LCLM read list: An LCLM Read List shall return a list of the Port/Node_Name pairs currently in the LCL of the Nx_Port starting at Index. The number of entries that are returned in the LS_ACC to the LCL read list is determined by the maximum frame length of the LCL LS_ACC (256 bytes). The end of the list should be determined using the value returned in Max Index. Responses may be one of the following:

- a) 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 to one and the LCL Management bit for this Port/Node_Name is set to zero, 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”.

- b) If there are no Port/Node_Names in the LCL, the Nx_Port shall return a LS_ACC with a Payload length of hex '1C' and no Port/Node_Names listed.

Clear LCL: An LCLM clear shall clear all entries in the LCL. Responses may be one of the following:

- a) 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 to one and the LCL Management bit for this Port/Node_Name is set to zero, 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".
- b) If there are no Port/Node_Names in the LCL, the Nx_Port shall return a LS_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 an Nx_Port with LCL from one configuration to another. This should only be used by an Nx_Port that has knowledge that it should be allowed to rebuild the LCL.

12.3.2.46.2 Protocol

- a) Login Control List Management (LCLM) Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.46.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the requesting source Nx_Port. The D_ID field designates the Nx_Port receiving the LCLM Request.

Payload: The format of the LCLM Payload is shown in table 173.

Table 173 - LCLM Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	LCLM (hex '25')	reserved	Payload Length (hex '1C')	
1	Reserved		LCLM command	Global Capabilities
2	MSB	N_Port_Name		
3		(8 bytes) LSB		
4	MSB	Node_Name		
5		(8 bytes) LSB		
6	Reserved		Index	

- a) **Payload length:** Bytes 2 to 3 of word 0 contain the Payload length of this ELS. This shall be set to hex '1C'.

- b) **LCLM Command:** Byte 2 of word 1 contains a value that indicates one of the LCLM commands as shown in table 174.

Table 174 - LCLM Commands

LCLM Command	Value
Read LCL entry	hex '00'
Add LCL entry	hex '01'
Delete LCL entry	hex '02'
Read LCL list	hex '03'
Clear LCL	hex '04'

- c) **Global Capabilities:** Byte 3 of Word 1 contains global capabilities. The format of the Global capabilities byte is described in table 175.

Table 175 - Global capabilities

Item	Word	Bits
Reserved	1	7 to 1
LCL Management	1	0

Bit zero is defined as LCL Management. This bit identifies that the Port/Node_Name in this LCLM may manage the LCL. If any entry in the LCL has this bit set to one, then only those entries that have this bit set to one are allowed to perform LCLM commands, except the Clear LCL command. A Management Server (hex 'FF FF FA') may use the clear LCL command regardless of the LCL management bit setting. If no entry in the LCL has this bit set to one then any Nx_Port may perform LCLM commands.

- d) **N_Port and node Name:** The values in these fields are LCLM command dependent as shown in table 176.

Table 176 - N_Port and node Name Field Values

Command	Value
Read LCL entry	N_Port and node Names for which the global capabilities are requested
Add LCL entry	N_Port and node Names to be added to the LCL
Delete LCL entry	N_Port and node Names to be added to the LCL
Read LCL list	Shall contain all zeros
Clear LCL	Shall contain all zeros

- e) **Index:** Bytes 2 to 3 of Word 6 is a 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 indicate 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'.

12.3.2.46.4 Reply sequence

LS_RJT: LS_RJT signifies rejection of the LCLM Request

LS_ACC: LS_ACC signifies successful completion of the LCLM Request. The format of the LS_ACC Payload is shown in table 177.

Table 177 - LCLM LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	reserved	Payload Length	
1	Reserved		LCLM command	Global Capabilities
2	MSB	N_Port_Name		
3		(8 bytes)		
4	MSB	Node_Name		
5		(8 bytes)		
6	List Length		Index	
7	MSB	N_Port_Name 1 in list		
8		(8 bytes)		
9	MSB	Node_Name 1 in list		
10		(8 bytes)		
..	..			
3+(n x 4)	MSB	N_Port_Name n in list		
4+(n x 4)		(8 bytes)		
5+(n x 4)	MSB	Node_Name n list		
6+(n x 4)		(8 bytes)		

a) Payload length: Bytes 2 to 3 of word 0 contain a value that specifies the length of the LCLM LS_ACC Payload. The right-most four bits shall be '1100' b. The value specified shall be greater than or equal to 28, and less than or equal to 252.

b) LCLM Command: Byte 2 of word 1 contains a value that indicates one of the following LCLM commands that this response corresponds to as defined in table 174.

c) Global Capabilities: Byte 3 of Word 1 contains the Global capabilities. The format of the Global capabilities byte is described in table 178. This is reserved and set to zero for the "read LCL list" and "clear LCL list" commands.

Table 178 - Global capabilities

Item	Word	Bits
Reserved	1	7 to 1
LCL Management	1	0

Bit zero is defined as LCL Management. This bit Identifies that the Port/Node_Name in this LCLM may manage the LCL. If any entry in the LCL has this bit set to one, then only those entries that have this bit set to one are allowed to perform LCLM commands to this Nx_Port. If no entry in the LCL has this bit set to one then any Nx_Port may perform LCLM commands.

d) List Length: Byte 0 to 1 of Word 6 is a 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.

e) Index: Byte 2 to 3 of Word 6 is a 16 bit 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 LS_ACC is reporting. The first entry is at index hex '0'. The LS_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'.

f) N_Port and node Name: The values in these fields correspond to the N_Port and node Name of the N_Port that is replying to the LCLM Request.

g) Port/Node_Name List: For the Read LCL command, Word 7 to the end of the LCLM LS_ACC Payload contains the Port/Node_Name list of the Nx_Port. The List contains a list of the Port/Node_Name pairs that are currently registered in the Nx_Port's LCL. For all other commands, word 5 of the Payload shall be the last word.

12.3.2.47 Read Port Status Block (RPS)

12.3.2.47.1 Description

The RPS ELS shall request an FC_Port to return port status associated with the port specified in the Payload of this frame. The ability to request status information for FC_Ports other than the FC_Port addressed in the frame header is useful in the absence of a management server.

12.3.2.47.2 Protocol

- a) Read Port Status Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.47.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting the Link Error Status Block. The D_ID field designates the destination Nx_Port or the Domain Controller (hex 'FF FC' || Domain_ID) to which the request is being made.

Payload: The format of the Payload is shown in table 179.

Table 179 - RPS Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RPS (hex '56')	Reserved		Flag
1	MSB	Port Selection (8 bytes)		
2				LSB

a) Flag: The values for the flag field are given in table 180. No more than one bit shall be set in the Flag field at a time. If no bits are set to one, the port selection field is ignored and the RPS Request is for the port identified in the frame header D_ID.

Table 180 - Flag field bit definitions

Value	Meaning
hex '00'	The port identified by the D_ID of the request
hex '01'	The port identified by the physical port number in the port selection field
hex '02'	The port identified by the Port_Name in the port selection field
Reserved	Other values

b) Port selection: Contains either an N_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 an Nx_Port or Fx_Port contained in a unit. Physical port numbers may be obtained using the RPL ELS, RNID ELS or by implicit means.

12.3.2.47.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the RPS command.

LS_ACC: The format of the LS_ACC Payload is shown in table 181.

Table 181 - RPS LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	Reserved		Flag
1	Reserved		Port Status	
2	MSB			
..	LESB			
7	(24 bytes)			
8	LSB			
8	MSB			
..	L_Port Extension			
15	(32 bytes)			
	LSB			

a) Flag: The values for the flag field are given in table 182.

Table 182 - Flag field bit definitions

Meaning	Bit(s)
Reserved	7 to 1
L_Port Extension Valid - When set to one, the L_Port Extension block is included in the RPS LS_ACC Payload. When set to zero, the RPS LS_ACC Payload ends with the LESB data.	0

b) **Port Status:** The values for the port status field are given in table 183.

Table 183 - Port Status

Item	Word	Bit
Reserved	1	15 to 6
point-to-point - When set to one, the port is connected point-to-point with another port.	1	5
Loop mode - When set to one, the port is in FC-AL mode.	1	4
Fabric present - When set to one, the port has detected an F_Port in a point-to-point connection or an FL_Port in loop mode.	1	3
Loss-of-Signal - When set to one, the port is not receiving an input signal. When set to zero, port is receiving an input signal or the status is not supported.	1	2
Loss-of-Synchronization - When set a one, the port is in the Loss-of-Synchronization State.	1	1
Link Reset Protocol- When set to one, the port is currently in the Link Reset Protocol.	1	0

c) **LESB:** See 21.8 for the definition of the Link Error Status Block (LESB).

d) **L_Port_Extension:** The definition of the L_Port_Extension field is given in table 184.

Table 184 - L_Port_Extension

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Reserved		L_Port Status	
1	Reserved			LIP AL_PS
2	LIP F7 Initiated Count			
3	LIP F7 Received Count			
4	LIP F8 Initiated Count			
5	LIP F8 Received Count			
6	LIP Reset Initiated Count			
7	LIP Reset Received Count			

e) L_Port_Status: The definition of the L_Port_Status field is given in table 185. If all bits in L_Port Status are zero, the port is not currently functional or does not support this status.(See [2], FC-AL-2) for definition of the terms used).

Table 185 - L_Port Status

Item	Word	Bit
OLD-PORT - When set to one, the Loop Port State Machine is in the OLD-PORT State.	0	15
MONITORING – When set to one, the Loop Port State Machine is in the MONITORING State.	0	14
ABITRATING - When set to one, the Loop Port State Machine is in the ABITRATING State.	0	13
OPEN - When set to one, the Loop Port State Machine is in the OPEN State.	0	12
OPENED - When set to one, the Loop Port State Machine is in the OPENED State.	0	11
XMITTED_CLOSE When set to one, the Loop Port State Machine is in the XMITTED_CLOSE State.	0	10
RECEIVED_CLOSE - When set to one, the Loop Port State Machine is in the RECEIVED_CLOSE State.	0	9
TRANSFER When set to one, the Loop Port State Machine is in the TRANSFER State.		8
Reserved		7 to 3
LIP(F7,xx) Loop Initialization in process - When set to one, the port is performing the Loop Initialization Process defined by FC-AL-2 initiated by a LIP(F7,xx).		2
LIP(F8,xx) Loop Initialization in process - When set to one, the port is currently performing the Loop Initialization Process defined by FC-AL-2 initiated by a LIP(F8,xx).		1
LIP(AL_PD,xx) Loop Initialization in process - When set to one, the port is currently performing the Loop Initialization Process defined by FC-AL-2 initiated by a LIP(AL_PD,xx).		0

f) LIP AL_PS: field contains the fourth character of the received LIP. A value of hex 'FF' indicates the port does not support this field.

g) LIP F7 Initiated Count: The number of loop initialization processes originated by the port with LIP (F7,xx) (Initialize LIP).

h) LIP F7 Received Count: The number of loop initialization processes initiated on the port by receiving (F7,xx) (Initialize LIP).

i) LIP F8 Initiated Count: The number of loop initialization processes originated by the port with (F8,xx) (Failure LIP).

j) LIP F8 Received Count: The number of loop initialization processes initiated on the port by receiving (F8,xx) (Failure LIP).

k) LIP Reset Initiated Count: The number of loop initialization processes originated by the port with (AL_PD,xx) (Reset LIP).

l) LIP Reset Received Count: The number of loop initialization processes initiated on the port by receiving (AL_PD,xx) (Reset LIP).

12.3.2.48 Read Port List (RPL)

12.3.2.48.1 Description

The RPL ELS 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 entries addressed by an entry number index.

12.3.2.48.2 Protocol

- a) Report Port List (RPL) Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.48.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting the port numbers information. The D_ID field designates the destination Nx_Port or the Domain Controller (hex 'FF FC' || Domain_ID) to which the request is being made.

Payload: The format of the RPL Payload is shown in table 186.

Table 186 - RPL Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RPL (hex '57')	Reserved		
1	Reserved		Maximum Size	
2	Reserved	Index		

a) Maximum Size: This field manages the size of the information returned in an RPL LS_ACC. The sender of an RPL may specify the maximum size of the RPL LS_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 shall be interpreted as shown in table 187.

Table 187 - Maximum Size values

Value	Meaning
hex '00 00'	No Maximum Size indicated; the RPL LS_ACC may be any size.
hex 'FF FF'	Reserved.
hex 'XX XX'	Any other value: The encoded value indicates the maximum number of words that shall be sent in the RPL LS_ACC, not inclusive of the command word 0.

b) 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.

12.3.2.48.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the RPL command

LS_ACC:

The RPL LS_ACC returns a list of the port number information. The format of the RPL LS_ACC Payload is shown table 188.

Table 188 - RPL LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	Reserved	Payload Length	
1	Reserved	List Length		
2	Reserved	Index		
3	MSB			
..	Port number block 1 (Each entry is 16 bytes)			
6				LSB
..				
(n x 4)-1	MSB			
..	Port number block n			
2+(n x 4)				LSB

- a) **Payload Length:** a value that specifies the length of the RPL LS_ACC Payload in bytes.
- b) **List Length:** The total number of Port Number Blocks accessible through the recipient port.
- c) **Index:** The entry number for the first Port Number Block in the LS_ACC Payload.
- d) **Port Number Blocks:** See table 189 for the contents of the Port Number Block. n is the number of Port Number Blocks in this LS_ACC.

Table 189 - Port Number Block

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Physical port number			
1	Reserved	Port_ID		
2	MSB		Port Name	
3			(8 bytes)	LSB

A Port_ID 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.

12.3.2.49 Report Port Buffer Conditions (RPBC)

12.3.2.49.1 Description

The RPBC ELS shall provide a method for a Port to report its buffer conditions. The normal response to an RPBC

ELS Sequence shall be a LS_ACC ELS Sequence. If the recipient Port does not support the RPBC ELS, it shall reply with an LS_RJT ELS Sequence with a reason code of “command not supported”.

12.3.2.49.2 Protocol

- a) Report Port Buffer Conditions (RPBC) Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.49.3 Request Sequence

Format: FT_1

Addressing: The S_ID designates the source port that is requesting port buffer conditions. The D_ID field designates the recipient that is to process the RPBC request and report buffer conditions.

Payload: The format of the RPBC Payload is shown in table 190.

Table 190 - RPBC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RPBC (hex '58')	Reserved		
1	ELS Buffer Parameters			
2	Reserved	Originator S_ID		

The ELS Buffer Parameter field contained in word 1 of the RPBC Payload relates to the ELS buffer conditions of the sender of the RPBC ELS and the contents is shown in table 191.

Table 191 - ELS Buffer Parameters Field

Bits	Field Name
31	Multi-frame ELS sequence supported
30 to 12	Reserved.
11 to 00	ELS Receive Data Field Size

The Multi-frame ELS sequence supported bit indicates that the port either supports multi-frame ELS's (bit 31=1) or does not support multi-frame ELS's (bit 31=0).

The ELS Receive Data Field Size field (word 1, bits 11 to 0) specifies the largest FT_1 ELS frame that may be received by the FC_Port responding with the RPBC LS_ACC. Values shall be a multiple of four bytes, and less than 128 or greater than 2 112 are invalid. For each class of service, the maximum usable ELS Receive Data Field Size is the lessor of the reported ELS Receive Data Field Size or the buffer-to-buffer Receive Data Field Size reported during Login.

NOTE If a 256 byte login is required when the buffer conditions of the destination port are unknown, a timeout may be avoided with the following procedure. Issue a login with the payload bit set to zero (116 byte login), followed by an RPBC ELS (if the Query Buffer Conditions bit is set), followed by a 256 byte login with the payload bit set to 1 (if both the ELS Receive Buffer Field Size field and the Buffer-to-buffer Receive Data_Field Size field are at least 256 bytes.)

12.3.2.49.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the RPBC command

LS_ACC: LS_ACC signifies acceptance of the RPBC Request and its Accept Payload. The format of the RPBC LS_ACC Payload is shown in table 192.

Table 192 - RPL LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	Reserved		
1	ELS Buffer Parameters			

The ELS Buffer Parameter field contained in byte word 1 of the RPBC LA_ACC Payload relates to the ELS buffer conditions of the sender of the RPBC_ACC ELS and the contents is shown in table 191.

12.3.2.50 Report node FC-4 Types (RNFT)

12.3.2.50.1 Description

The RNFT ELS 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 Nx_Port supports a given FC-4 protocol.

The RNFT requests a list of the FC-4 protocols supported by the responder, and the RNFT LS_ACC provides the requested list to the source of the of the RNFT.

A specific FC-4 may require that its Nx_Ports support RNFT, and therefore may conclude that a remote Nx_Port that returns LS_RJT with reason code of "Command not supported" does not support that FC-4.

If the recipient Nx_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" and a reason code explanation of "N_Port Login required".

12.3.2.50.2 Protocol

- Report node FC-4 Types (RNFT) Request sequence
- LS_ACC or LS_RJT Reply Sequence

12.3.2.50.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting the FC-4 Types information. The D_ID field designates the destination Nx_Port to which the request is being made.

Payload: The format of the request Payload is shown in table 193.

Table 193 - RNFT Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RNFT (hex '63')	Reserved	Maximum Size	
1	Reserved			Index

- Maximum Size:** Bytes 2 to 3 of Word 0 contain a 16-bit value that specifies the maximum length of the RNFT LS_ACC that the originator is able to accept. The value zero implies the RNFT LS_ACC may be any size.

b) Index: Byte 3 of Word 1 contains an 8-bit value 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 should be used to specify a subset of the entries when the entire list shall not fit into one reply. The index of the entry for a particular FC-4 Type may change at any time.

12.3.2.50.4 Reply Sequence

LS_RJT: LS_RJT signifies rejection of the RNFT command.

LS_ACC: LS_ACC Signifies that the destination Nx_Port has transmitted the requested data. The format of the LS_ACC Payload is shown in table 194.

Table 194 - RNFT LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	Reserved	Payload Length (M)	
1	Reserved	List Length	Reserved	Index
2	FC-4 Entry 1			
..				
N+1	FC-4 Entry N			

a) Payload Length: Bytes 2 to 3 of Word 0 contain a 16-bit value that specifies the length M of the RNFT LS_ACC Payload in bytes.

$$M = 8 + N \times 4$$

where

N is the number of FC-4 Entries contained in the Payload.

b) List Length: Byte 1 of Word 1 contains an 8-bit value 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.

c) Index: Byte 3 of Word 1 contains an 8-bit value that specifies the index of the first FC-4 Entry returned in the RNFT reply.

d) FC-4 Entries: The remainder of the Payload contains a list of N FC-4 Entry records.

e) FC-4 Entry: The FC-4 Entry record contains a FC-4 Entry and is shown in table 195.

Table 195 - RNFT FC-4 Entry

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
1	FC-4 Type	FC-4 Qualifier		

f) FC-4 Type: The FC-4 TYPE code of a FC-4 protocol that is supported by the sending Nx_Port. The values are defined in table 30.

g) FC-4 Qualifier: The FC-4 Qualifier may be used to distinguish between two protocols that use the same FC-4 TYPE code.

For FC-4 type codes that are reserved or assigned for specific use in this standard (hex '00' to hex '68'), the value of the FC-4 Qualifier shall be zero.

For Vendor specific 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 specific 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 specific FC-4 protocol to assure that the protocol has a unique qualified FC-4 Type.

12.3.2.51 Scan Remote Loop (SRL)

12.3.2.51.1 Description

The SRL ELS shall require a switch to scan attached loops to determine if any L_Ports have been disabled or removed. If the switch determines any L_Ports that are currently logged in with the fabric have been removed or disabled it shall update the name server and send an RSCN to all registered Nx_Ports.

The SRL Payload indicates whether the switch shall scan all attached loops or a single loop. If a single loop is to be scanned the Payload shall contain the FL_Port_ID of the loop to be scanned.

12.3.2.51.2 Protocol

- a) Scan Remote Loop Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.51.3 Request Sequence

Format: FT_1

Addressing: The S_ID field designates the source Nx_Port requesting Scan Remote Loop. The D_ID field designates the destination of the address identifier for the Domain Controller of the switch for which loops are being scanned. The format of the Domain Controller address is hex 'FF FC' || Domain_ID. Domain_ID is the Domain_ID of the switch being queried.

Payload: The format of the Payload is shown in table 196.

Table 196 - SRL Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	SRL (hex '7B')	Reserved		
1	Flag	Flag Parameter		

a) Flag: Byte 0 of word 1 indicates if the FL_Port shall be scanned, if all FL_Ports within the domain shall be scanned, or the scan period that all FL_Ports within the domain shall be scanned. The meaning of bits 0 to 7 is given in Table 197.

Table 197 - Flag field definitions

Value (hex)	Meaning	Flag Parameter
00	All the FL_Ports within the domain shall be scanned.	Ignored
01	Only the loop attached to the FL_Port addressed in the address identifier of the FL_Port field shall be scanned.	Address identifier of the FL_Port
02	Enable periodic scanning for all FL_ports.	Scan period ^a
03	Disable periodic scanning for all FL_ports.	Ignored
All Others	Reserved	
^a Scan period in seconds. If the scan period is set to zero the scan period is vendor specific. If the switch does not support this option it shall reject the SRL ELS with a reason code of Periodic Scan Not Supported. If the switch does not support the selected value it shall reject the SRL ELS with a reason code of Periodic Scan Value Not Allowed.		

b) Flag Parameter: See Table 197.

12.3.2.51.4 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the SRL Request

LS_ACC: LS_ACC signifies acceptance of the SRL Request. If the period scanning is enabled then the switch shall return the value of the periodic scanning period in the LS_ACC payloads Scan Period field, If the periodic scanning period is disabled then the switch shall set the LS_ACC payload Scan Period field to zero.

Table 198 - SRL LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	Scan Period		

12.3.2.52 Set Bit-error Reporting Parameters (SBRP)

12.3.2.52.1 Description

Set SBRP ELS shall be used to communicate a set of bit error reporting parameters to a Port or to all Ports in a particular Domain in a Fabric. There are 3 parameters, Error Interval, Error Window, and Bit-error Threshold. Error Interval is the time period over which bit error bursts are integrated to produce a single reported error. An Error Window is composed of one or more Error Intervals. The number of intervals in error occurring in an Error Window are counted and compared to a value derived from the Bit-error Threshold. If the count is equal or greater than derived value, a Registered Link Incident Report (RLIR) is generated with an Incident Code specifying Bit-error-rate threshold exceeded (see 21.2.7). At the end of the Error Window, the count is set to zero and the

process is repeated. See figure 16 for illustration of the parameters.

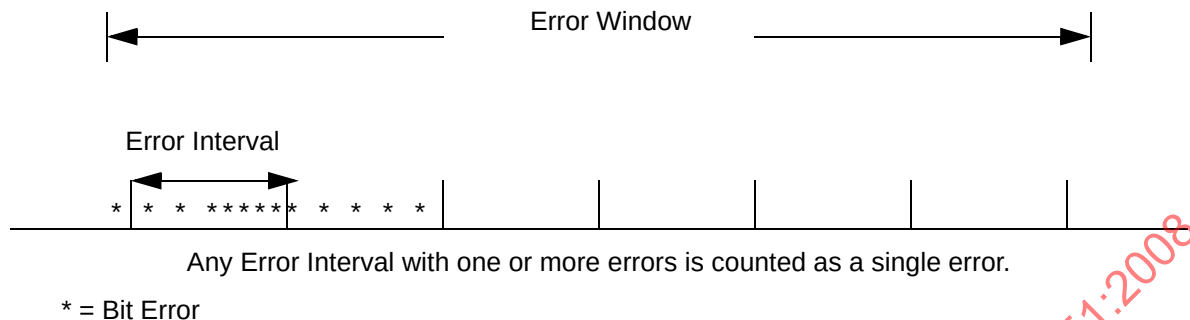


Figure 16 - Illustration of parameters

SBRP may be used to determine an acceptable set or the current set of bit error reporting parameters in the destination Port or Domain, without changing the settings.

The setting of parameters in a particular Port or Domain is done by a controlling entity that is out side the scope of this standard.

12.3.2.52.2 Protocol

- a) Set Bit-error Reporting Parameters (SBRP) Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.52.3 Request Sequence

Format: FT_1

Addressing: The S_ID designates the source Nx_Port requesting the bit error rate reporting parameters. The D_ID designates the destination Nx_Port or the Domain Controller (hex 'FF FC' || <Domain_ID>). Domain_ID is the Domain_ID of the recipient switch) to process the SB RP ELS.

Payload: The format of the SB RP Payload is shown in table 199.

Table 199 - SB RP Payload

Bits Word	31 .. 28	27 .. 24	23 .. 16	15 .. 12	11 .. 08	07 .. 03	02 .. 00
0	SBRP (hex '7C')		hex '00'	hex '00'		hex '00'	
1	Reserved						Set/ Report/ Default Flag
2	Error Window exponent	Error Window value		Error Interval exponent	Error Interval value		
3	Error Threshold						

- a) Word 1 - Set/Report/Report Default Flags

Bit 0 – Set Error Reporting Parameters: The bit is set to request that the destination set the Error Window, the Error Interval and the Bit-error Threshold parameters. If the destination is the Domain Controller then the request is for the switch to set all ports to the requested values.

Bit 1 – Report Error Reporting Parameters: The bit is set to request that the destination return the active parameters, currently being enforced.

Bit 2 – Report Default Parameters: The bit is set to request that the destination return the default parameters it is capable of enforcing. The default parameters to be used are those specified in 21.2.7 unless SBRP is supported and a SBRP is received.

- b) Word 2, Bits 31 to 28, Error Window exponent and Bits 27 to 16, Error Window value

The Error Window is a time duration described by a 16-bit value. Twelve bits, Bits 28 to 16, are used for the base value. Four bits, Bits 31 to 28 are encoded to form the exponent. The product of the base value and the exponent yields the time duration in seconds. The encoded exponent is defined as follows:

hex '0' represents 10^0

hex '1' represents 10^{-1}

hex '2' represents 10^{-2}

▪
▪
▪

hex 'F' represents 10^{-15}

(e.g., a base value of hex '001' multiplied by an exponent of hex '5' would yield a value of 1 times 10^{-5} or 10 μ s).

The Error Window has a tolerance of -0 to + 1 maximum Error Interval.

- c) Word 2, Bits 15 to 12, Error Interval exponent and Bits 11 to 0, Error Interval value

The Error Interval is a time duration in seconds and has the same, exact definitions for exponent and base value as Error Window. The Error Interval has a tolerance of – 50% to + 50%.

- d) Word 3, Bits 31 to 0, Error Threshold

The Error Threshold is the basis for a comparison value. The manner in which it is used shall be a function of the speed at which the Link is operating. The operating speed of the Port, in gigabits per second, shall be rounded to an integer number. The product of the Error Threshold value and the integer number of the speed shall be used in the comparison (e.g., if the Error Threshold is set to '00000100' b, at 1 gigabit per second the comparison value would be hex '00 00 01 00' or decimal 256; at 10 gigabits per second the comparison value would be hex '00 00 0A 00' or decimal 2560).

12.3.2.52.4 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the SBRP Request.

LS_ACC: LS_ACC signifies acceptance of the SBRP Request and presents SBRP data. The format of the LS_ACC Payload is shown in table 200.

The LS_ACC Payload conveys either:

- the current Bit Error Rate Reporting Parameters (i.e., the requestor is querying for currently set parameters) or,
- the accepted Bit Error Rate Reporting Parameters (a request to set the parameters was accepted) or,

- c) acceptable Bit Error Rate Report Parameters (a request to set the parameters was rejected because the parameters requested are not within the range supported by the port or switch).

Table 200 - SBRP LS_ACC Payload

Bits Word	31	30 .. 28	27 .. 24	23 .. 16	15 .. 12	11 .. 08	07 .. 00
0	hex '02'			hex '00'	hex '00'		hex '00'
1	SBRP Request Accepted	Reserved					
2	Error Window exponent	Error Window value			Error Interval exponent	Error Interval value	
3	Error Threshold						

a) Word 1, Bit 31, SBRP Request Accepted: When a request to set the reporting parameters is indicated (request Payload word 1 bit 0 is set to one) and the recipient accepts the requested parameters, it shall set the SBRP Request Accepted bit to zero (Accepted) and echo the requested Error Window, Interval and Threshold parameters in words 2 and 3.

When a request to set the reporting parameters is indicated (request Payload word 1 bit 0 is set to one) and the recipient does not accept the parameters as requested it shall set the SBRP Request Accepted bit to one (Rejected) and return in words 2 and 3 those parameters that were acceptable (values unenforceable by the recipient shall be set to zero).

When a request to report the parameters is indicated (request Payload word 1 bit 1 is set to one) the recipient shall set the SBRP Request Accepted bit to zero (Accepted) and report the current Bit Error Rate porting parameters being enforced. If no Bit Error Rate reporting is being enforced, words 2 and 3 shall be set to zero.

When a request to report the default parameters is indicated (request Payload word 1 bit 2 is set) the recipient shall set the SBRP Request Accepted bit to zero (Accepted) return the default Bit Error Rate porting parameters it is capable of enforcing.

12.3.2.53 Report Port Speed Capabilities (RPSC)

12.3.2.53.1 Description

The RPSC ELS shall provide a method for a Port to report its current and potential link operating speeds. The normal response to an RPSC ELS Sequence shall be a LS_ACC ELS Sequence. If the recipient Port does not support the RPSC ELS, it shall reply with an LS_RJT ELS Sequence with a reason code of "command not supported".

12.3.2.53.2 Protocol

- RPSC Request Sequence
- LS_ACC or LS_RJT Reply Sequence

12.3.2.53.3 Request Sequence

Format: FT_1

Addressing: The S_ID designates the source port that is requesting port speeds. The D_ID field designates the recipient that is to process the RPSC request. If the recipient is a Domain Controller, the request is for all the ports within that domain.

Payload: The format of the RPSC Payload is shown in table 201.

Table 201 - RPSC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	RPSC (hex '7D')	hex '00'	hex '00'	hex '00'

12.3.2.53.4 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the RPSC Request

LS_ACC: LS_ACC signifies acceptance of the RPSC Request and its Accept Payload. The format of the LS_ACC Payload is shown in table 202.

Table 202 - RPSC LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	hex '02'	Reserved	Number of entries	
1	Port Speed Capabilities (Port 1)		Port Operating Speed (Port 1)	
2	Port Speed Capabilities (Port 2)		Port Operating Speed (Port 2)	
..				
n	Port Speed Capabilities (Port n)		Port Operating Speed (Port n)	

- a) Number of Entries: Specifies the number of Port entries in the Payload.
- b) Port Speed Capabilities (bits 31 to 16): Identifies the operating speed capabilities of the port.
 - A) Bit 31 – 1 Gb capable
 - B) Bit 30 – 2 Gb capable
 - C) Bit 29 – 4 Gb capable
 - D) Bit 28 – 10 Gb capable
 - E) Bits 27 through 17 – reserved
 - F) Bit 16 - Unknown
- c) Port Operating Speed (bits 15 to 0): Identifies the current operating speed if set.
 - A) Bit 15 – 1 Gb Operation
 - B) Bit 14 – 2 Gb Operation
 - C) Bit 13 – 4 Gb Operation
 - D) Bit 12 – 10 Gb Operation
 - E) Bits 11 through 2 – reserved
 - F) Bit 1 - Unknown
 - G) Bit 0 - Speed not established.

12.3.2.54 Read Exchange Concise (REC)

12.3.2.54.1 Description

This ELS shall be used only for purposes specific to an FC-4. The REC (Read Exchange Concise) Extended Link Service requests an Nx_Port to return Exchange information for the RX_ID and OX_ID originated by the S_ID specified in the Payload of the request Sequence. The S_ID specified in the Payload of the request Sequence may differ from address identifiers of both the source and destination of the REC request itself. The specification of OX_ID and RX_ID shall be provided for the destination Nx_Port to locate the status information requested. A Responder destination Nx_Port shall use the RX_ID and verify that the OX_ID is consistent, unless the RX_ID is unassigned (i.e., RX_ID = hex 'FFFF'). If the RX_ID is unassigned in the request, the Responder shall identify the Exchange by means of the S_ID specified in the Payload of the request Sequence and OX_ID. An Originator

Nx_Port shall use the OX_ID and verify that the RX_ID is consistent.

If the destination Nx_Port of the REC request determines that the Originator S_ID, OX_ID, or RX_ID are inconsistent, then it shall reply with an LS_RJT Sequence with a reason code of “Unable to perform command request” and a reason code explanation of “Invalid OX_ID-RX_ID combination”.

The value of the Parameter field in the frame header of an REC ELS and an LS_ACC in response to an REC ELS shall be specified by the FC-4 that sends the frame. The Relative offset present bit in the frame header of an REC ELS or an LS_ACC in response to an REC ELS shall be set to zero.

12.3.2.54.2 Protocol

- a) Read Exchange Concise (REC) Request Sequence
- b) LS_ACC or LS_RJT Reply Sequence

12.3.2.54.3 Request Sequence

Format: FT_1

Addressing: The S_ID designates the source port that is requesting exchange information. The D_ID field designates the recipient that is requested to provide exchange information.

Payload: The format of the REC Payload is shown in table 203.

Table 203 - REC Payload

Bits Word	31 .. 24 Byte 0	23 .. 16 Byte 1	15 .. 08 Byte 2	07 .. 00 Byte 3
0	REC (hex '13')	hex '00'	hex '00'	hex '00'
1	Reserved	Exchange originator S_ID		
2	OX_ID		RX_ID	

12.3.2.54.4 Reply Sequence

LS_RJT: LS_RJT signifies the rejection of the REC Request

LS_ACC: LS_ACC signifies acceptance of the REC Request and returns the requested exchange information. The

format of the LS_ACC Payload is shown in table 204.

Table 204 - REC LS_ACC Payload

Bits Word	31 .. 24 Byte 0	23 .. 16 Byte 1	15 .. 08 Byte 2	07 .. 00 Byte 3
0	LS_ACC (hex '02')	hex '00'	hex '00'	hex '00'
1	OX_ID		RX_ID	
2	Reserved	Originator Address Identifier		
3	Reserved	Responder Address Identifier		
4	FC4VALUE			
5	E_STAT			

The Originator Address Identifier field shall be set to the address identifier of the Originator of the exchange about which information was requested.

The Responder Address Identifier field shall be set the address identifier of the Responder of the exchange about which information was requested.

The value of the FC4VALUE field shall be specified by the FC-4 that sends the LS_ACC to a REC ELS.

E_STAT shall be as defined in 17.8.1 for the E_STAT field in the Exchange Status Block for the exchange about which information was requested. The bits specifying whether the Exchange is complete (Bit 29) and whether the responder holds Sequence Initiative (Bit 30) shall be valid; the setting of other bits may not be valid.

12.3.3 Extended Link Service Reply Sequences

12.3.3.1 Overview

An ELS Reply Sequence shall signify that the ELS 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 ELS definition.

For any ELS 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.

12.3.3.2 LS_ACC

The Link Service Accept (LS_ACC) ELS reply Sequence shall notify the transmitter of an ELS request that the ELS 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 ELS request.

Format: FT_1

Protocol: LS_ACC is the reply Sequence for several ELSs as indicated in the applicable clause.

Addressing: The D_ID field designates the source of the ELS 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

ELS requests.

12.3.3.3 Reply Sequence

none

12.3.3.4 Link Service Reject (LS_RJT)

12.3.3.4.1 Description

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 that may be unique to a specific Link Service request (e.g., 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).

12.3.3.4.2 Payload

Addressing: The D_ID field designates the source of the ELS 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 17 and tables 205 and 206). The first error condition encountered shall be the error reported.

NOTE The applicable ELSs column in table 206 is not necessarily complete (i.e., A given reply may contain a Reason Code Explanation other than what is indicated for it in the table).

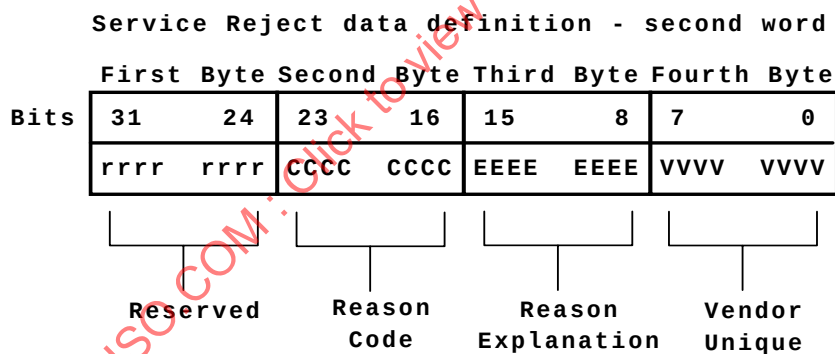


Figure 17 - LS_RJT format

Table 205 - LS_RJT Reason Codes

Encoded Value (Bits 23 to 16)	Description	Explanation
hex '01'	Invalid LS_Command code	The LS_Command code in the Sequence being rejected is invalid.
hex '03'	Logical error	The request identified by the LS_Command code and Payload content is invalid or logically inconsistent for the conditions present.
hex '05'	Logical busy	The Link Service is logically busy and unable to process the request at this time.
hex '07'	Protocol error	This indicates that an error has been detected that violates the rules of the ELS Protocol that are not specified by other error codes.
hex '09'	Unable to perform command request	The Recipient of a Link Service command is unable to perform the request at this time.
hex '0B'	Command not supported	The Recipient of a Link Service command does not support the command requested.
hex 'OE'	Command already in progress	
hex 'FF'	Vendor specific error (See bits 7 to 0)	The Vendor specific error bits may be used by Vendors to specify additional reason codes.
Others	Reserved	

12.3.3.4.3 Reply Sequence

none

Table 206 - LS_RJT Reason Code Explanation

Encoded Value (Bits 15 to 8)	Description	Applicable ELSS
hex '00'	No additional explanation	ABTX, ADVC, ESTS, FLOGI, PLOGI, LOGO, RCS, RES, REC, RLS, RSS, RTV, RSI, PRLI, PRLO, TPLS, TPRLO, GAID, FACT, FDISC, NACT, NDISC, 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, REC, RRQ, RSI, RSS
hex '17'	Invalid OX_ID-RX_ID combination	ABTX, RES, REC, RRQ, RSI, RSS
hex '19'	Command (request) already in progress	ABTX, PLOGI, RSI
hex '1E'	N_Port Login Required	ABTX, RCS, RES, REC, RSS, RSI, ESTS, ESTC, AVDC, PRLI, PRLO, TPLS, TPRLO, QoS, RVCS, PDISC, FDISC, ADISC, RNFT, RLIR, LIRR, SRL
hex '1F'	Invalid N_Port_ID	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

Table 206 - LS_RJT Reason Code Explanation (Continued)

Encoded Value (Bits 15 to 8)	Description	Applicable ELSS
hex '2A'	Unable to supply requested Data	ADVC, ESTS, RCS, RES, RLS, RSS, RTV
hex '2C'	Request not supported	ADVC, ESTS, 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_ID Type	Get Alias_ID
hex '31'	Alias_ID not activated (no resources available)	Fabric Activate Alias ID, N_Port Activate Alias ID
hex '32'	Alias_ID not activated (invalid Alias_ID)	Fabric Activate Alias ID, N_Port Activate Alias ID
hex '33'	Alias_ID not deactivated (doesn't exist)	Fabric Deactivate Alias ID, N_Port Deactivate Alias ID
hex '34'	Alias_ID not 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 cannot 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
hex '46'	Login Extension not Supported	PLOGI, FLOGI
hex '50'	Periodic Scan Value not Allowed	SRL
hex '51'	Periodic Scanning not Supported	SRL
Others	Reserved	

12.4 FC-4 Link Service

An FC-4 Link Service request solicits a destination Port (Fx_Port or Nx_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 beyond the scope of this standard. 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 (Nx_Port), transfer of Sequence Initiative, and a reply Sequence from the Responder (Nx_Port or Fx_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 Recovery_Qualifier reuse as specified in clause 17. The following rules regarding Sequence and Exchange management apply to FC-4 Link Services in addition to the rules specified in clause 17:

- a) FC-4 Link Services shall only be Exchanges originated following N_Port Login.
- b) the Originator of the Exchange shall use the Discard multiple Sequences Exchange Error Policy for all FC-4 Link Service Exchanges.
- c) 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.
- d) 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.
- e) if the Sequence Initiator aborts a Sequence using ABTS (Abort Sequence Protocol) due to receiving an ACK with the Abort Sequence bits (5 to 4) set to '01' b, the Sequence Initiator shall retry the Sequence after the Basic Accept is received for the aborted Sequence one time only.

R_CTL bits 31 to 28 (Word 0) are set = to '0011' b 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 15.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

13 Classes of service

13.1 Introduction

Five Classes of service are specified in this standard. These classes of service are distinguished primarily by the methodology with which the communication circuit is allocated and retained between the communicating Nx_Ports and by the level of delivery integrity required for an application.

A given Fabric or Nx_Port may support one or more of the following classes of service:

- a) Class 1 - Dedicated Connection
- b) Class 2 - Multiplex
- c) Class 3 - Datagram
- d) Class 4 - Fractional
- e) Class 6 - Connected Multicast

Classes 1, 2, and 3 may be supported with any of the three topologies. Classes 4 and 6 shall only be supported with the Fabric topology. Intermix is specified as an option of Class 1 and Class 6 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 19).

In all classes of service, for each frame received, the Fabric shall do one of the following:

- a) deliver only one instance of the frame to any single Nx_Port
- b) issue a F_BSY (except Class 4)
- c) issue a F_RJT
- d) discard the frame without issuing any response.

13.2 Class 1 - Dedicated connection

13.2.1 Function

Class 1 is a service that provides dedicated connections. A Class 1 Connection is opened between a pair of Nx_Ports, by the Nx_Port wishing to initiate the connection sending a frame with a SOFc1 delimiter. An acknowledgement (ACK) is returned by the other Nx_Port to establish the Connection (see clause 20). Once a Connection is established, it shall be retained and guaranteed by the Fabric, until one of participating Nx_Ports requests removal of the Connection. See 7.4 for special conditions in which a Connection may be removed.

NOTE This standard does not specify any method or configuration for establishing a connection between Nx_Ports of different speeds.

A Class 1 Connection is requested by an Nx_Port with another Nx_Port via transmission of a frame containing a SOFc1 (Class 1/SOFc1). The Fabric, if present, allocates a circuit between the requesting Nx_Port and the destination Nx_Port. The destination Nx_Port transmits an ACK indicating its acceptance to the requesting Nx_Port. Upon successful receipt of the ACK at the requesting Nx_Port, the Connection is established (see 20.4.1). The Fabric retains the allocated circuit between the two Nx_Ports, until one of the Nx_Ports requests the dedicated connection be removed.

Even if a Fabric is not present, the requesting Nx_Port and the destination Nx_Port follow the same protocol.

Class 1 Delimiters as specified in 13.2.3 are used to establish and remove the dedicated connection and to initiate and terminate one or more Sequences within the dedicated connection.

13.2.2 Rules

The following rules apply to exclusive Class 1 Connections. See 13.5 for additional rules applicable to Intermix. To provide a Class 1 Connection, the transmitting and receiving Nx_Ports, and the Fabric shall obey the following rules:

- a) Except for some Link Service Protocols (see clause 12) an Nx_Port requesting a Class 1 Connection is required to have previously logged in with the Fabric and the Nx_Ports with which it intends to communicate. To login explicitly, the requesting Nx_Port shall use Fabric and N_Port Login protocols (see clause 15).
- b) The Fabric is responsible for establishing a Connection at the request of an Nx_Port and retaining it until one of the communicating Nx_Ports explicitly requests removal of the Connection or one of the links attached to the Nx_Ports participates in a Primitive Sequence protocol (see 7.4). To establish or remove the dedicated connection, the requesting Nx_Port shall use the Class 1 Delimiters as specified in 13.2.3.
- c) After transmitting a Class 1/SOF_{c1} frame, the Nx_Port requesting the Connection shall not transmit additional frames to the destination Nx_Port, until it receives from that destination Nx_Port, an ACK that shall establish the Connection.
- d) Once a Connection is established between two Nx_Ports, each Nx_Port shall send frames only to the other Nx_Port in the Connection until the Connection is removed. All these frames shall contain respective S_ID and D_ID for these Nx_Ports.
- e) A destination Nx_Port shall acknowledge delivery of every valid Data frame with an ACK₁ or the entire Sequence with a single ACK₀.
- 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 17.3.6).
- g) Each Nx_Port of an established Connection may originate multiple Exchanges and initiate multiple Sequences within that Connection. The Nx_Port originating an Exchange shall set the OX_ID in accord with 9.11 and the Responder of the Exchange shall set the RX_ID in accord with 9.12. The Sequence Initiator shall assign a SEQ_ID, for each Sequence it initiates in accord with 17.
- h) Communicating Nx_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. All Class 1 frames except Class 1/SOF_{c1}, participate only in end to-end flow control. A Class 1/SOF_{c1} frame participates in both end-to-end and buffer-to-buffer flow controls.
- 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 Nx_Port and destined to one of the Nx_Ports engaged in the Connection (see 11.2.3.2 and 11.2.3.4).
- j) The destination Nx_Port specified in Class 1/SOF_{c1} frame may respond with a busy or a reject with a valid reason code to this frame. Once the dedicated connection is established, the destination Nx_Port shall not issue a busy but may issue a reject (see 11.2.3).
- k) The Credit established during Login by interchanging service Parameters shall be honored. At the beginning of a Connection, the End-to-end Credit_Count is re initialized to the value determined during Login. Class 1/SOF_{c1} frames shall share the Credit for connectionless service with Class 2 frames and Class 3 frames. All Class 1 frames shall share End-to-end Credit with Class 2 frames (see 18.3).
- l) The Fabric shall guarantee full bandwidth availability between the connected Nx_Ports (see 4.5).
- m) Frames within a Sequence are tracked on a Sequence_Qualifier and SEQ_CNT basis (see 9.2.1).
- n) An Nx_Port or Fx_Port shall be able to recognize SOF delimiters for all classes of service, whether or not the Nx_Port or Fx_Port supports all classes of service, and provide appropriate responses for all classes of service with appropriate delimiters.
 - A) if an Nx_Port that supports only Class 1 receives a Class 2 frame, and
 - a) is not engaged in a dedicated connection, the Nx_Port shall issue a P_RJT with appropriate Class 2 delimiters and obey buffer-to-buffer flow control rules.
 - b) is engaged in a dedicated connection, the Nx_Port response is unpredictable.
 - B) If an Nx_Port that supports only Class 1 receives a Class 3 frame, and
 - c) is not engaged in a dedicated connection, the Nx_Port shall discard the frame and obey the buffer-to-buffer flow control rules.

- d) is engaged in a dedicated connection, the Nx_Port shall discard the frame and not follow the buffer-to-buffer flow control rules.
- C) if an Fx_Port that supports only Class 1 receives a Class 2 frame, and
 - e) is not engaged in a dedicated connection, the Fx_Port shall issue a F_RJT with appropriate Class 2 delimiters and obey buffer-to-buffer flow control rules.
 - f) is engaged in a dedicated connection, the Fx_Port response is unpredictable.
- D) If an Fx_Port that supports only Class 1 receives a Class 3 frame, and
 - g) is not engaged in a dedicated connection, the Fx_Port shall discard the frame and obey buffer-to-buffer flow control rules.
 - h) is engaged in a dedicated connection, the Fx_Port response is unpredictable.
- E) If a Class 4 or Class 6 Frame is received and only Class 1 is supported, see 13.6.8 or 13.7.2 respectively.
- o) If an Nx_Port does not support Class 1 and receives a Class 1/SOF_{c1} frame, the Nx_Port shall issue a P_RJT with a SOF_{n1} and EOF_{dt} with a reason code of "Class not supported". If an Fx_Port does not support Class 1 and receives a Class 1/SOF_{c1} frame, the Fx_Port shall issue a F_RJT with a SOF_{n1} and EOF_{dt} with a reason code of "Class not supported".
- p) If an Fx_Port, not engaged in a dedicated connection, receives a frame with a SOF_{i1} or SOF_{n1}, the Fx_Port response is unpredictable. However, the buffer-to-buffer control shall remain unaffected.

13.2.3 Delimiters

A dedicated connection is requested by transmitting a Data frame using a 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. 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.

13.2.4 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 Nx_Port. Subsequent frames, after a dedicated connection is established, are limited only by the maximum Data Field size supported by the destination Nx_Port.

13.2.5 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 that do not terminate a Sequence, shall end with EOF_n.

All Class 1 frames shall follow end-to-end flow control rules (see 18.4.1). The Class 1 SOF_{c1} frame shall follow both end-to-end and buffer-to-buffer flow control rules (see 18.5.2).

13.2.6 Stacked connect-requests

Stacked connect-requests is a feature that may be provided by the Fabric (see 20.5.3 and clause 26).

13.3 Class 2 - Multiplex

13.3.1 Function

This operating environment provides connectionless service with notification of non-delivery between two Nx_Ports. This service allows one Nx_Port to transmit consecutive frames to multiple destinations without

establishing a dedicated connection with any specific Nx_Port. Conversely, this service also allows one Nx_Port to receive consecutive frames from one or more Nx_Ports without having established dedicated connections with any of them.

A Class 2 service is requested by an Nx_Port on a frame by frame basis. The Fabric, if present, routes the frame to the destination Nx_Port. If the Nx_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 13.2.3. Since Class 2 is connectionless, the question of service removal does not arise.

13.3.2 Rules

To provide Class 2 service, the transmitting and receiving Nx_Ports, and the Fabric shall obey the following rules:

- a) Except as explicitly stated in for a given Link Service protocol, an Nx_Port supporting Class 2 service is required to have logged in with the Fabric and the Nx_Ports with which it intends to communicate, either explicitly or implicitly. To login explicitly, the requesting Nx_Port shall use Fabric and N_Port Login protocols.
- b) The Fabric routes the frames without establishing a dedicated connection between communicating Nx_Ports. To obtain Class 2 service from the Fabric, the Nx_Port shall use the Class 2 Delimiters as specified in 13.3.3.
- c) An Nx_Port may send consecutive frames to one or more destinations. This enables an Nx_Port to demultiplex multiple Sequences to a single or multiple destinations concurrently (see 13.3.3).
- d) A given Nx_Port may receive consecutive frames from different sources. Each source may send consecutive frames for one or more Sequences.
- e) A destination Nx_Port shall provide an acknowledgement to the source for each valid Data frame received. As in Class 1, the destination Nx_Port shall use ACK for the acknowledgement (see 13.3.5). If unable to deliver ACK, the Fabric shall return a F_BSY or F_RJT.
- 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 17.3.6).
- g) An Nx_Port may originate multiple Exchanges and initiate multiple Sequences with one or more destination Nx_Ports. The Nx_Port originating an Exchange shall set the OX_ID in accord with 9.11 and the Responder of the Exchange shall set the RX_ID in accord with 9.12. The Sequence Initiator shall assign a SEQ_ID, for each Sequence it initiates in accord with 17.6.2.
- h) Each Fx_Port (the local and the remote) exercises buffer-to-buffer flow control with the Nx_Port to which it is directly attached. Communicating Nx_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.
- i) If the Fabric is unable to deliver the frame to the destination Nx_Port, 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 Nx_Port by P_BSY or P_RJT.
- j) A busy or reject may be issued by an Fx_Port or the destination Nx_Port with a valid reason code. (see 11.2)
- 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.
- l) The Credit established during Login by interchanging Service Parameters shall be honored (see 18.3 for more on Credit). Class 2 may share the Credit for connectionless service with Class 3 and Class 1/SOFc1 frames.
- m) Effective transfer rate between any given Nx_Port pair is dependent upon the number of Nx_Ports a given Nx_Port is demultiplexing to and multiplexing from.
- n) Frames within a Sequence are tracked on a Sequence_Qualifier and SEQ_CNT basis (see 9.2.1).
- o) An FC_Port shall be able to recognize SOF delimiters for all classes of service, whether or not the FC_Port supports all classes of service, and provide appropriate responses for all classes of service with appropriate delimiters. An Nx_Port that supports only Class 2 shall issue a P_RJT for Class 1 and Class 6 SOFc1 frames with appropriate Class 1 delimiters and discard Class 3 frames, while obeying the

buffer-to-buffer flow control, rules in both cases. An Fx_Port that supports only Class 2 shall issue a F_RJT for Class 1 and Class 6 SOF_{c1} frames with appropriate Class 1 delimiters and discard Class 3 frames, while obeying the buffer-to-buffer flow control, rules in both cases. If a Class 4 or Class 6 frame is received and only Class 2 is supported, see 13.6.8 or 13.7.2, respectively.

- p) The Class 2 PREF field is a Class of Service specific use of the CS_CTL field. When PREF is set to zero, the Fabric shall deliver the frame normally. When PREF is set to one, the Fabric may deliver the frame to the destination Nx_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.

13.3.3 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.

13.3.4 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 Nx_Port. Each frame is routed through the Fabric, if present, as a separate entity.

13.3.5 Flow control

Class 2 service uses both buffer-to-buffer and end-to-end flow controls. R_RDY is used for buffer-to-buffer flow control. R_RDY is transmitted by the Fx_Port to the Nx_Port originating the Class 2 frame to indicate that a buffer is available for further frame reception at the Fx_Port. R_RDY is transmitted by the destination Nx_Port to the attached Fx_Port to indicate that a buffer is available for further frame reception at the destination Nx_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 18.5.2).

13.4 Class 3 - Datagram

13.4.1 Function

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 communicating Nx_Ports. The Fabric, if present, and the destination Nx_Port are allowed to discard Class 3 frames without any notification to the transmitting Nx_Port. This service allows one Nx_Port to transmit consecutive frames to multiple destinations without establishing a dedicated connection with any specific Nx_Port. Conversely, this service also allows one Nx_Port to receive consecutive frames from one or more Nx_Ports without having established dedicated connections with any of them.

A Class 3 service is requested by an Nx_Port on a frame by frame basis. The Fabric, if present, routes the frame to the destination Nx_Port. If the Nx_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 13.4.3 Since Class 3 is connectionless, the question of service removal does not arise.

13.4.2 Rules

To provide Class 3 service, the transmitting and receiving Nx_Ports, and the Fabric shall obey the following rules:

- a) Except as explicitly stated in for a given Link Service protocol, an Nx_Port supporting Class 3 service is required to have logged in with the Fabric or the Nx_Ports, either explicitly or implicitly. To login explicitly, the requesting Nx_Port shall use Fabric and N_Port Login protocols (see clause 15).
- b) The Fabric routes the frames without establishing a dedicated connection between communicating Nx_Ports. To obtain Class 3 service from the Fabric, the Nx_Port shall use the Class 3 Delimiters as specified in the Class 3 Datagram Rules (see clause 13.4.3).
- c) A given Nx_Port may send consecutive frames to one or more destinations. This enables an Nx_Port to demultiplex multiple Sequences to single or multiple destinations concurrently.
- d) A given Nx_Port may receive consecutive frames from one or more source Nx_Ports. Each source Nx_Port may send consecutive frames for one or more Sequences.
- e) A destination Nx_Port shall not provide acknowledgement (ACK) to the source for any valid frame received.
- 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 17.3.6).
- g) An Nx_Port may originate Exchanges and initiate Sequences with one or more destination Nx_Ports. The Nx_Port originating an Exchange shall set the OX_ID in accord with 9.11 and the Responder of the Exchange shall set the RX_ID in accord with 9.12. The Responder may assign an RX_ID in the first Sequence it transmits. The Sequence Initiator shall assign a SEQ_ID for each Sequence it initiates in accord with 17.6.2.
- h) The local Fx_Port exercises buffer-to-buffer flow control with the transmitting Nx_Port. The remote Fx_Port exercises buffer to-buffer flow control with the receiving Nx_Port. R_RDY is used for buffer-to-buffer flow control.
- i) If the Fabric is unable to deliver the frame to the destination Nx_Port, the frame is discarded and the source is not notified. If the destination Nx_Port is unable to receive the frame, the frame is discarded and the source is not notified.
- j) The buffer-to-buffer Credit is used for buffer-to-buffer flow control. End-to-end Credit is not used (see 18.3). Class 3 may share the Credit for connectionless service with Class 2 and Class 1/SOF_{c1} frames.
- k) Effective transfer rate between any given Nx_Port pair is dependent upon the number of Nx_Ports a given Nx_Port is demultiplexing to and multiplexing from.
- l) Neither the Fx_Port nor Nx_Port shall issue busy or reject to Class 3 frames.
- m) Frames within a Sequence are tracked on a Sequence_Qualifier and SEQ_CNT basis (see 9.2.1).
- n) An Nx_Port or Fx_Port shall be able to recognize SOF delimiters of all classes of service, whether or not the Nx_Port or Fx_Port supports all classes of service, and provide appropriate responses for all classes of service with appropriate delimiters. An Nx_Port that supports only Class 3, shall issue a P_RJT for Class 1 and Class 6 SOF_{c1} frames or Class 2 frames with appropriate Class 1, Class 6, or Class 2 delimiters respectively while obeying the buffer-to-buffer flow control rules. An Fx_Port that supports only Class 3 shall issue a F_RJT for Class 1 and Class 6 SOF_{c1} frames or Class 2 frames with appropriate Class 1, Class 6, or Class 2 delimiters respectively, while obeying the buffer-to-buffer flow control rules. If a Class 4 or Class 6 frame is received and only Class 3 is supported, see 13.6.8 or 13.7.2, respectively.
- o) An Nx_Port may obtain the delivery status of Class 3 Sequences transferred by using Abort Sequence protocol (see 21.7.2.2) and thus verify the integrity of the delivered Sequences.
- p) The Class 3 PREF field is a class specific use of the CS_CTL field. When PREF is set to zero, the Fabric shall deliver the frame normally. When PREF is set to one, the Fabric may deliver the frame to the destination Nx_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.

13.4.3 Delimiters

Sequences are initiated by transmitting a frame started by a SOF_{i3}. A SOF_{n3} starts subsequent frames within a Sequence. A Sequence is terminated with a Data frame ended by EOF_t. An EOF_n terminates all frames other than the last frame within the Sequence.

13.4.4 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 Nx_Port. Each frame is routed through the Fabric, if present, as a separate entity.

13.4.5 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 18.5.2).

13.4.6 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 has no method to communicate it to the Sequence Initiator due to absence of ACK in Class 3. However, using Abort Sequence protocol (see 17.3.11 and 21.7), the Sequence Initiator may 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.

13.5 Intermix

13.5.1 Introduction

Intermix is an option of Class 1 and Class 6 services that allows interleaving of Class 2 and Class 3 frames during an established Class 1 Connection (Class 6 uses a Class 1 Connection) between two N_Ports. During a Class 1 Connection, an 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. An N_Port that supports Intermix shall be capable of both transmitting and receiving intermixed frames.

In a point-to-point topology, both interconnected N_Ports shall be required to support Intermix if Intermix is to be used. In the presence of a Fabric, both the N_Port and the Fabric shall be required to support Intermix if Intermix is to be used in that link. Fabric support for Intermix requires that the full Class 1 bandwidth during a dedicated connection be maintained.

NOTE Intermix permits the potentially unused Class 1 bandwidth to be available for transmission of Class 2 and Class 3 frames.

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 which of the following is used to manage the Class 2 or Class 3 frame that is in transit:

- a) 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
- b) if a Class 2 or Class 3 frame is being aborted (EOFa) on receipt of a Class 1 frame, the Class 1 frame shall be buffered to the extent of the time required to append an EOFa to the Class 2 or 3 frame in transit.

13.5.2 Rules

An 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 an N_Port shall provide the Intermix Service Parameter during Login to indicate its Intermix capability to the communicating Port.
- b) If the Fabric supports Intermix, an N_Port supporting Intermix may 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.
- 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, provided the Fabric supports Intermix.
- d) If the Fabric that supports Intermix receives a Class 2 or Class 3 frame destined to an 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.
- e) If the Fabric that does not support Intermix receives a Class 2 or Class 3 frame from a third N_Port destined to an 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 source of the Class 2 frame after determining that the frame is not deliverable. The Fabric shall discard the Class 3 frame without any response to the sender.
- f) If an 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.
- 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.
- h) Class 1 frames have priority over Class 2 and Class 3 frames. Class 2 and Class 3 frames may be inter-mixed 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.
- 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.
- j) In a point-to-point topology, an N_Port that 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.

13.5.3 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.

13.5.4 Flow control

The flow control for each class is governed separately by individual class. Intermix does not impose any additional rules on flow control.

13.6 Class 4 - Fractional

13.6.1 Function

Class 4 (see clause 25) 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.

13.6.2 Procedures

The following procedures specify the behavior for the various phases of a Class 4 circuit.

The flow chart in figure 18 outlines the procedures involved in Class 4 circuit management.

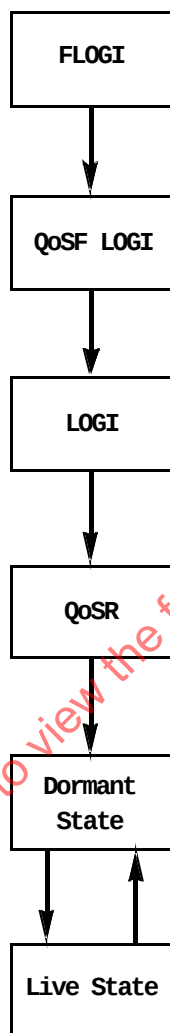


Figure 18 - Class 4 circuit – Management

- a) FLOGI - Each N_Port shall perform Fabric login indicating support for Class 4.
- b) Each N_Port shall perform N_Port Login with the QoSF.
- c) PLOGI - 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.
- d) One of the communicating N_Ports shall start the QoSR link service request with the QoSF. When successful, the Class 4 circuit is placed in the Dormant State (see 25.3.5).
- e) 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 communicating N_Ports (see 25.3.5.3). The Class 4 circuit may also be removed by the Fabric (see 25.3.5.5).

- f) While in the Live State, normal interaction between the communicating N_Ports takes place across the Class 4 circuit, moderated by the Fabric.
- g) When data transfers are complete or are to be suspended, the Class 4 circuit may be moved back to the Dormant State (see 25.3.5.4).

13.6.3 Login

Before any operations may 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 shall then perform N_Port Login with each other. N_Ports shall Login with the QoS_F prior to any QoS_R by the N_Port.

13.6.4 Circuit Setup

An N_Port requests setup of a Class 4 circuit to another N_Port by sending a Quality of Service request (QoS_R) ELS command to the QoS_F. The requesting N_Port becomes the circuit initiator (CTI) and the target N_Port the circuit recipient (CTR). The QoS_R identifies the communicating N_Ports and the required QoS parameters.

The QoS_R is granted if the QoS_F establishes a path between the communicating N_Ports with the requested QoS parameters and if the CTR also accepts the QoS parameters. If the QoS_R is accepted then a Class 4 circuit is established in the Pending State (see 25.3.5) when the LS_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 is able to establish a path between the communicating N_Ports with the requested QoS parameters but the CTR is unable to accept the QoS_R.

13.6.5 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_Ports.

A Class 4 circuit in the Dormant State is activated and enters the Live State when a SOF_{c4} delimited frame is sent or received.

While in the Live State, the Fabric monitors bandwidth utilization, 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.

13.6.6 Circuit Deactivation

A Class 4 circuit, in the Live State, is deactivated, and enters the Dormant State, when an EOF_{dt} or EOF_{dti} 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 EOF_{dt} and SOF_{n4}. The Fabric shall use this mechanism to deactivate Class 4 circuits when the N_Port or Fx_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).

13.6.7 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 Nonexistent State, when an EOF_{rt} or EOF_{rti} 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_F 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, 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 EOF_{rt} and SOF_{n4}. If the Class 4 circuit is alive, then the Fabric shall first deactivate the Class 4 circuit before removing the circuit.

13.6.8 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. 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 permitted Fabric exceptions occur. Permitted 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 25.5 An N_Port shall only send a SOF_{c4} 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 25.5 The Class 4 circuit may be re-activated at a later time by using the procedure in 25.3.5.3.
- e) To remove a Class 4 circuit, the requesting N_Port shall use the Class 4 delimiters as specified in 25.5.
- f) A destination N_Port shall acknowledge delivery of each valid Data frame.
- 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. The Nx_Port originating an Exchange shall set the OX_ID in accord with 9.11 and the Responder of the Exchange shall set the RX_ID in accord with 9.12. The Sequence Initiator shall assign a SEQ_ID, for each Sequence it initiates in accord with 17.6.2.
- 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. All Class 4 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 an N_Port transmits frames. 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 13.6.13).
- 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 (see 25.3.5).
- s) The N_Port shall discard all SOFi4 and SOFn4 delimited frames received on a Class 4 circuit in the Dormant State (see 25.3.5).
- t) An Fx_Port shall discard all SOFi4 and SOFn4 delimited frames received unless the Class 4 circuit is in the Live State (see 25.3.5).
- u) EE_C4_Credit shall remain unaffected by any discarded Class 4 frames.
- v) If VC_Credit is equal to zero 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 LS_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 Fx_Port, not supporting Class 4, receives a Class 4 circuit activation frame, the Fx_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.

13.6.9 Interleaving other classes of service in 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.

During Class 4 operation, Class 2 and Class 3 frames use R_RDY flow control, and Class 4 frames use VC_RDY flow control.

13.6.10 Class 4 delimiters

A Class 4 circuit in the Dormant State is activated to Live State by using a 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 a SOFi4 delimiter.

All other frames for a Class 4 circuit in the Live State shall begin with a 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.

13.6.11 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 ELS 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.

13.6.12 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 an N_Ports receiver. A small EE_C4_Credit value helps to decrease peak demands.

13.6.13 Buffer-to-buffer flow control

Summary rules for VC credit management:

- a) N_Ports shall track available credit independently for each VC.
- b) Maximum available credit (VC_Credit) on a VC shall be limited to a value of less than 256 ($VC\_Credit\_Limit < 256$).
- c) An N_Port shall not transmit any Class 4 frames on a given VC unless VC_Credit is available for the VC.
- d) One credit shall be consumed for every Class 4 frame transmitted on a VC ($VC\_Credit \leftarrow VC\_Credit - 1$).
- e) 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 1 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 other classes of service.

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 follows:

- a) For the CTR, to the value of the Live VC credit limit field in the QoS SR, if the VC is alive.
- b) For the CTI, to the value of the Live VC credit limit field in the LS_ACC frame to the QoS SR, if the VC is alive.
- c) Maximum available credit shall be zero for nonexisting VC or VCs in the Pending State (VC_Credit_Limit = 0).

NOTE 2 The CTI and the CTR start with an available VC_Credit of 0 when a new VC is setup.

- d) Maximum available credit shall be one for VCs in the Dormant State (VC_Credit_Limit = 1).
- e) Maximum available credit shall be set to the Live_VC_Credit_Limit for VCs in the Live State.
- f) Available VC_Credit shall be zero 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 outbound 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 an N_Port should be used to provide congestion control for the Fabric by keeping unwanted frames in the N_Ports until the fabric is able to meet the QoS parameters.

13.7 Class 6 - Connected Multicast

13.7.1 Function

Class 6 is a service that provides dedicated connections for Reliable multicast (see 23.3) for a Fabric topology. An N_Port for one or more destination N_Ports requests a Class 6 connection. An acknowledgment is returned by the destination N_Ports to a multicast server that 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.

A Class 6 connection is requested by an N_Port to one or more destination N_Ports via transmission of a frame containing a SOF_{c1}. A Class 6 SOF_{c1} is identical to a Class 1 SOF_{c1}. 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 that 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 13.2.3, 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.

13.7.2 Rules

The following rules apply to exclusive Class 6 Connections. 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, an 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 an N_Port and retaining it until the requesting N_Port explicitly requests removal of the Connection by setting the End_Connection bit in the F_CTL field (bit 18), 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 13.2.3.
- c) After transmitting a Class 6/SOF_{c1} 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 that 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.
- 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 17.3.6).
- g) The Connection Initiator N_Port of a Class 6 Connection shall be the multicast Source Nx_Port. The multicast Source Nx_Port may originate multiple Exchanges and initiate multiple Sequences within that Connection. The multicast Source Nx_Port shall set the OX_ID in accord with 9.11. RX_ID is not used in Class 6 and shall be set to hex 'FF FF' for all Class 6 frames. The Sequence Initiator shall be the multicast Source. It shall assign a SEQ_ID, for each Sequence it initiates in accord with 17.6.2.
- 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. All Class 6 frames, except Class 6/SOF_{c1}, participate only in end-to-end flow control. A Class 6/SOF_{c1} 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 11.2.3.2 and 11.2.3.4) except if intermix is enabled.
- j) The destination N_Ports specified in the Class 6/SOF_{c1} 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 23.3.4). Once the dedicated connection is established, the destination N_Ports shall not issue a busy but may issue a reject (see 11.2.3).
- k) The End-to-end Credit established during login by interchanging service Parameters shall be honored (see 18.3). 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 re initialized to the value determined during Login. A Class 6/SOF_{c1} frame shall share the End-to-end Credit with Class 2 and Class 3 frames.
- l) The Fabric shall guarantee full bandwidth availability to the connected N_Ports.
- m) Frames within a Sequence are tracked on a Sequence_Qualifier and SEQ_CNT basis (see 9.2.1).
- n) An N_Port or F_Port shall be able to recognize SOF delimiters for all classes of service, whether or not the N_Port or F_Port supports all classes of service, and provide appropriate responses for all classes of service with appropriate delimiters.
 - A) If an N_Port that supports only Class 6 receives a Class 2 frame, and
 - a) 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.
 - b) is engaged in a dedicated connection, the N_Port response is unpredictable.
 - B) If an N_Port that supports only Class 6 receives a Class 3 frame, and
 - a) is not engaged in a dedicated connection, the N_Port shall discard the frame and obey the buffer-to-buffer flow control rules.

- b) is engaged in a dedicated connection, the N_Port shall discard the frame and not follow the buffer-to-buffer flow control rules.
- C) If an F_Port that supports only Class 6 receives a Class 2 frame, and
 - a) 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.
 - b) is engaged in a dedicated connection, the F_Port response is unpredictable.
- D) If an F_Port that supports only Class 6 receives a Class 3 frame, and
 - a) is not engaged in a dedicated connection, the F_Port shall discard the frame and obey buffer-to-buffer flow control rules.
 - b) is engaged in a dedicated connection, the F_Port response is unpredictable.
- o) If an N_Port does not support Class 6 and receives a Class 6/SOF_{c1} frame, the N_Port shall issue a P_RJT with a SOF_{n1} and EOF_{dt} with a reason code of "Class not supported". If an 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 an 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.
- q) The effect of preemption of one destination N_Port on the remaining N_Ports is implementation-dependent.

13.7.3 Delimiters

A dedicated connection is requested by transmitting a Data frame using a 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. 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.

13.7.4 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.

13.7.5 Flow control

All Class 6 frames shall follow end-to-end flow control rules (see 18.4.1). The Class 6/SOF_{c1} frame shall follow both end-to-end and buffer-to-buffer flow control rules (see 18.5.2).

13.7.6 Stacked Connect-requests

Stacked connect-requests is a feature that may be provided by the Fabric (see 20.5.3).

14 Name_Identifier Formats

14.1 Introduction

Name_Identifier are used to identify entities in Fibre Channel such as an N_Port, Node, F_Port, Fabric or other Fibre Channel objects. The Name_Identifier for an entity shall be unique within the address domain of configuration, Fabric, point to point, or private loop. The scope of the uniqueness varies with the format type.

The NAA field (bits 31 to 28 of Word 0) within the Name_Identifier specifies its format and length. A list of supported formats is given in table 207.

Table 207 - NAA identifiers

Words 0, bits 31 to 28	NAA	Length
0000' b	Name not present	
'0001' b	IEEE 48-bit address	64
'0010' b	IEEE extended	64
'0011' b	Locally assigned	64
'0100' b	32-bit IP address	64
'0101' b	IEEE Registered	64
'0110' b	IEEE Registered Extended	128
'0111' b to '1011' b	Reserved	
'1100' b	EUI-64 Mapped	64
'1101' b	EUI-64 Mapped	64
'1110' b	EUI-64 Mapped	64
'1111' b	EUI-64 Mapped	64

An NAA field value of "Name not present" ('0000'b) indicated that the Name Value field does not contain an valid Name_Identifier, and shall be ignored.

14.2 IEEE 48-bit address

When the Name_Identifier format is IEEE 48-bit address, the name value 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. ULA Bytes 0, 1, and 2 are generated using the IEEE Company_ID. Reference Annex J for information on obtaining an IEEE Company_ID. ULA Bytes 3, 4, and 5 represent a unique value provided by the identified company.

The least significant two bits of byte 0 are the Individual/Group Address (I/G) bit and the Universally or Locally Administered Address (U/L) bit. These bits shall be zero when a ULA is used in a Name_Identifier. Table 208 shows how the bytes of an ULA shall be mapped to two words in the Name_Identifier.

A 48-bit IEEE address Name_Identifier is a Worldwide Name.

Table 208 - IEEE 48-bit address format

Bits Word	31 .. 28	27 .. 24	23 .. 16	15 .. 10	9	8	07 .. 00
0	'0001' b	hex '0 00'		ULA Byte 0	U/ L	I/ G	ULA Byte 1
1	ULA Byte 2		ULA Byte 3	ULA Byte 4		ULA Byte 5	

Example -

A company has an IEEE Company_ID value:

hex 'AC DE 48'

This value is combined with a unique value generated by the identified company of hex '00 00 80' to create a ULA of:

hex 'AC DE 48 00 00 80'

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'

14.3 IEEE extended

When the Name_Identifier format is IEEE extended, the name value field shall contain the 48-bit IEEE address preceded by a 12 bit value that is an extension to the company assigned address portion of the 48-bit address that shall form a unique 60-bit value. The 48-bit IEEE address shall be as defined for the IEEE 48-bit address Name_Identifier format. This format is described in table 209.

An IEEE extended Name_Identifier is a Worldwide Name.

Table 209 - IEEE extended format

Bits Word	31 .. 28	27 .. 24	23 .. 16	15 .. 10	9	8	07 .. 00
0	'0010' b	Vendor Specific		ULA Byte 0	U/ L/ G		ULA Byte 1
1	ULA Byte 2		ULA Byte 3	ULA Byte 4			ULA Byte 5

Example -

A company has an IEEE Company_ID value:

hex 'AC DE 48'

This value is combined with a unique value generated by the identified company of hex '00 00 80' to create a ULA of:

hex 'AC DE 48 00 00 80'

Using this ULA and a vendor specified value of hex 'B17', the following 64-bit Fibre Channel IEEE extended identifier format is created:

hex '2B 17 AC DE 48 00 00 80'

14.4 Locally assigned

When the Name_Identifier format is locally assigned, the name value field shall be assigned in a manner determined by the administration of the Fabric in which it is assigned. This format is described in table 210.

A locally assigned Name_Identifier is unique within the Fabric wherein it is assigned.

Table 210 - Locally assigned format

Bits Word	31 .. 28	27 .. 24	23 .. 16	15 .. 08	07 .. 00
0	'0011' b	Locally administered value			
1	Locally administered value				

14.5 32-bit IP address

When the Name_Identifier format is IP, the name value shall contain a 32-bit IP address. Assignment of IP addresses shall conform to accepted Internet Protocol conventions and be unique within the Fabric wherein it is assigned. This format is specified in table 211.

A 32-bit IP address Name_Identifier is unique within the Fabric wherein it is assigned.

Table 211 - 32-bit IP address format

Bits Word	31 .. 28	27 .. 24	23 .. 16	15 .. 10	9	8	07 .. 00
0	'0100' b	Reserved					
1	IP Address Byte 0		IP Address Byte 1		IP Address Byte 2		IP Address Byte

14.6 IEEE registered

When the Name_Identifier format is IEEE registered, the name value field shall contain the 24-bit IEEE Company_ID in canonical form, as specified by IEEE, followed by a 36-bit unique vendor specified identifier (VSID). This format is described in table 212.

An IEEE registered Name_Identifier is a Worldwide Name.

Table 212 - IEEE Registered

Bits Word	31 .. 28	27 .. 24	23 .. 16	15 .. 8	07 .. 04	03 .. 00
0	'0101' b	IEEE Company_ID				VSID (35 to 32)
1	VSID (31 to 0)					

Example -

A company has an IEEE Company_ID value:

hex 'AC DE 48'

The VSID value selected by the identified company is hex 'B 17 34 F6 2D'.

The resulting Fibre Channel IEEE registered format is:

hex '5A CD E4 8B 17 34 F6 2D'

14.7 IEEE registered extended

When the Name_Identifier format is IEEE registered extended, the name value field shall contain the 24-bit IEEE Company_ID in canonical form, as specified by IEEE, followed by a 36-bit unique vendor specified id (VSID). An additional 64-bit vendor specified identifier extension is defined. Only the first 64 bits of the IEEE registered extended format are used as the Name_Identifier and accessible by methods defined in this standard. The vendor specified identifier extension may have meaning to and be accessible by an FC-4 implementation. The format of IEEE registered extended is described table 213.

An IEEE registered extended name may be assigned to any Fibre Channel related object.

An IEEE registered extended Name_Identifier is a Worldwide Name.

Table 213 - IEEE Registered Extended

Bits Word	31 .. 28	27 .. 24	23 .. 16	15 .. 8	07 .. 04	03 .. 00
0	'0110' b	IEEE Company_ID				VSID (35 to 32)
1	VSID (31 to 0)					
2	VSID Extension (63 to 32)					
3	VSID Extension (31 to 0)					

Example -

A company has an IEEE Company_ID value:

hex 'AC DE 48'

The VSID value selected by the identified company is hex 'B 17 34 F6 2D' and the VSID extension is hex '12 34 56 78 AB CD EF'.

The resulting Fibre Channel IEEE registered extended format is:

hex '6A CD E4 8B 17 34 F6 2D'

14.8 Other uses of IEEE registered Company_ID

The Company_ID is used in the RNFT LS_ACC (see 12.3.2.53.3).

14.9 EUI-64 Mapped

14.9.1 General

When the Name_Identifier format is EUI-64 Mapped, The NAA field shall contain either hex '0C', hex '0D', hex '0E', or hex '0F'. The name value field shall contain a modified 22-bit IEEE Company_ID, as specified in following paragraphs, followed by a 40-bit unique VSID.

The EUI-64 name is so mapped to account for the 4 additional bits allocated to the VSID. The general mapping scheme is to right shift the first byte of the IEEE Company_ID, moving bits 7 to 2 to positions 5 to 0 of the WWN Byte 0. Bits 1 to 0 of are the Universal/Local and Individual/Group bits, presumed to always be '00' b. Bits 7 to 6 of the WWN Byte 0 are set to '11' b, and the byte is prepended to the rest of the name. The format of EUI-64 Mapped

Name_Identifier is described in table 214.

Table 214 - EUI-64 Mapped Name_Identifier Format

Bits Word	31...30	29...24	23...16	15...8	7...0
0	'11'b	IEEE Company_ID (modified)			VSID (39 to 33)
1	VSID (32 to 0)				

14.9.2 EUI-64 to WWN Mapping Rules

Refer to table 215, Bit Position Map. The following mapping rules apply:

- WWN.NAA 3 and WWN.NAA 2 are set = 1.
- EUI.OUI 23 to 18 are mapped to WWN.OUI 21 to 16.
- EUI.OUI 15 to 0 are mapped one for one to WWN.OUI 15 to 0.
- EUI.VSID 39 to 0 are mapped one for one to WWN.VSID 39 to 0.

14.9.3 Encapsulated MAC-48 and EUI-48 translation

Encapsulated MAC-48 and EUI-48 names may be translated using the same rules as the EUI-64 names. Uniqueness shall be preserved.

Table 215 - Bit Position Map

Byte Position	Bit Position in Byte	Bit Position in Name	EUI Values	WWN Values
0	7	63	OUI 23	1
	6	62	OUI 22	1
	5	61	OUI 21	OUI 23
	4	60	OUI 20	OUI 22
	3	59	OUI 19	OUI 21
	2	58	OUI 18	OUI 20
	1	57	OUI 17 / L/U	OUI 19
	0	56	OUI 16 / I/G	OUI 18
1	7 to 0	55 to 48	OUI 15 to 8	OUI 15 to 8
2	7 to 0	47 to 40	OUI 7 to 0	OUI 7 to 0
3	7 to 0	39 to 32	VSID 39 to 32	VSID 39 to 32
4	7 to 0	31 to 24	VSID 31 to 24	VSID 31 to 24
5	7 to 0	23 to 16	VSID 23 to 16	VSID 23 to 16
6	7 to 0	15 to 8	VSID 15 to 8	VSID 15 to 8
7	7 to 0	7 to 0	VSID 7 to 0	VSID 7 to 0

15 Login and Service Parameters

15.1 Introduction

The Login procedure is a method by which an Nx_Port establishes its operating environment with a Fabric, if present, and other destination Nx_Ports with which it communicates. Fabric Login and N_Port Login are both accomplished with a similar procedure using different D_IDs and possibly different S_IDs.

Login between an Nx_Port and the Fabric or between two Nx_Ports is long-lived. The number of concurrent Nx_Ports with which an Nx_Port may be logged in with is a function of the Nx_Port facilities available. There is no one to one relationship between Login and Class 1 dedicated connections.

Login between an Nx_Port and the Fabric or between two Nx_Ports may use an explicit or implicit methods. When Login is referred to throughout other sections of this standard, either the explicit or implicit procedure may be used. Implicit Login is assumed to provide the same functionality as Explicit Login.

Explicit Login is accomplished using a Login (FLOGI or PLOGI) ELS (see 12.3.2.7) within a new Exchange to transfer the Service Parameters (contained in the Payload) of the Nx_Port initiating the Login Exchange. The LS_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 Nx_Ports by means other than the explicit use of the Login ELS. Specific methods of implicit Login are not defined in this standard.

Implicit Fabric Login, unlike Explicit Fabric Login, does not require the support, within an Nx_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 an Nx_Port.

An Nx_Port determines its own native N_Port_ID through explicit or implicit Login by

- a) the Fabric, if present,
- b) implicit definition, or
- c) assignment in the PLOGI Sequence transmitted to a destination Nx_Port attached in a point-to-point topology.

Nx_Ports may collect Address Identifiers from other potential destination Nx_Ports from:

- a) a name server function, if present,
- b) implicit definition, or
- c) an alternate initialization procedure.

N_Port Login shall not affect the state of Class 4 connections in the Fabric.

15.2 Default Login values

Prior to Login or following Logout, the following default set of Service Parameters shall apply (Class 1, 2, 3 and 6):

- a) Concurrent Sequences = 1
- b) Total Concurrent Sequences = 1
- c) End-to-end Credit = 1
- d) buffer-to-buffer Credit = 1 (0 for an L_Port)
- e) Receive_Data_Field Size = 128
- f) X_ID interlock required
- g) ACK_1
- h) Discard multiple Sequences Error Policy
- i) Relative offset not used
- j) E_D_TOV Resolution = 0

- k) BB_SC_N = 0
- l) Other optionally supported features shall not be used or required.

Following Login with the destination, an Nx_Port shall use the Service Parameters obtained through Login.

The Class 4 Service Parameters prior to N_Port Login shall be as specified in 15.2. The default Login values for Class 6 shall be the same as Class 1.

15.3 Fabric Login

15.3.1 Introduction

Login with the Fabric is required for all Nx_Ports, regardless of the class supported. Communication with other Nx_Ports shall not be attempted until the Fabric Login procedure is complete.

- a) Fabric Login accomplishes the following functions:
 - b) It determines the presence or absence of a Fabric.
 - c) If a Fabric is present, it provides the Nx_Port with the specific set of operating characteristics associated with the entire Fabric, F_Port_Name and Fabric_Name.
 - d) If a Fabric is present, it provides the Fabric with the specific set of operating characteristics, N_Port_Name and Node_Name of the Nx_Port
 - e) If a Fabric is present, the Fabric shall optionally assign or shall confirm the N_Port_ID of the Nx_Port that initiated the Login.
 - f) If a Fabric is present, it initializes the buffer-to-buffer Credit.

15.3.2 Explicit Fabric Login

15.3.2.1 Introduction

The explicit Fabric Login procedure shall require an Nx_Port to transmit a Fabric Login (FLOGI) Link Service ELS (see 12.3.2.7).

Explicit Fabric Login replaces previous Service Parameters. The Login procedure shall follow the Exchange and Sequence management rules as specified in clause 17. Frames within a Sequence shall operate according to R_RDY Primitive Signal, ACK, and Link_Response rules specified in 11.2.

15.3.2.2 Explicit Fabric Login Request

The Nx_Port shall transmit the FLOGI in a new Exchange. The Payload of FLOGI contains the Service Parameters of the Nx_Port, a 64-bit N_Port_Name of the Nx_Port, and a 64-bit Node_Name. The Service Parameters are as specified for F_Port Login in 15.6. The applicability of the Service Parameters to Fabric Login are given in tables 217 and 222. The Nx_Port shall assign an OX_ID and set the D_ID to the well-known F_Port address (hex 'FF FF FE').

If the Nx_Port is unidentified, an Nx_Port shall set the S_ID in the FLOGI to hex '00 00 00' or hex '00 00 YY'. If the Nx_Port sets the S_ID to hex '00 00 00', the Nx_Port is requesting the Fabric assign all 24 bits of the N_Port_ID. If the Nx_Port sets the S_ID to hex '00 00 YY', the Nx_Port is requesting the Fabric assign the upper 16 bits, bits 23 to 8, and validate the lower 8 bits, bits 7 to 0, of the N_Port_ID. An example of the use of S_ID of hex '00 00 YY' is FC-AL-2. The lower 8 bits of the N_Port_ID are the AL_PA.

15.3.2.3 Responses to Explicit Fabric Login

The following are possible responses the Nx_Port may receive when transmitting a FLOGI:

- a) LS_ACC reply Sequence with OX_ID equal to the OX_ID of the FLOGI, and the Common Service N_Port/F_Port bit set to one (Fx_Port) - This is the normal response to a Fabric Login request. The D_ID of the

LS_ACC shall be the N_Port_ID assigned by the Fabric. If the S_ID in the FLOGI was hex '00 00 00', the D_ID shall be hex 'XX XX XX'. If the S_ID in the FLOGI was hex '00 00 YY', the D_ID shall be hex 'XX XX YY'. If the S_ID in the FLOGI was hex 'XX XX XX', the D_ID shall be same value of hex 'XX XX XX'. The Payload shall include the Service Parameters for the entire Fabric, a 64-bit F_Port_Name and 64-bit Fabric_Name. The Service Parameters are as specified for F_Port Login in 15.6. The applicability of the Service Parameters to Fabric Login are given in tables 217 and 222. The Nx_Port may continue operation with other Nx_Ports if the N_Port_ID, F_Port_Name, and Fabric_Name are the same as in a previous Fabric Login or proceed to N_Port Login.

- b) LS_ACC reply Sequence with OX_ID equal to the OX_ID of the FLOGI, and the Common Service Nx_Port/F_Port bit set to zero (Nx_Port) This indicates a point-to-point connection with another Nx_Port. The D_ID of the LS_ACC shall be the S_ID of the FLOGI. The Payload shall include the Service Parameters from the FLOGI with all classes mark invalid, a 64-bit N_Port_Name and 64-bit Node_Name of the connected Nx_Port. If the received N_Port_Name is less than its N_Port_Name, the Nx_Port proceeds to N_Port Login. If the received N_Port_Name is greater than its N_Port_Name, the Nx_Port waits for PLOGI from the attached N_Port.
- c) F_BSY with OX_ID equal to the OX_ID of the FLOGI. The D_ID shall be the S_ID of the FLOGI. The Fabric is busy. The Nx_Port may retry the FLOGI again later.
- d) P_BSY Sequence with OX_ID equal to the OX_ID of the FLOGI. The D_ID shall be the S_ID of the FLOGI. This indicates a point-to-point connection with another Nx_Port that is currently busy. The Nx_Port may proceed to N_Port Login after a delay to allow the destination Nx_Port to become not busy.
- e) F_RJT Sequence with OX_ID equal to the OX_ID of the FLOGI. The D_ID shall be the S_ID of the FLOGI. The Fabric has rejected the FLOGI request. The reason code contained in the Payload determines the Nx_Port's action. If the reason code is "class not supported", the Nx_Port may originate a FLOGI in a different class. If the reason code is "Invalid S_ID", the Nx_Port may originate a FLOGI with a different S_ID:
 - A) If the S_ID of the rejected FLOGI was hex '00 00 00' or hex '00 00 YY', the Nx_Port may select a 24 bit value, hex 'XX XX XX', for its N_Port_ID by a method outside this standard and originate a FLOGI with this value in the S_ID.
 - B) If the S_ID of the rejected FLOGI was hex 'XX XX XX', the Nx_Port may select a value of '00 00 00' or '00 00 yy', or a new value 'XX XX XX' for its N_Port_ID by a method outside this standard and originate a FLOGI with this value in the S_ID.

If the reason code in the reject is other than the reason codes listed here, the Nx_Port should respond accordingly.

- f) P_RJT Sequence, with OX_ID equal to the OX_ID of the FLOGI. The D_ID shall be the S_ID of the FLOGI. This indicates a point-to-point connection with another Nx_Port. The reason code contained in the Payload determines the Nx_Port's action. If the reason code is "class not supported", the Nx_Port may proceed to N_Port Login in a different class than used for the FLOGI. For other reason codes, the Nx_Port should respond accordingly.
- g) FLOGI Sequence. This indicates a point-to-point connection with another Nx_Port. The Payload shall include a 64-bit N_Port_Name and 64-bit Node_Name of the connected Nx_Port. If the received N_Port_Name is less than its N_Port_Name, the Nx_Port proceeds to N_Port Login. If the received N_Port_Name is greater than its N_Port_Name, the Nx_Port waits for PLOGI from the attached N_Port.
- h) LS_RJT Sequence with OX_ID equal to the OX_ID of the FLOGI. The D_ID of the LS_RJT shall be the N_Port_ID assigned by the Fabric. If the S_ID in the FLOGI was hex '00 00 00', the D_ID shall be hex 'XX XX XX'. If the S_ID in the FLOGI was hex '00 00 YY', the D_ID shall be hex 'XX XX YY'. If the S_ID in the FLOGI was hex 'XX XX XX', the D_ID shall be hex 'XX XX XX'. The reason code contained in the Payload determines the Nx_Port's action. The Nx_Port may alter the Service Parameters based on the reason code and originate a new FLOGI.
- i) No Response may indicate a delivery error, for example, error on the physical transport. The Nx_Port shall perform error recovery per clause 21. The Nx_Port may originate a new FLOGI after recovery.
- j) If the received N_Port_Name is equal to its N_Port_Name, then the Nx_Port is connected to itself and this case is outside the scope of this standard. The FLOGI is discarded.

15.3.2.4 Relogin with the Fabric

During a Login with the Fabric, if the Nx_Port was previously logged in with the fabric and the N_Port_ID, F_Port_Name, and the Fabric_Name are the same as the previous login, the Nx_Port may continue current

communications with other Nx_Ports that it has established logins with; if the Nx_Port detects that the N_Port_ID, F_Port_Name, and/or the Fabric_Name has changed since the last Fabric Login, the Nx_Port shall implicitly logout with all Nx_Ports and wait an R_A_TOV timeout period before initiating or accepting communication with other Nx_Ports. The timeout period shall start when the Nx_Port detects the change. After waiting the timeout period, new N_Port Logins are required before the Nx_Port may communicate with other Nx_Ports.

15.3.3 SOF delimiters

Fabric Login shall only be performed using Class 1, 2 or 3. Since the Fabric may not support all three classes, the FLOGI Sequence may require retry with a different SOF delimiter for each of the following classes:

- a) Class 1 - SOFc1
- b) Class 2 - SOFi2
- c) Class 3 - SOFi3

Fabric Login is valid for all supported classes as indicated by the validity bits in the FLOGI LS_ACC Reply Sequence.

SOFc1 may be used to attempt fabric login or fabric relogin. SOFi1 is not allowed.

Class 6 shall not be used for fabric login since the D_ID field requires a Alias_ID multicast address.

Selection of the SOF delimiter for the FLOGI Sequence is based on the Classes supported by the originating Nx_Port. The FLOGI Sequence is transmitted and the appropriate action is specified in 15.3.2.3. If an F_RJT with reason code "Class of Service not supported by entity at hex 'FF FF FE'" is received, another supported delimiter shall be attempted until the Login procedure is complete or until all supported delimiter types have been attempted. If all supported delimiter types have been attempted and the Fabric has rejected all or timed out, the Fabric and Nx_Port are incompatible and outside intervention is required.

Class 4 shall not be used for fabric login. Class 4 is dependent on Classes 1, 2, or 3 for fabric login of an N_Port. The Class 4 N_Port Login service parameters used during Fabric Login (in FLOGI or LS_ACC to FLOGI) shall be as specified in 25.2. The Fabric shall remove all Class 1, 4 and 6 connections to an N_Port when the N_Port performs a Fabric Login.

15.3.4 Frequency

Login between an Nx_Port and the Fabric should be long-lived. If Implicit Logout with the Fabric has occurred, it is necessary to perform a new Login with the Fabric (see 15.5.3).

15.3.5 Fabric Login completion - Originator

The Originator of the FLOGI request considers Fabric Login to have ended when

- a) in Class 1, the Originator has transmitted the ACK (EOF_t or EOF_{dt}) to the LS_ACC;
- b) in Class 2, the Originator has transmitted the ACK (EOF_t) to the LS_ACC; or
- c) in Class 3, the Originator has received the LS_ACC.

When Login is ended, the values of buffer-to-buffer Credit are initialized.

15.3.6 Fabric Login completion - Responder

The Responder of the FLOGI request considers Fabric Login to have ended when

- a) in Class 1, the Responder has received the ACK (EOF_t or EOF_{dt}) to the LS_ACC;
- b) in Class 2, the Responder has received the ACK (EOF_t) to the LS_ACC; or
- c) in Class 3, the Responder has transmitted the LS_ACC.

When Fabric Login has ended successfully, the values of buffer-to-buffer Credit are initialized.

15.4 N_Port Login

15.4.1 Introduction

N_Port Login follows the Fabric Login procedure. If a Fabric is present, as determined by performing the Fabric Login procedure, an Nx_Port proceeds with N_Port Login according to 15.4.2.2. If a Fabric is not present, as determined by performing the Fabric Login procedure, an Nx_Port proceeds with N_Port Login according to 15.4.2.4.

N_Port Login accomplishes the following functions:

- a) It provides each Nx_Port with the other Nx_Port's operating characteristics, N_Port_Name and Node_Name.
- b) If a Fabric is not present, it assigns the native N_Port_ID for both Nx_Ports.
- c) It initializes the end-to-end Credit.
- d) In point-to-point topology or between NL_Ports on the same loop, buffer-to-buffer Credit is initialized

N_Port Login between two Nx_Ports is complete when each Nx_Port has received the Service Parameters of the other Nx_Port. This may be accomplished by either implicit or explicit N_Port Login.

An Nx_Port is required to Login with each Nx_Port with which it intends to communicate. This includes reserved and well-known address identifiers since they are considered to be N_Ports (see 9.4.2).

NOTE 1 It is not required that an Nx_Port provide the same Login information with each destination Nx_Port or with the Fabric. However, an Nx_Port should avoid using contradictory or conflicting parameters with different Login destinations.

The N_Port Common Service Parameters during N_Port Login are specified in 15.6.2 (See table 217 for applicability). The N_Port Class Service Parameters during N_Port Login are specified in 15.6.5 (See table 222 for applicability). Both the Common Service Parameters and Class Service Parameters apply to each Nx_Port during N_Port Login.

NOTE 2 When an Nx_Port (A) receives a PLOGI from another Nx_Port (B), Nx_Port (A) should verify that it is not already logged in with an Nx_Port (C) with the same N_Port_Name but different Nx_Port native address identifier and Node_Name. If so, it should consider the prior Login to be ended and all open Sequences that it originated with or received from the destination Nx_Port are terminated before accepting the new Login. Such a situation may arise if configuration changes have occurred.

N_Port Login provides each Nx_Port with the other Nx_Port's Service Parameters. Knowledge of a destination Nx_Port's receive and transmit characteristics is required for data exchanges. Service Parameters of destination Nx_Ports are saved and used when communication with those Nx_Ports is initiated. The Service Parameters interchanged between two Nx_Ports may be asymmetrical. Saving the Service Parameters of destination Nx_Ports with which an Nx_Port communicates requires Nx_Port resources. These resources should be released using the destination N_Port Logout procedure (see 15.5).

Due to the resetting behavior of a PLOGI (for example, termination of all open exchanges with the destination port), a port shall only send a PLOGI to a destination port if it is not logged in with the destination port. Examples of why a port is not logged in include:

- a) it has determined that a configuration change has occurred;
- b) it has lost knowledge of the login parameters with the destination port;
- c) the destination port has responded with a frame that indicates an error condition (for example, LOGO, P_RJT);
- d) the local port has logged out the destination port, either implicitly or explicitly, due to resource constraints; and
- e) the destination port failed to respond after 2 times R_A_TOV has expired.

A configuration change shall be determined by comparing the Port_Name, Node_Name, and Address_Identifier received in the ACC from an ADISC or PDISC with the values previously established during the previous login process. A configuration change has occurred if either N_Port_Name or N_Port_ID do match and any of the three parameters do not match.

15.4.2 Explicit N_Port Login

15.4.2.1 Introduction

The explicit N_Port Login procedure shall require an Nx_Port to transmit a PLOGI Link request Sequence.

Explicit N_Port Login replaces previous Service Parameters. The Login procedure shall follow the Exchange and Sequence management rules as specified in clause 17. Frames within a Sequence shall operate according to R_RDY Primitive Signal, ACK, and Link_Response rules specified in 11.2.

A well-behaved Nx_Port shall Logout with another Nx_Port prior to initiating a new N_Port Login. However, if an Nx_Port receives or transmits a PLOGI request with another Nx_Port, it shall abnormally terminate open Sequences and respond to any new Sequences with that Nx_Port as though a Logout had been previously performed. During the N_Port Login procedure, other communication with the destination Nx_Port shall not be initiated or accepted. Once the N_Port Login procedure has been successfully completed, communication between the Nx_Ports may be initiated or accepted, (for example, if Nx_Port(A) performs a PLOGI request with Nx_Port(B) and Nx_Port(B) transmits the LS_ACC reply, then either Nx_Port(A) or Nx_Port(B) may initiate communication for other protocols. Nx_Port(B) shall not be required to transmit a PLOGI request Sequence to Nx_Port(A) unless it wishes to invalidate or alter the existing Login parameters).

15.4.2.2 N_Port Login - Fabric present

The destination Nx_Port explicit Login procedure requires transmission of a N_Port Login (PLOGI) Link Service Sequence. The PLOGI is sent within an Exchange with an assigned OX_ID, the D_ID of the destination Nx_Port and a S_ID of originating Nx_Port. The Payload of this Sequence contains the Service Parameters, N_Port_Name, and Node_Name of the Nx_Port originating the PLOGI Sequence. The N_Port Service Parameters are as specified in 15.6. The applicability of the Service Parameters to N_Port Login are given in tables 217 and 222.

The normal reply Sequence to a PLOGI Link Service Sequence by an Nx_Port is a LS_ACC Link Service Reply Sequence within the Exchange identified by the OX_ID of the Login Sequence and the RX_ID assigned by the Responder with a D_ID of the originating Nx_Port (PLOGI Sequence) and a S_ID of the responding Nx_Port. The Payload of the LS_ACC contains the Service Parameters of the responding Nx_Port.

15.4.2.3 Responses to N_Port Login - Fabric present

The following are possible responses the Nx_Port may receive in response to transmitting a PLOGI with a Fabric present:

- a) LS_ACC reply Sequence with OX_ID equal to the OX_ID of the PLOGI, and the Common Service N_Port/F_Port bit = 0 (Nx_Port) - This is the normal response to a N_Port Login request. The D_ID of the LS_ACC shall be the S_ID from the PLOGI. The S_ID of the LS_ACC shall be D_ID from the PLOGI. The Payload shall include the Service Parameters for the destination Nx_Port, a 64-bit N_Port_Name and a 64-bit Node_Name. The N_Port Service Parameters are as specified for in 15.6. The applicability of the Service Parameters to N_Port Login are given in tables 217 and 222. The Nx_Port may begin normal communication with the remote N_Port.
- b) F_BSY with OX_ID equal to the OX_ID of the PLOGI. The D_ID shall be the S_ID of the PLOGI. The Fabric is busy. The Nx_Port may retry the PLOGI again later.
- c) F_RJT Sequence with OX_ID equal to the OX_ID of the PLOGI. The D_ID shall be the S_ID of the PLOGI. The Fabric has rejected the PLOGI request. The reason code contained in the Payload determines the Nx_Port's action. If the reason code is Invalid D_ID, N_Port Login is not possible with the addressed Nx_Port. The Nx_Port may attempt Login with other destination Nx_Ports. For other reason codes, the Nx_Port should respond according to the code.

- d) P_BSY Sequence with OX_ID equal to the OX_ID of the PLOGI. The D_ID shall be the S_ID of the PLOGI. The destination Nx_Port is busy. The Nx_Port may retry the PLOGI again later.
- e) P_RJT Sequence with OX_ID equal to the OX_ID of the PLOGI. The D_ID shall be the S_ID of the PLOGI. The reason code contained in the Payload determines the Nx_Port's action. If the reason code is "Class not supported", the Nx_Port may attempt PLOGI in a different class. For other reason codes, the Nx_Port should respond according to the code.
- f) PLOGI Sequence. The D_ID is the N_Port_ID of receiving Nx_Port. The S_ID is the N_Port_ID of the originating Nx_Port. The OX_ID is as assigned by the originating Nx_Port. The Payload shall include a 64-bit N_Port_Name and 64-bit Node_Name of the Nx_Port originating the PLOGI. This indicates a collision with N_Port Login from the destination Nx_Port. If the received N_Port_Name is less than the receiving Nx_Port's N_Port_Name, the Nx_Port sends LS_RJT to the originating Nx_Port with reason code Login already in progress. If the received N_Port_Name is greater than its N_Port_Name, the Nx_Port processes the received PLOGI.
- g) LS_RJT Sequence with OX_ID equal to the OX_ID of the PLOGI. The D_ID of the LS_ACC shall be the N_Port_ID of the destination Nx_Port. The reason code contained in the Payload determines the Nx_Port's action. The Nx_Port may alter the Service Parameters based on the reason code and originate a new PLOGI.
- h) No Response may indicate a delivery error, for example, error on the physical transport. The Nx_Port shall perform error recovery per clause 21. The Nx_Port may originate a new PLOGI after recovery.
- i) If the received N_Port_Name is equal to its N_Port_Name, then the Nx_Port is connected to itself and this case is outside the scope of this standard.

15.4.2.4 N_Port Login - No Fabric present

This procedure is based on the Nx_Port discovering the Fabric is not present during an attempted Fabric Login. (see 15.3.2.3) The destination N_Port explicit Login procedure requires transmission of a PLOGI Link Service Sequence in a new Exchange.

Only one Nx_Port in the point-to-point connection is required to transmit a PLOGI. If the N_Port_Names are exchanged during Fabric Login, the Nx_Port with the highest N_Port_Name shall transmit the PLOGI.

If either Nx_Port does not have access the N_Port_Name of the connected Nx_Port, it may send a PLOGI. The processing requirements for responses received after transmitting a PLOGI resolves the condition of both Nx_Ports transmitting PLOGI.

An Nx_Port in a point-to-point configuration transmits PLOGI within a new Exchange. The S_ID shall be different than the D_ID. The Payload of this Sequence contains the Service Parameters, N_Port_Name, and Node_Name of the Nx_Port originating the PLOGI Sequence. The N_Port Service Parameters are as specified for in 15.6. The applicability of the Service Parameters to N_Port Login are given in tables 217 and 222.

15.4.2.5 Responses to N_Port Login - No Fabric present

The following are possible responses the Nx_Port may receive in response to transmitting a PLOGI in a point-to-point configuration:

- a) LS_ACC reply Sequence with OX_ID equal to the OX_ID of the PLOGI, and the Common Service N_Port/F_Port bit = 0 (Nx_Port). The D_ID of the LS_ACC shall be the S_ID from the PLOGI. The S_ID shall be the destination Nx_Port's N_Port_ID assigned by the D_ID in the PLOGI. The Payload shall include the Service Parameters for the destination Nx_Port, a 64-bit N_Port_Name and a 64-bit Node_Name. The N_Port Service Parameters are as specified for in 15.6. The applicability of the Service Parameters to N_Port Login are given in tables 217 and 222. This is the normal response to a N_Port Login request. The Nx_Port may begin normal communication with the remote N_Port.
- b) P_BSY Sequence with OX_ID equal to the OX_ID of the PLOGI. The D_ID shall be the S_ID of the PLOGI. The destination Nx_Port is busy. The Nx_Port may retry the PLOGI again later.
- c) P_RJT Sequence with OX_ID equal to the OX_ID of the PLOGI. The D_ID shall be the S_ID of the PLOGI. The reason code contained in the Payload determines the Nx_Port's action. If the reason code is "Class

not supported”, the Nx_Port may attempt N_Port Login in a different Class. For other reason codes, the Nx_Port should respond according to the code.

- d) LS_RJT Sequence with OX_ID equal to the OX_ID of the PLOGI. The D_ID of the LS_ACC shall be the N_Port_ID of the destination Nx_Port. The reason code contained in the Payload determines the Nx_Port's action. The Nx_Port may alter the Service Parameters based on the reason code and originate a new PLOGI.
- e) PLOGI Sequence. The D_ID is the Port_Identifier of receiving Nx_Port. The S_ID is the N_Port_ID of the originating Nx_Port. The OX_ID is as assigned by the originating Nx_Port. The Payload shall include a 64-bit N_Port_Name and 64-bit Node_Name of the Nx_Port originating the PLOGI. This indicates a collision with N_Port Login from the destination Nx_Port. If the received N_Port_Name is less than the receiving Nx_Port's N_Port_Name, the Nx_Port sends LS_RJT to the originating Nx_Port with reason code Login already in progress. If the received N_Port_Name is greater than its N_Port_Name, the Nx_Port processes the received PLOGI.
- f) No Response may indicate a delivery error, for example, error on the physical transport. The Nx_Port shall perform error recovery per clause 21. The Nx_Port may originate a new PLOGI after recovery.
- g) If the received N_Port_Name is equal to its N_Port_Name, then the Nx_Port is connected to itself and this case is outside the scope of this standard.

15.4.3 SOF delimiters

N_Port Login is only supported in Class 1, 2, and 3. Since the destination Nx_Port may not support all these classes for Login, the PLOGI Sequence may require retransmission in a different Class with the appropriate SOF in the same manner described for Fabric Login (see 15.3.3). Login is valid for all supported classes as indicated by the validity bits in the PLOGI LS_ACC Reply Sequence.

15.4.4 Frequency

The frequency of N_Port Login is installation dependent based on the frequency of configuration changes that may alter the N_Port_ID within an installation. Service Parameters of other Nx_Ports are retained until the next N_Port Login or until N_Port Logout (implicit or explicit) is performed.

15.4.5 N_Port Login completion - Originator

The Originator of the PLOGI request considers Login to have ended when

- a) in Class 1, the Originator has transmitted the ACK (EOF_t or EOF_{dt}) to the LS_ACC;
- b) in Class 2, the Originator has transmitted the ACK (EOF_t) to the LS_ACC; or
- c) in Class 3, the Originator has received the LS_ACC.

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.

15.4.6 N_Port Login completion - Responder

The Responder of the PLOGI request considers Login to have ended when

- a) in Class 1, the Responder has received the ACK (EOF_t or EOF_{dt}) to the LS_ACC;
- b) in Class 2, the Responder has received the ACK (EOF_t) to the LS_ACC; or
- c) in Class 3, the Responder has transmitted the LS_ACC.

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.

15.4.7 N_Port Login frame flow

See figures B.5, B.6, and B.7 for examples of frame flow for N_Port Login for Classes 1, 2, and 3, respectively.

15.5 Logout

15.5.1 Introduction

The destination Logout procedure provides a method for removing service between two Nx_Ports or for removing an N_Port_ID that was previously assigned by an F_Port. Logout releases resources associated with maintaining Service with a destination Nx_Port. Implicit Logout may occur between an Nx_Port and the Fabric (see 15.5.3), or an N_Port may use the LOGO ELS to request the removal of a virtual N_Port_ID previously assigned by the fabric

NOTE Except for the removal of a previously-assigned N_Port_ID, explicit Fabric Logout is not needed since the Fabric has no resources dedicated to an Nx_Port that could be usefully made available.

15.5.2 Explicit N_Port Logout

Logout is accomplished by transmitting a Logout (LOGO) Link Service request Sequence (see 12.3.2.8) to a destination Nx_Port. The Logout procedure is complete when the responding Nx_Port transmits a LS_ACC Link Service reply Sequence.

To explicitly Logout, the initiating Nx_Port shall terminate other open Sequences that it initiated with the destination Nx_Port prior to performing Logout, otherwise, the state of other open Sequences is unpredictable. If an Nx_Port receives a Logout request while another Sequence is open that was initiated from the requesting Nx_Port, it may reject the Logout request using an LS_RJT (Link Service Reject).

After an explicit Logout is performed with an Nx_Port, the default Login Service Parameters specified in 15.2 shall be functional if Login was explicit. After an explicit Logout is performed with an Nx_Port, the implicit Login Service Parameters shall be functional if Login was implicit.

15.5.3 Implicit Logout

If an Nx_Port receives or transmits the NOS or OLS Primitive Sequence, it shall be implicitly logged out from the Fabric, if present, or attached Nx_Port in a point-to-point topology. Fabric Login shall be performed following implicit Logout. Communication with other Nx_Ports shall not be accepted until the Fabric Login procedure is complete (implicit or explicit).

During Login with the Fabric, if the Nx_Port detects that the N_Port_ID, F_Port_Name and/or the Fabric_Name has changed since the last Fabric Login and the Clean Address bit is zero, the Nx_Port shall implicitly logout with all Nx_Ports and wait an R_A_TOV timeout period before initiating or accepting communication with other Nx_Ports. The timeout period shall start when the Nx_Port detects the change. After waiting the timeout period, new N_Port Logins are required before the Nx_Port may communicate with other Nx_Ports.

If an Nx_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 an Nx_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 Fx_Port during Fabric Login is associated with its Login request.

15.6 Service Parameters

15.6.1 ELS and LS_ACC Payload

Table 216 defines the Payload format for the FLOGI and PLOGI ELSs and the LS_ACCs. The definitions of the parameters are applicable to PLOGI, FLOGI, PLOGI LS_ACC and FLOGI LS_ACC unless stated otherwise.

NOTE 1 There are no separate Class 6 login Service Parameters as Class 1 service parameters are used instead. Service Parameter options specify FC-2 capability.

NOTE 2 The Link Service may further limit values supplied during Login as specified by individual Upper Level Protocols.

Table 216 - FLOGI, PLOGI or LS_ACC Payload

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	LS_Command code			
1	MSB	Common Service Parameters (16 bytes)		
..				
4		LSB		
5	MSB	Port_Name		
6				
7	MSB	Node_ or Fabric_Name		
8				
9	MSB	Class 1 and Class 6 Service Parameters (16 bytes)		
..				
12		LSB		
13	MSB	Class 2 Service Parameters (16 bytes)		
..				
16		LSB		
17	MSB	Class 3 Service Parameters (16 bytes)		
..				
20		LSB		
21	MSB	Class 4 Service Parameters (16 bytes)		
..				
24		LSB		
25	MSB	Vendor Version Level (16 bytes)		
..				
28		LSB		
29	MSB	Services Availability ^a (8 bytes)		
30				
31	Login Extension Data Length ^a			
32	MSB	Login Extension Data ^a		
..				
61		LSB		
62		Clock Synchronization QoS ^a (8 bytes)		
63				

^a These fields are only present when the Payload bit (see 15.6.2.4.17) is set to one. When the Payload bit is set to zero, these fields are not present in the Payload (i.e., the Payload is 116 bytes long).

15.6.2.1 Applicability

Table 217 - Common Service Parameter applicability

Service Parameter	Word	Bits	PLOGI and PLOGI LS_ACC Parameter applicability				FLOGI Parameter applicability				FLOGI LS_ACC Parameter applicability			
			Class				Class				Class			
			1 ^a	2	3	4	1 ^a	2	3	4	1 ^a	2	3	4
FC-PH Version - Obsolete	0	31 to 16	n	n	n	n	n	n	n	n	n	n	n	n
Buffer-to-Buffer Credit	0	15 to 0	y	y	y	n	y	y	y	n	y	y	y	n
Common Features	1	31 to 16												
Continuously increasing relative offset	1	31	y	y	y	y	n	n	n	n	n	n	n	n
Clean Address	1	31	n	n	n	n	n	n	n	n	y	y	y	y
Random relative offset	1	30	y	y	y	y	n	n	n	n	n	n	n	n
Valid Vendor Version Level	1	29	y	y	y	y	y	y	y	y	n	n	n	n
Multiple N_Port_ID Assignment	1	29	n	n	n	n	n	n	n	n	y	y	y	y
N_Port/F_Port	1	28	y	y	y	y	y	y	y	y	y	y	y	y
BB_Credit Management	1	27	y	y	y	y	y	y	y	y	n	n	n	n
E_D_TOV Resolution	1	26	y ^b	y ^b	y ^b	y ^b	n	n	n	n	y	y	y	y
Multicast supported by Fabric	1	25	n	n	n	n	n	n	n	n	y	y	y	y
Broadcast supported by Fabric	1	24	n	n	n	n	n	n	n	n	y	y	y	y
Hunt Group routing supported by Fabric	1	23	n	n	n	n	n	n	n	n	y	y	y	y
Query Data Buffer conditions	1	22	y	y	y	y	y	y	y	y	y	y	y	y
Security bit (see FC-SP)	1	21	-	-	-	-	-	-	-	-	-	-	-	-

Legend
 “y” indicates yes, applicable (i.e., has meaning);
 “n” indicates no, not applicable (i.e., has no meaning)

^a The Class 1 Service Parameters shall be used for Class 6. Each has the same applicability as Class 1.
^b E_D_TOV resolution and the corresponding value are only meaningful in a point-to-point topology and when doing PLOGI with an NL_Port on the same loop.

Table 217 - Common Service Parameter applicability (Continued)

Service Parameter	Word	Bits	PLOGI and PLOGI LS_ACC Parameter applicability				FLOGI Parameter applicability				FLOGI LS_ACC Parameter applicability			
			Class				Class				Class			
			1 ^a	2	3	4	1 ^a	2	3	4	1 ^a	2	3	4
Clock Synchronization Primitive Capable	1	20	y	y	y	y	y	y	y	y	y	y	y	y
R_T_TOV Value	1	19	y	y	y	y	y	y	y	y	y	y	y	y
Dynamic Half Duplex Supported	1	18	y	y	y	y	y	y	y	y	y	y	y	y
SEQ_CNT	1	17	y	y	y	y	n	n	n	n	n	n	n	n
Payload bit	1	16	y	y	y	y	y	y	y	y	y	y	y	y
BB_SC_N	1	15 to 12	y	y	y	n	y	y	y	n	y	y	y	n
Buffer-to-Buffer Receive Data Field Size	1	11 to 0	y	y	y	y	y	y	y	y	y	y	y	y
Nx_Port Total Concurrent Sequences	2	31 to 16	y	y	y	y	n	n	n	n	n	n	n	n
Relative offset by Info Category	2	15 to 0	y	y	y	y	n	n	n	n	n	n	n	n
R_A_TOV	2	31 to 0	n	n	n	n	n	n	n	n	y	y	y	y
E_D_TOV Value	3	31 to 0	y ^b	y ^b	y ^b	y ^b	n	n	n	n	y	y	y	y
Legend “y” indicates yes, applicable (i.e., has meaning); “n” indicates no, not applicable (i.e., has no meaning)														
^a The Class 1 Service Parameters shall be used for Class 6. Each has the same applicability as Class 1. ^b E_D_TOV resolution and the corresponding value are only meaningful in a point-to-point topology and when doing PLOGI with an NL_Port on the same loop.														

15.6.2.2 Payload

The Common Service Parameters Payload for FLOGI is shown in table 218.

Table 218 - Common Service Parameters - FLOGI

Word	Bits	31 .. 24	23 .. 16	15 .. 12	11 .. 08	07 .. 00
0		FC-PH Version - Obsolete		Buffer-to-buffer Credit		
1		Common Features (see table 217)		BB_SC_N	Buffer-to-buffer Receive Data Field size	
2		Reserved		Reserved		
3		Reserved		Reserved		

The Common Service Parameters Payload for PLOGI and PLOGI LS_ACC is shown in table 219.

Table 219 - Common Service Parameters - PLOGI and PLOGI LS_ACC

Bits Word	31 .. 24	23 .. 16	15 .. 12	11 .. 08	07 .. 00
0	FC-PH Version - Obsolete		Buffer-to-buffer Credit		
1	Common Features (see table 217)		BB_SC_N	Buffer-to-buffer Receive Data Field size	
2	Reserved	Total Concurrent Sequences	Relative offset by Information Category		
3	E_D_TOV				

The Common Service Parameters Payload for FLOGI LS_ACC is shown in table 220.

Table 220 - Common Service Parameters - FLOGI LS_ACC

Bits Word	31 .. 24	23 .. 16	15 .. 12	11 .. 08	07 .. 00
0	FC-PH Version - Obsolete		Buffer-to-buffer Credit (Fx_Port)		
1	Common Features (see table 217)		BB_SC_N	Buffer-to-buffer Receive Data Field size	
2	R_A_TOV				
3	E_D_TOV				

15.6.2.3 Buffer-to-buffer Credit

The buffer-to-buffer Credit field (word 0, bits 15 to 0) defines the number of buffers available for holding Class 1 and Class 6 connect-request, Class 2, or Class 3 frames received. An FC_Port tracks buffer-to-buffer Credit as a single entity for all frames subject to buffer-to-buffer flow control (see clause 18). Values in the buffer-to-buffer Credit field have the same format as in the end-to-end Credit Class Service Parameter (see 15.6.5.9).

For N_Port Login, this field shall only be meaningful for an Nx_Port in a point-to-point topology and between two NL_Ports on the same loop.

15.6.2.4 Common Features

15.6.2.4.1 Continuously increasing relative offset

0 = not supported

1 = supported

If the continuously increasing relative offset bit (word 1, bit 31) is set to one, the Nx_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. This bit shall only be applicable to those Information Categories in which an Nx_Port supports relative offset (i.e., word 3, bits 15 to 0). See 9.13 and 19.5 for the use of continuously increasing relative offset.

This bit shall be applicable to a Sequence Initiator in addition to a Sequence Recipient for all Classes of Service supported by the Nx_Port.

15.6.2.4.2 Clean Address

0 = No information
1 = Clean Address

The Clean Address bit (word 1, bit 31) provides an indication to an Nx_Port as to whether the address it was assigned by the Fabric had been previously used by another device within R_A_TOV. If this bit is set to zero, the assigned address may or may not have been used by a previous device within R_A_TOV. If this bit is set to one, the assigned address has not been used by any other device within R_A_TOV, or has been assigned to the current device for a previous FLOGI and not been changed within R_A_TOV. This bit is only meaningful in the FLOGI LS_ACC, it is not meaningful in the FLOGI request.

15.6.2.4.3 Random relative offset

0 = not supported
1 = supported

The random relative offset bit (word 1, bit 30) indicates that the Nx_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. This bit shall only be applicable to those Information Categories in which an Nx_Port supports relative offset (i.e., word 3, bits 15 to 0). See 19.5 for the use of random relative offset.

This bit shall be applicable to a Sequence Initiator in addition to a Sequence Recipient for all Classes of Service supported by the Nx_Port.

15.6.2.4.4 Valid Vendor Version Level

0 = not valid
1 = Valid

In PLOGI, PLOGI LS_ACC, and FLOGI, if the Valid Vendor Version Level bit (word 1, bit 29) is set to one, the Vendor Version Level (words 25 through 28 in table 216) contains valid information. If it is set to zero, the Vendor Version Level field is not meaningful.

15.6.2.4.5 Multiple N_Port_ID Assignment

0 = not supported
1 = supported

The Multiple N_Port_ID Assignment bit (word 1, bit 29) shall be set to one to indicate that the F_Port supplying this parameter is capable of assigning multiple N_Port_IDs to the attached N_Port using the FDISC ELS. The Multiple N_Port_ID Assignment bit shall be set to zero to indicate that the F_Port is not capable of assigning multiple N_Port_IDs to the attached N_Port. This bit is only meaningful in the FLOGI LS_ACC, it is not meaningful in the FLOGI request.

15.6.2.4.6 N_Port/F_Port

0 = Nx_Port
1 = Fx_Port

The N_Port/F_Port Bit (word 1, bit 28) bit shall be set to zero for PLOGI, PLOGI LS_ACC and FLOGI. It shall be set to one for FLOGI LS_ACC.

15.6.2.4.7 BB_Credit Management

0 = BB_Credit management specified in 18.5 shall be used.
1 = Alternate BB_Credit management specified in 18.5.10 shall be used.

The BB_Credit Management bit (word 1, bit 27) specifies the type of BB_Credit Management to be used.

15.6.2.4.8 E_D_TOV Resolution

0 = 1 ms

1 = 1 ns

The E_D_TOV resolution bit (word 1, bit 26) indicates the resolution of the E_D_TOV timer. If the bit is set to zero, the timer shall be in increments of 1 ms. If the bit is set to one, the timer shall be in increments of 1 ns. See 21.2.1.3 for the definition of E_D_TOV.

15.6.2.4.9 Multicast

0 = Multicast not supported by the Fabric

1 = Multicast supported by the Fabric

The Multicast bit (word 1, bit 25) indicates whether multicast is supported by the Fabric or not. It is only meaningful in the FLOGI LS_ACC. For multicast requirements see clause 23.

15.6.2.4.10 Broadcast

0 = Broadcast not supported by the Fabric

1 = Broadcast supported by the Fabric

The Broadcast bit (word 1, bit 24) indicates whether broadcast is supported by the Fabric or not. It is only meaningful in the FLOGI LS_ACC. For broadcast requirements see 23.4.

15.6.2.4.11 Hunt Group

0 = Hunt Groups not supported.

1 = Hunt Groups supported.

The Hunt Group bit (word 1, bit 23) indicates whether or not the Fabric supports Hunt Group routing. If it is set to zero, the Fabric shall not support Hunt Group routing. If it is set to one, the Fabric shall support Hunt Group routing. (see clause 22)

15.6.2.4.12 Query Buffer Conditions

0 = No buffer conditions to report.

1 = Buffer conditions to report.

The Query Buffer Conditions bit (word 1, bit 22) indicates whether or not the FC_Port has unusual buffer conditions to report. A Report Port Buffer Conditions (RPBC) ELS may be issued to gather the buffer conditions. An FC_Port shall only set the Query Buffer Conditions to 1 if the FC_Port supports the RPBC ELS, and any of the following conditions are true:

- a) The ELS Receive Buffer Field Size field is different that the Buffer-to-buffer Receive Data_Field Size field in the common service parameters, or
- b) multi-frame ELSs are not supported.

15.6.2.4.13 Clock Synchronization Primitive Capable

0 = not Clock Synchronization Primitive Capable

1 = Clock Synchronization Primitive Capable

The Clock Synchronization Primitive Capable bit (word 1, bit 20) indicates support for the Primitive method of Clock Synchronization (see 27.3). If the bit is set to zero, the Primitive method of Clock Synchronization is not supported.

If the bit is set to one, the meaning is as defined in table 221.

Table 221 - Clock Synchronization Applicability

	Meaning if bit set to one
Clock Synchronization Server (N_Port Login only)	The Clock Synchronization Server is capable of generating clock synchronization Primitive Signals.
Other FC_Ports (N_Port Login only)	The FC_Port is capable of receiving the clock synchronization Primitive Signals and acting upon them.
Fabric (FLOGI LS_ACC only)	The Fabric is capable of receiving the clock synchronization Primitive Signals and acting upon them.

15.6.2.4.14 R_T_TOV value

0 = Default value of 100 ms.

1 = Short value of 100 µs.

The R_T_TOV value (word 1, bit 19) indicates support for the short value of R_T_TOV. If this bit is set to zero, the default value of 100 ms is specified. If it is set to one, the value of 100 µs is specified.

15.6.2.4.15 Dynamic Half Duplex Supported

0 = DHD not supported

1 = DHD supported

The Dynamic Half Duplex Supported bit (word 1, bit 18) indicates support for Dynamic Half Duplex. If it is set to zero, Dynamic Half Duplex is not supported. If it is set to one, the FC_Port supports the reception of the DHD primitive.

NOTE DHD is applicable to FC-AL topologies. See [2], FC-AL-2, for a description of DHD behavior.

15.6.2.4.16 SEQ_CNT

0 = Normal rules apply to SEQ_CNT

1 = Continuously Increasing SEQ_CNT shall be used.

SEQ_CNT (word 1, bit 17) indicates the requirement on SEQ_CNT. If the bit is set to zero, normal rules shall apply to SEQ_CNT usage. If the bit is set to one, the Nx_Port is guaranteeing that it shall transmit all frames within an Exchange using a continuously increasing SEQ_CNT. Each Exchange shall start with SEQ_CNT set to zero 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 Nx_Port in the Exchange shall have no effect on the transmitted SEQ_CNT (see 9.10).

15.6.2.4.17 Payload Bit

0 = Payload length is 116 bytes

1 = Payload length in bytes is 256 plus four times the value of the Login Extension Length field

The Payload bit (word 1 bit 16) indicates the length of the FLOGI or PLOGI Payload. If it is set to zero, the Payload length shall be 116 bytes. If it is set to one, the Payload length in bytes shall be 256 plus four times the value of the Login Extension Length field.

If Payload bit is 0 in PLOGI or FLOGI, then LS_ACC sent in response shall have the Payload bit set to 0.

15.6.2.5 BB_SC_N

The Buffer-to-buffer State Change Number (BB_SC_N) field (word 1, bits 15 to 12) specifies the Buffer-to-buffer State Change Number. It 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. See 18.5.11 for a description of the BB_Credit recovery process.

15.6.2.6 Buffer-to-buffer Receive Data_Field size

The buffer-to-buffer Receive Data_Field Size field (word 1, bits 11 to 0) specifies the largest FT_1 frame Data_Field Size that may be received by the Nx_Port supplying the Service Parameters as a Sequence Recipient for:

- a) a connect-request (SOFc1),
- b) a Class 2 Data frame, or
- c) a Class 3 Data frame

The value shall be a multiple of four bytes. Values less than 256 or greater than 2^{112} are invalid. An Fx_Port shall support a Data_Field size of at least 256 bytes.

15.6.2.7 Total Concurrent Sequences

Total Concurrent Sequences field (word 3, bits 23 - 16) specifies the total number of Concurrent Sequences for all classes that the Nx_Port is capable of supporting as a Recipient.

The Total Concurrent Sequences specified by an Nx_Port shall be less than or equal to the sum of the Concurrent Sequences supported on a Class by Class basis (for example, an Nx_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 open may be fifteen).

15.6.2.8 Relative offset by category

The relative offset by category field (word 2, bits 15 to 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 the others are set to zero, Information Category '1110' b and '0010' b frames shall be capable of using relative offset as a Sequence Recipient or a Sequence Initiator). See 9.3.3 for definition of the Information Category field.

15.6.2.9 R_A_TOV

The R_A_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.

15.6.2.10 E_D_TOV

When the E_D_TOV Resolution bit (word 1, bit 26) is set to zero, the E_D_TOV value shall be specified as a count of 1 ms increments. When the E_D_TOV Resolution bit is set to one, the E_D_TOV value shall be specified as a count of 1 ns increments (for example, based on the setting of the E_D_TOV Resolution bit, a value of hex '00 00 00 0A' specifies a time period of either 10 ms or 10 ns).

For PLOGI, the E_D_TOV value in the LS_ACC to the PLOGI shall be greater than or equal to the value in the PLOGI. The E_D_TOV value in the LS_ACC shall be the value used by each Nx_Port. R_A_TOV shall be a value twice the E_D_TOV value in a point-to-point topology. See 21.2.1.3 for definition of E_D_TOV.

15.6.3 N_Port_Name

The N_Port_Name is an eight-byte field (words 5 to 6) that identifies an FC_Port. Each Nx_Port shall provide a unique N_Port_Name within the address domain of the Fabric and associated Nx_Ports. Bits 63 to 60 specify the

format of the N Port Name. The formats are defined in clause 14.

15.6.4 Node or Fabric Name

Node Name is applicable to PLOGI, PLOGI LS ACC and FLOGI. Fabric Name is applicable to FLOGI LS ACC.

The Node_ or Fabric_Name is an eight-byte field (words 7 to 8) that identifies a Node or Fabric for identification purposes, such as diagnostics that is independent and unrelated to network addressing. Each Node or Fabric shall provide a unique name within the address domain of the Fabric and associated Nx_Ports. Bits 63 to 60 specify the format of the name. The formats are defined in clause 14.

15.6.5 Class Service Parameters

15.6.5.1 Applicability

Table 222 defines the applicability, by class as well as by PLOGI, FLOGI, PLOGI LS_ACC and FLOGI LS_ACC, of the Class Service Parameters to N_Port and Fabric Login. The Class 1 and Class 6 Service parameters are given in words 9 to 12. The Class 2 Service parameters are given in words 13 to 16. The Class 3 Service parameters are given in words 17 to 20. The Class 4 Service parameters are given in words 21 to 24. The words given in the second column and in the following subclauses are relative to the start of the specific class service parameters field (see table 222)..

Table 222 - Class Service Parameters Applicability

Service Parameter	Word	Bits	PLOGI and PLOGI LS_ACC Parameter applicability				FLOGI Parameter applicability				FLOGI LS_ACC Parameter applicability			
			Class				Class				Class			
			1 ^a	2	3	4	1 ^a	2	3	4	1 ^a	2	3	4
Class Validity	0	31	y	y	y	y	y	y	y	y	y	y	y	y
Service Options	0	30 to 16												
Intermix Mode	0	30	y	n	n	n	y	n	n	n	y	n	n	n
Stacked Connect-Requests	0	29 to 28	n	n	n	n	n	n	n	n	y	n	n	n
Sequential delivery	0	27	n	n	n	n	n	y	y	n	n	y	y	n
Simplex dedicated connection - obsolete	0	26	n	n	n	n	n	n	n	n	n	n	n	n
Camp-On - obsolete	0	25	n	n	n	n	n	n	n	n	n	n	n	n
Buffered Class 1 - obsolete	0	24	n	n	n	n	n	n	n	n	n	n	n	n
Priority/Preemption	0	23	y	y	y	y	y	y	y	y	y	y	y	y
Preference	0	22	n	y	y	n	n	y	y	n	n	y	y	n
DiffServ QoS	0	21	n	y	y	n	n	y	y	n	n	y	y	n
Reserved	0	20 to 16	n	n	n	n	n	n	n	n	n	n	n	n
Initiator Control	0	15 to 0												
X_ID Reassignment - Obsolete	0	15 to 14	n	n	n	n	n	n	n	n	n	n	n	n

Legend
 “y” indicates yes, applicable (i.e., has meaning);
 “n” indicates no, not applicable (i.e., has no meaning)

^a The Class 1 Service Parameters shall be used for Class 6. Each has the same applicability as Class 1.

Table 222 - Class Service Parameters Applicability (Continued)

Service Parameter	Word	Bits	PLOGI and PLOGI LS_ACC Parameter applicability				FLOGI Parameter applicability				FLOGI LS_ACC Parameter applicability			
			Class				Class				Class			
			1 ^a	2	3	4	1 ^a	2	3	4	1 ^a	2	3	4
Initial Responder Process_Associator	0	13 to 12	y	y	y	y	n	n	n	n	n	n	n	n
ACK_0 capable	0	11	y	y	n	y	n	n	n	n	n	n	n	n
ACK_N Capable - Obsolete	0	10	n	n	n	n	n	n	n	n	n	n	n	n
ACK generation assistance	0	9	y	y	n	y	n	n	n	n	n	n	n	n
Data compression capable - obsolete	0	8	n	n	n	n	n	n	n	n	n	n	n	n
Data compression history buffer size - obsolete	0	7 to 6	n	n	n	n	n	n	n	n	n	n	n	n
Data Encryption Capable - obsolete	0	5	n	n	n	n	n	n	n	n	n	n	n	n
Clock Synchronization ELS capable	0	4	y	y	y	y	y	y	y	y	y	y	y	y
Reserved	0	3 to 0	n	n	n	n	n	n	n	n	n	n	n	n
Recipient Control	1	31 to 16												
ACK_0 Capable	1	31	y	y	n	y	n	n	n	n	n	n	n	n
ACK_N Capable - Obsolete	1	30	n	n	n	n	n	n	n	n	n	n	n	n
X_ID interlock	1	29	y	y	n	y	n	n	n	n	n	n	n	n
Error policy support	1	28 to 27	y	y	y	y	n	n	n	n	n	n	n	n
Categories per Sequence	1	25 to 24	y	y	y	y	n	n	n	n	n	n	n	n
Data compression capable - obsolete	1	23	n	n	n	n	n	n	n	n	n	n	n	n
Data compression history buffer size - obsolete	1	22 to 21	n	n	n	n	n	n	n	n	n	n	n	n
Data decryption capable – obsolete	1	20	n	n	n	n	n	n	n	n	n	n	n	n
Clock Synchronization ELS capable	1	19	y	y	y	y	y	y	y	y	y	y	y	y
Reserved - Fabric Specific	1	18 to 16	n	n	n	n	n	n	n	n	n	n	n	n
Reserved	1	15 to 12	n	n	n	n	n	n	n	n	n	n	n	n
Legend “y” indicates yes, applicable (i.e., has meaning); “n” indicates no, not applicable (i.e., has no meaning)														
^a The Class 1 Service Parameters shall be used for Class 6. Each has the same applicability as Class 1.														

Table 222 - Class Service Parameters Applicability (Continued)

Service Parameter	Word	Bits	PLOGI and PLOGI LS_ACC Parameter applicability				FLOGI Parameter applicability				FLOGI LS_ACC Parameter applicability			
			Class				Class				Class			
			1 ^a	2	3	4	1 ^a	2	3	4	1 ^a	2	3	4
Receive Data Field Size	1	11 to 0	y	y	y	y	n	n	n	n	n	n	n	n
Reserved	2	31 to 24	n	n	n	n	n	n	n	n	n	n	n	n
Concurrent Sequences	2	23 to 16	y	y	y	y	n	n	n	n	n	n	n	n
Nx_Port End-to-end Credit	2	15 to 0	y	y	n	n	n	n	n	n	n	n	n	n
Reserved	3	31 to 24	n	n	n	n	n	n	n	n	n	n	n	n
Open Sequences per Exchange	3	23 to 16	y	y	y	y	n	n	n	n	n	n	n	n
Reserved	3	15 to 0	n	n	n	n	n	n	n	n	n	n	n	n
CR_TOV	3	31 to 0	n	n	n	n	n	n	n	n	y	n	n	n
Legend “y” indicates yes, applicable (i.e., has meaning); “n” indicates no, not applicable (i.e., has no meaning)														
^a The Class 1 Service Parameters shall be used for Class 6. Each has the same applicability as Class 1.														

15.6.5.2 Payload

The Payload Class Service Parameters using FLOGI is shown in table 223.

Table 223 - Class Service Parameters - FLOGI

Bits Word	31 .. 24	23 .. 16	15 .. 12	11 .. 08	07 .. 00
0	Service Options		Initiator Control		
1	Recipient Control		Reserved		
2	Reserved	Total Concurrent Sequences	Nx_Port End-to-End credit		
3	Reserved	Open Sequences per Exchange	Reserved		

The Payload Service Parameters using PLOGI and PLOGI LS_ACC is shown in table 224.

Table 224 - Class Service Parameters - PLOGI and PLOGI LS_ACC

Bits Word	31 .. 24	23 .. 16	15 .. 00
0	Service Options		Initiator Control
1	Recipient Control		Reserved Receive Data Field Size
2	Reserved	Total Concurrent Sequences	Nx_Port End-to-End credit
3	Reserved	Open Sequences per Exchange	Reserved

The Payload Parameters using FLOGI LS_ACC is shown in table 225.

Table 225 - Class Service Parameters - FLOGI LS_ACC

Bits Word	31 .. 16	15 .. 00
0	Service Options	Reserved
1	Recipient Control	Reserved
2	Reserved	Reserved
3	CR TOV	

15.6.5.3 Class validity

0 = Invalid - Class not supported

1 = Valid - Class supported

The Class validity bit (word 0, bit 31) shall indicate whether this Class is supported or not. If the Class validity bit is set to zero, this set of sixteen bytes shall be ignored. If the Class validity bit is one, this Class shall be supported.

The Class 1 and Class 6 Service parameters are given in words 9 to 12. The Class 2 Service parameters are given in words 13 to 16. The Class 3 Service parameters are given in words 17 to 20. The Class 4 Service parameters are given in words 21 to 24.

NOTE There is no separate set of class service parameters for Class 6. The Class 1 service parameters are used for Class 6.

15.6.5.4 Service options

15.6.5.4.1 Introduction

The service options shall specify optional features of a class of service supported by the port supplying the service parameters.

15.6.5.4.2 Intermix Mode

0 = Intermix not requested

1 = Intermix requested

The Intermix Mode bit (word 0, bit 30) only has meaning in Class 1 and 6. It has no meaning in Classes 2, 3, and 4.

All N_Ports and Fabrics supporting Class 1 shall support exclusive connections. An 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, as appropriate, in response to Class 2 frames and connect-request Data frames (SOFc1) 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 N_Ports and Fabrics. When an N_Port performs Login with a Fabric, it shall request support for Intermix by setting bit the Intermix Mode bit to one. If the Fabric responds with bit 30 set to one in the LS_ACC, Intermix shall be functional.

For Fabric Login, table 226 specifies the meaning of the combination of the Intermix Mode bit for the requesting N_Port and the applicable F_Port.

Table 226 - Intermix Mode Support

Nx_Port	Fx_Port	Meaning
0	0	Neither supports
0	1	Fabric is capable of supporting, Intermix not functional
1	0	Nx_Port support requested, Fabric does not support
1	1	Nx_Port requested, Fabric is capable of supporting, Intermix is functional

For N_Port Login, this bit indicates that Intermix is functional between the N_Port setting this bit 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.

See 13.5 for Intermix requirements.

15.6.5.4.3 Stacked Connect-requests

The Stack Connect-request bits (word 0, bits 29 to 28) are used in the Fabric Login procedure. They are only meaningful for Class 1 in the FLOGI LS_ACC. They are not meaningful for other classes and other Class 1 login frames. Table 227 specifies the meaning of the combination of word 0 bits 29 and 28.

Table 227 - Stacked Connect-request support Login Bits

Word 0, bits 29 to 28	Meaning
'00' b	Stacked Connect-request not supported
'01' b	Lock-Down Mode
'10' b	Transparent Mode
'11' b	Reserved

Support for stacked connect-requests is optional in both a Fabric and Nx_Port. Both an Nx_Port's and Fx_Port's behavior change if stacked connect-requests are functional (See 20.5.3 and clause 26).

15.6.5.4.4 Sequential delivery

0 = Out of order delivery allowed
1 = Sequential delivery requested

The Sequential delivery bit (word 0, bit 27) is only meaningful for the class 2 and 3 N_Port Class Service Parameters in Fabric Login. Out of order frame delivery in class 2 and 3 is the default function by a Fabric. It is not meaningful for Classes 1, 4 or 6 or for N_Port Login.

If this bit is set to one by an Nx_Port, it is requesting that all frames delivered to the Nx_Port requesting this function be delivered in the same order in which the frames were transmitted by the source Nx_Port. If this bit is set to one by the Fx_Port in the LS_ACC, the Fx_Port shall deliver Class 2 and 3 frames in the same order as transmitted from any one Nx_Port.

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).

If this bit is set to one, the Fabric guarantees the order of delivery of both Data and Link_Control (Class 2 only) frames to the Nx_Port requesting this feature in the same order in which the frames were transmitted. Table 228 specifies the meaning of the combination of the Sequential delivery bit for the requesting Nx_Port and the applicable Fx_Port.

Table 228 - Sequential delivery support

Nx_Port	Fx_Port	Meaning
0	0	Neither supports
0	1	Fabric is capable of supporting, Sequential delivery is functional
1	0	Nx_Port support requested, Fabric does not support
1	1	Nx_Port requested, Fabric is capable of supporting, Sequential delivery is functional

15.6.5.4.5 Priority/Preemption

0 = Priority/Preemption is not supported

1 = Priority/Preemption is supported

See 9.5.3.

The Priority/Preemption bit (word 0, bit 23) has meaning in Class 1, 2, 3, 4 and 6.

When an Nx_Port performs Login with a Fabric, it shall request support for Priority and Preemption by setting the Priority and Preemption bit (word 0, bit 23) to one. If is set to one in the LS_ACC, then both the Nx_Port and Fabric have agreed that Priority and Preemption is available for use.

The set of values specified in table 229 give the meaning of the combination of the Priority and Preemption bit.

Table 229 - Fabric Login Priority and Preemption Support

Nx_Port	Fx_Port	Meaning
0	0	Neither supports Priority and Preemption
0	1	Fabric is capable of supporting Priority and Preemption
1	0	Nx_Port support requested, Fabric does not support Priority and Preemption
1	1	Nx_Port requested, Fabric is capable of supporting Priority and Preemption, available for use

15.6.5.4.6 Preference

15.6.5.4.6.1 Nx_Port

0 = non-zero CS_CTL may be tolerated

1 = non-zero CS_CTL shall be tolerated

When an Nx_Port performs Login with another Nx_Port, it shall indicate tolerance for non-zero CS_CTL within the Class of Service by set the Preference bit (word 0, bit 22) to one. The other Nx_Port indicates tolerance for non-zero CS_CTL by setting this bit to one in the LS_ACC. An Nx_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 an Nx_Port never intends to set the PREF bit to any value other than zero, the Nx_Port may still have reason to set the login Preference bit to one. Setting the bit to one indicates to the other Nx_Port that the Nx_Port shall accept frames with a non-zero CS_CTL field value.

In Class 2 and 3, if this bit is set to one, the Nx_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 Nx_Port shall ignore the PREF field (See 13.3.2, p and 13.4.2, p).

This Preference Bit has no meaning in Class 1, 4 and 6.

15.6.5.4.6.2 Fx_Port

0 = Normal delivery

1 = Preferred delivery functional

If the Preference bit (word 0, bit 22) is set to one by an Nx_Port, then it is requested that all frames transmitted by the Nx_Port requesting this function be delivered according to the setting of the PREF field in the CS_CTL field of the Frame_Header. If this bit is set to one by the Fx_Port, the Fx_Port is indicating that it shall deliver Class 2 and 3 frames transmitted by the requesting Nx_Port according to the setting of the PREF field.

NOTE An Fx_Port that responds with bit 22 set to zero may not itself support Preferred delivery, but other Fabric Elements in the path to the destination may support it. An Nx_Port may attempt Preferred delivery even if the Fx_Port does not indicate support.

If this bit is set to one, the Fabric shall deliver both Data and Link_Control (class 2 only) frames according to the setting of the PREF field in the CS_CTL field of the frame header.

The Preference bit is not meaningful for Class 1, 4 and 6.

Table 230 summarizes the function of the PREF bit for both Class 2 and Class 3.

Table 230 - Class 2 and 3 Preference Bit Function

Nx_Port Word 0, Bit 22	Fx_Port Word 0, Bit 22	Meaning
0	0	Preferred delivery by the Fabric may be functional
0	1	Preferred delivery by the Fabric shall be functional
1	0	Nx_Port support requested, Preferred delivery by the Fabric may be functional
1	1	Nx_Port requested, Fabric agrees, Preferred delivery by the Fabric shall be functional

Table 231 summarizes the relationship between Preferred delivery and sequential delivery for both Class 2 and

Class 3.

Table 231 - Relationship between Preferred delivery and sequential delivery

Preference Functional	Sequential delivery Functional	Meaning
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

15.6.5.4.7 DiffServ QoS

15.6.5.4.7.1 N_Port Login

0 = DiffServ QoS not supported

1 = DiffServ QoS supported

When an Nx_Port performs Login with another Nx_Port, it shall indicate support for Differentiated Services QoS by setting the DiffServ QoS bit (word 0, bit 21) to one. The other Nx_Port indicates support for Differentiated Services QoS by setting this bit to one in the LS_ACC. Support of Differentiated Services QoS as an Exchange Originator means that the CS_CTL/Priority Enable bit (F_CTL bit 17) is set to zero to indicate that the DSCP field of the CS_CTL field in the frame header (word 1, bits 29 to 24) specifies the Differentiated Services QoS policy. Support of Differentiated Services QoS as an Exchange Responder means that the CS_CTL/Priority Enable bit (F_CTL bit 17) is set to zero, indicates that the DSCP field of the CS_CTL field in the frame header (word 1, bits 29 to 24) specifies the Differentiated Services QoS policy (see 9.5.2).

The DiffServ QoS bit only has meaning for Classes 2 and 3.

15.6.5.4.7.2 F_Port Login

0 = DiffServ QoS not Supported

1 = DiffServ QoS Supported

When doing Fabric Login, an Nx_Port shall request support for Differentiated Services QoS by setting the DiffServ QoS bit (word 0, bit 21) to one. If the LS_ACC reply from the Fx_Port has this bit set to one, both the Nx_Port and Fx_Port have agreed that Differentiated Services QoS is available for use.

The set of values specified in table 232 give the meaning of the combination of Word 0, bit 21 between the requesting Nx_Port and the responding Fx_Port (see 9.5.2).

The DiffServ QoS bit only has meaning for Classes 2 and 3.

Table 232 - DiffServ QoS bit definition

Nx_Port	Fx_Port	Description
0	0	Neither supports Differentiated Services QoS
0	1	Fx_Port is capable of supporting Differentiated Services QoS
1	0	Nx_Port support requested, Fx_Port does not support Differentiated Services QoS
1	1	Nx_Port requested, Fx_Port is capable of supporting Differentiated Services QoS, Differentiated Services QoS is available for use

15.6.5.5 Initiator control

15.6.5.5.1 Introduction

The Initiator Control Flags shall specify which protocols, policies or functions the Sequence Initiator function in the Nx_Port supplying the Service Parameters requests of the recipient or is capable of as a Sequence initiator.

15.6.5.5.2 Initial Process_Associator

The definition of the Initial Process_Associator bits (word 0, bits 13 to 12) is shown in table 233.

Table 233 - Initial Process_Associator Bits Definition

Word 0, bits 13 to 12	Meaning
'00' b	Initial Process_Associator not supported
'01' b	Initial Process_Associator supported
'10' b	Reserved
'11' b	Initial Process_Associator required (and supported)

Initial Process_Associator required indicates that the Nx_Port supplying this parameter requires an Association_Header at certain Sequence boundaries (see 10.4) that contains a specific initial value in the Process_Associator field. An Nx_Port that 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. If an Initial Process_Associator is required or supported, then X_ID interlock also is required in Class 1 and 2.

If the Responder Nx_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 Nx_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 Nx_Ports are unable to communicate.

These bits only have meaning for PLOGI and PLOGI LS_ACC.

15.6.5.5.3 ACK_0 capability

0 = ACK_0 incapable

1 = ACK_0 capable

The ACK_0 capability bit (word 0, bit 11) specifies if the Nx_Port supplying these Class Service Parameters is 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 an Nx_Port receives and processes ACK frames in response to Data frame transmission. ACK_0 support is applicable to acknowledged class of service Sequences.

The conditions under which ACK_0 is support is defined in table 234 and described in the following text.

Table 234 - ACK_0 Support Conditions (Initiator Control)

Nx_Port A Word 0, Bit 11	Nx_Port B Word 1, Bit 31	Nx_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 Nx_Port (for example, Nx_Port A) is capable of receiving ACK_0 as a Sequence Initiator (word 0, bit 11 set to one) and the other Nx_Port (for example, Nx_Port B) is capable of transmitting ACK_0 as a Sequence Recipient (word 1, bit 31 set to one), ACK_0 is supported when Nx_Port A is the Sequence Initiator and Nx_Port B is the Sequence Recipient. Otherwise, ACK_0 shall not be supported while Nx_Port A is the Sequence Initiator and Nx_Port B is the Sequence Recipient. ACK_0 usage shall take precedence over ACK_1.

If ACK_0 is supported, Nx_Port A shall transmit Data frames to Nx_Port B with infinite buffering in effect and shall be able to receive ACK_0 for Sequences transmitted. If ACK_0 is supported, Nx_Port B shall support infinite buffering and be capable of transmitting ACK_0 to acknowledge an entire Sequence (See 11.2.2.3.3 and 18.4.3.2).

ACK_0 capability may be asymmetrical for a single Nx_Port (i.e., an Nx_Port may be capable processing ACK_0 as a Sequence Initiator, but not be capable of ACK_0 transmission as a Sequence Recipient). Similarly, an Nx_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 this standard (for example, if an FC-4 only transmitted Video_Data from Nx_Port A to Nx_Port B (for example, a frame buffer), Nx_Port A need only support ACK_0 as a Sequence Initiator, whereas, Nx_Port B need only support ACK_0 as a Sequence Recipient). See 15.6.5.5.4 for additional requirements regarding the use of ACK_0.

15.6.5.5.4 ACK generation assistance

0 = No ACK generation assistance is provided to Sequence Recipient.

1 = ACK generation assistance is provided to Sequence Recipient.

See 11.3.

15.6.5.5.5 Clock synchronization ELS capable

0 = Initiator does not have clock synchronization ELS capability

1 = Initiator has clock synchronization ELS capability

The Clock synchronization ELS capable bit (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 this bit is set to zero, the Nx_Port does not support the ELS method of Clock Synchronization. If this bit is set to one, the Nx_Port is capable of generating the CSU ELS frames. See 27.2.

15.6.5.6 Recipient control

15.6.5.6.1 Introduction

The Recipient Control Flags shall specify which protocols, policies or functions the Recipient Initiator function in the Nx_Port supplying the Service Parameters when acting as a receiver of Data frames.

15.6.5.6.2 ACK_0 capability

0 = ACK_0 incapable

1 = ACK_0 capable

The ACK_0 capability bit (word 1, bit 31) specifies that the Nx_Port supplying these Class Service Parameters may or may not be 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 an Nx_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 or service Sequences.

The conditions under which ACK_0 is support is defined in table 235 and described in the following text.

Table 235 - ACK_0 Support Conditions (Recipient Control)

Nx_Port A Word 0, Bit 11	Nx_Port B Word 1, Bit 31	Nx_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 Nx_Port (for example, Nx_Port A) is capable of receiving ACK_0 as a Sequence Initiator (Word 0, Bit 11 set to one) and the other Nx_Port (for example, Nx_Port B) is capable of transmitting ACK_0 as a Sequence Recipient (Word 1, Bit 31 set to one), then ACK_0 may be used when Nx_Port A is the Sequence Initiator and Nx_Port B is the Sequence Recipient. Otherwise, ACK_0 shall not be supported while Nx_Port A is the Sequence Initiator and Nx_Port B is the Sequence Recipient. If ACK_0 is supported, as indicated, Nx_Port A shall transmit Data frames to Nx_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, Nx_Port B shall support infinite buffering and be capable of transmitting ACK_0 to acknowledge an entire Sequence (See 11.2.2.3.3 and 18.4.3.2).

ACK_0 capability may be asymmetrical for a single Nx_Port (i.e., an Nx_Port may be capable processing ACK_0 as a Sequence Initiator, but not be capable of ACK_0 transmission as a Sequence Recipient. Similarly, an Nx_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 an Nx_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 this standard (for example, if an FC-4 only transmitted Video_Data from Nx_Port A to Nx_Port B (a frame buffer), then Nx_Port A need only support ACK_0 as a Sequence Initiator, whereas, Nx_Port B need only support ACK_0 as a Sequence Recipient).

15.6.5.6.3 X_ID interlock

0 = X_ID interlock not required
1 = X_ID interlock required

X_ID interlock (word 1, bit 29) only applies to Class 1 and Class 2. This bit indicates that the Nx_Port supplying this parameter requires that an interlock be used during X_ID assignment 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 (See 17.5.5).

15.6.5.6.4 Error policy supported

The definition of the Error policy supported bits (word 1, bits 28 to 27) is shown in table 236.

Table 236 - Error Policy Bits Definition

Word 1, bits 28 to 27	Meaning
'00' b	Only discard policy supported
'01' b	Reserved
'10' b	Both discard and process policies supported
'11' b	Reserved

These bits are set to specify the types of support possible for missing frame conditions processed by the receiver Nx_Port. The policy used for a given Exchange shall be specified as discard or process by the Exchange Originator (see 9.7.14 and 21.6.3).

In either Discard or Process policy, when a missing frame error is detected, the expected SEQ_CNT 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 Nx_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 17.3.10). In one discard policy, only a single Sequence shall be discarded, whereas in the other discard policy, multiple Sequences shall be discarded (see 9.7.14 and 21.6.3).

Process policy support specifies that the Nx_Port is to 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 21.6.3).

15.6.5.6.5 Categories per Sequence

The definition of the Categories per Sequence bits (word 1, bits 25 to 24) is shown in table 237.

Table 237 - Categories per Sequence Bits Definition

Word 1, bits 25 to 24	Meaning
'00' b	1 Category/Sequence
'01' b	2 Categories/Sequence
'10' b	Reserved
'11' b	More than 2 Categories/Sequence

The setting of these bits shall specify that the Recipient is capable of processing one, two, or more than two Information Categories (R_CTL bits 27 to 24 in the Frame_Header) in a single Sequence. Bits 25 to 24 are applicable to each Class of Service since support for an individual Class may offer different capabilities in the same Nx_Port.

When an Nx_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. An Nx_Port's capability for processing Information Categories in a single Sequence may prohibit that Nx_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.

15.6.5.6.6 Clock synchronization ELS capable

0 = Recipient does not have clock synchronization ELS capability

1 = Recipient has clock synchronization ELS capability

The Clock synchronization capable bit (word 1, bit 19) indicates support for the ELS method of Clock Synchronization. If this bit is set to zero, the Nx_Port or Fabric does not support the ELS method of Clock Synchronization. If this bit is set to one, the Nx_Port or Fabric is capable of receiving CSU ELS frames and acting upon them (See 27.2).

15.6.5.7 Receive Data_Field Size

The Receive Data_Field Size is a value (word 1, bits 11 to 0) that specifies the largest Data_Field Size for an FT_1 frame (see 8.5) that may be received by the Nx_Port supplying the Service Parameters as a Sequence Recipient. Values less than 256 or greater than 2 112 are invalid. Values shall be a multiple of four bytes. An Nx_Port shall support a Data Field size of at least 256 bytes.

In Class 1 and 6 the Receive Data_Field size represents the largest Data_Field size that an Nx_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 (see 15.6.2.6).

The Receive Data_Field size for Class 2, 3 and 4 shall be equal to or less than the Buffer-to-Buffer Receive Data_Field size specified in the Common Service Parameters.

15.6.5.8 Concurrent Sequences

Concurrent Sequences (word 2, bits 23 to 16) shall specify the number of Sequence Status Blocks available in the

Nx_Port supplying the Service Parameters for tracking the progress of a Sequence as a Sequence Recipient. The maximum number of Concurrent Sequences that may be specified is 255 per Nx_Port as a Recipient that may be allocated across all classes. The total number of Concurrent Recipient Sequences that may be open across all classes by a single Nx_Port is specified in the Common Service Parameter field (see 15.6.2.7). The number of concurrent sequences open is specified in table 238.

Table 238 - Concurrent Sequences field meaning

Word 2, bits 23 to 16	Meaning
hex '00'	Reserved
hex '01'	1 Sequence Status Register
...	...
hex 'FF'	255 Sequence Status Registers

NOTE 1 If each Nx_Port specified 255 Concurrent Sequences, then the maximum number of open Sequences between the two Nx_Ports is 510 (255 as Recipient and 255 as Initiator, for each Nx_Port).

In Class 1 and 6, 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 an Nx_Port shall support the maximum number of Concurrent Sequences specified during Login.

In Class 2 and 3, the SEQ_ID values shall range from 0 to 255. In Class 2 an Nx_Port may respond with a P_BSY to a frame initiating a new Sequence if Nx_Port resources are not available.

In Class 4, the SEQ_ID values shall range from 0 to L, inclusively, where L is the value of the Concurrent Sequence field. During Live Class 4 circuits a Nx_Port shall support the maximum number of Concurrent Sequences specified during Login.

NOTE 2 The ability to sustain the maximum number of concurrent sequences may be limited by the available VC_Credit.

15.6.5.9 End-to-end Credit

End-to-end Credit (word 2, bits 14 to 0) is the maximum number of Data frames that may be transmitted by an Nx_Port without receipt of an 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 Nx_Port supplying the Service Parameters. End-to-end Credit is not applicable to Class 3 since ACK frames are not used.

In order to ensure frame identification integrity, end-to-end Credit is defined as a 15-bit field while SEQ_CNT is a 16-bit field. This ensures that end-to-end Credit never exceeds one-half of the maximum SEQ_CNT. Bit 15 shall be set to zero. See 18.4.

Values in the end-to-end Credit Field have the meanings defined in table 239.

Table 239 - End-to-end Credit Field Meaning

Word 2, Bits 14 to 0	Meaning
'0000000 00000000' b	Reserved
'0000000 00000001' b	1 Buffer
.	.
'1111111 11111110' b	32 766 Buffers
'1111111 11111111' b	32 767 Buffers

15.6.5.10 Open Sequences per Exchange

The value of open Sequences per Exchange field (word 3, bits 23 to 16) shall specify the maximum number of Sequences that may be open at the Recipient at one time between a pair of Nx_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. The value of X is specified in bits 23 to 16 (see clause 21).

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.

The open Sequences per Exchange field is valid for PLOGI and PLOGI LS_ACC only.

15.6.5.11 CR_TOV

The CR_TOV value (word 3) 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 26.8.2 for CR_TOV usage. CR_TOV is applicable to Class 1 and 6. CR_TOV is not applicable to other Classes. CR_TOV is only meaningful in FLOGI LS_ACC.

15.6.6 Vendor Version Level

Vendor Version Level field (words 25 to 28) specifies vendor-specific information. If the Valid Version Level bit in the Common Service Parameters field (word 1, bit 29) is set to one, the Vendor Version Level field contains valid information. If the Valid Version Level bit is set to zero, the Vendor Version Level field is not meaningful.

15.6.7 Services Availability

15.6.7.1 Introduction

This field returns information regarding the Fabric's ability to route to the defined well-known addresses. It is meaningful only for FLOGI LS_ACC. Only bits 10 - 3 of word 30 are meaningful. Word 29 and bits 31 to 11 and 2 to 0 of word 30 are reserved (see 9.4.2).

15.6.7.2 Multicast Server

When set to one, the Multicast Server bit (word 30, bit 10) shall indicate that the Fabric supports routing to the well-known Multicast Server address identifier (hex 'FF FF F5'). When set to zero, this bit shall indicate that the Fabric does not support routing to the well-known Multicast Server address identifier.

15.6.7.3 Clock Synchronization Server

When set to one, the Clock Synchronization Server bit (word 30, bit 9) shall indicate that the Fabric supports routing to the well-known Clock Synchronization Server address identifier (hex 'FF FF F6'). When set to zero, this bit shall indicate that the Fabric does not support routing to the well-known Clock Synchronization Server address identifier.

15.6.7.4 Security Key Distribution Server

When set to one, the Security Key Distribution Server bit (word 30, bit 8) shall indicate that the Fabric supports routing to the well-known Key Distribution Server address identifier (hex 'FF FF F7'). When set to zero, this bit shall indicate that the Fabric does not support routing to the well-known Key Distribution Server address identifier.

15.6.7.5 Alias Server

When set to one, the Alias Server bit (word 30, bit 7) shall indicate that the Fabric supports routing to the

well-known Alias Server address identifier (hex 'FF FF F8'). When set to zero, this bit shall indicate that the Fabric does not support routing to the well-known Alias Server address identifier.

15.6.7.6 Quality of Service Facilitator

When set to one, the Quality of Service Facilitator bit (word 30, bit 6) shall indicate that the Fabric supports routing to the well-known Quality of Service Facilitator address identifier (hex 'FF FF F9'). When set to zero, this bit shall indicate that the Fabric does not support routing to the well-known Quality of Service Facilitator address identifier.

15.6.7.7 Management Server

When set to one, the Management Server bit (word 30, bit 5) shall indicate that the Fabric supports routing to the well-known Management Server address identifier (hex 'FF FF FA'). When set to zero, this bit shall indicate that the Fabric does not support routing to the well-known Management Server address identifier.

15.6.7.8 Time Server

When set to one, the Time Server bit (word 30, bit 4) shall indicate that the Fabric supports routing to the well-known Time Server address identifier (hex 'FF FF FB'). When set to zero, this bit shall indicate that the Fabric does not support routing to the well-known Time Server address identifier.

15.6.7.9 Directory Server

When set to one, the Directory Server bit (word 30, bit 3) shall indicate that the Fabric supports routing to the well-known Directory Server address identifier (hex 'FF FF FC'). When set to zero, this bit shall indicate that the Fabric does not support routing to the well-known Directory Server address identifier.

15.6.8 Login Extension

15.6.8.1 General

If a port does not support a Login request with the Payload bit set to one, the port shall reject the Login request with an LS_RJT. The reason code shall be set to Logical Error with a reason code explanation of "Invalid Payload Length".

If a port does not support a Login request with a non-zero Login Extension Data Length field, the port shall reject the Login request with an LS_RJT. The reason code shall be set to Logical Error with a reason code explanation of "Login Extension not supported".

If a port receives a Login request containing a page code that it does not support, the port shall reject the Login request with an LS_RJT. The reason code shall be set to Logical Error with a reason code explanation of "Login Extension not supported".

If a port does not support a LS_ACC reply Sequence with the Payload bit set to one, the port shall perform an explicit Logout with the port that sent the reply.

15.6.8.2 Login Extension Data Length

If the Login Extension Data Length field (word 31) is non-zero, a Login Extension follows the normal payload. The Login Extension Data Length field indicates the length of the Login Extension field in words. The Payload bit (see 15.6.2.4.17) shall be set to one if this field is non-zero.

15.6.8.3 Login Extension format

The Login Extension field is a sequence of zero or more Login Extension Pages. The length in words of the sequence of Login Extension Pages shall be equal to the value of the Login Extension Length field. The format of a Login Extension Page is given in Table 240. Word 0 of the first Login Extension Page, if present, is word 64 of the

FLOGI/PLOGI/ACC payload.

Table 240 - Login Extension Page format

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Page Length (n)		Reserved	Page Code
1	Page Code Specific Data			
...				
n to 1				

Page Length Field: The length in words of the following page, including the word containing the page length and page code fields.

Page Code Field: The Page Code field specifies the type of page as shown in table 241.

Table 241 - Page Code Definitions

Page Code	Meaning
hex '00' to hex 'EF'	Reserved
hex 'F0'	Vendor Specific
hex 'F1' to hex 'FF'	Reserved

Vendor Specific Page: The format of the Vendor Specific Page is shown in table 242.

Table 242 - Vendor Specific Page format

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
0	Page Length (n)		Reserved	Page Code (hex 'F0')
1	(MSB)			
2	Vendor Identification Data			(LSB)
3	Vendor Specific Data			
...				
n to 1				

Vendor Identification field: The value of the Vendor Identification field shall be a Vendor Identifier as defined in [10], SPC-3. If the length of the Vendor Identifier is less than eight bytes, it shall be left aligned within the Vendor Identification field. The format and interpretation of the Vendor Specific Data field is vendor specific to the vendor identified by the value of the Vendor Identification field.

Vendor Specific Data Field: The Vendor Specific Data field contains vendor specific data and shall be padded to a word boundary.

15.6.9 Clock Synchronization Quality of Service

15.6.9.1 N_Port Login

15.6.9.1.1 Applicability

The Clock Synchronization Quality of Service (QoS) field in PLOGI ELS or LS_ACC (words 62 to 63) is only

meaningful when sent to or received from the Clock Synchronization Server (hex 'FF FF F6'). This field contains meaningful information only if either the Clock Synchronization Primitive Capable bit of the Common features field (word 1, bit 20) is set to one; or if the Clock Synchronization ELS capable bit of one of the N_Port Class Service Parameter Recipient control fields (word 1, bit 19) is set to one. If this field does not contain meaningful information, it shall be set to zero (See 27.2 and 27.3).

The Clock Synchronization QoS field is defined in table 243.

Table 243 - N_Port Clock Synchronization QoS

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
62	CS_QoS_Request	CS_Accuracy	CS_Implemented_MSB	CS_Implemented_LSB
63	CS_Update_Period			

15.6.9.1.2 CS_QoS_Request

For PLOGI and FLOGI Request, the meaning is defined in table 244. This field is not meaningful for PLOGI LS_ACC and shall be set to zero.

Table 244 - FLOGI/PLOGI CS_QoS_Request

Word 62, Bits 31 to 24	Meaning
hex '00'	The CS_Accuracy, CS_Implemented_MSB, CS_Implemented_LSB, and CS_Update_Period fields are not meaningful in the FLOGI/PLOGI Request.
hex '01'	The CS_Accuracy, CS_Implemented_MSB, CS_Implemented_LSB, and CS_Update_Period fields contain the requested Quality of Service parameters.
hex '02' – hex 'FF'	Reserved

15.6.9.1.3 CS_Accuracy (Mantissa and Exponent)

This field contains the CS_Accuracy_Mantissa (word 62, bits 23 to 21) and CS_Accuracy_Exponent (bits word 62, 20 to 16).

When sent to the Fabric during FLOGI, these bits indicate the accuracy that the Fabric is requested to maintain in passing along to the clients the clock synchronization value it receives from the Clock Synchronization Server (hex 'FF FF F6').

When sent to the Clock Synchronization Server (hex 'FF FF F6'), these bits indicate the requested accuracy of the clock synchronization value as it leaves the server port.

When received from the Clock Synchronization Server (hex 'FF FF F6'), these bits indicate the accuracy of the clock synchronization value as it leaves the server port. Specifically, the server shall supply a CS_Accuracy value such that the Clock Synchronization value is always within the range of:

$$T_{\text{reference}} \pm (0.5 + \text{CS\_Accuracy\_Mantissa} \times 2^{-4}) \times 2^{(\text{CS\_Accuracy\_Exponent}-30)},$$

where

- a) T_reference is the clock reference value internal to the server
- b) CS_Accuracy_Mantissa is a value from '000' b to '111' b'
- c) CS_Accuracy_Exponent is a value from '00000' b to '11111' b'

Example #1, if CS_Accuracy Mantissa and Exponent are set to '001' b and '01011' b, respectively, the Clock Synchronization value as it exits the server shall always be within the range of:

$$T_{\text{reference}} \pm 1,073 \mu\text{sec}$$

Example #2, if CS_Accuracy Mantissa and Exponent are set to '111' b and '11000' b, respectively, the Clock Synchronization value as it exits the server shall always be within the range of:

$$T_{\text{reference}} \pm 14,65 \text{ msec}$$

15.6.9.1.4 Clock Synchronization Implemented MSB

The Clock Synchronization Implemented MSB field (word 62, bits 13 to 8) is a 6-bit value. Word 62, bits 15 to 14 are reserved and shall be set to zero.

When sent to the Clock Synchronization Server (hex 'FF FF F6') during PLOGI, these bits indicate the requested most significant bit position within the 64-bit Clock Count field in the CSU ELS Payload.

When received from the Clock Synchronization Server (hex 'FF FF F6') this field represents the most significant bit position within the 64-bit Clock Count field that contains meaningful information.

NOTE The value in the Clock Count field shall wrap around to zero when an overflow occurs from the Clock Synchronization Implemented MSB.

(for example, a value of '110111' b indicates that the MSB of byte 1 of the Clock Count field is the highest bit that contains meaningful information).

15.6.9.1.5 Clock Synchronization Implemented LSB

The Clock Synchronization Implemented LSB field (word 62, bits 13 to 8) field is a 6-bit value. Word 62, bits 7 to 6 are reserved and shall be set to zero.

When sent to the Clock Synchronization Server (hex 'FF FF F6') during PLOGI, these bits indicate the requested least significant bit position within the 64-bit Clock Count field in the CSU ELS Payload.

When received from the Clock Synchronization Server (hex 'FF FF F6'), this field represents the least significant bit position within the 64-bit Clock Count field that contains meaningful information.

(for example, a value of '001000' b indicates that the LSB of byte 6 of the Clock Count field is the lowest bit that contains meaningful information).

15.6.9.1.6 Clock Synchronization Update Period

When sent to the Clock Synchronization Server (hex 'FF FF F6'), the Clock Synchronization Update Period field (word 63) indicates the requested time, in microseconds, between consecutive updates from the Clock Synchronization server.

When received from the Clock Synchronization Server (hex 'FF FF F6'), it represents the time, in microseconds, between consecutive updates from the Clock Synchronization server.

This field is not meaningful for FLOGI and shall be set to zero.

15.6.9.2 Fabric Login

15.6.9.2.1 Applicability

The Clock Synchronization Quality of Service field contains meaningful information only if either Word 1, bit 20 -

Clock Synchronization Primitive Capable of the Common features field is set to one, or Word 1, bit 19 - Clock Synchronization ELS capable of the Recipient control field is set to one. If this field does not contain meaningful information, it shall be set to zero (See 27.2 and 27.3).

The Fx_Port Clock Synchronization Quality of Service field is illustrated in table 245.

Table 245 - Fx_Port Clock Synchronization QoS

Bits Word	31 .. 24	23 .. 16	15 .. 08	07 .. 00
62	Reserved	CS_Transfer_Accuracy	CS_Implemented_MSB	CS_Implemented_LSB
63	Reserved			

15.6.9.2.2 CS_Transfer_Accuracy

The CS_Transfer_Accuracy field contains CS_Transfer_Accuracy_Mantissa (word 62, bits 23 to 21) and CS_Transfer_Accuracy_Exponent (word 62, bits 20 to 16).

These bits indicate the accuracy that the Fabric maintains in passing along to the clients the clock synchronization value it receives from the Clock Synchronization Server. Specifically, the Fabric shall supply a CS_Transfer_Accuracy value such that the Clock Synchronization value supplied to the clients is always within the range of:

$$(T_{\text{server}} + T_{\text{fabric\_delay}}) \pm (0.5 + \text{CS\_Accuracy\_Mantissa} \times 2^{-4}) \times 2^{(\text{CS\_Accuracy\_Exponent}-30)}$$

where:

- T_server is the value received from the Clock Synchronization Server
- T_fabric_delay is the time from when a given value was received from the server until the corresponding value is delivered to the client
- CS_Accuracy_Mantissa is a value from '000' b to '111' b'
- CS_Accuracy_Exponent is a value from '00000' b to '11111' b'

Example #1, if CS_Transfer_Accuracy Mantissa and Exponent are set to '001' b and '01011' b, respectively, the Clock Synchronization value supplied to the clients shall always be within the range of:

$$(T_{\text{server}} + T_{\text{fabric\_delay}}) \pm 1,073 \mu\text{s}$$

Example #2, if CS_Transfer_Accuracy Mantissa and Exponent are set to '111' b and '11000' b, respectively, the Clock Synchronization value supplied to the clients shall always be within the range of:

$$(T_{\text{server}} + T_{\text{fabric\_delay}}) \pm 14,65 \text{ ms}$$

15.6.9.2.3 Clock Synchronization Implemented MSB

The Clock Synchronization Implemented MSB field (word 62, bits 13 to 8) is a 6-bit value that represents the most significant bit position within the 64-bit Clock Count field that contains meaningful information. Word 62, bits 15 to 14 are reserved and shall be set to zero (for example, a value of '110111' b indicates that the MSB of byte 1 of the Clock Count field is the highest bit that contains meaningful information).

This field refers to the capabilities of the Fabric in transferring the clock synchronization value that was received from the Clock Synchronization Server (hex 'FF FF F6') to the clients. It does not refer to the capabilities of the Clock Synchronization Server itself.

15.6.9.2.4 Word 0, Bits 7 to 0 Clock Synchronization Implemented LSB

This Clock Synchronization Implemented LSB field (word 62, bits 5 to 0) is a 6-bit value that represents the least significant bit position within the 64-bit Clock Count field that contains meaningful information (for example, a value of '001000' b indicates that the LSB of byte 6 of the Clock Count field is the lowest bit that contains meaningful information). Word 62, bits 7 to 6 are reserved and shall be set to zero.

This field refers to the capabilities of the Fabric in transferring the clock synchronization value that was received from the Clock Synchronization Server (hex 'FF FF F6') to the clients. It does not refer to the capabilities of the Clock Synchronization Server itself.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

16 Process Login/Logout

16.1 Process Login

16.1.1 Introduction

The Process Login (PRLI) ELS request shall be used to establish the operating environment between a group of related processes at the originating Nx_Port and a group of related processes at the responding Nx_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.

A Process Login remains in affect until a process logout occurs. The number of concurrent Process Logins in effect at an Nx_Port is a function of the Nx_Port facilities available. Process Login is separate from N_Port Login. Process Login may be either implicit or explicit.

If the recipient Nx_Port is not logged in with the requesting port it shall reply with a LOGO ELS Sequence, or with an LS_RJT ELS Sequence with a reason code of "Unable to perform command request" and a reason code explanation of "N_Port Login required". There are 2 types of 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 in this standard.
- Explicit Process Login is accomplished by using the PRLI ELS 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 19). The combination of the D_ID, Responder Process_Associator, S_ID and Originator Process_Associator identify the image pair (see figure 20). Either a single group or multiple groups of related processes may exist behind an Nx_Port. A single group of related processes behind an Nx_Port may be denoted by either an N_Port_ID only or N_Port_ID and Process_Associator.

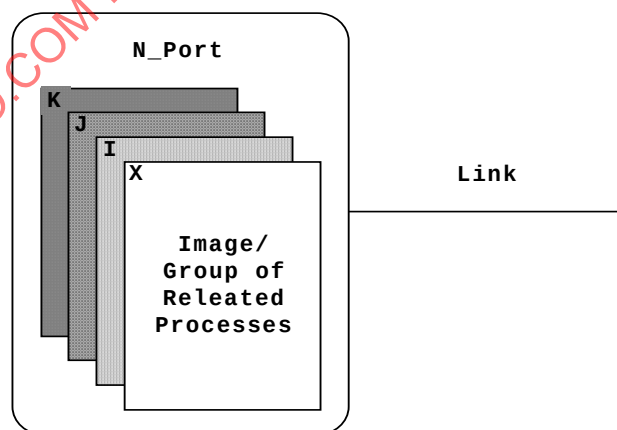


Figure 19 - Image/Group of Related Processes

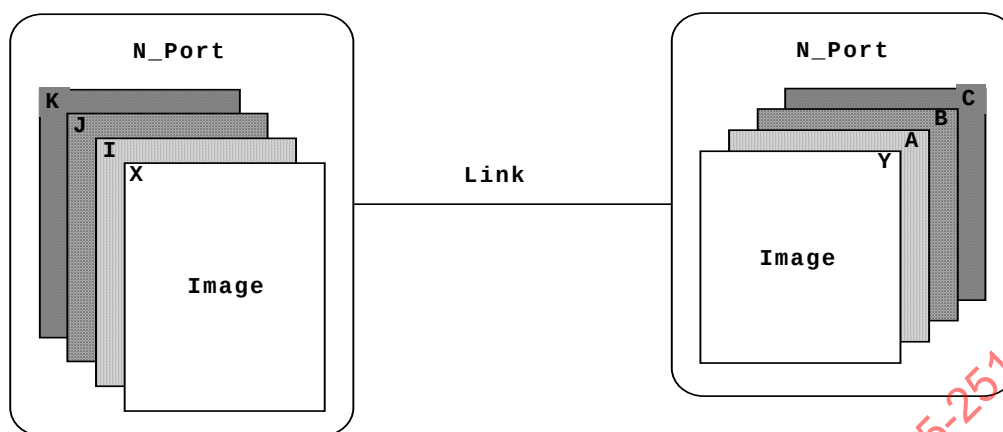


Figure 20 - Image pairs

PRLI, if required, is performed after N_Port Login is successful and prior to other FC-4 transfers. Examples of use of the Process Login function may include image initialization, image re-configuration, or when the Nx_Port receives an indication that the image pair no longer exists. PRLI allows each image behind an Nx_Port to separately manage its resources.

PRLI may be used to establish an operating environment between any of the following combinations of Nx_Port facilities:

- a) two Nx_Ports
- b) one Nx_Port and one Nx_Port image
- c) two Nx_Port images

Multiple image pairs may be established with a single PRLI request and LS_ACC 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 Nx_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 should be accomplished by specifying multiple Service Parameter pages within a single PRLI request. However, a destination Nx_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 LS_ACC 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 113).

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 open Sequences shall be terminated by invoking Abort Sequence (ABTS) protocol. Following the completion of ABTS protocol, all open 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 Nx_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 LS_ACC reply Sequence indicating that the image pair has been established. The Nx_Port responding to the PRLI request does not consider the image pair to be established until the necessary action is taken at the Nx_Port to establish the image pair, and an LS_ACC 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 recipient Nx_Port is not logged in with the requesting port it shall reply with a LOGO ELS Sequence, or with an LS_RJT ELS Sequence with a reason code of "Unable to perform command request" and a reason code explanation of "N_Port Login required". 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, P_BSY, F_BSY, P_RJT, or F_RJT 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 Nx_Port shall not assume that the requested action has or has not taken place. If the Nx_Port to the PRLI request receives no valid response, the Nx_Port should retry the request. The number of retries is system dependent.

16.1.2 PRLI/PRLO Relationships

16.1.2.1 Introduction

Process_Associator (PA) images may exist in the following relationships. Any of these relationships may be established as a default condition using mechanisms not specified by this standard.

16.1.2.2 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 should allow control over the enabling and disabling the specified services for the entire Nx_Port.

16.1.2.3 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). There is no relationship between PAs identified through one Nx_Port of a host system and another Nx_Port from the same host. 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 may allow control over the enabling and disabling the access of each Originator Process to the Responder Nx_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.

16.1.2.4 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. There is no

relationship between PAs identified through one Nx_Port of a host system and another Nx_Port from the same host. 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 may allow control over the enabling and disabling the access of each Responder Process to the Originating Nx_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.

16.1.2.5 PA required by originator and responder

Communication should only take place between an Originator and a Responder when valid PAs are specified. The PA may additionally be used by the FC-4 in a FC-4 dependent manner.

The PRLI request pages shall have valid Originator and Responder PA values defined and shall have the Establish Image Pair bit set to one in order to create image pairs in a 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.

16.1.3 Mode of operation

16.1.3.1 Informative mode

Service Parameter information is exchanged enabling subsequent negotiation for image pair establishment.

16.1.3.2 Binding mode

Information is exchanged that explicitly establishes a relationship between processes in the communicating Nx_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 (e.g., channel or device for FCP-2, initiator or target for FCP-2 and similar values) Requirements indicate those FC-4 properties that shall be agreed upon by both nodes for operation with a FC-4.

16.1.4 Protocol

16.1.4.1 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 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 LS_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 Nx_Port and one image at the Responder Nx_Port.

16.1.4.2 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 LS_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 Nx_Port and the Responder Nx_Port.

16.1.4.3 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 LS_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 Nx_Port and one image at the Responder Nx_Port.

16.2 Process Logout

The Process Logout (PRLO) ELS request shall be used to request invalidation of the operating environment between an image at the initiating Nx_Port and an image at the recipient Nx_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 TYPE-specific Service Parameter settings may be reset or image pairs removed with a single PRLO request and LS_ACC reply Sequence set. Other TYPE-specific Service Parameter settings or image pairs associated with the same or different image pairs or Nx_Ports shall be unaffected. After TYPE-specific Service Parameter settings are reset or image pair is removed, Information Units may not be sent or received for the specified FC-4 TYPE using that image pair and Nx_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 should be accomplished by specifying multiple Logout Parameter pages within a single PRLO request. However, a destination Nx_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 in this standard. An LS_ACC 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 LS_ACC response is sent.

The Nx_Port originating the PRLO request shall not consider an image pair to be removed until it receives an LS_ACC reply Sequence. The Nx_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 LS_ACC reply Sequence is sent. An Nx_Port that receives a P_BSY, F_BSY, P_RJT, or F_RJT reply in response to a PRLO request may retry the PRLO request. The number of retries is system dependent.

Unless the requesting Nx_Port receives a valid response to a PRLO request, that Nx_Port shall not assume that the requested action has or has not taken place. If an Nx_Port to the PRLO request receives an invalid response, the Nx_Port should retry the request. The number of retries is vendor unique.

If the recipient Nx_Port is not logged in with the requesting port it shall reply with a LOGO ELS Sequence, or with an LS_RJT ELS Sequence with a reason code of "Unable to perform command request" and a reason code explanation of "N_Port Login required".

PRLO operates by referencing particular PA image pairs. Pages may be mixed in any combination in a PRLO request. The PRLO LS_ACC 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.

If a ULP attempts to communicate over an image pair that has not been established or has been abnormally terminated, the communication 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.

17 Exchange, Sequence and sequence count management

17.1 Introduction

17.1.1 Data frame transfer

Transfer of information between two Nx_Ports is based on transmission of a Data frame by a source Nx_Port with an ACK response frame, as appropriate (Class 1, 2, 4 and 6 only), from the destination Nx_Port or Nx_Ports receiving the Data frame to acknowledge Data frame delivery.

17.1.2 Sequence

A Sequence is a set of one or more related Data frames transmitted unidirectionally from one Nx_Port to another Nx_Port within an Exchange. The relationship between Sequences and Exchanges is shown in figure 21. In Class 1, 2, 4 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 by the Sequence Initiator. A Sequence shall only be initiated when an Nx_Port holds the Sequence Initiative for a given Exchange.

17.1.3 Streamed Sequences

This standard allows an Nx_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 9.8 for more information regarding the assignment of SEQ_IDs for additional rules when streaming Sequences.

17.1.4 SEQ_CNT

Each frame within a Sequence contains a SEQ_CNT that 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 SEQ_CNT that is set equal to the Data frame SEQ_CNT to which it is responding (Classes 1, 2, 4 and 6).

17.1.5 Exchange

An Exchange is the fundamental mechanism for coordinating the interchange of information and data between two Nx_Ports. All Data transmission shall be part of an Exchange. This discusses Exchanges between Nx_Ports. This standard does not address the means to manage Exchanges across multiple Nx_Ports within a node.

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 Nx_Ports but not simultaneously (i.e., Data flows in one direction at a time within an Exchange for a single Nx_Port pair). 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 Nx_Port (i.e., a Sequence Initiator shall complete transmission of Data frames for a Sequence before initiating another Sequence for the same Exchange). In Classes 1 and 6, 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 Classes 1, 2 and 6. Unless stated otherwise by the upper level, Class 3 Sequences shall not be transmitted in the same Exchange with Class 1, 2, 4 or 6 Sequences. The ability to send or receive Class 3 Sequences in the same Exchange as Class 1, 2, 4 or 6 Sequences is not a requirement of this standard. A Sequence Initiator shall not stream Sequences that are in different classes of service.

NOTE In Class 2, when Sequences are streamed, a Recipient Nx_Port may see multiple active Sequences for the same Initiator because of out of order delivery.

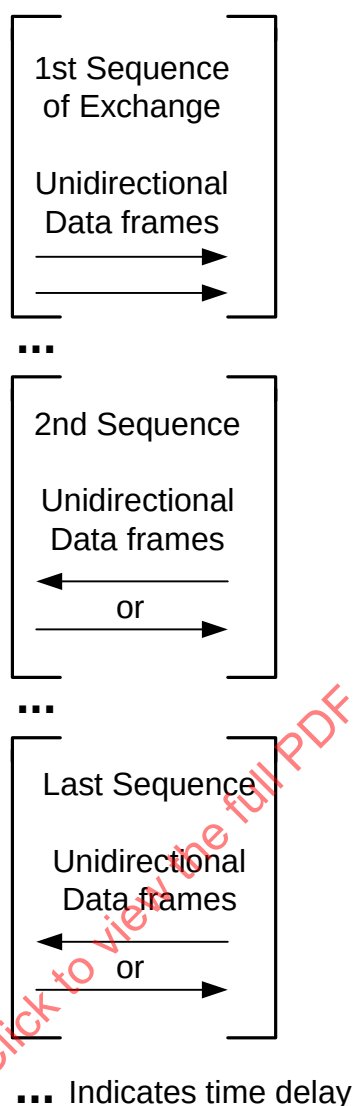


Figure 21 - Exchange - Sequence relationship

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 OX_ID by the Originator and an RX_ID by the Responder. 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 (e.g., an Nx_Port's X_ID).

17.1.6 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 Nx_Port, or it may retain the initiative to transmit the next Sequence.

17.2 Applicability

For Classes 1,2, 3, 4 and 6, FC-2 manages:

- a) Activation and deactivation of Exchanges,
- b) Initiation and termination of Sequences,
- c) Assignment of X_IDs,
- d) Sequence Initiative,
- e) Assignment of SEQ_IDs,
- f) Segmentation and Reassembly,
- g) Sequences,
- h) SEQ_CNT of frames,
- i) Detection of frame Sequence errors, and

In addition to the above, for classes 1, 2, 4, and 6 FC-2 manages notification of frame Sequence errors.

In addition to the above, for Class 4, FC-2 manages:

- a) Manages Available VC credit,
- b) Selects frames to send per VC with Available VC_Credit greater than zero,
- c) Notification of frame Sequence errors.
- d) Sequence Initiator shall assign SEQ_IDs from 0 to 255.
- e) Sequence Recipient assigns the SEQ_ID to a Sequence Status Block.

For Classes 1 and 6, the Sequence Initiator shall assign SEQ_IDs from 0 to L where L is the number of Concurrent Sequences supplied by the Recipient Nx_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 that 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.

A Class 6 Sequence requires a Class 6 Multicast Connection for the duration of the Sequence. When the Connection is terminated, all open Class 6 Sequences are terminated. A Sequence, that has been abnormally terminated, is left in an indeterminate state at the Upper Level Protocol.

17.3 Exchange rules

17.3.1 Exchange management

The following rules apply to 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 and 6 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 this standard. Certain specific cases of Basic Link Services and Extended Link Services do, however, specify collision cases (e.g., FLOGI, PLOGI, ABTX, and RSI).

17.3.2 Exchange origination

The following rules apply to Exchange origination:

- a) An Exchange being originated for ELSs before Login is complete or for the purpose of Login shall follow default Login parameters and special ELSs rules specified in clause 15.
- b) A new Exchange, other than ELSs, may be originated if three conditions are met:
 - A) The originating Nx_Port has performed Login with the destination Nx_Port.
 - B) The originating Nx_Port has an OX_ID and Exchange resources available for use.
 - C) The originating Nx_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 to 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'.
- f) 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. Class 4 requires X_ID Interlock.
- g) If the Responder (Login) does not require X_ID interlock, the Originator may transmit additional frames of the Sequence. In Class 1, 2 and 6, 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.
- h) In Class 1, 2, and 4, the Sequence Initiator shall receive at least one ACK from the Recipient before the Initiator attempts to initiate subsequent Sequences for the Exchange.
- i) The rules specified in Sequence initiation and termination specify the method for assigning X_IDs.

17.3.3 Sequence delimiters

For a more complete description of Data frame and Link_Control delimiters see tables 42 and 46. The following

rules summarize the management of frame delimiters within a Sequence:

- a) A Sequence shall be initiated by transmitting the first frame with a SOF_{ix}, or SOF_{cx}.
- b) Intermediate frames within a Sequence shall be transmitted with SOF_{nx} and EOF_n.
- c) The Sequence shall be complete when an EOF_t, EOF_{dt} for Classes 1, 4 and 6 or EOF_{rt} (for Class 4) 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.

17.3.4 Sequence initiation

The following rules apply to Sequence initiation:

- a) A new Sequence may be initiated if three conditions are met:
 - A) The initiating Nx_Port holds the initiative to transmit (Sequence Initiative).
 - B) The initiating Nx_Port has a SEQ_ID available for use.
 - C) The total number of active Sequences initiated by the initiating Nx_Port with the Recipient Nx_Port does not exceed any of the following:
 - a) total concurrent Sequences (see 15.6.2.7),
 - b) concurrent Sequences per class (see 15.6.5.8),
 - c) open Sequences per Exchange (see 15.6.5.10).
- b) A SOF_{ix} (or a SOF_{cx}) 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 Nx_Port 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 (e.g., 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 Fx_Port and destination Nx_Port capabilities as specified by Login.
- g) In Classes 1 and 6, the connect-request frame shall be limited in size to the lesser of the Fx_Port and destination Nx_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 17.8.2 in response to a Read Sequence Status ELS 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 specified in clause 18 and Connection rules specified in clause 20.

17.3.5 Sequence management

The Sequence Recipient and the Sequence Initiator shall 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, 2, 4 or 6, 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 that follow the rules as defined in 17.3.6.
- f) Frame transmission shall follow Flow Control Rules as defined in clause 18.
- g) The Data Field size of each frame of the Sequence shall be less than or equal to:

- A) the maximum size specified by the Fabric (SOF_{c1}, Class 2, or 3), if present, or
- B) the maximum size specified by the destination Nx_Port in the Service Parameters defined during Login, whichever is smaller.
- C) the maximum size specified by the QoS_R for the VC in the Class 4 circuit.
- 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.

17.3.6 SEQ_CNT

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 SEQ_CNT of each frame of a Sequence:

- a) The SEQ_CNT 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 SEQ_CNT of each subsequent Data frame within the Sequence shall be incremented by one from the previous Data frame.
- c) The SEQ_CNT of the first Data frame of a streamed Sequence shall be incremented by one from the last Data frame of the previous sent Sequence.
- d) The SEQ_CNT of the first Data frame of a non-streamed Sequence may be incremented by one from the last Data frame of the previous sent Sequence or may be zero.
- e) The SEQ_CNT of each Link_Response shall be set to the SEQ_CNT of the Data frame to which it is responding (Class 1, 2, 4 and 6).
- f) The SEQ_CNT of each ACK_1 frame shall be set to the SEQ_CNT of the Data frame to which it is responding (Class 1, 2, 4 and 6). See 18.4.3.3 for ACK_1 rules.
- g) The SEQ_CNT of each ACK_0 frame shall be set to the SEQ_CNT of the last Data frame transmitted (End_Sequence = 1) for the Sequence (Class 1, 2, 4 and 6). See 18.4.3.2 for ACK_0 rules.
- h) If infinite buffers and ACK_0 is being used for Sequences in which the SEQ_CNT may wrap, frame uniqueness is not being ensured (See 15.6.5.9 and 18.4.3.2 for ACK_0 rules).
- i) Within an acknowledged class of service, the SEQ_CNT of any frame shall not be reused until that frame is acknowledged.

17.3.7 Normal ACK processing

The following rules apply to 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 Nx_Ports support multiple ACK forms, ACK_0 usage shall take precedence over ACK_1. ACK_0 use may be asymmetrical between two Nx_Ports (See 15.6.5.5 and 15.6.5.6).
- 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 '00'. b. ACK_0 shall not participate in end-to-end Credit management. An additional ACK_0 shall be used within a Sequence to
 - A) perform X_ID interlock or
 - B) respond a connect-request
- 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 (EOF_t, EOF_d (Classes 1, 4 and 6 only), EOF_r (Class 4 only)) of a Sequence shall be transmitted according to the rules for normal Sequence completion in the absence of detected errors.
- f) ACK_1 or an ACK_0 frame, shall be transmitted during X_ID interlock, (see 17.5.5) and in response to a connect-request (see 20.4.2).
- g) If a Sequence Recipient receives a Data frame in Class 2 or 4 that falls within the SEQ_CNT range of a Recovery_Qualifier, it shall discard the Data Field of the frame and shall not deliver 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 or 4 that falls within the SEQ_CNT range of a Recovery_Qualifier, it shall discard and ignore the ACK frame.
- i) See 18.4.3.2 and 18.4.3.3 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 that prompts the initiative to transmit an ACK frame (i.e., 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.

17.3.8 Normal Sequence completion

The following rules apply to 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) = 1, and any of the following F_CTL bits are set as indicated:
 - A) Sequence Initiative (Bit 16) = 1,
 - B) Continue Sequence Condition (Bits 7 to 6) = 11,
 - C) Last Sequence (Bit 20) = 1, or
 - D) End_Connection (Bit 18) = 1.
- c) A Sequence Event shall be defined when the End_Sequence bit (Bit 19) = 1, and the Continue Sequence Condition bits (Bits 7 to 6) in F_CTL are set to '00' b, '01' b, or '10' b in the absence of an Exchange Event.
- d) In Classes 1 and 6 the Sequence Initiator shall consider a Sequence as deliverable and complete when it receives the final ACK for the Sequence (ACK with EOF_t or EOF_{dt} 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 EOF_t delimiter). However, the Sequence Initiator shall account for all ACKs before reusing the SEQ_ID for this Exchange.
- f) For Class 3 Sequences, this standard provides no deliverability guarantees.
- g) A Class 1, 2, 4 or 6 Sequence shall be considered complete by the Sequence Recipient if
 - A) all Data frames are received,
 - B) no Sequence errors are detected, and
 - C) 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
 - A) all Data frames are received,
 - B) no Sequence errors are detected, and
 - C) An EOF_t terminates the last Data frame.
- i) For a normally completed Sequence in Classes 1 or 6, the Sequence Recipient shall transmit an ACK frame (ACK_1 or ACK_0) with EOF_t (or EOF_{ndt} or EOF_{rt}) 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. The choice of transmitting EOF_{dt} or EOF_{rt} depends on the setting of R_C when E_C = 1.
 - A) 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.
 - B) 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 i).
- 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:
 - A) 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
 - B) save the value of End_Sequence (Bit 19) and Continue Sequence Condition (Bits 7 to 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 that 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 Classes 1, 2, 4 and 6 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 shall be transmitted immediately ('01' b), soon ('10' b), or delayed ('11' b). If Continue Sequence bits are set to '11' b, 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
 - A) Sequence Initiative bit in F_CTL to 0 to hold Sequence Initiative.
 - B) Sequence Initiative bit in F_CTL to 1 to transfer Sequence Initiative.
- p) In Classes 1, 2, 4 and 6, 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 or 6 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 20.7.3).
- r) 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 17.8.1 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.
- s) In the last Data frame of a Class 4 Sequence, the Sequence Initiator may indicate its desire to deactivate the Class 4 circuit by setting the End_Connection (bit 18) = 1 and R_C (bit 8) = 0 in order to begin the procedure to deactivate the Class 4 circuit (see clause 25).
- t) In the last Data frame of a Class 4 Sequence, the Sequence Initiator may request its removal of the Class 4 circuit by setting the End_Connection (bit 18) = 1 and R_C (bit 8) = 1 in order to begin the procedure to remove the Class 4 circuit (see clause 25).
- u) In the last Data frame of a Class 4 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.

17.3.9 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, 4 or 6, 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, 4 or 6, 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 1 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 that 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 this standard. 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 2 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.

17.3.10 Sequence errors - Class 1, 2, 4 and 6

17.3.10.1 Rules common to all Discard policies

Either the Sequence Initiator or the Sequence Recipient may detect errors within a Sequence.

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 an Nx_Port include:
 - A) detection of a missing frame based on SEQ_CNT,
 - B) detection of a missing frame based on a timeout (E_D_TOV),
 - C) detection of an error within a frame (P_RJT),
 - D) reception of a Reject frame (F_RJT or P_RJT) or
 - E) 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 ('10' b) in F_CTL. (see 21.7.3)
- 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 that 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 ('01' b) 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 ('10' b), 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 12.2.2).
- h) If an Initiator receives an ACK with the Abort Sequence Condition bits ('01' b) 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 12.2.2).
- 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 12.2.2) if the Sequence has not been terminated by the Sequence Recipient or Fabric using an EOFt (EOFd) on the RJT.
- j) In Class 1 and Class 6, if the Sequence Initiator detects a Sequence timeout (see 21.2.4), it shall:

- A) abort the Sequence using ABTS,
- B) perform the Link Reset Protocol if all active Sequences have timed out or no end-to-end Credit is available, or
- C) read status from the Recipient if the current state of a Sequence is uncertain in order to determine the existence of an error condition.

End-to-end credit is not required in order to exercise option A; however, if ABTS is sent in absence of end-to-end credit, it is possible that the ABTS frame may be lost, forcing further error recovery process.

- k) In Class 2 or Class 4, if the Sequence Initiator detects a Sequence timeout (see 21.2.4), it shall:
 - A) abort the Sequence using ABTS,
 - B) transmit Link Credit Reset to the Recipient if no end-to-end Credit is available, or
 - C) read status from the Recipient if the current state of a Sequence is uncertain in order to determine the existence of an error condition.

End-to-end credit is not required in order to exercise option A; however, if ABTS is sent in absence of end-to-end credit, it is possible that the ABTS frame may be lost, forcing further error recovery process.

17.3.10.2 Discard multiple Sequences Error Policy

These rules apply to Discard multiple Sequences Error Policy and Discard multiple Sequences with Retransmission Error Policy.

- a) In Classes 1 and 6, if a Sequence Recipient detects a missing frame error, or detects an internal malfunction for a Sequence within an Exchange that requested Discard multiple Sequences Error Policy, it shall request that the Sequence be aborted by setting the Abort Sequence Condition bits ('01' b) 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 ('01' b) 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 (End_Sequence), 18 (End_Connection or Deactivate Class 4 circuit), 17 (Priority Enable), 16 (Sequence Initiative) and 14 (Invalidate X_ID - obsolete) 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 to 4 ('01' b). (see 21.7.2).
- b) In Classes 1 and 6, if a Sequence Recipient detects a missing frame error, or detects an internal malfunction for a Sequence within an Exchange that requested Discard multiple Sequences with Retransmission Error Policy, it shall request that the Sequence be aborted and immediately retransmitted by setting the Abort Sequence Condition bits ('11' b) 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 that requires retransmission, the Sequence Recipient shall follow the rules for Discard multiple Sequences Error Policy (bits 5 to 4 set to '01' b).
- d) For errors detected other than missing frame, the Abort Sequence Condition bits ('11' b) 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 (SOF_{ix}) with the F_CTL Retransmission bit set to one, or an ABTS frame is received. If an ACK is transmitted for the last Data frame of a Sequence, F_CTL bits 19 (End_Sequence), 18 (End_Connection or Deactivate Class 4 circuit), 17 (Priority Enable), 16 (Sequence Initiative) and 14 (Invalidate X_ID - obsolete) 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 to 4 ('11' b).
- 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 to 4 set to '01' b). 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 21.7.2).
- f) In Class 2 or 4, if a Sequence Recipient detects a missing frame error, transmits a P_RJT, or detects an internal malfunction for a Sequence within an Exchange that requested Discard multiple Sequences Error Policy, it shall request that the Sequence be aborted by setting the Abort Sequence Condition bits ('01' b)

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 ('01' b) 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 '01' b (see 21.7.2). The last ACK for the Sequence in error shall be transmitted as described under normal Sequence completion.

- g) In Classes 1 and 6, if a Sequence Initiator receives an ACK with the Abort Sequence Condition bits ('11' b) 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 one until it has received at least one ACK that 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 ELSs request (see 21.7.2.3).

17.3.10.3 Discard a single Sequence Error Policy

In Classes 1, 2, 4 and 6, if a Sequence Recipient detects a missing frame error, or detects an internal malfunction for a Sequence within an Exchange that requested Discard a single Sequence Error Policy, it shall request that the Sequence be aborted by setting the Abort Sequence Condition bits to '01' b 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 '01' b 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 (EOFt) to the Sequence is transmitted, F_CTL bits 19, 18, 17, 16, and 14 settings on the Data frame shall be ignored and shall be set to zero in the ACK frame, and bits 5 to 4 shall be set to '01' b in the ACK frame (see 21.7.2).

17.3.10.4 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 ('01' b) 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.

17.3.11 Sequence errors - Class 3

17.3.11.1 Rules common to all discard policies

The Sequence Recipient may only detect errors within a Sequence.

In both Discard policies, the Sequence Recipient shall discard Sequences in the same manner as in Classes 1, 2 and 6 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 an Nx_Port are:
 - A) detection of a missing frame based on timeout, or
 - B) 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.

17.3.11.2 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.

17.3.12 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
 - A) it receives the ACK (EOFT),
 - B) BA_ACC is received to an ABTS frame,
 - C) a Read Sequence Status or Read Exchange Status indicates normal or abnormal completion,
 - D) the Link Reset Primitive Sequence is received or transmitted during a dedicated connection,
 - E) the Exchange is aborted using the ABTX Link Service request, or
 - F) a Logout Link Service request is completed.
- b) A Sequence shall be considered open and active, and a Sequence Status Block 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 a Sequence Status Block 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 '00' b, 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.

17.3.13 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 = 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 ELSs 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 Nx_Port involved in the Exchange (either Originator or Responder). A Recovery_Qualifier range timeout may be required in Class 2, 3 or 4.

17.3.14 Exchange Status Rules

The following rules summarize handling of Exchange Status Block processing:

- a) An Exchange shall be considered active, and an Exchange Status Block opened, by the Originator after transmission of the first frame of the first Sequence.
- b) An Exchange shall be considered active, and an Exchange Status Block opened, by the Sequence Recipient when any frame in the first Sequence is received.
- c) An Exchange shall be remain open until
 - A) the last Sequence of the Exchange completes normally,
 - B) a timeout period of E_D_TOV has elapsed since the last Sequence of the Exchange completed abnormally, or
 - C) the Exchange is successfully aborted with ABTX (that 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 reuse. 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 this standard.
- e) The contents of an Exchange Status Block may be made available (RES request) until the resource is allocated to another Exchange.

17.4 Exchange management

An Exchange is managed as a series of Sequences of data frames. The Originator of the Exchange shall transmit the initial Sequence. F_CTL bits within the Frame_Header identify and manage 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.

17.5 Exchange origination

17.5.1 Introduction

The key facilities, functions, and events involved in the origination of an Exchange by both the Originator and Responder are diagrammed in figure 22. An Exchange for Data transfer may be originated with a destination Nx_Port following N_Port Login. Login provides information necessary for managing an Exchange and Sequences (e.g., class, the number of Concurrent Sequences, Credit, and Receive Data Field Size). An Exchange is originated through the initiation of a Sequence. The rules in 17.3.2 specify the requirements for originating an Exchange.

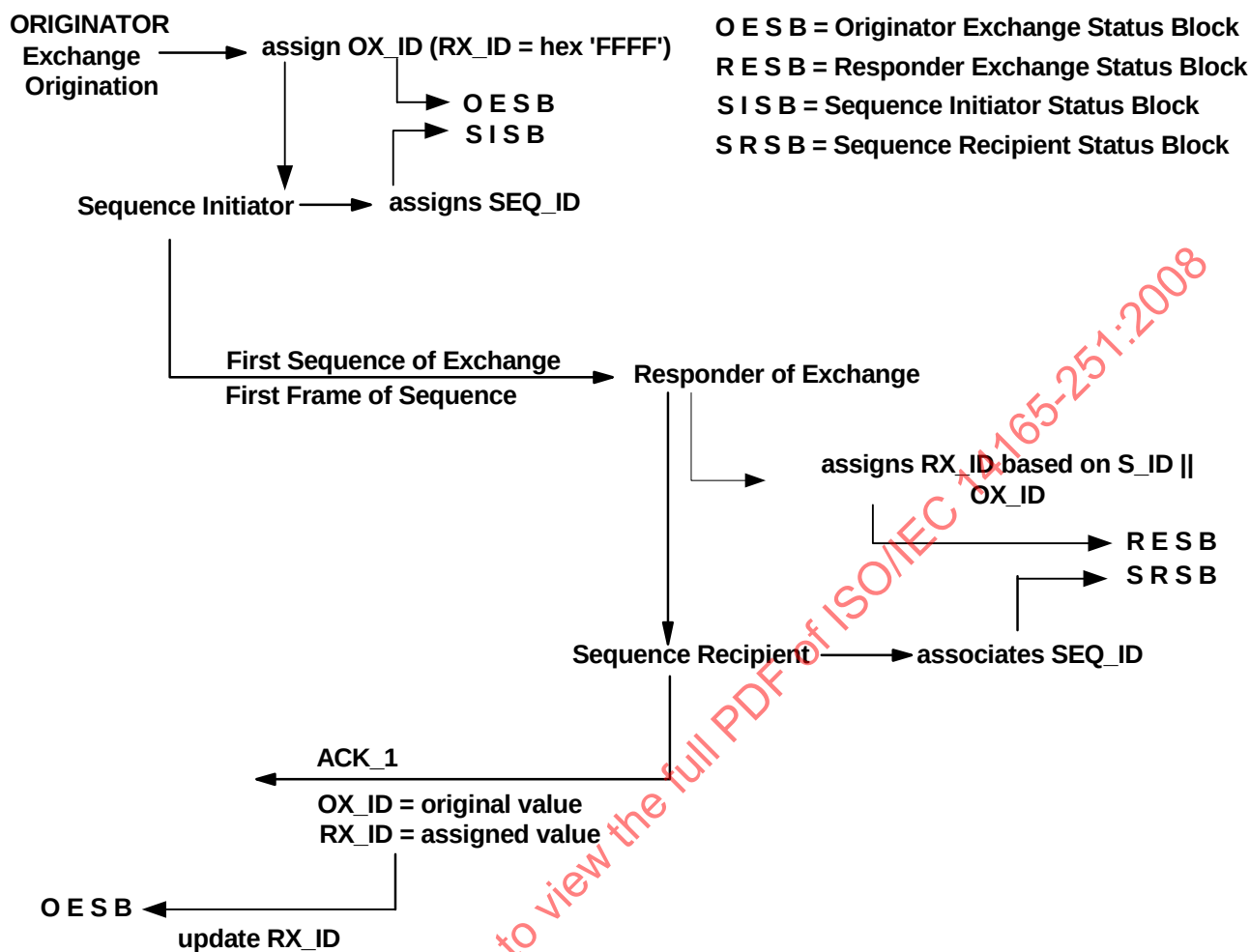


Figure 22 - Exchange origination

17.5.2 Exchange Originator

When an Exchange is originated by an Nx_Port, that Nx_Port shall assign an OX_ID unique to that Originator or Originator-Responder pair. An Originator Exchange Status Block is allocated and bound to the Exchange and other link facilities in that Nx_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 17.6.2 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 OX_ID, together with Originator-Responder pair information (if required) provides the mechanism for tracking Sequences for multiple concurrent Exchanges that 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 Nx_Port. The Originator may choose to qualify the OX_ID using the Originator-Responder pair.

17.5.3 Exchange Responder

The destination Nx_Port shall be designated as the Responder for the duration of the Exchange. When the destination Nx_Port receives the first Sequence of the Exchange, that Nx_Port shall assign an 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 17.6.2 for more information on unique Sequence_Qualifiers.

In Class 1, 2 or 4, 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 Nx_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 Responder in the Responder Exchange Status Block shall track the status of the Exchange.

17.5.4 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, 2 and 4). 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.

17.5.5 X_ID interlock

X_ID interlock is only applicable to Classes 1, 2 and 4. When an Nx_Port initiates a Sequence with an Nx_Port that has specified during Login that X_ID interlock is required and the Recipient's X_ID is invalid or unassigned, the initiating Nx_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.

17.6 Sequence management

17.6.1 Open and active Sequences

From the standpoint of the Sequence Initiator, a Sequence is active for the period of time from the allocation of the SSB for the sequence until the end of the last Data frame of the Sequence is transmitted. In Class 1 and Class 6, 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 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 this standard, which provides no deliverability guarantees for Class 3 Sequences. In Class 4, the Sequence Initiator considers the Sequence open until the ACK with EOF_t (EOF_{dt} or EOF_{rt}) is received, the Sequence is aborted by performing the ABTS Protocol, or the Sequence is abnormally terminated. In Classes 1, 2, 4 and 6, from the standpoint of the Sequence Recipient, a Sequence is open and active 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 Classes 1, 2, 4 and 6, from the standpoint of the Sequence Recipient, a Sequence is active 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 open and active from the time the initiating Data frame is received until all Data frames up to the frame containing EOF_t have been received.

17.6.2 Sequence_Qualifier management

The Sequence Initiator assigns a SEQ_ID that is unique between the Nx_Port pair of the Initiator and the Recipient at the time it is assigned and until the Sequence is complete. When the Sequence 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 17.3.4). 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 21.7.2), however, SEQ_ID shall not be included in the Recovery_Qualifier.

17.6.3 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:

- a) Sequence initiative, and
- b) Sequence termination.

17.6.4 Transfer of Sequence Initiative

The Sequence Initiator controls which Nx_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 Classes 1, 2, 4 and 6, the Sequence Recipient shall not consider Sequence Initiative to have been passed until the Sequence that 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 Classes 1, 2, 4 and 6, 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.

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.

17.6.5 Sequence Termination

17.6.5.1 Introduction

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:

- a) In Classes 1, 2, 4 and 6 the Sequence Recipient transmits an ACK frame of ACK_1 or ACK_0 with EOF_t (or EOF_d or EOF_r as appropriate) in response to the last Data frame received for the Sequence.
- b) In Class 3, the Sequence Initiator transmits the last Data frame of the Sequence with EOF_t.

17.6.5.2 Class 1 and Class 6

When EOF_t or EOF_d has been transmitted or received by each Nx_Port, the appropriate Exchange Status Block associated with the Sequence shall be updated in each Nx_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.

17.6.5.3 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 Nx_Ports.

When the Sequence is completed by each Nx_Port, the appropriate Exchange Status Block associated with the Sequence shall be updated in each Nx_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 that 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 21.7.4).

17.6.5.4 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 Nx_Port, the appropriate Exchange Status Block associated with the Sequence shall be updated in each Nx_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.

17.6.5.5 Class 4

When EOFt, EOFdt, or EOFrt has been transmitted or received by each Nx_Port, the appropriate Exchange Status Block associated with the Sequence shall be updated in each Nx_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.

17.6.5.6 Continue Sequence Condition

When an Nx_Port terminates a Sequence yet holds the Sequence Initiative, the Nx_Port may include additional information regarding the timing of the next Sequence. The Nx_Port uses the Continue Sequence Condition bits in F_CTL to indicate that the next Sequence is being transmitted immediately ('01' b), soon ('10' b), or delayed ('11' b). The estimate of time is based on the time to remove and reestablish a Class 1 and Class 6 Connection, regardless of which class is being used.

17.6.5.7 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.

17.7 Exchange termination

17.7.1 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. Abnormal termination

17.7.2 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 as specified in 11.2.3 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 in this standard.

When a Class 4 Virtual Circuit is terminated by the END Link_Control Command, all exchanges active at that time that are using the Virtual Circuit shall be terminated. The Fabric, or either Nx_Port may originate the END Link_Control Command.

17.8 Status blocks

17.8.1 Exchange Status Block

An Exchange Status Block (see table 246) 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 Nx_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 Nx_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 246 - Exchange Status Block

Item	Size - Bytes
Priority (if enabled) (LSB 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 (see 15.6.5.10) x 8
Newest Sequence Status (SSB format-first 8 bytes)	8

The E_STAT Field shall be defined as follows:

- a) Bit 31 - ESB owner
0 = Originator
1 = Responder
- b) Bit 30 - Sequence Initiative
0 = Other port holds initiative
1 = This port holds initiative
- c) Bit 29 - Completion
0 = open
1 = complete
- d) Bit 28 - Ending Condition
0 = normal
1 = abnormal
- e) 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.
- f) Bit 26 - Recovery_Qualifier
0 = none
1 = Active
- g) Bits 25 to 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
- h) 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.
- i) 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.
- j) 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.
- k) Bits 20 to 0 - Reserved

17.8.2 Sequence Status Block

A Sequence Status Block (see table 247) is a logical construct used to track the progress of a single Sequence by an FC_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 FC_Port.

Table 247 - 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 to 24	1
Frame header Word 4, bits 31 to 16	2
RX_ID	2
reserved	2

Frame header Word 1, Bits 31 to 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 to 16: This field shall contain the OX_ID.

The S_STAT field shall be defined as follows:

- a) Bit 15 - Sequence context
0 = Initiator
1 = Recipient
- b) Bit 14 - active
0 = not active
1 = active
- c) Reserved
- d) Bit 12 - Ending Condition
0 = normal
1 = abnormal
Bits 11 to 7 clarify the reason for an abnormal ending condition.
- e) Bits 11 to 10 ACK, Abort Sequence condition
0 0 = continue
0 1 = Abort Sequence requested
1 0 = Stop Sequence requested
1 1 = Abort with Retransmission requested
- f) Bit 9 - ABTS protocol performed
0 = ABTS not completed
1 = ABTS completed by Recipient
- g) Bit 8 - Retransmission performed
0 = Retransmission not completed
1 = Retransmission completed by Recipient

- h) Bit 7 - Sequence time-out
0 = Sequence not timed-out
1 = Sequence timed-out by Recipient (E_D_TOV)
- i) Bit 6 - P_RJT transmitted
0 = P_RJT not transmitted
1 = P_RJT transmitted
- j) Bits 5 to 4 Class
0 0 = reserved
0 1 = Class 1
1 0 = Class 2
1 1 = Class 3
- k) Bit 3 - ACK (EOF_t) transmitted
0 = ACK (EOF_t) not transmitted
1 = ACK (EOF_t or EOF_{dt}) transmitted
- l) Bits 2 to 0 - Reserved

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

18 Flow control management

18.1 Introduction

Flow control is the FC-2 control process to pace the flow of frames between Nx_Ports, an Nx_Port and the Fabric and within the Fabric to prevent overrun at the receiver. Flow control is managed using end-to-end Credit, end-to-end Credit_CNT, ACK_1, buffer-to-buffer Credit, buffer-to-buffer Credit_CNT, and R_RDY along with other frames.

Flow control is managed between Nx_Ports (end-to-end) and between FC_Ports (buffer-to-buffer). Flow control management has variations dependent upon the class.

Class 1 and 6 frames use end-to-end flow control. Class 1 and 6/SOF_{c1} frames 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 248 shows the applicability of the flow control mechanisms to each class. Buffer-to-buffer flow control management for Class 4 is specified in clause 25. Class 4 uses VC_RDYs as specified in clause 25 and end-to-end for flow control.

Table 248 - Flow control applicability

Flow Control methodology and mechanism	Class 1 and 6 without SOF _{c1}	Connect-request frame with SOF _{c1}	Class 2	Class 3	Class 4
end-to-end	Yes	Yes	Yes	No	Yes
buffer-to-buffer	No	Yes	Yes	Yes	No (uses VC_RDYs)
ACK_1	Yes	Yes	Yes	No	Yes
ACK_0	One per Sequence	Yes	One per Sequence	No	One per Sequence
R_RDY	No	Yes	Yes	Yes	No (uses VC_RDYs)
F_BSY (DF) ^a	No	Yes	Yes	No	No
F_BSY (LC) ^b	No	No	Yes	No	No
F_RJT	No	Yes	Yes	No	Yes
P_BSY	No	Yes	Yes	No	No
P_RJT	Yes	Yes	Yes	No	Yes
^a F_BSY (DF) indicates F_BSY response to a Data frame. ^b F_BSY (LC) indicates F_BSY response to a Link_Control frame except P_BSY					

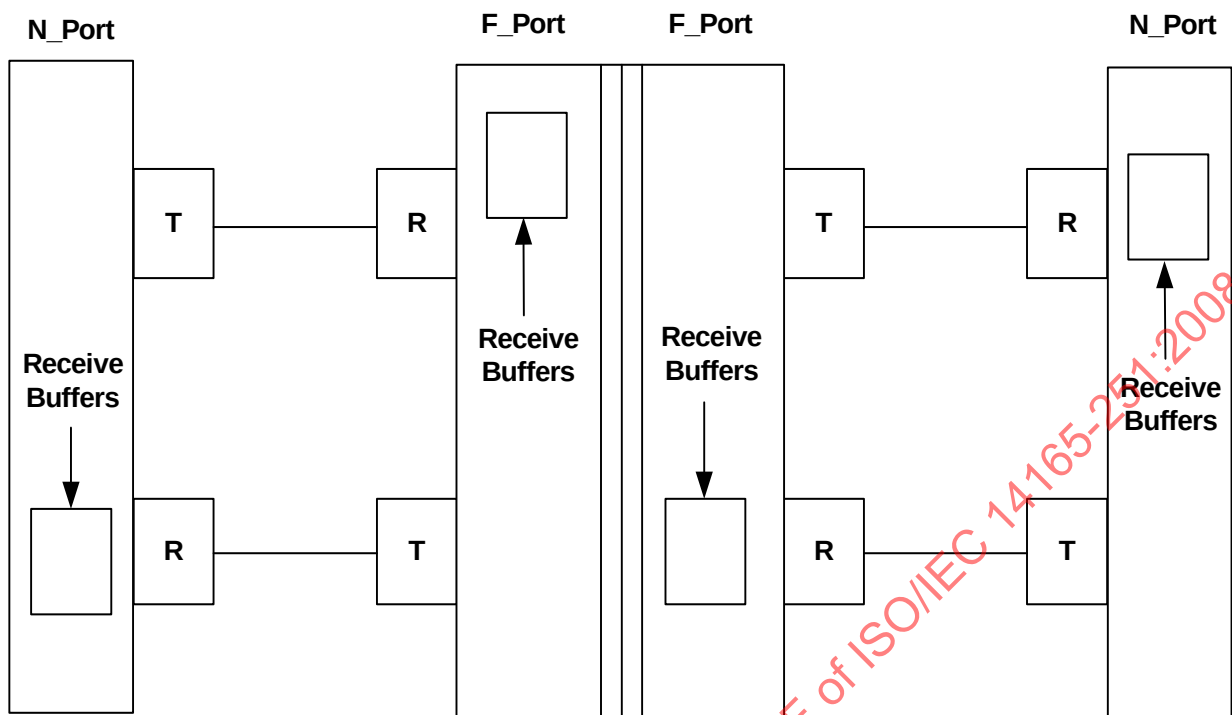


Figure 23 - Physical flow control model for Classes 1, 2, 3 and 6

18.2 Physical flow control model for Classes 1, 2, 3 and 6

The physical flow control model for Classes 1, 2, 3 and 6 is illustrated in figure 23 (See clause 25 for the Class 4 physical flow control model). The model consists of following physical components:

- Each Nx_Port with Class 1 and 6/SOFc1 and Connectionless receive buffers.
- Each Fx_Port to which an Nx_Port is attached, with its receive buffers for Connectionless service. Class 1 buffers internal to the fabric used for Intermix are transparent to the FC-2 flow control.

Class 1 and 6 buffering requirements during Intermix are specified in 13.5. Class 2, Class 3 and Class 1 SOFc1 frames shall share connectionless buffers. Table 249 shows the use of buffers for end-to-end and buffer-to-buffer flow controls.

Buffer-to-buffer flow control management for Class 4 is specified in clause 18.

Table 249 - Buffer participation

Participating buffers	end-to-end flow control	buffer-to-buffer flow control
Class 1 ^a and 6 buffers	Class 1 and 6 frames	-
connectionless buffers	Class 2 frames	Class 2, Class 3 and Class 1 and 6/SOFc1 frames
Class 4 buffers	Class 4 frames	Uses VC_RDYs
^a Participation of Class 1 buffers in the Fabric during Intermix is transparent to flow control.		

18.3 Credit and Credit_Count

18.3.1 Introduction

Credit is the number of receive buffers available to a transmitting FC_Port. Two types of Credits used in flow control are:

- a) End-to-end Credit (EE_Credit) - between communicating Nx_Ports
- b) buffer-to-buffer Credit (BB_Credit) - between adjacent FC_Ports

18.3.2 Credit_Count management

The Sequence Initiator manages the Credit_Count. The Credit_Count management is internal to the Sequence Initiator and is transparent to the other port involved in the Sequence. The Credit_Count may be managed either by increasing or decreasing starting with an appropriate value.

18.3.3 Management by increasing the Credit_Count

To manage the increasing Credit_Count, the Credit_Count is initialized to zero and increased by a specified value (see tables 250, 251, and 252) 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 zero.

18.3.4 Management by decreasing the Credit_Count

To manage the decreasing Credit_Count, the Credit_Count is initialized to the value of the Credit and decreased to the minimum value of 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 zero. (see 18.5.10)

18.3.5 Credit_Count types

Corresponding to two types of Credits listed above, two types of Credit_Counts used are:

- a) End-to-end Credit_Count (EE_Credit_CNT)
- b) buffer-to-buffer Credit_Count (BB_Credit_CNT)

18.3.6 Usage

The Nx_Port transmitting Class 1, 2 or 6 Data frames shall use the end-to-end Credit allocated by the receiving Nx_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 an FC_Port is transmitting Data frames or Link_Control frames, it shall use BB_Credit allocated by the receiving FC_Port for buffer-to-buffer flow control and manage the corresponding BB_Credit_Count.

18.4 End-to-end flow control

18.4.1 End-to-end management rules

End-to-end flow control is an FC-2 control process to pace the flow of frames between Nx_Ports. An Nx_Port pair in Class 1, 2, 4 or 6 uses end-to-end flow control.

End-to-end flow control is performed with EE_Credit_CNT and EE_Credit as the controlling parameter.

End-to-end management rules are given in following subclauses for those cases where no error occurs.

Management of EE_Credit_CNT is summarized in table 250. The EE_Credit recovery is specified in 18.4.9.

Table 250 - End-to-end flow control management

Activity	EE_Credit_CNT (Nx_Port only)
Nx_Port transmits a Classes 1, 2, and 6 Data frame	Increment EE_Credit_CNT by one
Nx_Port transmits an LCR	And reinitializes End-to-end Credit for the destination Nx_Port to its Login value.
Nx_Port receives F_BSY (DF), F_RJT, P_BSY, or P_RJT	Decrement EE_Credit_CNT by one
Nx_Port receives F_BSY (LC)	N/A
Nx_Port receives ACK_1 (Parameter field: History bit = 1, ACK_CNT = 1)	Decrement EE_Credit_CNT by one
Nx_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 11.2.2.3)
Nx_Port receives ACK_0 (Parameter field: History bit = 0, ACK_CNT = 0)	N/A (see 11.2.2.3)
Nx_Port receives Data frame (Class 1, Class 2 or Class 3)	N/A
Nx_Port receives an LCR	N/A ^a
Nx_Port transmits a Class 3 Data frame	N/A
Nx_Port transmits P_BSY or P_RJT	N/A
Nx_Port transmits ACK	N/A
Legend: N/A = Not applicable	
^a On receipt of LCR, the Sequence Recipient resets buffer (see 11.2.4)	

18.4.2 Sequence Initiator

The following rules apply to the Sequence initiator:

- The Sequence Initiator is responsible for managing EE_Credit_CNT across all active Sequences.
- The Sequence Initiator shall not transmit a Data frame other than the ABTS Basic Link Service unless the allocated EE_Credit is greater than zero and the EE_Credit_CNT is less than this EE_Credit.
- In Class 1, 2 or 6, the value of the EE_Credit_CNT = 0 at the end of N_Port Login, N_Port Relogin, or Link Credit Reset (see 11.2.4).
- In Class 1 and 6, the EE_Credit_CNT is set to one, on transmitting Class 1 and 6/SOF_{c1} frame. It is incremented by one for each subsequent Class 1 and 6 Data frame transmitted. In the case of ACK_0 usage, EE_Credit_CNT management is not applicable.
- The EE_Credit_CNT is incremented by one for each Class 2 Data frame transmitted. In the case of ACK_0 usage, EE_Credit_CNT management is not applicable.
- 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.
- 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 lower SEQ_CNTs.

edged 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 11.2.2), but does not perform any EE_Credit_CNT management.
- i) For an ACK_1 with EOF_t or EOF_{dt} 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.

18.4.3 Sequence Recipient

18.4.3.1 General

The Sequence Recipient is responsible for acknowledging valid Data frames received (see 11.2.2.3).

The Sequence Recipient may use ACK_0 and ACK_1 as determined during N_Port Login (see 15.6). The Sequence Recipient rules for using ACK_0 and ACK_1 are different and are listed for a non-streamed Sequence first, followed by additional rules for streamed Sequences.

18.4.3.2 ACK_0

If ACK_0 is used (see 11.2.2), 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.
- c) Both the History bit and the ACK_CNT of the Parameter field shall be set to zero.
- 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 Classes 1 and 6 and with EOF_t in Class 2.

18.4.3.3 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 zero if no ACK is pending for any previous SEQ_CNT for the Sequence (see 11.2.2.3).
- c) In Classes 1 and 6, 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 Classes 1 and 6 shall have the End_Sequence bit set to 1, History bit set to zero 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.
 - A) 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 17.3.10). 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 zero and shall contain EOF_t.
 - B) 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 may transmit ACK_1 (with End_Sequence bit set to zero) 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.

18.4.3.4 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. The Sequence Recipient supporting the Process Policy shall transmit the last ACK without setting the Abort Sequence bits (see 17.3.10.4).

18.4.3.5 Streamed Sequences

Each of the streamed Sequences shall follow all the rules for a non-streamed Sequence as defined in 18.4.3.1 and 18.4.3.3 above. 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.

18.4.4 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 may 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 or 6 EE_Credit represents the number of Class 1 or 6 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 Nx_Port through the EE_Credit field of the Service Parameters. The minimum or default value of EE_Credit is one.

EE_Credit is used as a controlling parameter in end-to-end flow control.

18.4.5 EE_Credit_CNT

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.

18.4.6 EE_Credit management

EE_Credit management involves an Nx_Port establishing and revising EE_Credit with the other Nx_Port it intends to communicate with, for Class 1, 2, 6 or all three.

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 LS_ACC Sequence may also be used as a stand-alone procedure to revise the EE_Credit. The Service Parameters interchanged during N_Port Login provide the Class 1 or Class 2 EE_Credit separately in their respective Credit fields.

A Sequence Initiator, during N_Port Login obtains EE_Credit from the Nx_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, at the end of initialization, Login or Relogin. The EE_Credit_CNT is incremented, decremented or left unaltered as specified by the flow control management rules (see 18.4.1). The EE_Credit_CNT shall not exceed the EE_Credit value to avoid possible overflow at the receiver except that the EE_Credit_CNT may exceed the EE_Credit value as a result of transmitting an ABTS Basic Link Service.

The Sequence Initiator shall allocate the total Nx_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 Nx_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 LS_ACC response has been received to the Abort Sequence frame.

The Nx_Port is responsible for managing EE_Credit_CNT using EE_Credit as the upper bound on a per Nx_Port

basis except that the EE_Credit_CNT may exceed the EE_Credit value as a result of transmitting an ABTS Basic Link Service.

18.4.7 End-to-end flow control model

The end-to-end flow control model is illustrated in figure 24. 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.

- a) The Sequence Recipient provides a number of Class 1, Class 2 and/or Class 6 receive buffers.
- b) The Sequence Initiator obtains the allocation of Class 1, Class 2 and/or Class 6 receive buffers, as Class 1 Class 2 and Class 6 Accredits, respectively. That allocation is distributed among all the open Sequences for a specific Sequence Recipient.
- c) The Sequence Initiator manages the end-to-end flow by managing Class 1, Class 2 and Class 6 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 when the ACK_1 frame is received. F_BSY(LC) is a valid response only to Class 2.

For more details on incrementing and decrement EE_Credit_CNT see table 250.

18.4.8 End-to-end class dependency

18.4.8.1 End-to-end Credit allocation

Each Sequence Recipient may allocate the same Class 1 or Class 6 Nx_Port EE_Credit value to each Nx_Port it is logging into. This Class 1 or Class 6 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 Nx_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 this standard.

18.4.8.2 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 currently active, with that Sequence Recipient's EE_Credit as the upper bound.

A Class 3 Nx_Port does not perform EE_Credit_CNT management.

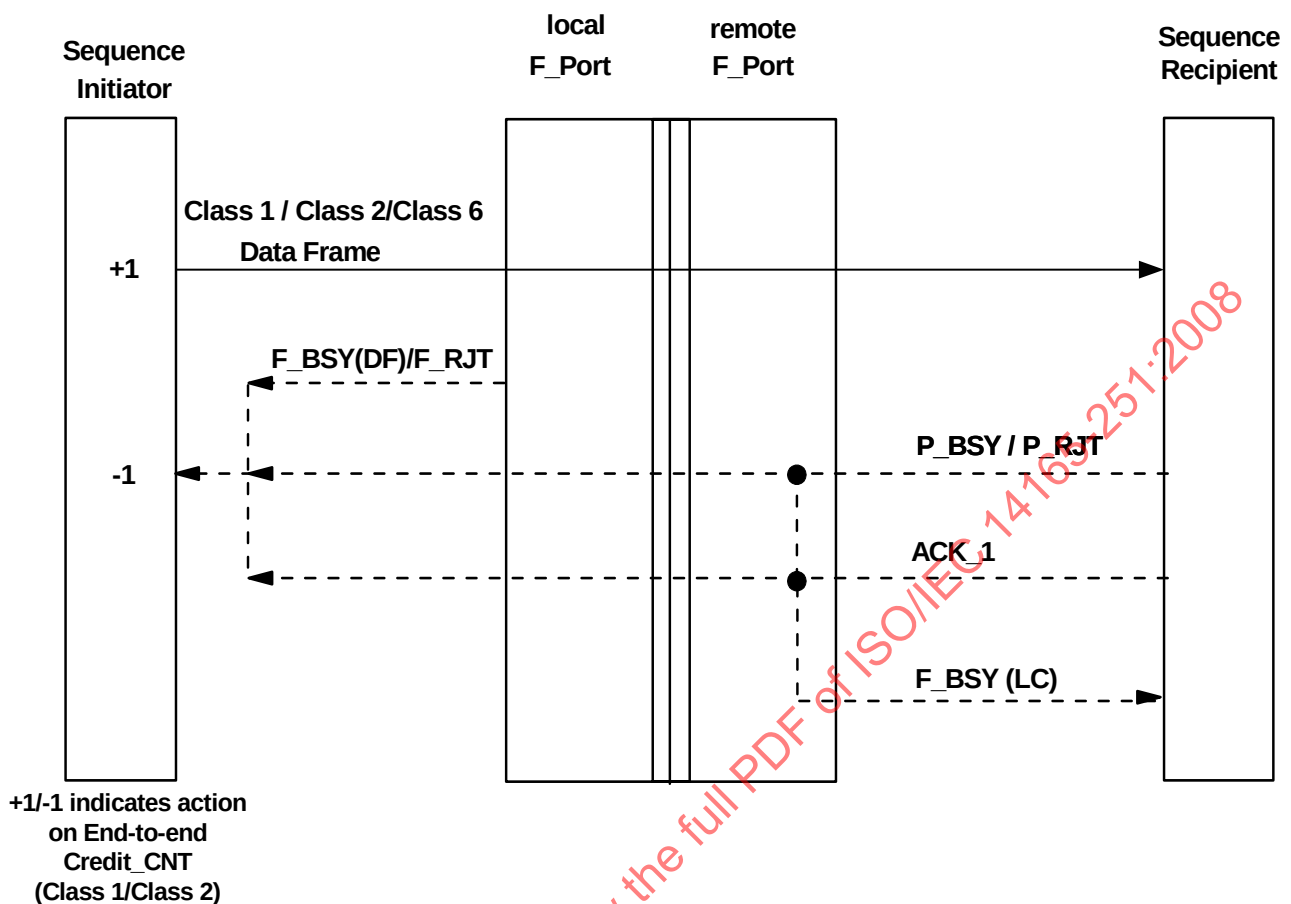


Figure 24 - End-to-end flow control model

18.4.9 EE_Credit recovery

In Class 1, Class 2 and Class 6, the Sequence Initiator may recover EE_Credit when a Sequence is terminated, by any unacknowledged Data frames associated with the Sequence being terminated. Termination may be normal or abnormal.

In Class 1, Class 2 and Class 6, 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 should recover EE_Credit.

If an ACK is received that 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_CNT.

Class 1 and Class 6 EE_Credit is also recovered when a dedicated connection is removed by either EOFdt or by the Link Reset Protocol.

In Class 1, Class 2 and Class 6, EE_Credit may also be recovered by an Nx_Port through Relogin.

Class 2 EE_Credit may also be reinitialized to N_Port Login value using Link Credit Reset (see 11.2.4.2).

18.4.10 Class 4

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 EOF_t delimiter. The ACK frame used to deactivate a Class 4 circuit and terminate a Sequence shall use the EOF_d delimiter. The ACK frame used to remove a Class 4 circuit and terminate a Sequence shall use the EOF_r delimiter. All other ACK frames that do not terminate a Sequence shall use an EOF_n delimiter.

18.4.11 Procedure to estimate end-to-end Credit

18.4.11.1 Introduction

An estimate of the minimum end-to-end Credit between an Nx_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 that support this procedure are optional. This procedure shall be performed after Login between this Nx_Port pair. Login determines a number of Service Parameters (e.g., the maximum frame size that each Nx_Port is capable of receiving).

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.

The procedure shall be invoked by the Link Service support of the source Nx_Port and responded to by the Link Service support of the destination Nx_Port. Since the ELS requests used to perform this procedure are optional, LS_RJT may be received to any request (except ESTC which has no reply) with a reason code of "Command not supported".

To perform this procedure for Class 1, a Class 1 Connection shall have been established before the procedure is performed.

18.4.11.2 Procedure steps

18.4.11.2.1 General

This procedure is optional and consists of following three request Sequences.

- a) Establish Streaming Sequence
- b) Estimate Credit Sequence
- c) Advise Credit Sequence

The procedure is illustrated with these request Sequences and their respective reply Sequences in figure 25.

The procedure shall be performed for Class 1 or Class 2 with respective delimiters, as clause 13.

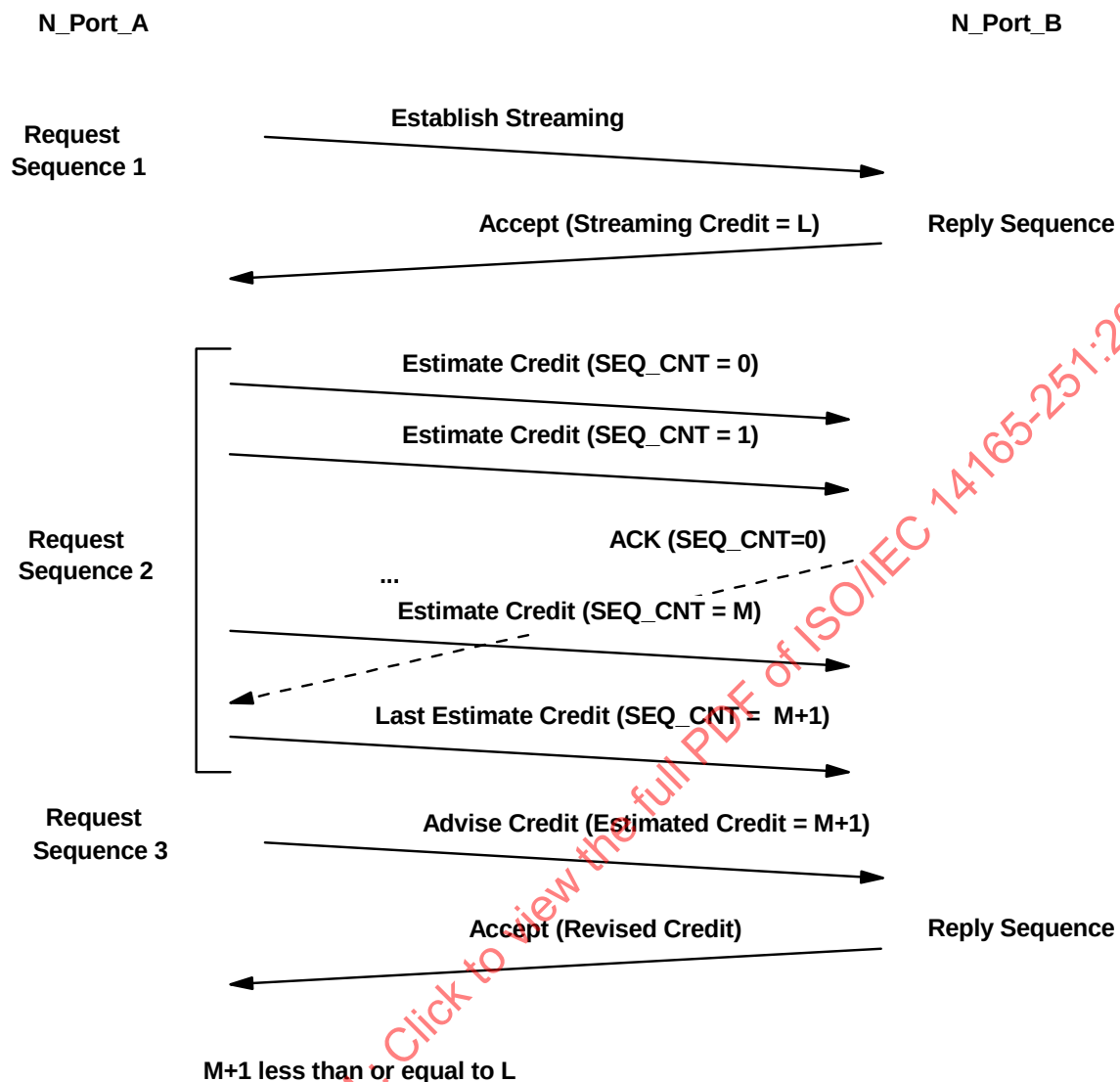


Figure 25 - Procedure to estimate end-to-end Credit

18.4.11.2.2 Establish Streaming Sequence

The ESTS ELS (see 12.3.2.6) shall be used to obtain an end-to-end Credit large enough to perform continuous streaming from a source Nx_Port to a destination Nx_Port. This Sequence provides an opportunity for the destination Nx_Port to communicate the maximum end-to-end Credit it shall provide for the purposes of streaming. This temporary allocation is termed Streaming Credit (L).

This Sequence shall be used between an Nx_Port pair after the Nx_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.

- 1) The source shall transmit the ESTS ELS.
- 2) The destination shall reply with a LS_ACC frame.
- 3) The Class Validity bit set to one (see 15.6.5.3) shall identify the Class that is requesting streaming Credit.
- 4) The Payload of LS_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 to 0 of each class group). The receiver shall ignore the other fields.

18.4.11.2.3 Estimate Credit Sequence

The Estimate Credit (see 12.3.2.5) ELS shall be performed immediately following the completion of the Establish Streaming Sequence. This Sequence requires the use of ACK_1 and may not be executed by all Nx_Ports.

- a) The source Nx_Port shall stream ESTC (see 12.3.2.5) frames consecutively until it receives the first ACK (ACK_1) from the destination Nx_Port that set the Abort Sequence bits (F_CTL bits 5 to 4) to '10' b. The source shall not exceed the Streaming Credit obtained during the Establish Streaming Sequence.
- b) If the source does not receive 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 to 4) set to '10' b.
- c) The size of the Data Field of the ESTC frame shall be the normal size frames transmitted by a 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 Nx_Port shall respond with ACK for Data frames received.
- g) If the highest SEQ_CNT transmitted by the source Nx_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, this value is less than the optimal Credit required to utilize the maximum bandwidth of the fibre.
- h) The source Nx_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.

18.4.11.2.4 Advise Credit Sequence

The Advise Credit (see 12.3.2.3) shall be performed immediately following completion of the Estimate Credit Sequence. The source Nx_Port that performed the Estimate Credit Sequence shall advise the destination Nx_Port of the Estimated Credit in ADVC Data Field. The destination Nx_Port shall reply using a LS_ACC frame, with a revised end-to-end Credit value in its Payload. This value is determined by the destination Nx_Port based on its buffering scheme, buffer management, buffer availability and Nx_Port processing time. This is the final value to be used by the source Nx_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, the destination may choose to reallocate the estimated end-to-end Credit. If the Streaming Credit is smaller than needed for continuous streaming, the source Nx_Port is bound to run short of end-to-end Credit and the source Nx_Port may advise that value as the Estimated Credit.

- a) The source Nx_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 set to one. The receiver shall ignore the other fields.
- c) The destination Nx_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 Nx_Port for end-to-end Credit.
- d) The destination Nx_Port replies with a LS_ACC frame that successfully completes the Protocol. The LS_ACC Sequence shall contain the end-to-end Credit allocated to the source Nx_Port. The Payload of LS_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 the Class Validity bit set to one. The receiver shall ignore the other fields.

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.

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 may 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 ADVC Link request Sequence.

18.5 Buffer-to-buffer flow control

18.5.1 Introduction

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

- a) Sequence Initiator and local Fx_Port
- b) remote Fx_Port and Sequence Recipient Nx_Port
- c) E_Ports within the Fabric
- d) Sequence Initiator and Sequence Recipient Nx_Ports in point-to-point topology

18.5.2 Buffer-to-buffer management rules

Buffer-to-buffer flow control rules are as follows:

- a) Each FC_Port is responsible for managing its own BB_Credit_CNT.
- b) The sending FC_Port shall not transmit a Class 2, Class 3, or Class 1 and 6/SOF_{c1} 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 FC_Port is responsible for maintaining BB_Credit_CNT less than BB_Credit.
- c) Each Nx_Port and Fx_Port sets BB_Credit_CNT value to zero at the end of Fabric Login, or Fabric Relogin.
- d) Each FC_Port increments BB_Credit_CNT by one for each Class 2, Class 3, or Class 1 and 6/SOF_{c1} frame transmitted and decrements by one for each R_RDY received.
- e) Recognition of SOF_{c1}, SOF_{x2}, or SOF_{x3} shall be responded to by a transmission of an R_RDY when the buffer becomes available, regardless of whether the frame is valid, invalid, or discarded.

Managing BB_Credit_CNT is given in table 251. BB_Credit_CNT for E_Ports and B_Ports is specified in [7], FC-SW-3.

Table 251 - Buffer-to-buffer flow control management

Activity	BB_Credit_CNT
FC_Port transmits any Class 2, Class 3, Class 1 or 6 frame with SOF _{c1} (including F_BSY(DF), F_BSY(LC), F_RJT, P_BSY, P_RJT or LCR)	Increment BB_Credit_CNT by one
FC_Port receives R_RDY	Decrement BB_Credit_CNT by one
FC_Port receives any Class 2, Class 3, or Class 1 or 6 frame with SOF _{c1} (including F_BSY(DF), F_BSY(LC), F_RJT, P_BSY, P_RJT or LCR)	N/A
FC_Port transmits R_RDY	N/A

18.5.3 BB_Credit

BB_Credit represents the number of receive buffers supported by an FC_Port for receiving Class 1 and 6/SOF_{c1}, Class 2, or Class 3 frames. BB_Credit values of the attached FC_Ports are mutually conveyed to each other during the Fabric Login through the Credit field of common Service Parameters (see 15.6.2.3). The minimum or default value of BB_Credit is one.

BB_Credit is used as the controlling parameter in buffer-to-buffer flow control.

18.5.4 BB_Credit_Count

BB_Credit_CNT is defined as the number of unacknowledged or outstanding frames awaiting R_RDY responses from the directly attached FC_Port. It represents the number of receive buffers that are occupied at the attached FC_Port. To track the number of frames transmitted for which R_RDY responses are outstanding, the transmitting FC_Port uses the BB_Credit_CNT.

18.5.5 BB_Credit management

BB_Credit management involves an FC_Port receiving the BB_Credit value from the FC_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 15.6.2).

The transmitting FC_Port is responsible to manage BB_Credit_CNT with BB_Credit as its upper bound.

18.5.6 Buffer-to-buffer flow control model

The buffer-to-buffer flow control model is illustrated in figure 26. The model includes flow control parameters, control variables for a Class 2, Class 3, or Class 1 and 6/SOF_{c1} 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 FC_Port provides a number of receive buffers for Connectionless service. Each Nx_Port obtains the allocation of receive buffers from the Fx_Port (or Nx_Port in case of point-to-point topology) to which it is attached, as BB_Credit. Each Fx_Port obtains the allocation of receive buffers from the Nx_Port to which it is attached, as total BB_Credit for Connectionless service.

18.5.7 Class dependent frame flow

The class dependent flow of frames accomplishing buffer-to-buffer flow control for the following cases are illustrated in the figures referenced:

- Class 1 and 6/SOF_{c1} frame with delivery or non-delivery to the Fabric (see figure 27). Possible responses from the Fx_Port or an Nx_Port to a Class 1 and 6/SOF_{c1} frame are illustrated.
- Class 1 and 6/SOF_{c1} frame with delivery or non-delivery to an Nx_Port (see figure 28). Possible responses from the Fx_Port and the destination Nx_Port to a Class 1 and 6/SOF_{c1} frame are illustrated.
- Class 2 with delivery or non-delivery to the Fabric (see figure 29). Possible responses from the Fx_Port or an Nx_Port within the Fabric to a Class 2 are illustrated.
- Class 2 with delivery or non-delivery to an Nx_Port (see figure 30). Possible responses from the Fx_Port and the destination Nx_Port to a Class 2 Data frame are illustrated.
- Class 3 (see figure 31). Possible responses from the Fx_Port and the destination Nx_Port to a Class 3 Data frame are illustrated.

18.5.8 R_RDY

For any Class 2, Class 3 Class 1/SOF_{c1} or Class 6/SOF_{c1} frames received at an FC_Port, an R_RDY is issued when a receive buffer is available.

18.5.9 BB_Credit_Count reset

BB_Credit_Count is reinitialized to the login value for both Nx_Port and Fx_Port on Fabric Relogin or after the Link Reset Protocol has been performed.

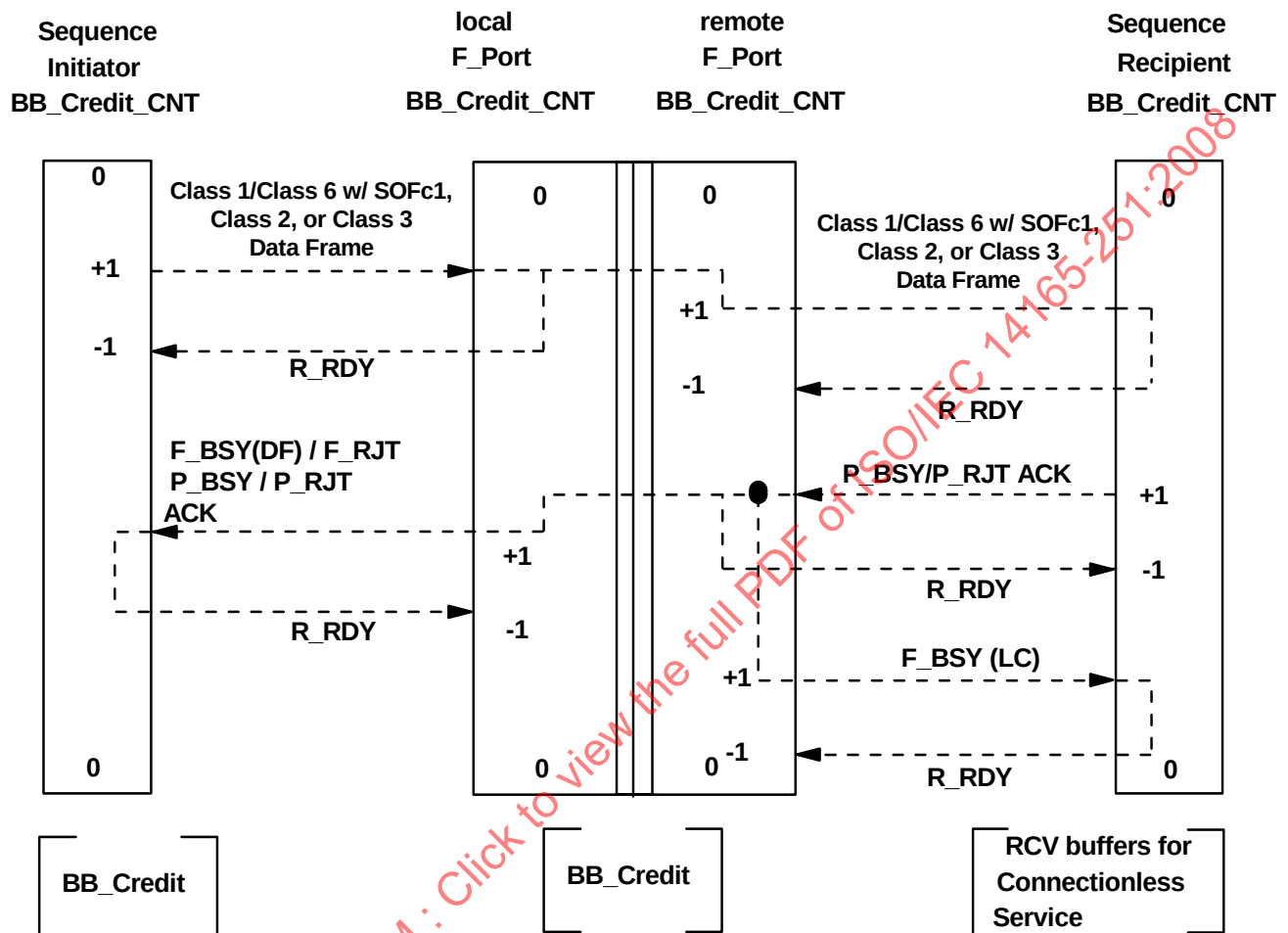


Figure 26 - Buffer-to-buffer flow control model

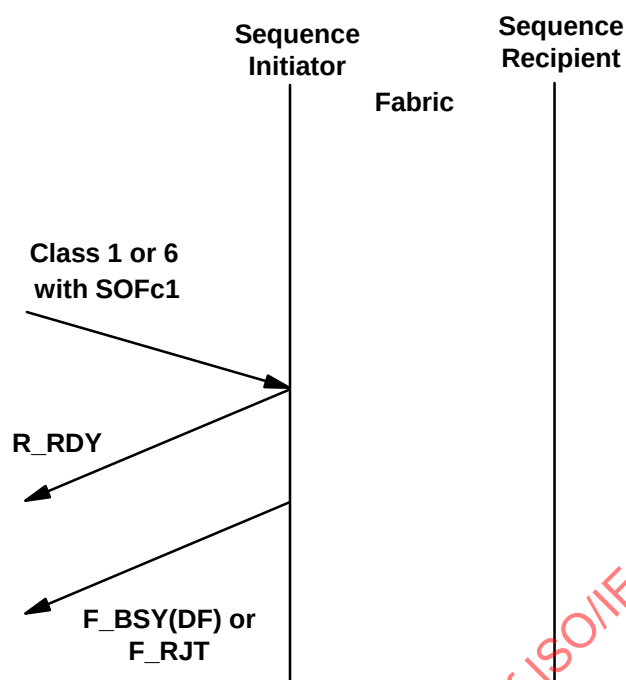


Figure 27 - Class 1 or 6/SOFc1 frame flow with delivery or non-delivery to the fabric

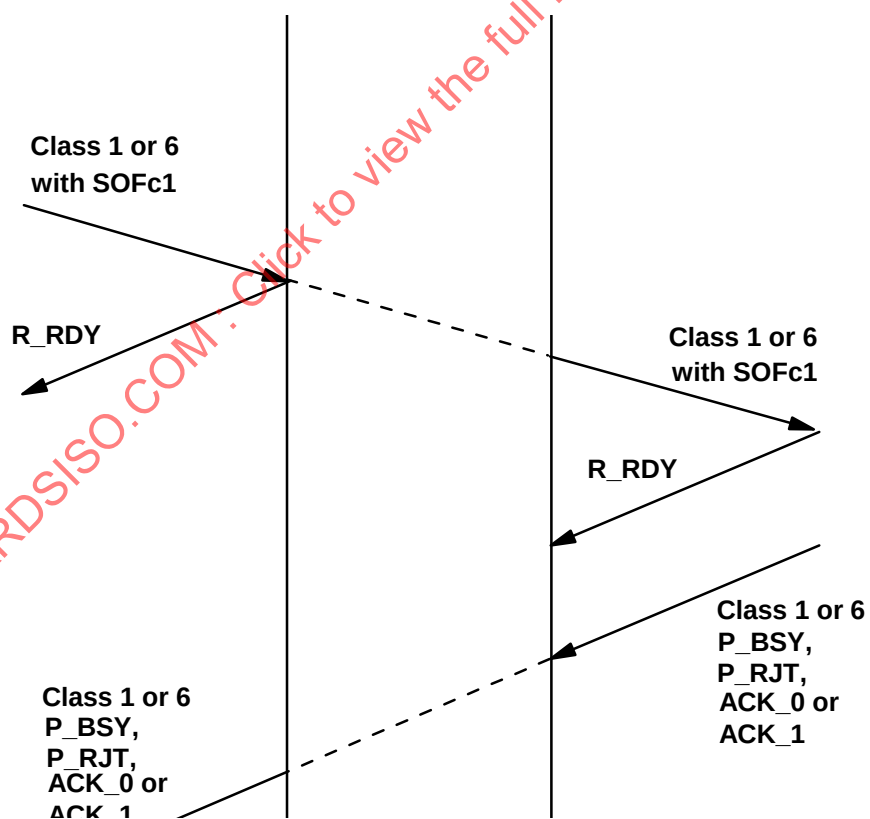


Figure 28 - Class 1 or 6/SOFc1 frame flow with delivery or non-delivery to an Nx_Port

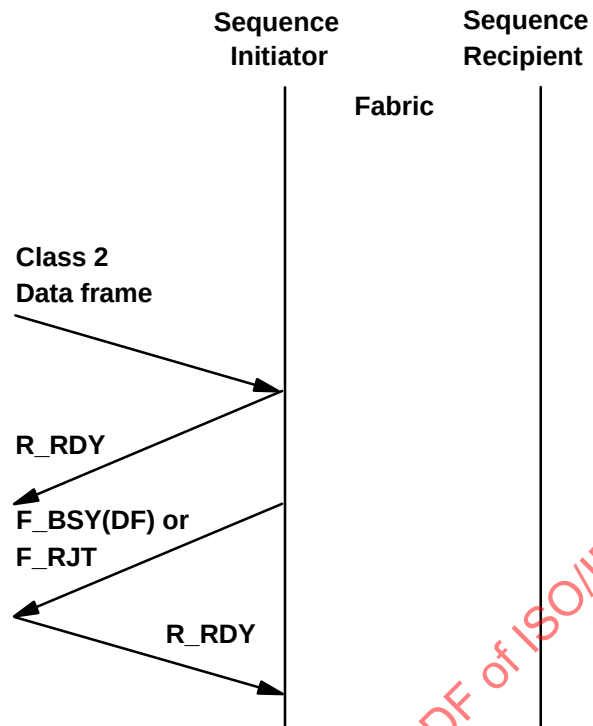


Figure 29 - Buffer-to-buffer - Class 2 frame flow with delivery or non-delivery to a Fabric

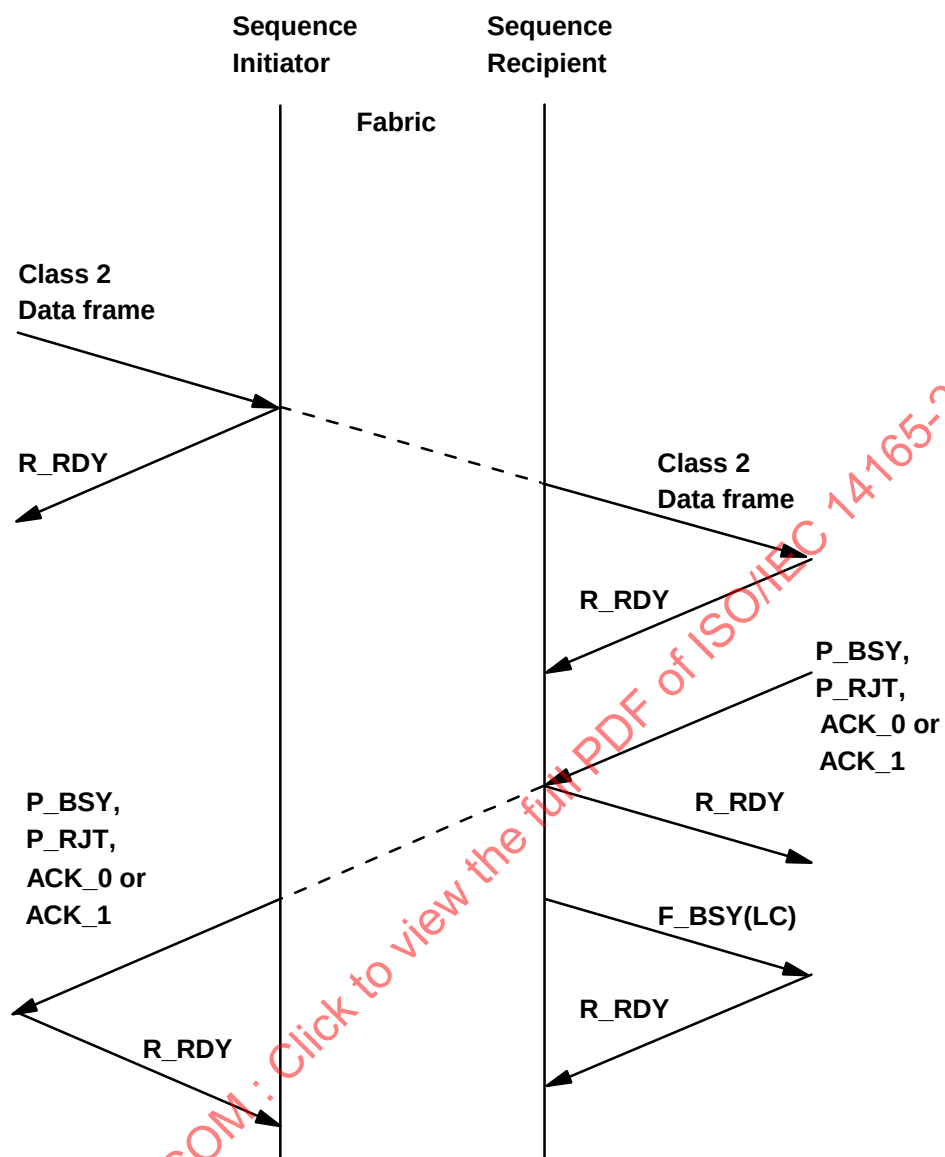


Figure 30 - Buffer-to-buffer - Class 2 frame flow with delivery or non-delivery to an Nx_Port

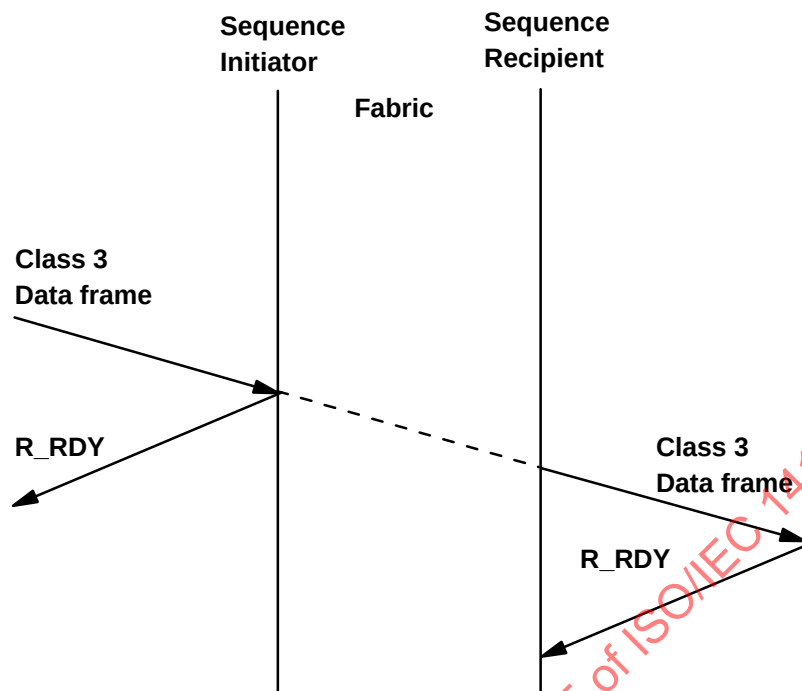


Figure 31 - Buffer-to-buffer - Class 3 frame flow

18.5.10 Alternate buffer-to-buffer Credit management

An alternate buffer-to-buffer credit management may be used instead of the one described in 18.5.

The use of alternate buffer-to-buffer Credit management shall be indicated by the Nx_Port through an N_Port Login Service Parameter during Fabric Login and N_Port Login (see 15.6.2.4.6).

Alternate BB_Credit management rules are summarized (See [2], FC-AL-2 for additional details):

- Each Port is responsible for managing the Alternate BB_Credit.
- During Login, BB_Credit shall be set to a value that represents the number of receive buffers that a FC_Port shall guarantee to have available as soon as a circuit is established. If this value is greater than zero, the FC_Port may start transmitting a frame without waiting for R_RDYs. If this value is equal to zero, the sending FC_Port shall wait to receive at least one R_RDY, before transmitting a frame.
- The receiving FC_Port shall transmit at least one R_RDY, representing the number of additional receive buffers available, before, during, or after the reception of frames.
- The transmitting FC_Port shall decrement BB_Credit by one for each frame transmitted and increment by one for each R_RDY received.
- For transmitting frames, the Available Credit shall be greater than zero. The Available Credit at any given time is expressed by the following equation:

$$\text{Available Credit} = \text{Login\_BB\_Credit} + (\text{Number of R\_RDYs received} - \text{Login\_BB\_Credit}) - \text{Number of frames transmitted}$$

where

- Number of R_RDYs received $\geq$ Login_BB_Credit.
- The parenthetical expression is applicable only if it is positive, otherwise it is zero.

18.5.11 BB_Credit Recovery

BB_Credit recovery as described in this clause shall only be performed when two FC_Ports, not operating in arbitrated loop mode, have logged in with each other and have agreed to a non-zero BB_SC_N value.

An FC_Port that supports BB_Credit Recovery, shall maintain the following BB_Credit Recovery counters:

- a) BB_SC_N is the log2 of BB_Credit Recovery modulus.
- b) BB_RDY_N counts the number of R_RDY primitives received modulo $2^{BB\_SC\_N}$.
- c) BB_FRM_N counts the number of frames received modulo $2^{BB\_SC\_N}$;

To initialize the counters, each FC_Port capable of supporting BB_Credit Recovery shall perform the following steps:

- a) Set the above counters (a, b and c) to zero before fabric login.
- b) After having transmitted 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 FC_Port in the login is zero, the FC_Port shall discard or set to zero the above counters (a, b and c). If the BB_SC_N value is not zero, the FC_Port shall set BB_SC_N (a above) to the greater of BB_SC_N from the other FC_Port's login parameters or its own BB_SC_N value, and set to zero all other BB_Credit Recovery counters (b and c above).
- c) After explicit or implicit logout, discard or set to zero all counters (a, b and c above).

To recover any lost BB_Credit, each FC_Port shall perform the following steps. Each step of initialization and operation shall be processed atomically, (i.e., each step shall be completed before any other BB_Credit recovery operation):

- a) Transmit a BB_SCs primitive if $2^{BB\_SC\_N}$ number of frames that require BB_Credit have been sent since the start of communication, or since the last time a BB_SCs primitive was sent.
- b) Transmit 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 may be calculated using the following equation:

$$BB\_Credits\ lost = (2^{BB\_SC\_N} - BB\_RDY\_N) \text{ modulo } 2^{BB\_SC\_N}.$$
The BB_Credit_CNT shall then be decremented by the number of BB_Credits lost. BB_RDY_N is then set to zero, before the next R_RDY is received.
- f) When a BB_SCs primitive is received, the number of BB_Credits the other FC_Port has lost may be calculated using the following equation:

$$BB\_Credits\ lost\ by\ other\ FC\_Port = (2^{BB\_SC\_N} - BB\_FRM\_N) \text{ modulo } 2^{BB\_SC\_N}.$$
One R_RDY shall be resent for each BB_Credit that is lost. BB_FRM_N is then set to zero, before the next frame is received.

When the two FC_Ports performing the login specifies different non-zero values of BB_SC_N, the larger value shall be used. If either FC_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.

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 are not recovered by the scheme outlined in 18.5.11. 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.

18.6 VC_RDY

In Class 4, once a Class 4 circuit is activated to provide fractional bandwidth service between communicating Nx_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 from an 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 an 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 an N_Port to cause a frame for the indicated VC to be transmitted.

18.7 BSY / RJT in flow control

In Class 1 and Class 6 end-to-end flow control, neither F_BSY, F_RJT, nor P_BSY occurs, except for a Class 1 or 6/SOFc1. 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 Nx_Port are shared by multiple source Nx_Ports that may be multiplexing Data frames. This Class 2 multiplexing requires the distribution of Class 2 Credit to each source Nx_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).

18.8 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 32). In response to an LCR, the Fabric shall issue an R_RDY and may issue a F_BSY or F_RJT. The destination Nx_Port shall issue an R_RDY and may issue a P_RJT (see 11.2.3.4). The destination Nx_Port shall not issue a P_BSY to an LCR.

Flow control model for an LCR frame is illustrated in figure 33 that covers the buffer-to-buffer flow control for all possible responses to an LCR.

After issuing the LCR, the Sequence Initiator shall re initialize its EE_Credit to the N_Port Login value for the destination Nx_Port. The Sequence Initiator shall not wait for any response before reinitialized this Credit (see 18.3.5).

18.9 Integrated Class 2 flow control

Integrated buffer-to-buffer and end-to-end flow controls applicable to Class 2 is illustrated in figure 34 for a Data frame from the Sequence Initiator and its response from the Sequence Recipient.

Integrated Class 2 flow control management is summarized in table 252.

18.10 Intermix

The flow control model for Intermix is represented by the superposition of end-to-end and buffer-to-buffer flow controls (See figures 24 and 26).

18.11 Point-to-point topology

All the flow control models specified 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.

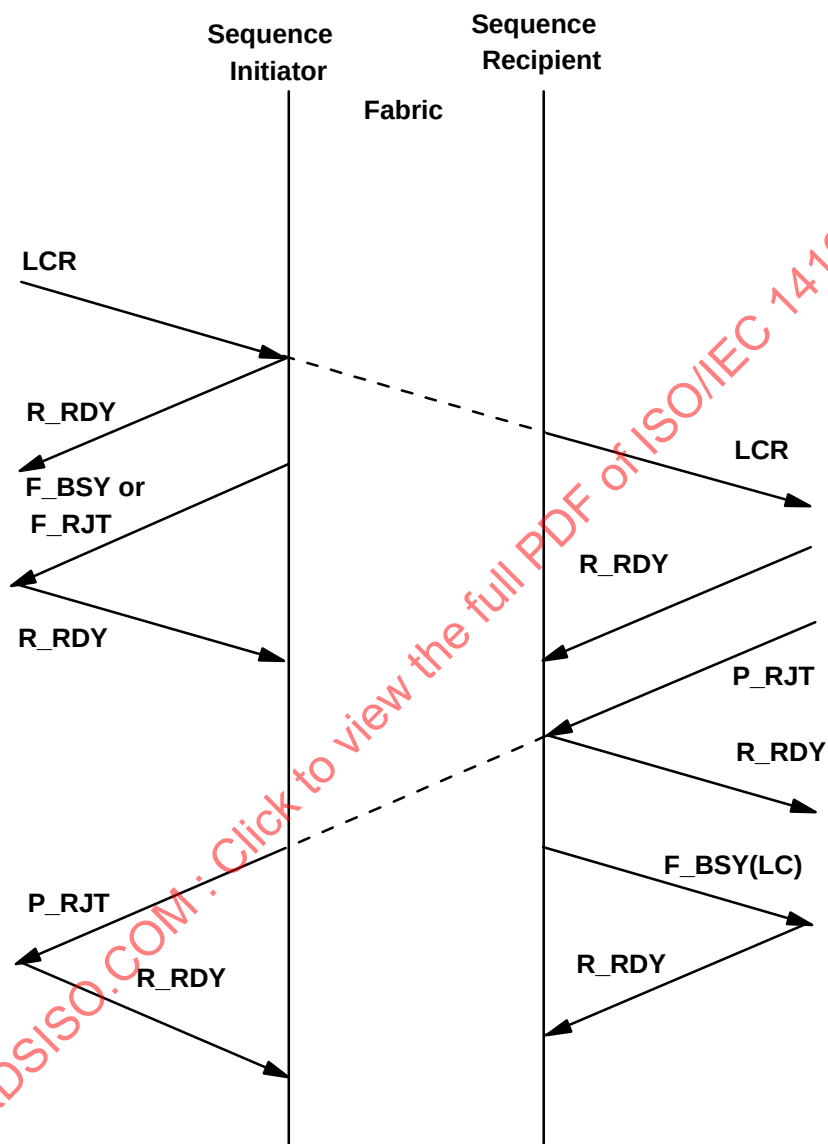


Figure 32 - LCR frame flow and possible responses

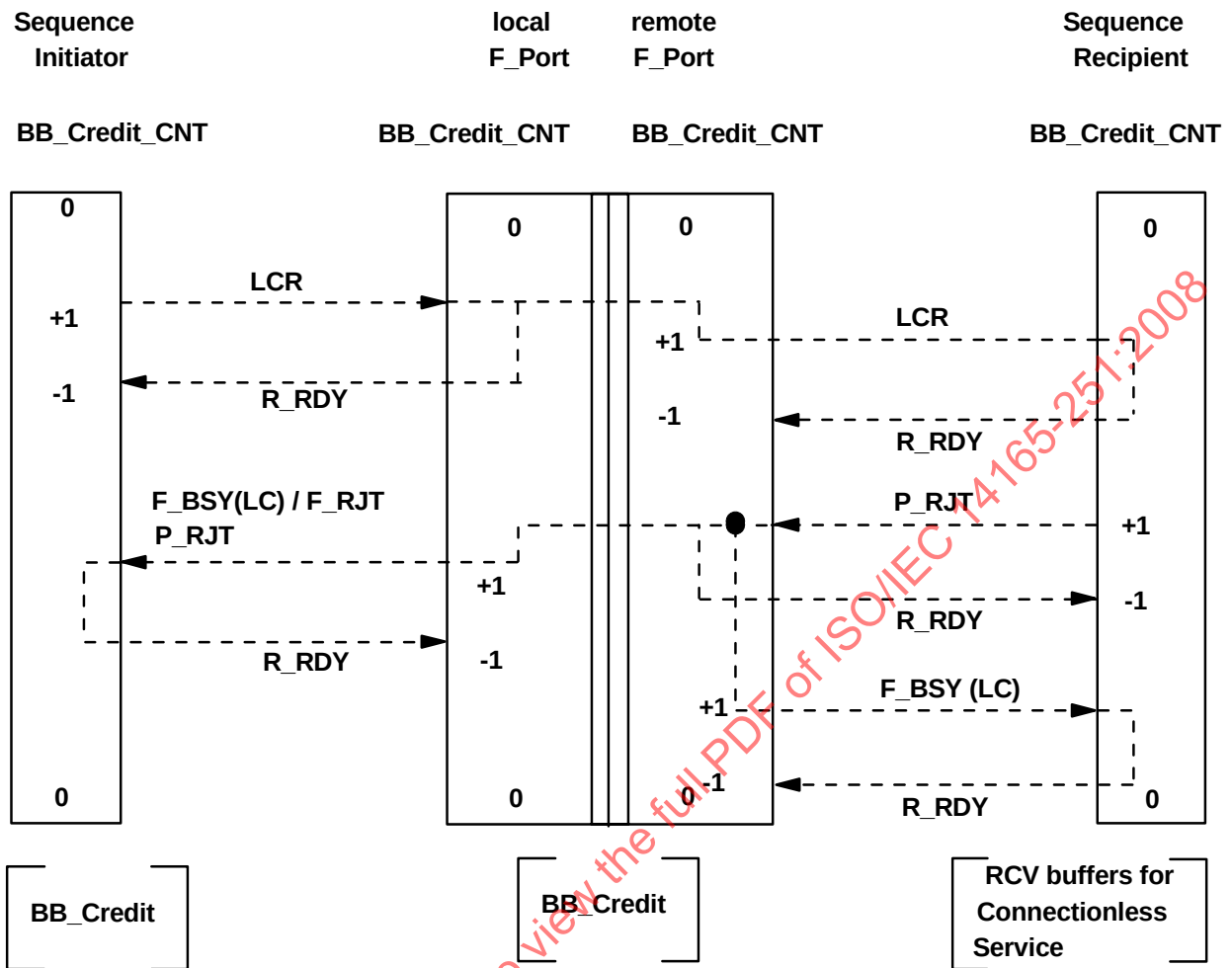


Figure 33 - LCR flow control model

Table 252 - Integrated Class 2 flow control management

Activity	Nx_Port		Fx_Port
	EE_Credit_CNT	BB_Credit_CNT	BB_Credit_CNT
Port transmits a Class 2 Data frame	Increment by one	Increment by one	Increment by one
Port transmits an LCR	Reinitializes to Login value	Increment by one	Increment by one
Port receives R_RDY	N/A	Decrement by one	Decrement by one
Port receives F_BSY(DF), F_RJT, P_BSY, or P_RJT	Decrement by one	N/A	N/A
Port receives F_BSY(LC)	N/A	N/A	N/A
Port receives ACK_1 (Parameter field: History bit = 1, ACK_CNT = 1)	Decrement by one	N/A	N/A
Port receives ACK_1 (Parameter field: History bit = 0, ACK_CNT = 1)	a) subtract 1 for current SEQ_CNT of the ACK_1 b) subtract all unacknowledged lower SEQ_CNTs (see 11.2.2.3)	N/A	N/A
Port receives ACK_0 (Parameter field: History bit = 0, ACK_CNT = 0)	N/A (see 11.2.2.3)	N/A	N/A
Port receives an LCR	N/A ^a	N/A	N/A
Port receives a Class 2 Data frame	N/A	N/A	N/A
Port transmits R_RDY	N/A	N/A	N/A
Port transmits F_BSY, F_RJT, P_BSY, P_RJT, or ACK	N/A	+1	+1
^a On receipt of LCR, the Sequence Recipient resets buffer (see 11.2.4)			

19 Segmentation and reassembly

19.1 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.

Mapping application data to Upper Level Protocols (ULPs) is outside the scope of this standard. ULPs maintain the status of application data transferred.

19.2 Sending end

19.2.1 Introduction

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 data block to be transferred within a Sequence.

19.2.2 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.

19.2.3 Data block

A data 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 data block shall be specified. A data block may map to all or part of relative offset space of the Information Category. An upper level shall specify an initial relative offset for each data block. The initial relative offset value may be zero or non-zero.

Data blocks may be divided into portions for transfer, only if the Sequence Recipient supports random relative offset. An upper level shall specify an initial relative offset for each portion. The initial relative offset value may be zero or non-zero. The initial relative offset values for portions of a given Information Category transmitted consecutively are allowed to be random.

A data block for a single Information Category may map to portions or all of relative offset space for the Information Category.

19.2.4 Sequence

The data blocks to be transferred within a single Sequence and the Information Category for each data block shall be specified to FC-2 by an upper level. An upper level shall specify the portions of data blocks to be transferred within a Sequence and the information Category for each portion to FC-2.

19.2.5 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.

19.3 FC-2

19.3.1 Mechanisms

FC-2 mechanisms to support this function are

- a) relative offset or SEQ_CNT,
- b) Sequence, and
- c) F_CTL bit indicating the presence of relative offset in the parameter field (see 9.7)

19.3.2 Relative offset

The Parameter field in the Frame_Header may be used to hold the value of relative offset that is the relative displacement of first byte of Payload related to an upper level-defined origin for a given Information Category.

19.3.3 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:

Application data _{i} = PL _{i} (m) || PL _{i} (m+1) || ... || PL _{i} (m+n) where
where

- a) PL (m) represents Payload of a frame with SEQ_CNT m
- b) n is the total frame count within the Sequence
- c) m is the lowest SEQ_CNT (zero or non-zero) and
- d) 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 65 536 arithmetic (i.e., SEQ_CNT = hex '00 00' follows SEQ_CNT = hex 'FF FF').

19.4 Login

The following Common Service Parameters related to segmentation and reassembly are interchanged during N_Port Login (see 15.6.2):

- a) relative offset by Information Category and
- 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.

19.5 Segmentation rules

Segmentation summary rules are listed and referred to table 253:

- 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 data blocks to be transferred as a Sequence, the Information Category for each data 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 data 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 data 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.
 - A) The Sequence Initiator shall set the relative offset present F_CTL bit to one.
 - B) 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 data block, namely, $RO_i(0) = \text{initial relative offset for the Information Category } i$
 - C) The Sequence Initiator shall use for all other frames of the data block, the relative offset computed as follows: $RO_i(n+1) = RO_i(n) + \text{Length of Payload}_i(n)$ where n is ≥ 0 and represents the consecutive frame count of frames for a given Information Category within a single Sequence.
 - D) Above steps A) through C) shall be repeated for each data 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.
 - A) The Sequence Initiator shall set the relative offset present F_CTL bit to one.
 - B) 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 data block, namely, $RO_i(0) = \text{initial relative offset for the Information Category } i$
 - C) The Sequence Initiator shall use for all other frames of the data block, the relative offset computed as follows: $RO_i(n+1) = RO_i(n) + \text{Length of Payload}_i(n)$ where n is ≥ 0 and represents the consecutive frame count of frames for the data block for a given Information Category within a single Sequence.
 - D) Above steps A) through C) shall be repeated for each data block within the Sequence.

19.6 Reassembly rules

Reassembly rules are listed and referred to table 253.

- 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 data 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 that the relative offset present bit (F_CTL bit 3) is set to zero and reassemble each of these Information Categories using SEQ_CNT. If the relative offset bit (F_CTL bit 3) 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."

- d) 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 that the relative offset present bit (F_CTL bit 3) is set to one and assemble each of these Information Categories using relative offset.
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.
- e) 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 that the relative offset present bit (F_CTL bit 3) is set to one and assemble each of these Information Categories using relative offset.
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.
- f) 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 "Invalid Parameter field".
- g) If the Sequence Recipient has indicated during Login relative offset support for one or more Information Categories and if the relative offset value in the Parameter field would cause transfer of data outside of vendor specific bounds enforced by the Sequence Recipient, the Sequence Recipient shall issue a P_RJT with a reason code of "Invalid Parameter field".

Table 253 - 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 Information Unit 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 data block: $RO_i(0)$ = initial relative offset for the data block specified All other frames of the data 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 Information Unit specific	Ignore parameter field SEQ_CNT shall be used
3	Random relative offset supported	F_CTL relative offset present bit = 1. Parameter field = relative offset. The Initial relative offset for a data block is permitted to be random.	Relative offset shall be used
		F_CTL relative offset present bit = 0 Parameter field meaning is protocol and Information Unit specific	Ignore parameter field SEQ_CNT shall be used

20 Connection management

20.1 Introduction

20.1.1 Establishing a Connection

Class 1 and Class 6 Service is based on establishing a dedicated connection between a source Nx_Port and a destination Nx_Port. The dedicated connection guarantees that the full bandwidth of the Link is available to each Nx_Port.

In order to establish a Class 1 and Class 6 dedicated connection, the source Nx_Port shall transmit a Data frame to a destination Nx_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 Nx_Port) or by the maximum Receive Data_Field size specified by the destination Nx_Port, whichever is smaller.

The Nx_Port shall receive an R_RDY Primitive Signal to indicate that the connect-request frame was received successfully and a buffer in the Fx_Port or Nx_Port is available. No additional frames shall be transmitted for the pending Connection until the ACK frame has been received. When the Nx_Port transmitting the connect-request receives an 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.

NOTE 1 A Connection is established from the Nx_Port's perspective (Connection Initiator) at the time the Nx_Port receives the ACK frame. It has no relation to the method or timing by which the Fabric actually forms the dedicated connection.

NOTE 2 It is recommended that a Fabric not introduce a phase discontinuity (see [8], FC-PI-2 and [1], 10GFC) while establishing a Class 1 Connection. No frame errors are introduced by such a discontinuity.

When a dedicated connection is established:

- a) Both Nx_Ports shall re initialize their end-to-end Class 1 and Class 6 Credit to the Login values.
- b) the Nx_Port initiating the connection shall be known as the Connection Initiator and the Nx_Port responding to the connect-request shall be known as the Connection Recipient.
- c) the Connection Initiator may continue the initial Sequence, and
- d) 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 20.5.4).

20.1.2 Removing a Connection

Removing a dedicated connection is accomplished by either Nx_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 Nx_Ports involved. Negotiation is required in order to avoid breaking the dedicated connection while frames are still flowing between the Nx_Ports.

In situations that do not require the use of the Link Reset Primitive Sequence, an Nx_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 Fx_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.

20.1.3 Preempting a dedicated connection

Interrupting and removing a Class 1 or Class 6 Dedicated Connection between Nx_Ports and making the Fabric

and Nx_Port resources previously consumed by the dedicated connection available is Preemption. A Preemption is initiated by an Nx_Port sourcing a Preemption Request frame (an SOF_x with the Preemption flag set, Word 1 bit 24 along with F_CTL bit 17 set to one) to the Fabric.

The preemptor Nx_Port, sourcing other than a Class 3 Preemption Request frame, 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 SOF_x having the appropriate S_ID and D_ID fields of the preemption request frame.

When the ACK is received either a new dedicated connection has been established or the Class 2 Preemption Request frame was successfully received by the destination Nx_Port. If an F_RJT is received, the Preemption Request was not successful 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 to 25 of the preemption request SOF_x header. If priority is used for preemption request resolution, a preemption request shall 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 the preemption of a dedicated connection all normal frame forwarding or Dedicated Connection rules (as appropriate) defined in this standard shall be followed.

20.1.4 Frame processing precedence

There are a number of different fields within a frame that affect the processing for establishing and removing dedicated connections. The general precedence hierarchy follows:

- a) Frame delimiters (SOF_{c1}, EOF_{dt}),
 - A) R_CTL
 - B) Data frame, or
 - C) Link_Control frame.
- b) F_CTL
 - A) Unidirectional Connection, when meaningful (bit 8),
 - B) End_Sequence (bit 19),
 - C) other bits when meaningful (e.g., if bit 19 = 1, check End_Connection) as specified in tables 35 and 36.

20.2 Applicability

Connection management applies to Class 1 and Class 6 Service. An Nx_Port supporting Class 1 or Class 6 Service may also support Class 2 and Class 3. Depending on the options supported by the Fabric, multiple class support by an Nx_Port may be complex. Because Class 1 and Class 6 involve dedicated connections, managing Class 1 and Class 6 usually overrides Class 2 or 3 management during periods of connection. This standard specifies the allowable responses on a class by class basis.

NOTE An Nx_Port engaged in Class 2 communication with another Nx_Port may experience Sequence timeouts if the other Nx_Port supports Class 1 and Class 2 without functioning in Intermix mode and a Class 1 Connection is created to the other Nx_Port.

20.3 Topology models

20.3.1 Introduction

An Nx_Port may be attached directly to another Nx_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 this standard.

20.3.2 Fabric model

The Nx_Port behavior described in this clause is based on a Fabric model in which an Fx_Port acts as the control point for establishing and removing Class 1 and Class 6 connections on behalf of the locally attached Nx_Port.

The following terminology is used in the discussion of Connection management. Nx_Port (A), (B), or (X) represent three separate Nx_Port_IDs where $A \neq B \neq X$. The side of the Fx_Port directly attached to the Nx_Port side is termed its Link side, whereas the side of the Fx_Port attached internally to the Fabric is termed its internal side.

The following Fx_Port characteristics are required behavior:

- a) When an Fx_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 an Fx_Port has accepted a connect-request from the Link, it reserves Fabric resources as it attempts to establish the requested dedicated connection.
 - A) the Fx_Port is Busy to other connect-requests from its internal side as destination Nx_Port.
 - B) the Fx_Port returns an F_BSY with EOFdt 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).
 - C) the Fx_Port returns an F_RJT with EOFdt to the Link if a reject condition is satisfied.
- c) After an Fx_Port has accepted a connect-request from the internal side:
 - A) it passes the connect-request to the Link as the destination Nx_Port.
 - B) it monitors its Link side for a proper confirmation response frame expected in response to the delivered connect-request.
 - C) it discards connect-request frames (SOFc1) 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 an Fx_Port encounters a collision case wherein connect-requests from both the internal side and the Link side arrive simultaneously, the Fx_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 is no longer used, the Fabric may notify each of the Fx_Ports attached to the Nx_Ports involved in an established or pending dedicated connection and each of the Fx_Ports may then transmit the Link Reset Primitive Sequence to its locally attached Nx_Port (i.e., initiate Link Reset Protocol).
See 5.5.4 for information on the effect of Primitive Sequences on Fx_Ports.

20.3.3 Point-to-point model

A point-to-point topology is indicated during the Login procedure. Two Nx_Ports arranged in a point-to-point connection may choose to:

- a) establish one dedicated connection for the duration of an operating period, or
- b) establish and remove dedicated connections dynamically, as the need to communicate arises.

20.4 Connect/disconnect rules

20.4.1 Connect-request rules

20.4.1.1 Source of connect-request

See 20.3 for the definition of Nx_Port (A), Nx_Port (B) and Nx_Port (X).

The following rules specify the connect-request procedure as the source (A) of the connect-request:

- a) An Nx_Port (A) shall initiate a connect-request using a Data (Device_Data or Link_Data) frame with an SOFc1 delimiter directed to destination Nx_Port (B). The connect-request frame is formed as follows:
 - A) an SOFc1 delimiter
 - B) a Data (Device_Data or Link_Data) frame
 - C) an S_ID of Nx_Port (A) and a D_ID of Nx_Port (B)
 - D) the E_C bit (F_CTL bit 18) shall be set to zero
 - E) the Unidirectional bit (F_CTL bit 8) shall be set to zero for a bi-directional Connection, and to one for a unidirectional Connection
 - F) an EOFn ending delimiter
- b) The Data Field of the connect-request shall be limited to the smaller of
 - A) the maximum buffer-to-buffer Receive_Data_Field size specified by the Fabric, if present, or
 - B) the maximum Receive_Data_Field size specified by the destination Nx_Port.
- c) After Nx_Port (A) transmits the connect-request frame, Nx_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 an ACK frame has responded to the connect-request frame. A proper response frame consists of an ACK_1 or ACK_0 frame with
 - A) an SOFn1 delimiter
 - B) an S_ID of Nx_Port (B), and a D_ID of Nx_Port (A)
 - C) an EOFn, or EOFt delimiter
- e) The destination Nx_Port, as an alternative response, may also send a P_BSY or P_RJT with:
 - A) an SOFn1 delimiter,
 - B) an S_ID of Nx_Port (B), and a D_ID of Nx_Port (A), and
 - C) an EOFdt delimiter.
- f) The Fabric, as an alternative response, may also send an F_BSY or F_RJT frame with:
 - A) an SOFn1 delimiter,
 - B) an S_ID of (B), and a D_ID of (A), and
 - C) an EOFdt ending delimiter.
- g) After a dedicated connection is established, Nx_Port (A) shall be the Connection Initiator and Nx_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, Nx_Port (A), may continue transmitting its initial Sequence and initiate other Sequences with SOFi1 up to Nx_Port (B)'s ability to support Concurrent Sequences.

20.4.1.2 Destination of connect-request

The following rules specify the connect-request procedure at the destination Nx_Port (B) of the connect-request:

- a) If a Data frame started by SOFc1 is received when Nx_Port (B) is not connected and Nx_Port (B) is busy, Nx_Port (B) responds with P_BSY with an EOFdt delimiter as specified in 20.4.1.1 item number e).
- b) If a Data frame started by SOFc1 is received when Nx_Port (B) is not connected and Nx_Port (B) rejects the connect-request, Nx_Port (B) responds with P_RJT with an EOFdt delimiter as specified in 20.4.1.1 item number e).
- c) If a Data frame started by SOFc1 is received when Nx_Port (B) is not connected and not busy, Nx_Port (B) responds with the proper response frame. A proper response frame is defined in 20.4.1.1 item number d). A dedicated connection is established with Nx_Port (A). Nx_Port (B) shall be the Connection Recipient and Nx_Port (A) shall be the Connection Initiator.
- d) With a Fabric present, if Nx_Port (B) receives a connect-request frame from Nx_Port (X) after a connect-request has been transmitted by Nx_Port (B), Nx_Port (B) requeues its own request for transmission at a later time and responds with a proper response frame to Nx_Port (X).

NOTE Nx_Port (B) requeues its original connect-request because the Fabric has discarded it. Nx_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 Nx_Port (B).

A dedicated connection is established with Nx_Port (X) with Nx_Port (B) as Connection Recipient.

- e) Without a Fabric present, if Nx_Port (B) accepts a connect-request frame from Nx_Port (A) after a connect-request has been transmitted by Nx_Port (B), Nx_Port (B) shall respond as follows:
 - A) if $A > B$, in value,
 - a) Nx_Port (B) requeues its own request for transmission at a later time,
 - b) responds to (A) with a proper response frame,
 - c) a dedicated connection is established with Nx_Port (A) with Nx_Port (B) as Connection Recipient, and
 - d) Nx_Port (B) may reinitiate its connect-request Sequence using SOFi1.
 - B) if $A < B$, in value,
 - a) Nx_Port (B) discards connect-request from Nx_Port (A), and
 - b) waits for a proper response frame.
- f) After a dedicated connection is established (i.e., the ACK to the connect-request is transmitted), Nx_Port (B) may begin initiating Sequences with SOFi1 up to the destination Nx_Port's ability to support Concurrent Sequences when the Connection is bi-directional.

20.4.2 Connection Rules

The following are the connection rules:

- a) 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.
- b) If a unidirectional Connection is established (F_CTL bit 8 = 1), it shall remain unidirectional until the Connection Initiator sets F_CTL bit 8 to zero, if ever, on the first or last frame of a Sequence (see 9.7.12) subsequent to the connect-request frame.
- c) If a unidirectional Connection is made bi-directional, it shall remain a bi-directional Connection for the life of the Connection.
- d) The Connection Recipient may request that a unidirectional Connection be made bi-directional by setting F_CTL bit 8 to zero on an ACK in response to a Data frame. Once F_CTL bit 8 is set to zero on an ACK, it shall remain set to zero for the life of the Connection.

20.4.3 Remove Connection rules

The following are the remove connection rules:

- a) 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.
- b) If an Nx_Port detects an abnormal condition that requires immediate removal of an existing Connection, the Nx_Port shall use the Remove Connection (RMC) Basic Link Service frame with the appropriate F_CTL bit settings that includes setting the E_C bit = 1 in order to remove the dedicated connection. All open 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, that require Link Reset.
- c) An Nx_Port shall transmit the E_C bit in F_CTL set to one on the last Data frame of a Sequence to indicate:
 - A) it is ready to end the Connection,
 - B) it shall not initiate any new Sequences, and
 - C) it requests the other Nx_Port to complete its active Sequences and not initiate any new Sequences.
- d) If an Nx_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.
- e) 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 EOF_{dt} delimiter.
- f) 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
 - A) does not initiate any new Sequences (unless C_S bit is received),
 - B) completes active Sequences,
 - C) transmits the E_C bit set to one on the last Data frame of its last active Sequence for the Connection.
- g) If an Nx_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,
 - A) the Connection Recipient shall respond with an ACK with an EOF_t delimiter, whereas
 - B) the Connection Initiator shall withhold transmitting ACK until after its Sequence is complete.
 - C) the Connection Initiator shall transmit the ACK with the EOF_{dt} delimiter.

20.5 Establishing a Connection

20.5.1 Introduction

A dedicated connection is established with an Nx_Port as the source of a connect-request (Connection Initiator) or as the destination Nx_Port (Connection Recipient) of a connect-request from another Class 1 or Class 6 Nx_Port.

20.5.2 Connection Initiator

When FC-2 receives a request from an FC-4 or upper level to initiate a Class 1 or Class 6 Sequence when a dedicated connection does not exist, the Nx_Port shall also establish a Class 1 or Class 6 Connection with the destination Nx_Port as part of the Sequence initiation.

The Nx_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:

- a) the maximum buffer-to-buffer Receive_Data_Field size specified by the Fabric, if present, or
- b) the maximum Receive_Data_Field size specified by the destination Nx_Port.

After the Nx_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 Nx_Port receives an R_RDY Primitive in response to the connect-request to indicate successful delivery to the Fx_Port or Nx_Port and that a buffer is available for a connectionless frame. If an N_Port is not operating in Intermix mode, the Nx_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 Nx_Port. A proper ACK frame is defined in 20.4.1.1 item number d).

Table 254 defines Event 1 as the connect-request and events 2 through 9 define the possible responses.

Table 254 - Responses to connect-request (SOF_{c1})

Event	SOF	D_ID	S_ID	Frame	EOF	Nx_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 Nx_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 EOF _{dt}	- dedicated connection established - Continue transmitting Sequence - Sequence ended, Connection established
7)	SOF _{c1}	A	B	Data frame	EOF _n	Nx_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 SOF_{n1} on ACK - dedicated connection established with B. - retransmit request associated with event 1 with SOF_{i1}
8)	SOF _{c1}	A	X	Data frame	EOF _n	Fabric is present. - Requeue request associated with event 1 (unless stacked connect-requests used) - Respond with SOF_{n1} on ACK_0 or ACK_1 - Dedicated connection established with X.
9)						- Timeout, no response frame. - Perform Link Reset Protocol. (see 7.6.3)

Event 1: A connect-request is transmitted by Nx_Port (A) with an SOF_{c1} delimiter with a destination of Nx_Port (B).

Event 2: An F_BSY indicates that the Fabric is unable to access the destination Nx_Port due to a busy condition internal to the Fabric. Try again later.

Event 3: A P_BSY indicates that the destination Nx_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 Nx_Port is unable to establish the dedicated connection. The reason code specifies the cause.

Event 6: A Dedicated connection has been established. Nx_Port (A) is Connected to Nx_Port (B).

Nx_Port (A) is the Connection Initiator and Nx_Port (B) is the Connection Recipient.

Nx_Port (A) may continue transmitting the Sequence initiated (EOF_n), or the Sequence that initiated the Connection may be complete (EOF_t).

Nx_Port (A) may initiate other Sequences with the same destination Nx_Port (B) up to the maximum number of Sequences defined by the Service Parameters obtained from (B) during Login.

The connected Nx_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 Nx_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 Nx_Port (A) discards the connect-request and waits for a response (Nx_Port (B) requeues request with SOF_{i1}).

In the case of a PTP topology where (A) < (B), Nx_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}). Nx_Port (A) responds as the destination of the connect-request with an appropriate ACK frame and becomes the Connection Recipient. Nx_Port (B) is the Connection Initiator.

Event 8: In the case of a Fabric present, Nx_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). Nx_Port (A) responds as the destination of the connect-request with an appropriate ACK frame and becomes the Connection Recipient. Nx_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 7.6.3).

See 20.4.1 for the rules that a source Nx_Port of a connect-request shall follow.

20.5.3 Stacked connect-requests

Stacked connect-requests are a feature that may be provided by a Fabric (see clause 26). An Nx_Port during Login with the Fabric determines support for Stacked connect-requests. If Stacked connect-requests are functional, an Nx_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 ELS command. Stacked connect-requests allow the Fabric to work on establishing a Connection while an Nx_Port is busy servicing another Connection and may enhance system performance. An Nx_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 (i.e., an ACK response shall be received within a CR_TOV timeout period or an F_BSY shall be returned from the Fabric to the Nx_Port. If either condition is not met within CR_TOV, the Nx_Port detects a Sequence timeout and Connection Recovery (see 20.8) is performed). A Fabric that supports stacked connect-requests shall deal with any associated race conditions that 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:

- a) transparent mode, or
- b) lock-down mode.

In transparent mode, when the SOF_{c1} Data frame is delivered to the destination Nx_Port, the return path of the bi-directional circuit is established in the same manner as Exclusive dedicated connections. As a result, the destination Nx_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 Nx_Port, the return path of the bi-directional circuit is not necessarily established to the source Nx_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 Nx_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 Fx_Port service parameters (see 15.6).

20.5.4 Unidirectional dedicated connection

When an Nx_Port transmits a connect-request, it may set F_CTL, bit 8 to one to indicate that a unidirectional dedicated connection is being established. Unidirectional means that only the Connection Initiator Nx_Port shall be able to transmit Data frames. The Connection Recipient Nx_Port 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 set F_CTL bit 8 equal to zero 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 = 0 on a meaningful Data frame (first or last) of a subsequent Sequence, the Connection becomes bi-directional for the remainder of the Connection.

To become bi-directional, a Connection Recipient of a unidirectional Connection may request that change by setting F_CTL bit 8 to zero 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.

20.5.5 Destination of connect-request

When Nx_Port (B) is not connected, but is available, and it receives a connect-request as the destination Nx_Port, Nx_Port (B) transmits the appropriate ACK frame (ACK_1 or ACK_0) to Nx_Port (A) which is requesting the connection. After the ACK frame has been transmitted with SOF_{n1}, a dedicated connection is established with Nx_Port (A) as the Connection Initiator and Nx_Port (B) as the Connection Recipient. After a Connection has been established, Nx_Port (B) may initiate Sequences with Nx_Port (A) using an SOF_{i1} delimiter.

If Nx_Port (B) is not connected, but is busy, and it receives a connect-request as the destination Nx_Port from Nx_Port (A), it responds with P_BSY with an EOF_{dt} delimiter.

See 20.4.1 for the rules that a destination Nx_Port of a connect-request shall follow.

20.6 Connected

When an Nx_Port is in a dedicated connection, it may receive Sequences as the Sequence Recipient as well as initiate Sequences as the Sequence Initiator.

When an Nx_Port has acted as a Connection Initiator and is in a Unidirectional Connection, it may only initiate Sequences and shall not receive Sequences. When an Nx_Port has been the Connection Recipient and is in a Unidirectional Connection, it may only receive Sequences and shall not initiate Sequences.

20.7 Removing a Connection

20.7.1 Introduction

Class 1 Connections may be removed in four ways:

- a) using the remove connection procedure that uses the E_C bit in F_CTL for negotiation,
- b) the Remove Connection command, that forces the recipient Nx_Port to transmit EOFdt,
- c) the Link Reset Protocol, or
- d) 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.

All Class 6 dedicated connections shall be Unidirectional dedicated connections.

20.7.2 When to remove a Connection

FC_FS does not specify the method to be employed in determining when to end a connection. Normally, an Nx_Port chooses to remove a dedicated connection when it has no Sequences to transmit to the Connected Nx_Port and it has a request to transmit Sequences to a different Nx_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 an Nx_Port from removing a Connection while using the E_C bit in the normal remove connection procedure.

NOTE In order to maximize system utilization, an Nx_Port should remove a dedicated connection when it has no Data to transmit in order to be available for Connection to another Nx_Port.

Two information bits are available in F_CTL to indicate when the next Sequence of an Exchange is estimated to be transmitted: immediately ('01' b), soon ('10' b), or delayed ('11' b). The indication of delayed transmission may be used as one condition to assist in determining the appropriate time to remove a dedicated connection.

20.7.3 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 Nx_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:

- a) This Nx_Port wishes to maintain the existing dedicated connection.
- b) This Nx_Port may transmit another Sequence within this Connection.
- c) This Nx_Port may wait for a reply Sequence within this Connection.

E_C shall be transmitted as one on a Data frame to indicate:

- a) This Nx_Port is ready to remove the existing dedicated connection.
- b) This Nx_Port has completed all active Sequences and agrees not to initiate another Sequence
- c) This Nx_Port is requesting the destination Nx_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 Nx_Port requests the receiving Nx_Port to transmit EOF_{dt} on the ACK response frame if the receiving Nx_Port has completed all active Sequences, otherwise, transmit an ACK frame with EOF_t. When the Nx_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.

20.7.4 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 Nx_Port receiving the Data frame has completed its last active Sequence of the existing Connection.

If both Nx_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.

20.8 Connection Recovery

20.8.1 Introduction

In case of link errors, the state of the existing dedicated connection may not be known. 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 7.6.3). Errors within a specific Sequence are processed according to the rules for handling Sequence errors (see 21.7).

For example, the following events provide instances when an Nx_Port is unable to determine the state of an existing dedicated connection and should perform the Link Reset Protocol. Nx_Port (A) believes that a dedicated connection exists with Nx_Port (B):

- a) Nx_Port (A) receives an SOFc₁ from Nx_Port (C),
- b) Nx_Port (A) receives an SOFi₁ from Nx_Port (C), or
- c) Nx_Port (A) receives an SOFn₁ from Nx_Port (C).

20.8.2 Link timeout

When the last active Sequence during a Class 1 or Class 6 dedicated connection has detected a Sequence timeout, a Link Timeout is detected. For more discussion on Link Timeout see 21.2.3. The Link Reset Protocol is performed as described in 7.6.3.

20.8.3 Corrupted connect-request

If an Nx_Port is not engaged in a Class 1 or Class 6 dedicated connection, and it receives a frame started by SOFi₁ or SOFn₁ and the frame is detected as invalid, the Nx_Port discards the frame and performs the Link Reset Protocol.

20.9 Connection Preemption

20.9.1 Applicability

This preemption process applies only to the preemption of existing Class 1 or Class 6 connections. Other preemption processes may be defined that apply to other connection oriented classes.

20.9.2 Topology Model

An Nx_Port having the ability to be connected to another Nx_Port through a Fabric may be preempted. Any Nx_Port, if enabled, may cause a connection to be preempted.

20.9.3 Rules for Preemption

20.9.3.1 Preemptor

The following rules specify the procedure for preempting a dedicated connection with respect to the Preemptor.

- a) An Nx_Port shall initiate a preemption request using a Data (Device_Data or Link_Data) frame directed to the desired destination Nx_Port. The SOF_x preemption request frame shall be formed as follows:
 - A) an appropriate SOF_x delimiter
 - B) a Data (Device_Data or Link_Data) frame
 - C) a S_ID of the Preemptor Nx_Port and a D_ID of the Preemption destination Nx_Port(s)
 - D) the E_C bit (F_CTL bit 18) shall be set to zero
 - E) the Priority (F_CTL bit 17) shall be set to one, and the Priority field (Word 1, bits 31 to 25) shall contain an appropriate priority value.
 - F) the Unidirectional bit (F_CTL bit 8) shall be set to zero for a bi-directional Connection, and set to one for a unidirectional Connection
 - G) the preemption flag bit (Word 1, bit 24) shall be set to one.
 - H) an appropriate EOF_x delimiter
- b) The Data Field of the preemption request frame shall be limited to the smaller of
 - A) the maximum buffer-to-buffer Receive_Data_Field size specified by the Fabric, or
 - B) the maximum Receive_Data_Field size specified by the destination Nx_Port.
- c) The preemptor Nx_Port shall follow all frame transfer or dedicated connection rules (as appropriate), including all buffer-to-buffer and end-to-end flow control and sequence and exchange rules as specified in this standard.
- d) Preemption request response frames, including ACK, P_BSY, P_RJT, F_BSY, or F_RJT frames shall have an S_ID having a value representing the Preemption Destination and a D_ID having a value representing the preemptor Nx_Port.

20.9.3.2 Preempted Source

The following rules specify the procedure for a Preempted Source. The Preempted Source is the Connection Initiator for a Class 1 or Class 6 connection.

A Connection Initiator Nx_Port's dedicated connection has been removed if a basic link service is received consisting of a PRMT command with:

- a) an SOF_{n1} delimiter,
- b) an S_ID of the Preemptor and a D_ID of the Preempted Source, and
- c) an EOF_{dt} delimiter

The Preempted source Nx_Port should notify its host that its dedicated connection has been preempted.

20.9.3.3 Preempted Destination(s)

The following rules specify the procedure for a Preempted Destination. The Preempted Destination is the Connection Recipient Nx_Port for a Class 1 or Class 6 connection.

A connection recipient Nx_Port's dedicated connection has been removed if a basic link service frame consisting of a PRMT command with:

- a) an SOF_{n1} delimiter,
- b) an S_ID of the Preemptor and a D_ID of the Preempted Destination, and
- c) an EOF_{dt} delimiter

NOTE The Preempted destination Nx_Port should notify its FC-4 that its dedicated connection has been preempted.

20.9.3.4 Preemption Destination(s)

The preemption destination shall follow all of the rules for frame transfer and connections as defined in this standard. Response frames, if required, shall contain an S_ID with the value representing the preemption destination Nx_Port and a D_ID with the value representing the preemptor Nx_Port.

20.10 Establishing a Connection Using Preemption

20.10.1 Introduction

A dedicated connection is established between a preemption requesting (Connection Initiator) Nx_Port and preemption destination(s) (Connection Recipient(s)) Nx_Port(s) following the successful completion of the Preemption process specified in 20.9.

NOTE Frame transfers may be completed following a connection preemption using a Class 2 or Class 3 preemption request frame. While the process for establishing a frame transfer using a preemption request is not explicitly defined in this standard, it follows the requirements for Class 2 or Class 3 and as defined in 20.9.

20.10.2 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 Nx_Port shall attempt to establish a Class 1 or Class 6 connection with the destination Nx_Port(s) using an SOF_{c1} preemption request frame (preemption flag = 1) as part of a Sequence initiation.

The Nx_Port initiates a preemption request using the rules defined in 20.9.

Table 255 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 255 - Responses to Preemption Requests

Event	SOF	D_ID	S_ID	Frame	EOF	Nx_Port Action
1)	SOF _{c1}	B	A	Data frame	EOF _n	1) Transmit Preemption Request 2) Wait for confirmation frame
2)	SOF _{n1}	A	B	F_RJT	EOF _{dt}	Preemption Failed, Fabric Rejected request.
3)	SOF _{n1}	A	B	F_BSY	EOF _{dt}	Preemption Failed, Fabric Busy.
4)	SOF _{n1}	A	B	P_BSY	EOF _{dt}	Busy Nx_Port, Connection Removed.
5)	SOF _{n1}	A	B	P_RJT	EOF _{dt}	Nx_Port Rejected, Connection Removed.
6)	SOF _{n1}	A	B	ACK_1/ ACK_0	EOF _n / EOF _{dt}	1) Dedicated connection Established 2) Continue Transmitting Sequence 3) Sequence/Connection Ended.
7)	SOF _{c1}	A	Any	Connect- request frame	EOF _n	1) Requeue Preemption Request Associated with Event 1. 2) Respond with SOF _{n1} on ACK_1, ACK_0, or P_BSY.
8)						Time-out, no response frame. Perform Link Reset Protocol (see 7.6.3)
9)	SOF _{n1}	PS or PD	A	PRMT	EOF _{dt}	Dedicated connection Preempted, Notify Host

Event 1: A preemption Request is transmitted by Nx_Port (A) with an SOF_{c1} delimiter with a destination address of Nx_Port (B) and the preemption flag = 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 Nx_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 Nx_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. Nx_Port (A) is connected to Nx_Port (B).

- Nx_Port (A) is the Connection Initiator and Nx_Port (B) is the Connection Recipient.
- Nx_Port (A) may continue transmitting the Sequence initiated (EOF_n), or the Sequence that initiated the Connection may be complete (EOF_t).
- Nx_Port (A) may initiate other Sequences with the same destination Nx_Port (B) up to the maximum number of Sequences defined by the Service Parameters obtained from (B) during Login.
- The connected Nx_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 Nx_Port (A) during Login.

Event 7: Nx_Port (A) shall requeue the SOF_{c1} sent to the Fabric in Event 1 and may respond to the SOF_{c1}

received with an ACK_1 or ACK_0, or P_BSY. Once this connection is completed Nx_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 7.6.3)

See 20.4.1 for the rules that a source Nx_Port of a connect-request shall follow.

Event 9: When a connected source (i.e., Connection Initiator) or destination (i.e., Connection Recipient), receives a PRMT, the dedicated connection has been preempted and no longer exists. The Nx_Port should notify its host that the Sequence(s) were abnormally terminated due to the connection being preempted.

20.10.3 Preemption Destination

If an SOFc1 preemption request frame is received by an Nx_Port previously connected (i.e., connected prior to being preempted), it shall transmit the appropriate ACK frame (ACK_1 or ACK_0) to the preemptor Nx_Port. After the ACK frame has been transmitted with the SOFn1, a dedicated connection is established with the preemptor Nx_Port as the Connection Initiator and this destination Nx_Port(s) as Connection Recipient(s). After a Connection has been established, this destination Nx_Port may initiate Sequences with the preemptor Nx_Port using an SOFi1 delimiter, if the Connection is bi-directional.

NOTE Class 6 dedicated connections are uni-directional.

When an unconnected Nx_Port receives a preemption request it shall respond normally for a Class 1 connection request, as specified in 20.5.5.

21 Error detection/recovery

21.1 Introduction

Link integrity and Sequence integrity are the two fundamental levels of error detection in this standard. 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 data block transmitted and received. Exchange management ensures that Sequences are delivered in the manner specified by the Exchange Error Policy (see 21.6.3). Each frame within a Sequence is tracked on the basis of Exchange_ID, Sequence_ID, and a SEQ_CNT 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. For Classes 1, 2, 3 and 6 see clause 20 for a more complete discussion on flow control, buffer-to-buffer Credit, and end-to-end Credit. For Class 4, see clause 25 for a more complete discussion on flow control, virtual circuit Credit, and EE_C4_Credit. See 25.4 for Class circuit error recovery.

21.2 Timeouts

21.2.1 Timeout periods

21.2.1.1 General

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%.

21.2.1.2 R_T_TOV

The Receiver_Transmitter timeout value (R_T_TOV) is used by the receiver logic to detect Loss-of-Synchronization (see 6.1.5.2). The default value for R_T_TOV is 100 ms. A shorter value of 100 µs is allowed. FC_Ports that use the shorter value indicate this by setting the R_T_TOV bit in the Common Service Parameters during Login. An FC_Port may determine another FC_Port's R_T_TOV value using the Read Timeout Value (RTV) ELS (see 12.3.2.13).

21.2.1.3 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 timeout value for detection of a response to a timed event (i.e., during Data frame transmission it represents a timeout value for a Data frame to be delivered, the destination Nx_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 Nx_Port processing parameters. The default value is 2 s. However, a valid E_D_TOV value shall also adhere to the proper relationship to the R_A_TOV value. When an Nx_Port performs Fabric Login, the Common Service Parameters provided by the Fx_Port specify the proper value for E_D_TOV.

When an Nx_Port performs N_Port Login in a point-to-point topology, the Common Service Parameters provided by each Nx_Port specify a value for E_D_TOV. If the two values differ, each Nx_Port shall use the longer time. An FC_Port may determine another FC_Port's value for E_D_TOV via the Read Timeout Value (RTV) ELS (see 12.3.2.13). Timeout values as specified in the Payload of the LS_ACC are counts of either 1 ms or 1 ns increments, depending on the resolution specified (e.g., 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:

- a) transmission of consecutive Data frames within a single Sequence,

NOTE 1 For Class 4, when very low guaranteed bandwidth is specified, the delivery time between Data frames within a Sequence may exceed E_D_TOV and a mechanism outside of this standard should be used for detecting a Sequence timeout.

- b) retransmission of a Class 2 Data frame in response to an F_BSY or P_BSY, and
- c) transmission of ACK frames from the point in time that the event that prompted the acknowledgment action.

NOTE 2 For Class 4, when very low guaranteed bandwidth is specified, the delivery time between ACK frames within a Sequence may exceed E_D_TOV and a mechanism outside of this standard should be used for detecting a Sequence timeout.

For all other cases, E_D_TOV shall expire before an action is taken. Two such examples include:

- a) Link timeout (see 21.2.3), and
- b) Sequence timeout (see 21.2.4).

21.2.1.4 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 Reinstate 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 s.

When an Nx_Port performs Fabric Login, the Common Service Parameters provided by the Fx_Port specify the value for R_A_TOV. When an Nx_Port performs N_Port Login in a point-to-point topology, the Common Service Parameters provided by each Nx_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. An FC_Port may determine another FC_Port's value for R_A_TOV via the RTV ELS (see 12.3.2.13).

When R_A_TOV is used to determine when to reuse an Nx_Port resource (e.g., Recovery_Qualifier), the resource shall not be reused until R_A_TOV has expired for all frames previously transmitted that 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 not specified by this standard may also be employed.

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 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 Nx_Ports.

21.2.2 Link Failure timeouts

A Link Failure is detected under the following timeout conditions:

- a) Loss-of-Signal (see 6.1.5.2),
- b) Loss-of-Synchronization > timeout period (R_T_TOV) (see 6.1.5.2),
- c) Link Reset Protocol timeout (> R_T_TOV) (see 7.6.3).

21.2.3 Link timeout

In Class 1 and Class 6, 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, Class 3 or Class 6 (SOF_{c1}), a Link timeout error shall be detected if one or more R_RDY Primitive Signals are not received within E_D_TOV after the buffer-to-buffer Credit_CNT has reached zero.

In Class 4, a Link timeout error shall be detected if one or more VC_RDY Primitive Signals are not received within E_D_TOV after the VC Credit_CNT has reached zero. A Link Timeout determination for Class 4 shall add E_D_TOV to the normal expected arrival time of VC_RDYs from the QoSF for each Virtual Circuit.

Recovery from Link timeout is accomplished by performing the Link Reset Protocol (see 7.6.3).

Link timeout values need to take Fabric characteristics into consideration. In Class 1 and Class 6, 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.

21.2.4 Sequence timeout

21.2.4.1 Introduction

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 Nx_Port resources, Sequence timeout is applicable to both discard and process error policies.

21.2.4.2 Classes 1, 2 and 6

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 Responder to Data frame transmission is an ACK response. The expected event for the Initiator is another Data frame for the same Sequence that is active and incomplete. Other events (e.g., 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 21.7.2.2). Detection of a Sequence timeout by the Sequence Initiator may also result in aborting the Exchange (see 12.2.2 and 12.3.2.2).

If a Sequence Recipient detects a Sequence timeout, it shall set the Abort Sequence Condition (F_CTL bits 5 to 4) in an ACK to an '01' b, or '11' b 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 12.2.2) by the Sequence Initiator, abnormal termination of a Sequence (see 21.7.2) by the Sequence Recipient, or aborting the Exchange by either the Sequence Initiator or Sequence Recipient (see 12.3.2.2).

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.

21.2.4.3 Class 3

In Class 3, the expected event for the Recipient is another Data frame for the same Sequence. Other events (e.g., 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.

NOTE 1 For environments that do not use a request/response protocol, the Sequence Initiator may periodically transmits an ABTS frame and 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 may 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.

NOTE 2 For environments that use a request/response protocol, ABTS should not be used to forward progress of a transmission. 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.

21.2.4.4 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 LCR Link_Control frame shall be transmitted by the Sequence Initiator, that has no end-to-end Credit, to the Sequence Recipient. The Sequence Initiator (indicated by the S_ID in the LCR frame) shall perform normal recovery for the Sequence that timed out (see 21.7).

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 an Nx_Port receives a 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.

21.2.5 OLS transmit timeout

When a Port is performing the Online-to-Offline Protocol (see 7.6.5), the Port shall timeout the transmission of OLS (5 ms) before reception of a Primitive Sequence in response.

NOTE 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. The intent is to ignore received data while transmitting OLS for 5 ms

21.2.6 Timer Enhancements for Classes 1 and 6

21.2.6.1 Introduction

This standard defined two timers, E_D_TOV and R_A_TOV, that are used for normal frame flow. E_D_TOV is used by both the Sequence Recipient (SR) and the Sequence Initiator (SI) to time FC-2 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 shall never emerge from the Fabric after an R_A_TOV time-out.

These timers should be kept small to minimize the interval required to recognize time-out errors. Under certain conditions these time-out values may 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 subclause defines the Connection Request time-out Value (CR_TOV) (see figure 35) that is used by the Connection Initiator, the Connection Recipient, and the Fabric to control the connect-request process when Stacked Connect-request is invoked.

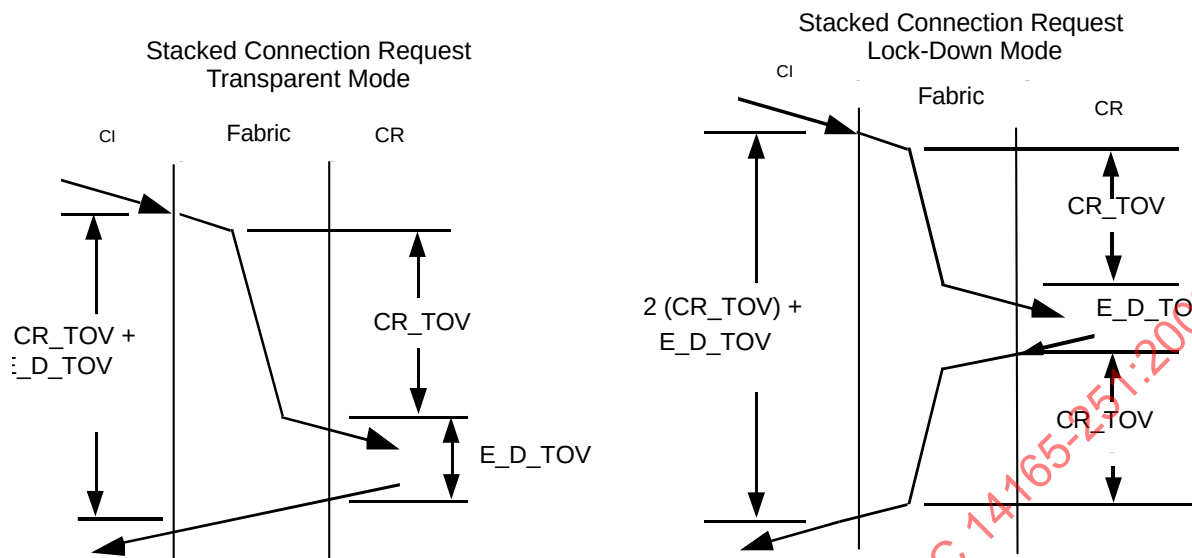


Figure 35 - Frame Flow Timers

21.2.6.2 Applicability

The Connection Initiator, Connection Recipient, and the Fabric shall use the CR_TOV timer to time connect-request frames that are transmitted as Stacked Connect-request.

21.2.6.3 Login

CR_TOV is specified by the Fx_Port during Fabric Login as part of the Class Service Parameters (see 15.6.5.11).

21.2.6.4 Value

The minimum CR_TOV value shall be R_A_TOV. The Fabric sets the maximum value of CR_TOV.

21.2.6.5 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.

21.2.6.6 Rules

21.2.6.6.1 Connection Initiator

The following are the rules for the Connection Initiator:

- Following the transmission of a connect-request frame invoking Stacked Connect-request, if the Connection Initiator receives an F_BSY in reply to that connect-request, the Connection Initiator shall retry that connect-request.
- The Connection Initiator 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 declares the connect-request frame to be lost and initiate error recovery.
- The Connection Initiator 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 declares the connect-request frame to be lost and initiate error recovery.

21.2.6.6.2 Connection Recipient

The following are the rules for the Connection Recipient:

- a) The Connection Recipient 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 Connection Recipient to have been lost.
- b) The Connection Recipient shall wait a minimum time period of $CR\_TOV + E\_D\_TOV$ following receipt of a connect-request frame that invoked Stacked Connect-request Lock Down mode, before considering the ACK to the Connection Recipient to have been lost.

21.2.6.6.3 Fabric

The following are the rules for the Fabric:

- a) The Fabric shall start the CR_TOV timer on receipt of a connect-request frame, that has invoked Stacked Connect-request.
- b) If the CR_TOV timer expires on a connect-request frame that has invoked Stacked Connect-request before the Fabric forwards it to the Connection Recipient, the Fabric shall discard the connect-request frame and Fabric transmit an F_BSY frame to the Connection Initiator.

21.2.7 Bit-Error-Rate Thresholding

21.2.7.1 Introduction

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 Report (RLIR) ELS shall be generated as required by the RLIR ELS (see 12.3.2.31).

The bit-error rate is measured during frame, Primitive Signal, 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.

21.2.7.2 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.

21.2.7.3 Error Bursts

A single error may result in several related errors occurring closely together that 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:

- a) An error burst is a time period 1,5 +/- 0,5 s long during which one or more invalid Transmission Words are recognized. This time may be exceeded due to infrequent unusual conditions.
- b) Only one error in an error burst is counted toward the bit-error threshold.

21.2.7.4 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 $\pm 0,3$ s.

The bit-error-counting process shall be restarted when Active State is entered and when a vendor-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.

21.3 Link error detection

21.3.1 Link Failure

The first level of link error detection is at the receiver. Link Failure is detected under the following conditions:

- a) Link Failure conditions (see 21.2.2), or
- b) Reception of the NOS Primitive Sequence (see 5.5.4.2).

Recovery from Link Failure is accomplished by performing the Link Failure Protocol (see 7.6.4).

21.3.2 Code violations

Code violations are link errors that result from an invalid transmission code point or disparity error. If a code violation occurs during Primitive Signals, 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 8.9), 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 table 8). In particular, even if the code violation is detected following the Frame_Header, fields in the header may not be valid.

21.3.3 Primitive Sequence protocol error

If an Nx_Port is in the Active State and it receives LRR, it shall detect a Primitive Sequence protocol error that is counted in the LESB.

21.4 Link error recovery

The Link recovery hierarchy is shown in figure 36.

The recovery protocols are nested and organized from the most serious to least serious link action.

- a) Link Failure Protocol (see 7.6.4)
- b) Link Initialization Protocol (see 7.6.2)
- c) Link Reset Protocol (see 7.6.3)

21.5 Link recovery - secondary effects

21.5.1 Class 1 and Class 6

During a Class 1 and 6 dedicated connection, transmission or reception of Primitive Sequences removes the connection. While an Nx_Port is transmitting a Primitive Sequence, it shall discard any subsequent Class 1 and

Class 6 frames received. For Class 1 intermix, see the discussion regarding Class 2 (see 21.5.2) and Class 3 (see 21.5.3). 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 the Nx_Port and Fx_Port involved.

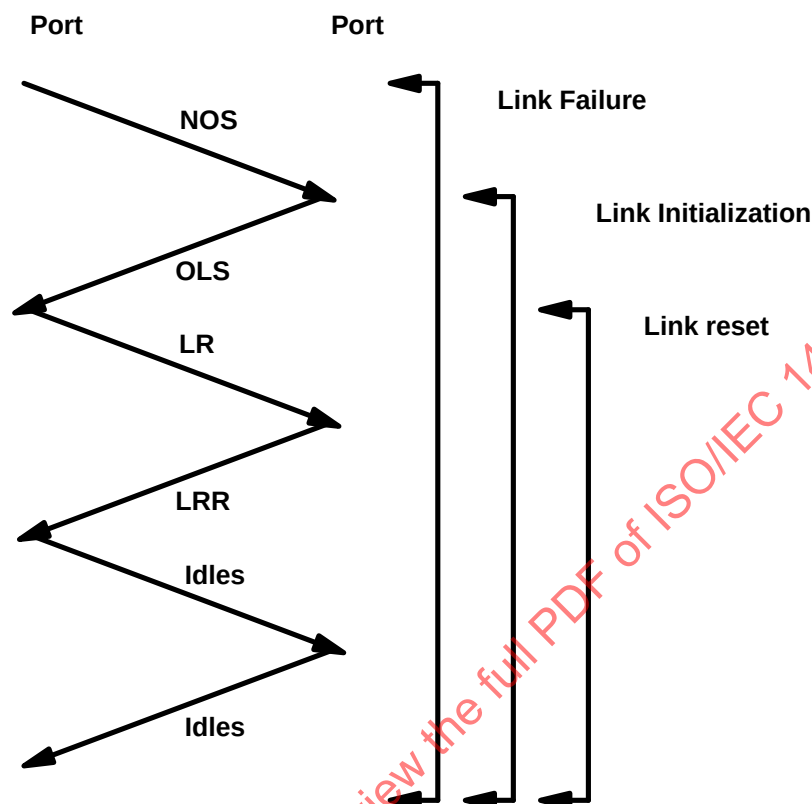


Figure 36 - Link recovery hierarchy

After leaving the Active State to perform a Primitive Sequence Protocol, all frame Sequences that are open 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 Nx_Port shall interrogate the other Nx_Port's Exchange Status Block to determine which Nx_Port has the Sequence Initiative (see 21.7.2).

21.5.2 Class 2

When Primitive Sequences are transmitted or received, the Fx_Port may discard any Class 2 frames held in its buffers. While an Nx_Port is transmitting a Primitive Sequence, it may discard any subsequent Class 2 frames received. After receiving the first Primitive Sequence of any Primitive Sequence Protocol, the buffer-to-buffer Credit shall be reset in the Nx_Port and the Fx_Port to the Login values. Both the Nx_Port and Fx_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.

21.5.3 Class 3

When Primitive Sequences are transmitted or received, the Fx_Port may discard any Class 3 frames held in its buffers. While an Nx_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 Nx_Port and the Fx_Port. Both the Nx_Port and Fx_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.

21.5.4 Class 4

The actions to be taken for Class 4 are specified in clause 7.

21.6 Exchange Integrity

21.6.1 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 Classes 1, 2, 4 and 6. Items applicable to Class 3 shall be specified explicitly.

21.6.2 Exchange management

An Exchange is managed according to the rules specified in 17.3.1. 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 Abort Sequence Condition bits (F_CTL bits 5 to 4). (see 9.7.14)

21.6.3 Exchange Error Policies

21.6.3.1 Introduction

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 that 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 17.3.10 for rules that discuss detailed requirements for each type of Discard and Process policy.

21.6.3.2 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.

21.6.3.3 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.

21.6.3.4 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 an Nx_Port to utilize the Data Field of invalid frames under certain restrictive conditions (see 8.9.2 and 8.9.3). The Process with infinite buffering Error Policy is intended for applications (e.g., 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.

21.6.3.5 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 and 6. It provides a method by which the Sequence Recipient may request immediate Sequence retransmission at the FC-2 level under the guidance or authorization of the FC-4 or upper level (see 21.7.2.3). Sequences shall be delivered to the FC-4 or upper level on a Sequence basis in the same order as transmitted.

21.6.4 Sequence integrity

Sequence management and integrity involves

- a) proper initiation of Sequences (see 17.3.4),
- b) proper control of the ordering of Sequences (SEQ_ID usage) with continuously increasing SEQ_CNT (see 17.3.6),
- c) proper control of Data frames by SEQ_CNT within single Sequence (see 17.3.6), and
- d) proper completion of a Sequence (see 17.3.8).

Frame identification (see 9.2.1) and Sequence identification (see 9.2.2) 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 17.3.5.

21.6.5 Sequence error detection

Sequence errors are detected in three ways

- a) detection of a missing frame (see 17.3.10 and 17.3.11),
- b) detection of a Sequence timeout (see 21.2.4), and
- c) detection of rejectable condition within a frame (see 11.2.3.4).

Detection of Sequence errors by the Recipient is conveyed in the Abort Sequence Condition bits (F_CTL 5 to 4) in an ACK frame or by a P_RJT frame (except Class 3). 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 17.8.1 and 17.3.14) and the Sequence Status Block (see 17.8.2 and 17.3.12), respectively.

21.6.6 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 that is beyond the scope of this standard.

21.7 Sequence recovery

21.7.1 Introduction

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. This standard specifies procedures to terminate or abort a Sequence, recover end-to-end Credit, handle missing or delayed frames, and synchronize both Nx_Ports with respect to Sequence and Exchange status. This standard does not require Sequence retransmission within the same Exchange in which a Sequence error is detected.

21.7.2 Abnormal Sequence termination

21.7.2.1 Introduction

There are multiple methods by which a Sequence may complete 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 17.3.8. The methods by which a Sequence completes abnormally include:

- a) 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.
- b) a Class 1 or 6 dedicated connection and Class 6 Multicast Connection is broken by transmission or reception of a Primitive Sequence while the Sequence is open.
- c) a Class 4 Virtual Circuit is broken by transmission or reception of a Primitive Sequence as specified in 7.6 when the Sequence is open.
- d) if the Exchange of which a Sequence is a member is abnormally terminated, each open Sequence shall also be abnormally completed (see 17.3.13).
- e) Logout (see 12.3.2.8).

A Sequence may complete normally with only a part of the Sequence being received by the Sequence Recipient in the Stop Sequence Protocol.

21.7.2.2 Abort Sequence Protocol

21.7.2.2.1 General Case

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 BA_ACC Payload along with a valid indication (see 12.2.3). The Payload of the BA_ACC 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 that have been delivered.

If the ABTS frame is transmitted in Class 2, 3 or 4, 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 Reinstate Recovery Qualifier Link Service request in order to purge the Recovery_Qualifier. Timing out the Recovery_Qualifier range for R_A_TOV allows both Nx_Ports to discard frames received in this range. This ensures the Data integrity of the Exchange.

NOTE Since streaming Sequences across multiple classes of service 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 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 or 6, then none is needed. If ABTS is transmitted in Class 2, 3 or 4, then a Recovery_Qualifier is established (e.g., if a Sequence Initiator has transmitted a Sequence in Class 1 or 6, 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 BA_ACC is transmitted and thereafter, until it receives a Reinstate Recovery Qualifier (RRQ) ELSS 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 BA_ACC is received.

In like manner, after the Sequence Initiator has received the BA_ACC to the ABTS frame, it shall discard any frames received for the Recovery_Qualifier range. In Class 2, 3 or 4, 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 BA_ACC, 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 17.3.8 and 17.3.11).

If a Recovery_Qualifier exists and is being timed out (R_A_TOV) and another Sequence error occurs that 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.

21.7.2.2.2 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 that 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 BA_ACC 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.

21.7.2.3 Class 1 or 6 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 the following

restrictions shall be placed on the use of Class 1 Sequence retransmission:

- a) Class 1 and Class 6 only (in-order delivery, with no frames delayed in the Fabric).
- b) the Sequence Initiator shall request Discard multiple Sequences with immediate retransmission as the Error Policy on the first Data frame of the Exchange.
- c) the Sequence Recipient shall not be required to support Sequence retransmission and may reply with Abort Sequence Condition bits set to '01' b if a missing frame error is detected.
 - A) 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 to 4) to '01' b to indicate a missing frame error.
 - B) 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 to 4) set to '01' b if a Data frame is received with the retransmit bit (F_CTL bit 9) set to one.
- d) 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 bits 5 to 4) to '11' b to indicate a missing Data frame error when:
 - A) 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 to zero.
 - B) the Sequence Recipient shall continue to transmit ACKs for the same SEQ_ID with the Abort Sequence Condition bits set to '11' b.
 - C) for any subsequent streamed Sequences, the Sequence Recipient shall set the Abort Sequence Condition bits to '01' b 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.
 - D) 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).
- e) 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 to 4) to '01' b in the corresponding ACK frame.
- f) when the Sequence Initiator receives an ACK with Abort Sequence Condition bits (F_CTL bits 5 to 4) set to '11' b, 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.
- g) 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 '11' b, it shall begin Sequence retransmission with the following restrictions:
 - A) 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.
 - B) the SEQ_ID shall match the SEQ_ID of an ACK frame with the Abort Sequence Condition bits set to '11' b.
 - C) the SEQ_ID being retransmitted is the first non-deliverable Sequence as determined by the Sequence Initiator,
 - D) 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.
 - E) 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.
 - F) 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.
 - G) 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.
 - H) 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 one, the Abort Sequence Condition bits set to '01' b, requesting ABTS, and the Sequence being retransmitted shall be non-deliverable.

- l) after the Sequence Initiator determines that the retransmitted Sequence is successfully delivered (ACK EOF_t), it shall transmit subsequent Sequences with F_CTL bit 9 set to zero.

21.7.2.4 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.

21.7.2.5 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 EOF_t 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_ID, 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.

21.7.3 Stop Sequence Protocol

The Stop Sequence Protocol shall be used by a Sequence Recipient 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 '10' 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 Sequence timeout protocols before transmitting the ACK to the frame with the End_Sequence bit set and shall recover Credit in the normal manner.

NOTE 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.

NOTE 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 is not 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 that includes the count of the number of bytes in the Sequence that were stored before the ULP buffer was exhausted.

21.7.4 End-to-end Credit loss

This standard does not define the method to be employed for Credit allocation to a destination Nx_Port. If destination Nx_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 Classes 1, 2,

4 and 6.

21.8 Link Error Status Block

The errors shown in table 256 are accumulated over time within an Nx_Port. The format shown is the format in which the LESB information shall be supplied in response to an RLS ELS. It does not require any specific hardware or software implementation. The errors accumulated provide a coarse measure of the integrity of the link over time. No means is provided to reset the LESB. It may to overflow and thus reset. The counts shall be 32 bit values.

Table 256 - Link Error Status Block format for RLS command

Bits Word	31 .. 00
0	Link Failure Count
1	Loss-of-Synchronization Count
2	Loss-of-Signal Count
3	Primitive Sequence Protocol Error
4	Invalid Transmission Word
5	Invalid CRC Count

NOTE 1 Informative guideline to manage LESB is provided in Annex E.

An Nx_Port may choose to log these events as well as other errors that occur on an Nx_Port specific basis, which is not defined in this standard.

NOTE 2 It is recommended that Fx_Ports also maintain an LESB and accumulate error events in a manner, which is not defined in this standard.

21.9 Detailed Error Detection/Actions

21.9.1 Errors detected

Table 257 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 that may be detected. It is not an exhaustive list. Column 1 of table 257 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 Classes 1, 2, 4 and 6). Column 4 identifies the action that the Sequence Recipient takes on Data frame errors detected for the Sequences being received or link integrity errors.

21.9.2 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.

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.

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

2) Transmit P_RJT frame

If a valid Data frame is received that contains information in the Frame_Header that is invalid or incorrect, a P_RJT frame shall be transmitted with the appropriate reason code as specified in 11.2.3.4. 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 or Class 6 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.

3) 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 (e.g. 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 that contains information in the Frame_Header that is invalid or incorrect, the frame shall be discarded.

Table 257 - 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 missing R_RDYs (buffer-to buffer credit = 0)	6	6
		6	6
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 (SOF _{c1} while connected)	2	2
	Abnormal frame termination	1	1
	EOF _a received	1	1
	Incorrect SOF or EOF (see tables 42 and 44)	1	1
Address Identifier errors	incorrect D_ID	2	2
	incorrect S_ID	2	2

Table 257 - Detailed errors and actions (Continued)

Error Category	Specific Error	Seq Init Action	Seq Recp Action
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
	Relative offset out of bounds	N/A	2
Data frame errors	buffer not available - Class 1	N/A	2,7
	buffer not available - Classes 2 and 4	N/A	4
	buffer not available - Class 3	N/A	1
	ABTS frame received	N/A	8
	missing frame error detection	N/A	13,7
ACK_1 frame errors	ABTS frame received	8,10	N/A
	missing frame error detection	13,8	N/A

4) Transmit P_BSY frame

An Nx_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 an Nx_Port receives a Class 1 and 6 connect-request frame and its internal link facilities are busy or unavailable, the Nx_Port responds by transmitting a P_BSY frame with the appropriate reason code (see 11.2.3.4.3) with EOFdt. 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 or 6 connect-request regardless of the disposition of the received frame.

5) Process Busy

When an F_BSY or P_BSY is received in response to a Class 1 or 6 connect-request, the Nx_Port may:

- attempt a connect-request to a different destination Nx_Port if such a request is pending,
- delay a period of time before reissuing the connect-request, or

- c) immediately reissue the connect-request.

The Fabric shall not queue a connect-request, if the destination Nx_Port has responded with a P_BSY to the connect-request.

The decision as to which action to take is dictated based on the conditions in the Nx_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 Nx_Port has the capability to retransmit, the Nx_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 an Nx_Port may choose to count the number of retries and decide to shutdown communication with a specific Nx_Port. When an F_BSY is received in response to an ACK frame (Class 2), the Nx_Port shall not retransmit the ACK frame.

6) Perform Link Reset Protocol

When an Nx_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 and 6, if an Nx_Port has timed out all open Sequences (E_D_TOV), a Link timeout is detected. When a Link timeout is detected, the Nx_Port or Fx_Port begins the Link Reset Protocol.

In addition, in Class 1 and 6 an Nx_Port may not know whether a dedicated connection exists or not. An Nx_Port may initiate the Link Reset Protocol in order to reach a known state.

7) 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 to 4.

'00' b = Continue Sequence

'01' b = Abort, perform ABTS

'10' b = Stop Sequence

'11' b = 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.

8) Perform Abort Sequence Protocol

When a Sequence Initiator detects a Sequence error or receives an appropriate Abort Sequence Condition ('01' b) in F_CTL bits 5 to 4 in an ACK for an active 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 21.7.2).

When a Sequence Recipient receives an ABTS frame, it shall respond as specified in 21.7.2.2 and 12.2.2.

9) 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 or 6 dedicated connection is removed by an EOF delimiter or a Primitive Sequence, the Nx_Port shall abnormally terminate any active Sequences (see 21.7.2).

10) 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 or 6 (see 21.7.2.3). For other Exchange Error Policies, each individual FC-4 shall specify the procedure for Sequence retransmission, if any.

11) Update LESB

The Link Error Status Block is updated to track errors not directly related to an Exchange.

12) Perform Link Failure Protocol

Transmission or reception of the Not Operational Primitive Sequence initiates the Link Failure Protocol.

13) Error Policy processing

When an error is detected within a Sequence, the Sequence is either processed normally (process policy), or discarded (discard policy). See 21.6.3 for additional information.

14) Read Status Block

Read Status Block (either Sequence or Exchange) is accomplished by originating an Exchange with the appropriate ELS request.

22 Hunt Group

22.1 Introduction

A Hunt Group is a collection of one or more Nx_Ports controlled by a Common Controlling Entity. A Hunt Group is addressed using a Hunt Group Identifier (HG_ID) that is managed as an Alias_ID. The Hunt Group shall be formed by the registration of a group of one or more Nx_Ports with the Alias Server. An Nx_Port may be a member of multiple Hunt Groups and may be addressed by either its base address identifier or any one of its HG_IDs. Other methods may be used to select one of N destination Nx_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. An Nx_Port using its base address identifier or HG_ID (Alias_ID) 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 Nx_Port and the Fabric.

The Alias Server handles the registration of Nx_Ports into a Hunt Group, and the deregistration of Nx_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 22.7.

Hunt Group operations are applicable to Classes 1, 2, 3, 4 and 6.

Hunt Group is an optional communication model. To support this model, Fabrics and Nx_Ports shall have the additional capability specified.

22.2 Function

The function of a Hunt Group is to provide a HG_ID (Alias_ID) that shall be used by the Fabric to select a single available Nx_Port out of the group of possible Nx_Ports as the destination (D_ID) of a frame. This increases the likelihood of locating a destination Nx_Port that is not engaged in a Class 1 or 6 connection. In Class 2, Hunt Groups may also be used to increase aggregate bandwidth, reduce latency, or both.

22.3 Communication Model

The Hunt Group communication model is an extension of Nx_Port to Nx_Port communication. Since a Hunt Group is a collection of Nx_Ports, the following combinations are supported in this model:

- a) Nx_Port to Nx_Port
- b) Nx_Port to Hunt Group
- c) Hunt Group to Hunt Group

All combinations are supported as part of the adaptive addressing protocol, specified in 22.7.

22.4 Applicability

22.4.1 Classes

Class 1, 2, 3, 4 or 6 may be used with Hunt Group. Common Controlling Entity based or Port based allocation of resources required shall vary by class (see 22.8).

22.4.2 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 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 (e.g., an N_Port_ID K1 in Class 4 implies specifically, for example , K1(5) where 5 is the Virtual Circuit ID. A HG_ID K in Class 4 implies specifically, for example , K1(5), K2(9), and K3(25)).

22.5 Formation

22.5.1 General

A Hunt Group is formed by an Nx_Port under control of the Common Controlling Entity performing Hunt Group Registration with the Alias Server. The Nx_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 Common Controlling Entity shall be responsible for allowing any of its Nx_Ports becoming a Hunt Group member and communicating common service parameters for such membership. The method by which the Common Controlling Entity informs its Nx_Ports is not defined by this standard.

During registration, the Alias Server shall inform each member Nx_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:

- a) Alias_ID assigned
- b) Assigned Alias_ID is associated with Hunt Group
- c) Node_Name associated with the list of Nx_Ports in the Hunt Group
- d) List of Nx_Ports in the Hunt Group.

The Fabric Controller is responsible for distribution of Alias_IDs for HG_ID and Fabric Nx_Port Service Parameters to the appropriate internal facilities and Fx_Ports within the Fabric. The method for distributing this information within the Fabric is not defined by this standard.

22.5.2 Registration/deregistration

Registration of a list of Nx_Ports into or deregistration of a list of Nx_Ports from a Hunt Group by a Common Controlling Entity is performed by originating an Exchange with the Alias Server. Any Nx_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 Nx_Port. The third party authorization for registration or deregistration is the responsibility of the Common Controlling Entity which is not defined by this standard.

The Exchange may be used to:

- a) create a Hunt Group with a list of member Nx_Ports,
- b) add a list of Nx_Ports to an existing Hunt Group, or
- c) delete a list of Nx_Ports of an existing Hunt Group.

HG_ID is managed as an Alias_ID. The detailed format and protocol for registration and deregistration are specified in clause 24.

The information transmitted by the Originator contains the following items:

- a) Node_Name: The Node_Name of the Common Controlling Entity.
- b) HG_ID: The HG_ID 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 (zero) or a value requested by the Common Controlling Entity.

- c) Nx_Ports: During creation of the Hunt Group, all member Nx_Ports are identified. During add or delete functions, member Nx_Ports being added or deleted are identified.
- d) Common and Class Service Parameters: During the creation of a Hunt Group, the Fabric Nx_Port Service Parameters shared by each member Nx_Port for all classes shall be included in the same format and meaning as during the N_Port to Fabric Login procedure (see clause 15).

22.5.3 Inquiry

The Hunt Group information may be available from the Directory Server, if Alias Server has communicated the information to the Directory Server (well-known address hex 'FF FF FC').

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.

22.5.4 HG_ID Removal

Removal of a Hunt Group by a Common Controlling Entity is performed by originating an Exchange with the Alias Server. Any Nx_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 Nx_Port. The details of the Exchange are specified in the in clause 24.

22.6 N_Port Login

N_Port Login shall be required on an address identifier basis. Therefore, Login shall be performed with the HG_ID prior to communication other than Basic and Extended Link Applications. Login with any instance of the Alias_ID of HG_ID completes Login with all members of the Hunt Group (e.g., 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). N_Port Login shall have been completed between the S_ID and D_ID N_Port_IDs prior to communication of Upper Level Protocols for the first Data frame of the Exchange.

Each member Nx_Port of a specific Hunt Group shall operate with a single, identical set of Common and Class Service Parameters with the Fabric and destination Nx_Ports with which it communicates.

22.7 Addressing Protocol

22.7.1 address identifier Protocol

An adaptive address identifier protocol is utilized when 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 22.3).

22.7.2 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.

22.7.3 Responder

In the ACK to that first Data frame, the Responder shall address the Originator with the same HG_ID that the Originator identified itself in the first Data frame. However, the Responder may choose to use either its N_Port_ID or the HG_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 of "Protocol error".

22.7.4 Address Resolution

After the first ACK has been specified, the identifiers for both Originator and Responder shall remain the same. The address resolution for different cases is summarized in the table 258.

Table 258 - Address Resolution

Originator First Data frame S_ID, D_ID	Responder ACK to First Data frame S_ID, D_ID	Resolution Originator ID, Responder ID
ON_ID, RN_ID	RN_ID, ON_ID	ON_ID, RN_ID
	RL_ID, ON_ID	ON_ID, RL_ID
OK_ID, RN_ID	RN_ID, OK_ID	OK_ID, RN_ID
	RL_ID, OK_ID	RL_ID, ON_ID
ON_ID, RL_ID	RN_ID, ON_ID	RN_ID, ON_ID
	RL_ID, ON_ID	RL_ID, ON_ID
OK_ID, RL_ID	RN_ID, ON_ID	RN_ID, ON_ID
	RL_ID, ON_ID	RL_ID, ON_ID
Legend: OK_ID - Originator N_Port_ID OK_ID - Originator Hunt Group Alias ID RN_ID - Responder N_Port_ID RL_ID - Responder Hunt Group Alias ID		

22.7.5 Class

The address resolution is applicable to Class 1, 2, 4 and 6, 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.

22.7.6 Class 4

The address resolution summarized in table 258 shall be valid for Class 4 also, with the Class 4 address implication specified in 22.4.2.

22.8 Resource management

22.8.1 Introduction

Since a Hunt Group may span multiple Nx_Ports, resources may be managed by the Common Controlling Entity and by Nx_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.

An Nx_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 Nx_Port. An Nx_Port may use any or all of the control entities for a given implementation.

When an Nx_Port supports multiple address identifiers or common address identifiers for S_ID, D_ID, or both, Sequence and frame identification alters Nx_Port behavior. The manner in which HG_ID are utilized within an Nx_Port leads to various logical models of Nx_Port resources.

Based on the organization of resources within an Nx_Port and Common Controlling Entity and the Class of Service, the Originator or Responder shall choose HG_ID or N_Port_ID for an Exchange.

22.8.2 HG_ID use

For an Originator or a Responder to use a HG_ID, the following resource requirements shall be met:

- a) Class 1 and 6
 - A) Exchange resources (ESB, OX_ID) shall be Common Control Entity managed
 - B) Sequence resources (SSB) may be Nx_Port managed or Common Controlling Entity managed
- b) Class 2
 - A) Exchange resources (ESB, OX_ID) shall be Common Controlling Entity managed
 - B) Sequence resources (SSB) shall be Common Controlling Entity managed
- c) Class 3
 - A) Exchange resources (ESB, OX_ID) shall be Common Controlling Entity managed
 - B) Sequence resources (SSB) shall be Common Controlling Entity managed
- d) Class 4
 - A) Exchange resources (ESB, OX_ID) shall be Common Controlling Entity managed
 - B) Sequence resources (SSB) shall be Common Controlling Entity managed

22.8.3 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 Common Controlling Entity managed or Nx_Port managed or shared between them. If shared, the SSB, ESB, X_ID control hierarchy shall be retained.

22.9 Rotary Group

Rotary Group may be used 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 Nx_Ports or Fx_Ports.

A Rotary Group is a collection of one or more Nx_Ports controlled by a Common Controlling Entity and a cooperative Fabric (see [7], FC-SW-3) in which any member of the Rotary Group may respond in a transparent way to non-participating Nx_Ports.

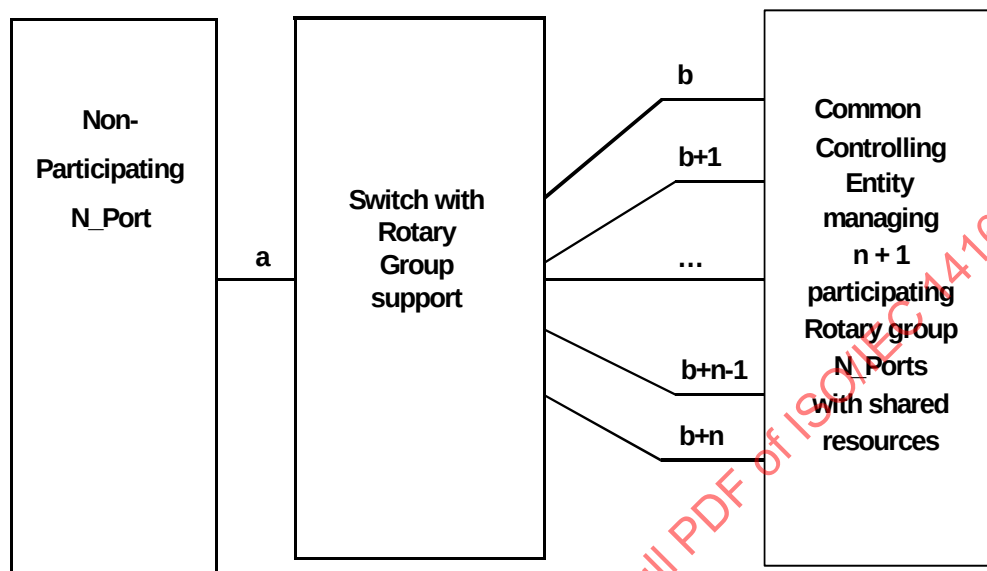
NOTE In this way, systems that handle more bandwidth than the attached Nx_Ports provides, may utilize multiple Nx_Port to Fabric connections to extend their achievable bandwidth.

Rotary Group are formed and managed on a local basis, both in the participating Nx_Ports host (Common Controlling Entity) and in the Fabric, by the System administration.

The overall objective of a Rotary Group is that other Nx_Ports or Directory Services interact with the Group in a completely transparent manner (i.e., Nx_Ports that are not members of the Group do not know or care about the existence or non existence of the Rotary Group). Non-participating Nx_Ports address any Nx_Port of the Rotary group by its native address. No matter which Nx_Port of the Rotary Group responds, the Non-Participating Nx_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 37).

In Class 1 and 6, 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 [7], FC-SW-3).

A Rotary Group member Nx_Port, under direction of its Common Controlling Entity, returns the address with which it was addressed as its source address (no matter which Rotary Nx_Port is actually comes from) in instances where it is the exchange responder. The Common Controlling Entity may use any available Nx_Port of its Rotary Group that is logged in to a destination Nx_Port for those exchanges where it is the exchange Originator. The Common Controlling Entity manages the resources (i.e., buffers, state tables) as a pool for all of the participating Rotary Group Nx_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 37 - Relationship of a Non-Participating to a Participating Nx_Port and switch

23 Multicast

23.1 Applicability

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 Nx_Port to many destination Nx_Ports.

Multicast operations are not applicable to Classes 1, 2 and 4.

23.2 Class 3 Multicast

23.2.1 Introduction

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 Nx_Ports to which a frame being multicast is delivered. A Multicast Group is managed by the Alias Server (hex 'FF FF F8') that handles the registration of Nx_Ports into a Multicast Group, and the de-registration of Nx_Ports from a Multicast Group. The Alias Server is not involved in the routing of multicast frames.

23.2.2 Registration and De-registration

The registration/de-registration by an Nx_Port as a member of a Multicast Group with the Alias Server (hex 'FFFFF8') is specified in clause 24. An Nx_Port may register itself as a member of one or more Multicast Groups. An Nx_Port may register a list of Nx_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.

23.2.3 Multicast Routing

The Fabric, based on recognition that the D_ID of the transmitted frame is an MG_ID Alias_ID, does all routing of multicast frames. The exact frame that entered the Fabric is replicated to every destination Nx_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 (e.g., 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_ID and accept the frame.

The Fabric shall reject Class 1, Class 2, and Class 4 frames with a D_ID equal to a MG_ID.

Figure 38 illustrates the multicast routing. Nx_Port B, C, D, and E are members of a Multicast Group. Nx_Port A is the message sender that may not be a member of the group. If it is a member, it may optionally receive the same message (see [6], FC-GS-4).

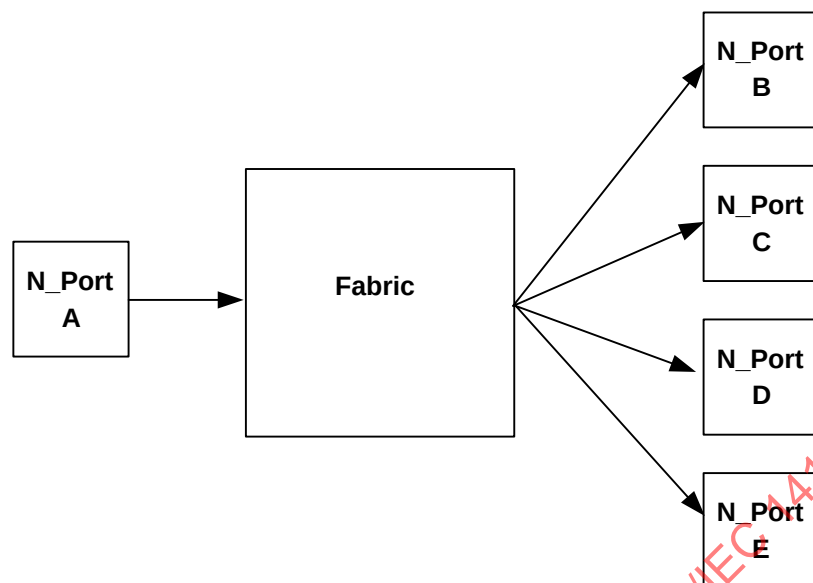


Figure 38 - Class 3 Multicast Routing

23.2.4 Class 3 Multicast Rules

- The Sequence Initiator of the Multicast message shall follow the uniqueness rules for OX_ID, SEQ_ID, and SEQ_CNT. RX_ID for the Sequence shall be set to hex 'FF FF'.
- D_ID in all frames shall be the Alias_ID for the Multicast Group. S_ID in all frames shall be the N_Port_ID 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.

23.3 Class 6 Multicast

23.3.1 Introduction

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 9. 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 not defined by this standard. 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 from participating N_Ports and generating a single response back to the multicast Source.

Class 6 frames use SOF/EOF delimiters identical to Class 1 frames. Normal connection setup and removal follows the rules for Class 1 connections specified in clause 20. Similarly, Class 6 connect request frames follow the rules for buffer-to-buffer flow control specified 18.5.

NOTE A system implementation may provide less than reliable multicast service by implementing a single Sequence with infinite credit.

23.3.2 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 39 illustrates the multicast routing. N_Ports B, C, D and E are members of a Multicast group. N_Port A is the multicast Source.

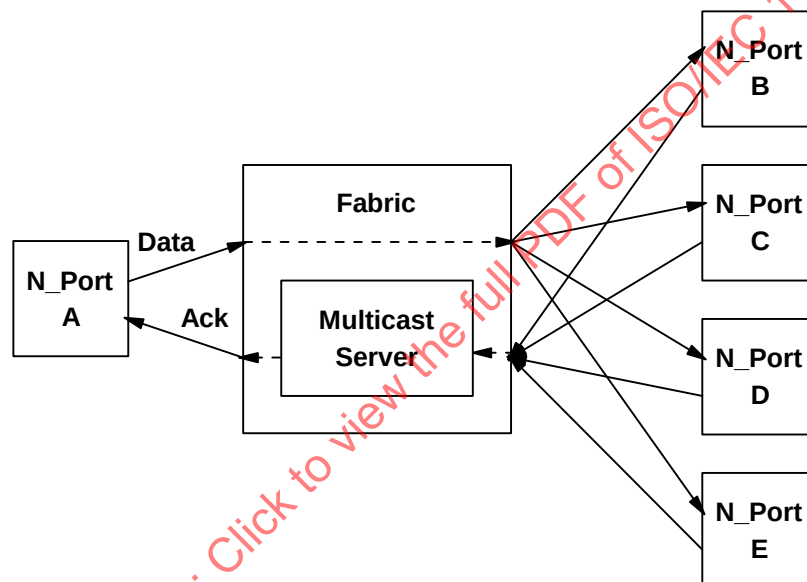


Figure 39 - Class 6 Multicast Routing

23.3.3 Class 6 Multicast Rules

The following are the Class 6 multicast rules:

- The multicast Source shall follow the uniqueness rules for OX_ID, SEQ_ID, and SEQ_CNT. X_ID interlock is not supported for Class 6 Exchanges. The multicast Source shall be the Connection Initiator for all multicast sequences. RX_ID is not used in Class 6 and shall be set to hex 'FF FF' for all Class 6 frames.
- 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_ID of the multicast Source and may also be an Alias_ID.
- 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.
- All Class 6 multicast frames shall be routed as uni-directional dedicated connections.
- 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.
- 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.

23.3.4 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 (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 51) 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 54) 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 EOF_{dt} delimiter and a "Multicast error terminate" reason code (see table 54), if one or more destinations respond with an ACK, BSY, or RJT frame with an EOF_{dt} 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 the Link Reset protocol is initiated, the Class 6 Connection is assumed to be terminated and the Multicast Server is no longer a participant.

Once a Class 6 connection is established, end-to-end credit is managed between the Connection Initiator and the multicast server.

23.3.5 Class 6 Multicast Recovery

If the multicast Source receives from the Multicast Server a busy or a reject with an EOF_{dt} 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 vendor-unique.

23.4 Broadcast

A frame addressed to the Well-known address for Broadcast (i.e., hex 'FF-FF-FF') is a Broadcast frame. The Fabric shall attempt to send the Broadcast frame to all possible Nx_Ports within zoning constraints. However, the Fabric may not be able to deliver to all Nx_Ports for any number of reasons (e.g., class mismatch or Nx_Port not operational).

Broadcast is considered a simplification of Class 3 multicast. No explicit registration or de-registration is required to send or receive Broadcast frames. An Nx_Port may discard a Broadcast frame.

An Nx_Port shall send and receive Class 3 Broadcast Data frames in the context of an implicit Broadcast Port Login. The implicit Broadcast Port Login is particular because it is not tied to any remote N_Port_Name and Node_Name, but it is tied to the destination address identifier hex 'FF FF FF'.

The implicit Broadcast Port Login specifies the service parameters to be used for broadcast communications. An FC-4 using the Broadcast functionality may specify the service parameters that it requires in the implicit Broadcast

Port Login. In absence of such a specification, the default login parameters specified in 15.2 shall be used.

23.5 Other

Other forms of multicast are available in topology specific configurations. For examples, See [2], FC-AL-2, for a description of Selective Replicate to perform dynamic multicasting.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

24 Alias_IDs

24.1 Introduction

Alias_ID is an address recognized by one or more Nx_Ports or the Alias Server, if the Nx_Port has registered with the Alias Server (hex 'FF FF F8') as a member of a group. Thus an Nx_Port may be able to recognize one or more Alias_IDs in addition to its N_Port_ID.

Presently defined Alias_IDs are:

- a) HG_ID, and
- b) Multicast Group Identifier.

A given Nx_Port may register with one or more Hunt Groups and/or with one or more Multicast Groups.

See clause 22 for Hunt Group and clause 23 for Multicast operations for the applicable classes of service.

NOTE Well-known addresses specified in table 21 are allocated for special functions (e.g. Directory Server) These special addresses are not to be confused with Alias_IDs.

24.2 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 (hex 'FF FF F8') may be internal or external to the Fabric. The following subclauses describe the registration/deregistration process in more detail.'

24.3 Alias Service protocol

Alias_ID registration and deregistration are managed through protocols containing a set of request/reply Information Units supported by the Alias Server. These requests and replies use constructs as defined in FC-GS-3.

24.4 Alias_ID 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 Nx_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 Nx_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_ID Group Service Parameters, rather than the Login Service Parameters (e.g., 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_ID Group, it shall recognize the Alias_ID as an Alias_ID 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.

24.5 Function Flow

Figure 40 illustrates the flow among the Originator Nx_Port, participating Nx_Ports, Alias Server, Directory Server, and Fabric Controller to create an Alias Group.

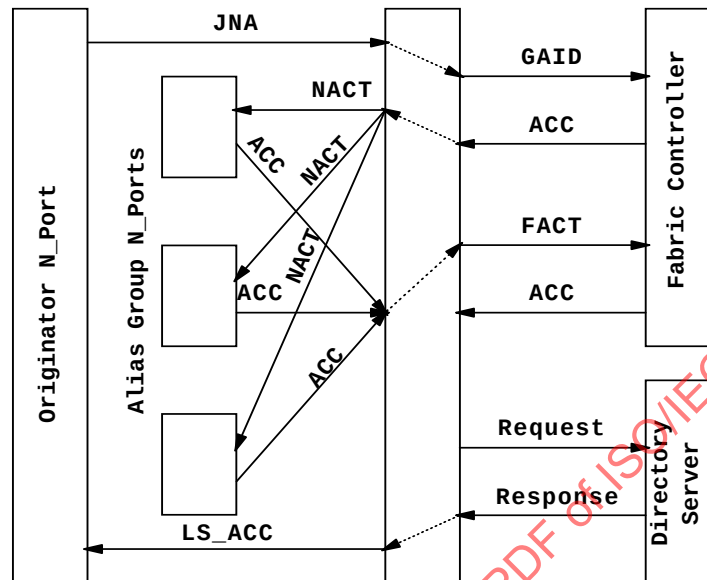


Figure 40 - Function Flow

24.6 PA Considerations

24.6.1 Hunt Groups

For Hunt Groups, there are no Initial Process Associator considerations, since there is a requirement that all members of a Hunt Group be within a single Common Controlling Entity that shall not be spread across images. Therefore, the same IPA may be used no matter which Nx_Port receives the frame.

24.6.2 Multicast Groups

For Multicast Groups, the considerations for multicasting to Nx_Ports that require an Initial Process Associator are minimal as any multicasting behind the Nx_Port is handled internally. It is not necessary to know the Initial Process Associator of each image behind an Nx_Port that belongs to a Multicast Group. Instead, if the Multicast Group requires an IPA, then the Sequence Initiator uses a Multicast Group Initial Process Associator (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 one. This indicates an MG_IPA.

Internally, images shall register with their Nx_Ports to join Multicast Groups, they do not register with the Alias Server (hex 'FF FF F8'). The MG_IPA becomes an Alias_ID for the actual Initial Process Associator of the image and is recognized as such by the Nx_Port as a result of the internal registration. When a frame is received, the Nx_Port multicasts the Payload to all images in the internal Multicast Group, based on the MG_IPA.

24.6.3 Broadcast

For Broadcast, there are no Initial Process Associator considerations. Since the intent of Broadcast is to deliver to all possible recipients, it is the responsibility of the Nx_Port receiving a broadcast frame to broadcast the Payload internally to all its images. Therefore, an Initial Process Associator shall not be included in a frame being Broadcast.

STANDARDSISO.COM : Click to view the full PDF of ISO/IEC 14165-251:2008

25 Class 4 – Fractional

25.1 Introduction

Class 4 Service is a connection oriented service for use with a Fabric (see 13.6). 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. An 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 figure 41. 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.

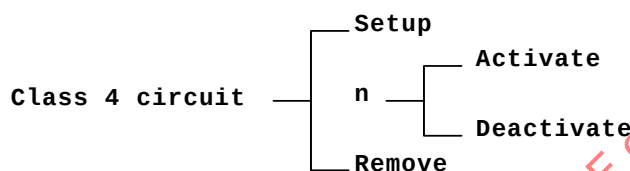


Figure 41 - Class 4 Circuit hierarchy

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 may 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.

Class 4 is applicable only to topologies where single N_Ports are attached to F_Ports.

25.2 Default N_Port Login service parameters

Prior to Login or following Logout, the following default set of Service Parameters shall apply:

- a) Class validity = 1 (Class supported)
- b) ACK_0 capable = 0 (ACK_0 incapable)
- c) ACK generation assistance = 0 (no ACK generation assistance)
- d) X_ID interlock = 1 (interlock required)
- e) Error Policy supported = 00 (discard policy only)
- f) Categories per Sequence = 1
- g) Receive Data Field size = 256
- h) Concurrent Sequences = 1
- i) Open Sequences per Exchange = 1
- j) Total Concurrent Sequences = 1
- k) End-to-end Credit = 1
- l) Relative offset not used
- m) E_D_TOV Resolution = 0
- n) BB_SC_N = 0
- o) Other optionally supported features shall not be used or required.

25.3 Communication Model

25.3.1 Introduction

Class 4 service provides a guaranteed bandwidth between communicating N_Ports. The ELSs associated with Class 4 sets up a pair of unidirectional VCs between communicating N_Ports (see figure 42) to guarantee bandwidth and frame transmission delay.

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 may 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 43). The amount of bandwidth needed for each VC is requested when the Class 4 circuit is setup.

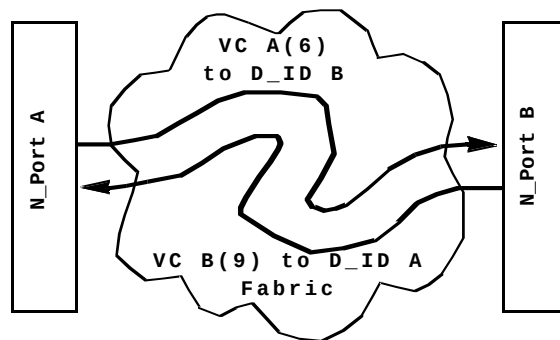


Figure 42 - Basic Class 4 circuit

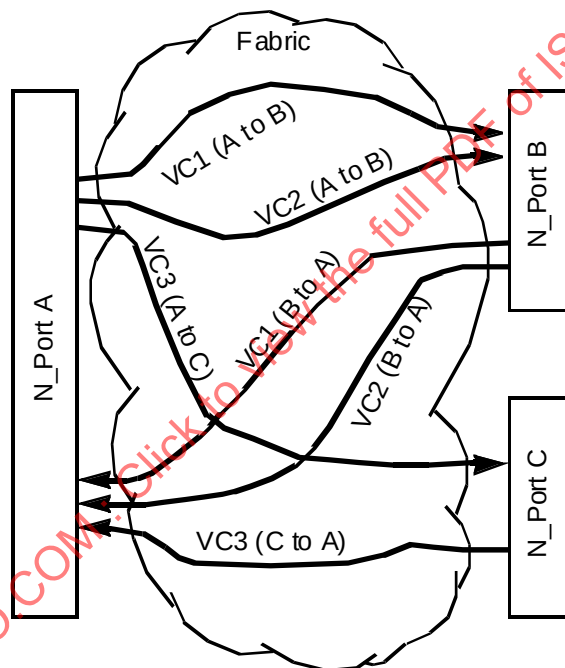


Figure 43 - Class 4 circuits – Example

25.3.2 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.

25.3.3 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.

25.3.4 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 43).

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 an N_Port does not transmit frames on a VC in excess of the requested maximum bandwidth limit.

25.3.5 Class 4 circuit

25.3.5.1 Introduction

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 42 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 25.3.5.2).

Table 259 illustrates how a Class 4 circuit should be setup between two N_Ports 'A' and 'B'. N_Port 'A' supports a data rate of 100 MBytes/sec, and 'B' a data rate of 25 MBytes/s. The VC from 'A' to 'B' (VC(1)) has a guaranteed minimum bandwidth of 15 MBytes/s and a maximum latency of 500 μ s. The VC from 'B' to 'A' (VC(6)) has a guaranteed minimum bandwidth of 3 MBytes/s and a maximum latency of 2 000 μ s. N_Port 'A' is left with an excess outbound bandwidth of 85 MBytes/s and an excess inbound bandwidth of 97 MBytes/s. The excess bandwidth for N_Port 'B' is 22 MBytes/s in the outbound direction and 10 MBytes/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.

Class 4 circuits and their VCs exist in one of four states:

- a) Nonexisting,
- b) Pending,
- c) Dormant, or
- d) Live.

A Ports State Diagram for a Class 4 circuit is shown in figure 44. The states and state transitions are described in the sub-clauses that follow.

Ports that support Class 4 shall, regardless of the state of Class 4 circuits, support interleaving of Class 2 and Class 3 in the Class 4 traffic frames based on the rules of those two classes of service.

Table 259 - Bandwidth Allocation Example

Class 4 Connection		N_Port		Rate (MB/s)	Max. Delay (μs)
		ID	VC_ID		
A ⇒ B		A	(1)	15	500
A ⇐ B		B	(6)	3	2 000
Free	A ⇒	A	N/A	85	N/A
	A ⇐		N/A	97	
	B ⇒	B	N/A	22	N/A
	B ⇐		N/A	10	

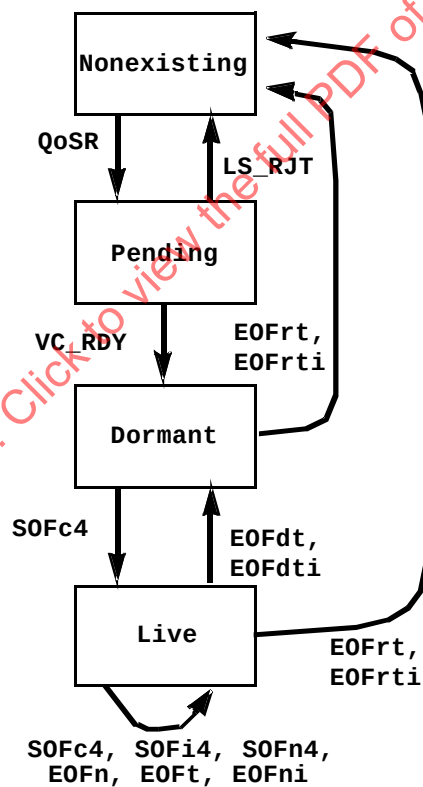


Figure 44 - Class 4 circuit - Port level state diagram

25.3.5.2 Setup

An N_Port requests setup of a Class 4 circuit to another N_Port by sending a Quality of Service request (QoS SR) ELS command to the QoSF. The requesting N_Port becomes the circuit initiator (CTI) and the target N_Port the circuit recipient (CTR). The QoS SR identifies the communicating N_Ports and the required QoS parameters.

The QoS SR is granted if the QoS F establishes a path between the communicating N_Ports with the requested QoS parameters and if the CTR also accepts the QoS parameters (see 12.3.2.38). If the QoS SR is accepted, then a Class 4 circuit is established in the Pending State when the LS_ACC Sequence is returned to the CTI (see figure 45(a)).

The QoS SR 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 (see figure 45(b)).

The QoS SR is rejected if the QoS F establishes a path between the communicating N_Ports with the requested QoS parameters, but the CTR is unable to accept the QoS SR (see figure 45(c)).

The CTI selects the QoS parameters for both VCs in the Class 4 circuit.

The QoS F 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 QoS F 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 QoS F shall reject the request (see figure 45(b)).

General conditions associated with a Class 4 circuit are:

- a) An 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.
- b) The bandwidth of any Class 4 circuit is restricted to the maximum bandwidth of the slowest link in the Class 4 circuit path.
- c) 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.
- d) The Quality of Service Request (QoS SR) ELS 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:

- a) The CTI shall set the VC_Credit_Limit, VC_Credit, EE_C4_Credit_Limit, and EE_C4_Credit to zero before it sends the QoS SR to the QoS F. The CTI shall enter the Pending State for the VC as it sends the QoS SR to the QoS F.
- b) The CTI shall release all resources assigned to the Class 4 circuit if an LS_RJT is received from the QoS F.

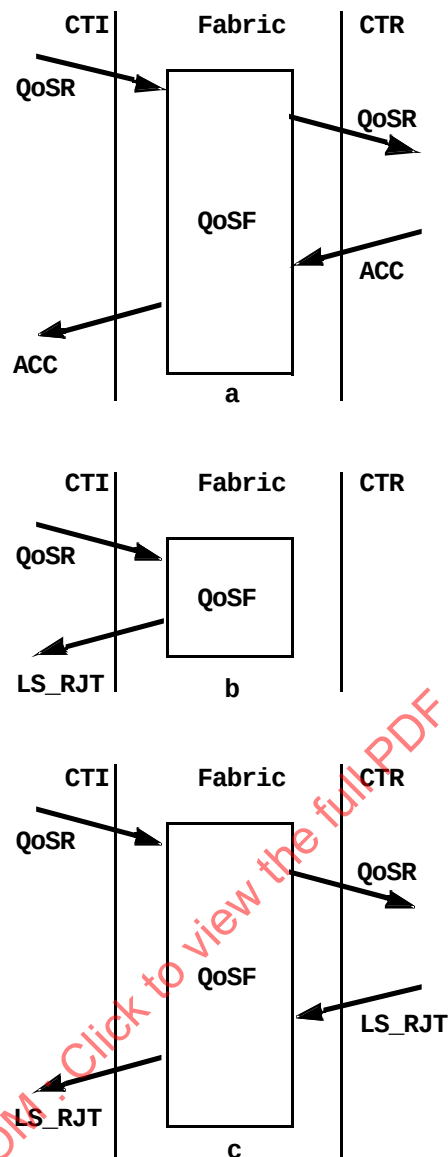


Figure 45 - Class 4 circuit – Setup

- c) The CTI shall, when it recognizes the LS_ACC Sequence to the QoS SR, set the Live_VC_Credit_Limit to the value obtained from the LS_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 LS_ACC Sequence. CTI VC_Credit remains at zero until a VC_RDY is received on the new Class 4 circuit.
- d) The CTI shall use the address identifier in the CTR address identifier field in the LS_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_ID or native, assigned to the CTR on the Class 4 circuit.

- e) 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:

- a) The CTR shall enter the Pending State before it transmits a LS_ACC Sequence to the QoS from the QoSF. Before the CTR transmits a LS_ACC Sequence the CTR shall:
 - A) set the VC_Credit_Limit to zero;
 - B) set the Live_VC_Credit_Limit to the value obtained from the QoS Sequence;
 - C) set the EE_C4_Credit_Limit to the value obtained from the QoS Sequence;
 - D) set the EE_C4_Credit to the value obtained from the QoS Sequence; and
 - E) set VC_Credit to zero.
- b) If the CTR is unable to accept the QoS from the QoSF it shall return an LS_RJT to the QoSF and enter the Nonexisting State.
- c) 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.
The CTI may elect to use any address identifier, Alias_ID or native, assigned to the CTI on the Class 4 circuit.
- d) 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 QoSF actions are:

- a) The QoSF shall set the F_Port, linked to the CTI, to the Pending State, once it recognizes the QoS from the CTI.
- b) The QoSF shall set the F_Port, linked to the CTI, to the Dormant State when it completes the QoS transaction with the CTI. The QoS transaction is complete in Class 1,2 or 4 when the QoSF receives the ACK to the LS_ACC Sequence, sent to the CTI. In Class 3 the QoS transaction is complete when the QoSF transmits the LS_ACC Sequence. The F_Port shall transmit one or more VC_RDY when the F_Port enters the Dormant State.
- c) The QoSF shall set the F_Port, linked to the CTR, to the Pending State, as it sends the QoS to the CTR.
- d) The QoSF shall set the F_Port, linked to the CTR, to the Dormant State when it completes the QoS transaction with the CTI. The QoS transaction is complete in Class 1,2 or 4 when the QoSF transmits the ACK to the LS_ACC Sequence, received from the CTR. In Class 3 the QoS transaction is complete when the LS_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.
- e) The QoSF shall send the CTI an LS_RJT if an LS_RJT is received from the CTR.
- f) The QoSF shall set the F_Port, linked to the CTR, to the Nonexisting State, once it recognizes the LS_RJT from the CTR.
- g) The QoSF shall send the CTI an LS_RJT if it is unable to accept the CTI's QoS request.
- h) The QoSF shall set the F_Port, linked to the CTI, to the Nonexisting State, before it sends an LS_RJT to the CTI.
- i) The QoSF shall send LS_ACC to the CTI only if a LS_ACC has been received from the CTR.
- j) The QoSF shall remove the Class 4 circuit (see 25.3.5.5) if the QoS transaction fails to complete.

The Class 4 circuit may be activated once it is setup (see 25.3.5.2).

25.3.5.3 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 46).

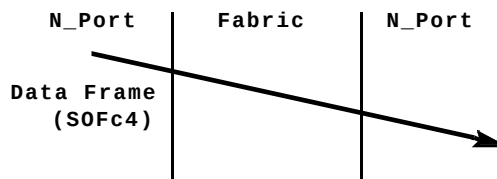


Figure 46 - Class 4 circuit – Activation

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 47).

Class 4 end-to-end flow control is managed by the communicating N_Ports using ACK frames.

25.3.5.4 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 25.4). The ACK response then carries the EOF_d delimiter (see figure 48).

The request to deactivate a Class 4 circuit may be sent by either of the communicating N_Ports.

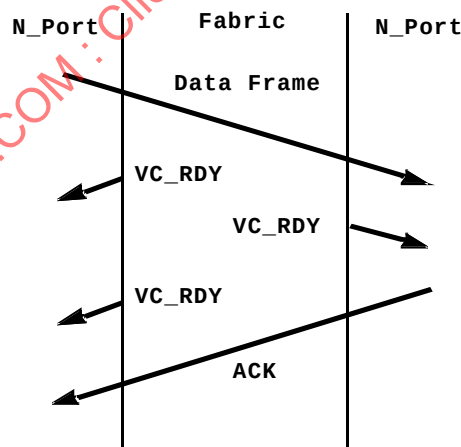


Figure 47 - Class 4 - Example of frame flow

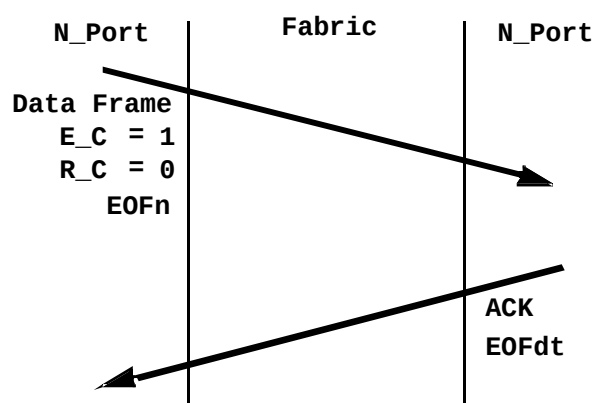


Figure 48 - Class 4 circuit - CTI or CTR initiated

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 49).

A deactivated Class 4 circuit may later be reactivated to the Live State (see 25.3.5.2).

The Fabric shall issue two END Link_Control commands, one on either VC, when the QoS 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 49). 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).

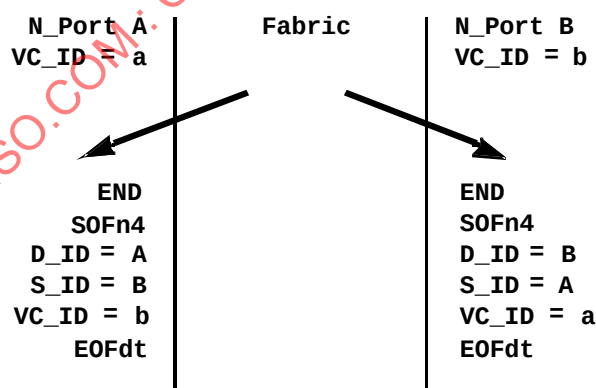


Figure 49 - Class 4 circuit - Fabric initiated deactivation

25.3.5.5 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 25.4). The ACK response then carries EOFrt delimiter (see figure 50).

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 EOF_{rt} or EOF_{rti} delimited frame, is received (see figure 49). A Class 4 circuit, in the Live State, is removed, and enters the Nonexisting State, when a circuit removal request, an EOF_{rt} or EOF_{rti} delimited frame, is sent (see figure 49).

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_F 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.

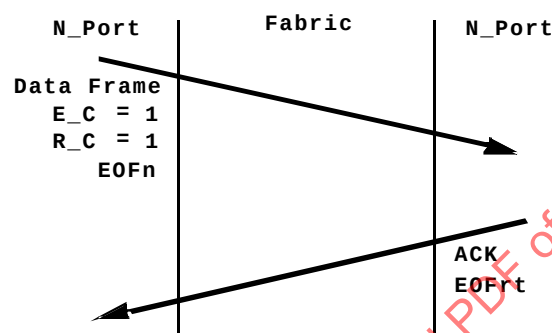


Figure 50 - Class 4 circuit - CTR or CTR initiated removal

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 EOF_{rt} and SOF_{n4} (see figure 51). If the Class 4 circuit is alive, then the Fabric shall first deactivate the Class 4 circuit before removing the circuit (see 25.3.5.4).

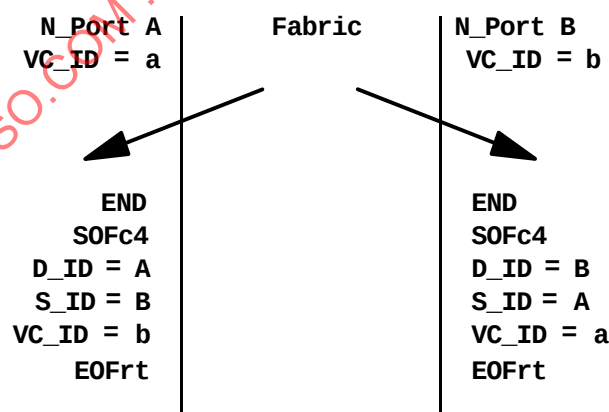


Figure 51 - Class 4 circuit - Fabric initiated removal

The Fabric shall use this mechanism to remove Class 4 circuits when:

- Fabric Rlogin is attempted,
- N_Port or the F_Port enters the Offline State,

- c) Link failure is detected at the N_Port,
- d) error recovery for the setup process (see 25.3.5.2), or
- e) internal Fabric failures of the Class 4 circuit path are detected (i.e., the Fabric is unable to provide the guaranteed quality of service).

An 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).

25.4 Deactivation, removal contention and error recovery

25.4.1 CTI has requested removal of Class 4 circuit

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 EOF_n or EOF_t, 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 (a) above).
- 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.

25.4.2 CTR has requested removal of a Class 4 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:

- a) 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 EOF_n, EOF_t or EOF_{rt}.
- b) Requests to deactivate the Class 4 circuit from the CTI shall be denied (see (a) above).
- c) Requests to remove the Class 4 circuit from the CTI shall be accepted (see (a) above).
- d) Data frame rejection (P_RJT) criteria for the CTR is unaffected by the Class 4 circuit removal request.
- e) 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.
- f) 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.

25.4.3 CTI has requested deactivation of a Class 4 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:

- 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 EOF_n, EOF_t or EOF_{rt}, preventing the CTI to respond to deactivation request from the CTR but allowing the CTI to respond to removal request from the CTR.
- b) Requests to deactivate the Class 4 circuit from the CTR shall be denied (see (a) above).
- c) Requests to remove the Class 4 circuit from the CTR shall be accepted (see (a) above).
- d) Data frame rejection (P_RJT) criteria for the CTI is unaffected by the Class 4 circuit deactivation request.
- e) 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.