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

dpANS X3.288-199x
X3T11/Project 1050D/Rev 3.1

FIBRE CHANNEL

GENERIC SERVICES

(FC-GS)

REV 3.1

working draft proposed
American National Standard
for Information Systems

August 7, 1996

Secretariat:
Computer & Business Equipment Manufacturers Association

ABSTRACT: This standard describes in detail the basic Fibre Channel services introduced in ANSI X3.230, FC-PH. In addition, this document describes any ancillary functions and services required to support the Fibre Channel services.

NOTE:

This is a draft proposed American National Standard of Accredited Standards Committee X3. As such, this is not a completed standard. The X3T11 Technical Committee may modify this document as a result of comments received during public review and its approval as a standard.

POINTS OF CONTACT:

Roger Cummings (X3T11 Chairman)
Distributed Processing Technology
140 Candace Drive
Maitland, Florida 32751
(407) 830-5522 Fax: (407) 260-5366
E-Mail: cummings_roger@dpt.com

I. Dal Allan (FC Working Group Chairman)
ENDL
14426 Black Walnut Court
Saratoga, CA 95070
(408) 867-6630 Fax: (408) 867-2115
E-Mail: dal.allan@mcimail.com

Carl Zeitler (X3T11 Vice Chairman)
IBM Corporation - AWD, MS 9570
11400 Burnet Road, Austin, TX 78758
(512) 838-1797 Fax: (512) 838-3822
E-Mail: zeitler@ausvm6.vnet.ibm.com

Steven L. Wilson (FC-GS Editor)
Amdahl Corporation, MS 119
1250 E. Arques Ave., Sunnyvale, CA 94088-3470
(408) 737-5953 Fax: (408) 746-7000
E-Mail: steve-wilson@ivorytower.amdahl.com

Acknowledgments:

As editor of FC-GS, I would like to acknowledge the following people for their contributions to FC-GS:

R. Bryan Cook (Principal Contributor: Alias Service)
IBM Corporation - P318
522 South Rd.
Poughkeepsie, NY 12601-5400
(914) 435-7914 FAX: (914) 432-9413
Email: bryan_cook@vnet.ibm.com

Khasin Teow (Principal Contributor: Common Transport, Native SNMP Mapping, Time Service)
Brocade Communications Systems, INC.
457 E. Evelyn Ave, Suite E.
Sunnyvale, CA. 94086
(408) 524-8600 Fax: (408) 524-8601
E-Mail: khasin@brocadesys.com

Elizabeth A. Vanderbeck (Principal Contributor: Directory Services)
IBM Corporation - P318
522 South Rd.
Poughkeepsie, NY 12601-5400
(914) 435-6242 FAX: (914) 432-9413
Email: beth_vanderbeck@vnet.ibm.com

ANSI ®
dpANS X3.288-199x

draft proposed American National Standard
for Information Systems —

Fibre Channel —
Generic Services (FC-GS)

Secretariat

Computer and Business Equipment Manufacturers Association

Approved199x

American National Standards Institute, Inc.

Abstract

This standard describes in detail the basic Fibre Channel services introduced in ANSI X3.230, FC-PH. In addition, this document describes any ancillary functions and services required to support the Fibre Channel services.

American National Standard

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

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

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

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

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

Published by

American National Standards Institute
11 W. 42nd Street, New York, New York 10036

Contents

Page

Tables

Figures

Foreword (This Foreword is not part of American National Standard dpANS X3.288-199x.)

The Fibre Channel Generic Services (FC-GS) standard describes in detail all of the basic Fibre Channel services introduced in ANSI X3.230, FC-PH. In addition, this document describes any ancillary functions and services required to support the Fibre Channel services.

This standard was developed by Task Group X3T11 of Accredited Standards Committee X3 during 1994. The standards approval process started in 1992. This document includes four annexes which are informative and are not considered part of the standard.

Requests for interpretation, suggestions for improvement or addenda, or defect reports are welcome. They should be sent to the X3 Secretariat, Computer and Business Equipment Manufactures Association, 1250 Eye Street, NW, Suite 200, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by Accredited Standard Committee on Information Processing Systems, X3. Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, the X3 Committee had the following members:

Richard Gibson, Chair
Donald C. Loughry, Vice-Chair
Joanne M. Flanagan, Secretary

Organization Represented	*Name of Representative*
Allen-Bradley Company	Ronald H. Reimer
	Joe Lenner (Alt.)
American Library Association	Paul E. Peters
American Nuclear Society	Geraldine C. Main
	Sally Hartzell (Alt.)
AMP, Inc.	Edward R. Kelly
	Edward Mikoski (Alt.)
Apple Computer, Inc.	Karen Higginbottom
Association of the Institute of Certification of Computer Professionals	Kennety Zemrowski
	Eugene M. Dwyer (Alt.)
AT&T	Thomas F. Frost
	Paul D. Bartoli (Alt.)
Boeing Company	Catherine Howells
	Gail Dohmen (Alt.)
Bull HN Information Systems, Inc.	David M. Taylor
Compaq Computer Corporation	James L. Barnes
	Keith Lucke (Alt.)
Digital Equipment Corporation Users Society	Stephen C. Jackson
	Mike Terrazas (Alt.)
Digital Equipment Corporation	Gary S. Robinson
	Del Shoemaker (Alt.)
Eastman Kodak	James D. Converse
	Michael Nier (Alt.)
Electronic Data Systems Corporation	Jerrold S. Foley
	Charles M. Durrett (Alt.)
GUIDE International	Frank Kirshenbaum
	Harold Kuneke (Alt.)
Hewlett-Packard	Donald C. Loughry
Hitachi America Ltd.	Kei Yamashita
	John Neumann (Alt.)
Hughes Aircraft Company	Harold L. Zebrack

IBM Corporation	Robert H. Follett
	Mary Anne Gray (Alt.)
Lawrence Berkeley Laboratory	David F. Stevens
	Robert L. Fink (Alt.)
National Communications Systems	Dennis Bodson
	George W. White (Alt.)
National Institute of Standards and Technology	Robert E. Rountree
	Michael D. Hogan (Alt.)
NCR Corporation	Thomas W. Kern
	A. R. Daniels (Alt.)
OMNICOM, Inc.	Harold C. Folts
	Cheryl C. Slobodian (Alt.)
Open Software Foundation	Fritz Schulz
Recognition Technology Users Association	Herbert F. Schantz
	G. Edwin Hale (Alt.)
SHARE, Inc.	Thomas B. Steel Jr.
	Gary Ainsworth (Alt.)
Sony Corporation	Michael Deese
Storage Technology Corporation	Joseph S. Zajaczkowski
	Samuel Cheatham (Alt.)
Sun Microsystems, Inc.	Scott K. Jameson
3M Company	Paul D. Jahnke
Unisys Corporation	Stephen P. Oksala
	John L. Hill (Alt.)
U. S. Department of Defense	William C. Rinehuls
	Thomas E. Bozek (Alt.)
	Fred Virtue (Alt.)
U. S. General Services Administration	Douglas K. Arai
	Larry L. Jackson (Alt.)
US West Corporation	Gary Dempsey
	Susan Capraro (Alt.)
USE, Inc.	Pete Epstein
Wang Laboratories, Inc.	Steve Brody
	Sarah Wagner (Alt.)
Wintergreen Information Services	John L. Wheeler
Xerox Corporation	Roy Pierce

Task Group X3T11 on Device Level Interfaces, which developed this standard, had the following participants:

Roger Cummings, Chair
Carl Zeitler, Vice-Chair
Steven L. Wilson, FC-GS Technical Editor

D. Allan	D. Hagerman	S. Mindemann	R. Snively
T. Anderson	T. Harper	A. Miura	D. Somes
K. Annamalai	N. Harris	M. Montana	J. Sostarich
C. Beck	V. Haydu	J. Mork	T. Sprenkle
B. Bellino	D. Hepner	J. Morris	J. Stai
C. Brill	J. Himes	J. Murdock	G. Stephens
K. Chan	E. Jacques	M. O'Donnell	B. Støvhase
K. Chennappan	T. Johnson	D. Olsen	A. Stone
H. Chin	S. Joiner	T. Palkert	T. Szostak
R. Cook	R. Kembel	M. Parvaresh	F. Tarverdians
J. Coomes	J. Knickerbocker	A. Patel	J. Thatcher
R. Cornelius	0. Kornblum	R. Pedersen	L. Thorsbakken
R. Dahlgren	M. Krzych	M. Peterson	D. Tolmie
S. Dean	D. LaFollette	G. Porter	J. Toy
J. Dedek	L. Lamers	R. Prentice	H. Truestedt
F. DeNap	J. Lear	G. Rara	S. van Doorn
M. Dorsett	M. Leib	J. Renwick	E. Vegdahl
B. Edge	R. Leibow	W. Rickard	T. Vrankar
S. Erler	K. Malavalli	E. Rodriguez	R. Wagner
E. Freeman	S. Malladi	R. Ronald	P. Walford
E. Frymoyer	C. Masog	G. Rossmann	N. Wanamaker
T. Fung	B. Masterson	P. Rupert	G. Warden

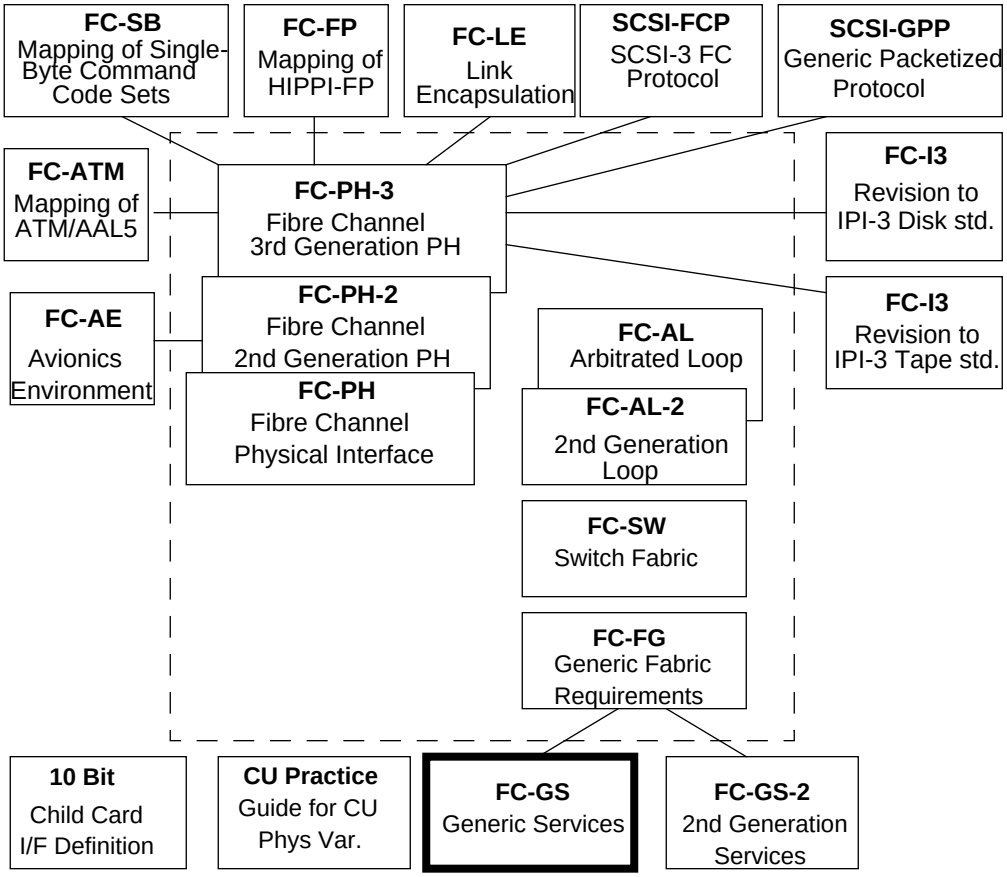
B. Gallagher
C. Grant
M. Griffin
E. Grivna
J. Guedj
D. Hackler
J. Mathis
V. Mattella
K. Mehta
G. Milby
M. Miller
F. Rutherford
P. Scott
M. Shiflett
L. Sloan
K. Smith
R. Whiteman
J. Williams
K. Witte
J. Young
B. Yunker

Introduction

The Fibre Channel-Generic Services draft defines a set of tools to aid in the configuration and management of systems based on the Fibre Channel suite.

The diagram below shows the relationship of this American National Standard (the highlighted rectangle) with other Fibre Channel documents.

.



draft proposed American National Standard for Information Systems —

Fibre Channel — Generic Services (FC-GS)

1 Scope

FC-GS describes in detail the basic Fibre Channel services introduced in ANSI X3.230, FC-PH.

The Fibre Channel services described in this document are:

– Basic Directory Services (DS)

– Native SNMP Mapping (NSM)

– Time Services (TS)

– Alias Server (AS)

In addition, to the aforementioned Fibre Channel services, the Common Transport (CT) protocol is described. The common transport service provides a common FC-4 for use by the Fibre Channel services.

2 Normative references

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

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

2.1 Approved references

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

2.2 References under development

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

X3 Project 901-D, *Information Technology-Fibre Channel-Enhanced Physical/(FC-PH-2)* (1)

X3 Project 258-D, *Information Technology-Fibre Channel-Fabric Requirements (FC-FG)* (1)

X3 Project 955-D, *Information Technology-Fibre Channel-Link Encapsulation (FC-LE)* (1)

1) For information about obtaining copies of this document or for more information on the current status of the document, contact the X3 Secretariat at http://www.x3.org or 202-626-5738.

3 Definitions and conventions

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

3.1 Definitions

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

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

3.1.3 attribute: A characteristic which describes an object and appears in an entry describing that object in the Directory

Information Base (e.g., N_Port ID is an attribute of an N_Port object).

3.1.4 attribute descriptor: That component of an attribute which indicates the class of information described by that attribute (e.g., N_ Port ID).

3.1.5 attribute value: Aparticular instance of the class of information specified by an attribute descriptor (e.g., N_Port ID = hex'123456').

3.1.6 dereferencing: Replacing the directory-alias name for an object by the object's distinguished name.

3.1.7 directory: A repository of information about objects which provides directory services to its users, thereby allowing access to the information.

3.1.8 directory agent (DA): A process which represents the user in accessing the directory.

3.1.9 directory-alias: A directory-alias, or directory-alias name, for an object is a name at least one of whose RDNs is that of a directory-alias entry.

3.1.10 directory-alias entry: A leaf entry of the class "directory-alias" which provides a pointer to an object entry. A directory-alias entry provides an alternate name for an object entry. Note: This term is equivalent to the X.500 term "alias". It has been renamed "directory-alias" in order to distinguish it from the term "alias" as defined by FC-PH.

3.1.11 directory information base (DIB): The complete set of information to which the Directory provides access and which includes all of the pieces of information which can be read or manipulated using the operations of the Directory.

3.1.12 directory information tree (DIT): The DIB considered as a tree whose vertices, other than the root, are the directory entries. Note: the term DIT is used instead of DIB only in contexts where the tree structure of the information is relevant.

3.1.13 directory schema: The set of rules and constraints concerning DIT structure, object class definitions, attribute descriptors and syntaxes which characterize the DIB.

3.1.14 directory server (DS): A process which is part of the directory; it provides DAs with access to the DIB.

3.1.15 distinguished name (DN): A name of an object, formed from the sequence of the RDNs of the object entry and each of its superior entries.

3.1.16 distinguished value: An attribute value in an entry which has been designated to appear in the relative distinguished name of the entry.

3.1.17 directory entry: A part of the DIB which contains information about an object.

3.1.18 endpoint: An object class which describes either a Fibre Channel Node or the process within the Node identified by an Initial Process Associator.

3.1.19 endpoint name: A Fibre Channel name identifier which is associated with either a Fibre Channel node or the process within a node identified by an Initial Process Associator.

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

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

3.1.22 Name_Identifier: A 64 bit identifier, with a 60 bit value preceded with a four bit Network_ Address_Authority_Identifier, used to identify physical entities in Fibre Channel such as N_

Port, Node, F_Port, or Fabric.

3.1.23 Network_Address_Authority (NAA): An organization such as CCITT or IEEE which administers network addresses.

3.1.24 Network_Address_Authority Identifier: A four bit identifier defined in FC-PH to indicate a Network_Address_Authority (NAA).

3.1.25 object class: An identified family of attributes which are relevant to objects of the given class; every object belongs to at least one object class.

3.1.26 relative distinguished name (RDN): A single attribute associated with an entry, chosen as an identifier for the entry.

3.1.27 symbolic name: A user-defined name for an object, up to 255 characters in length. The Directory does not guarantee uniqueness of its value.

3.1.28 unidentified N_Port: An N_Port which has not yet had its N_Port identifier assigned by the initialization procedure.

3.1.29 well-known addresses: A set of address identifiers defined in FC-PH to access global server functions such as a name server.

3.2 Editorial conventions

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

Numbered items in FC-GS do not represent any priority. Any priority is explicitly indicated.

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

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

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

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

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

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

Hexadecimal notation

Hexadecimal notation is used to represent fields. For example, a four-byte Process_Associator field containing a binary value of 00000000 11111111 10011000 11111010 is shown in hexadecimal format as x’00 FF 98 FA’.

3.3 Abbreviations, acronyms and symbols

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

4 Common transport for FC services (CT)

4.1 Overview

The Fibre Channel services share a common transport at the FC-4 level. The common transport provides each Fibre Channel service application (e.g. directory server) with a set of service parameters that facilitates the usage of FC-PH constructs. It also provides another level of multiplexing that will simplify the server-to-server communication for a distributed Fibre Channel service. Class 3 Service, if available in the operational environment, shall also be accessible by the CT user. It is important to note that Fibre Channel services do not require a high performance communication channel as do other high performance I/O protocols such as HIPPI, SCSI, SB, etc. The relationship of CT with respect to its upper level protocols (ULP) and FC-PH is illustrated in figure 1.

From a Fibre Channel service application (entity) point of view, it communicates with another entity by transmitting one or more information unites (IUs) over the Fibre Channel. The other entity may respond by transmitting respective response IUs. There are situations where an entity transmits an IU containing information about an event that is of interest to another entity, and no response IU is required. The role of an CT is to provide the necessary service and mapping to Fibre Channel such that many of the FC-PH constructs and idiosyncracies are shielded from its applications.

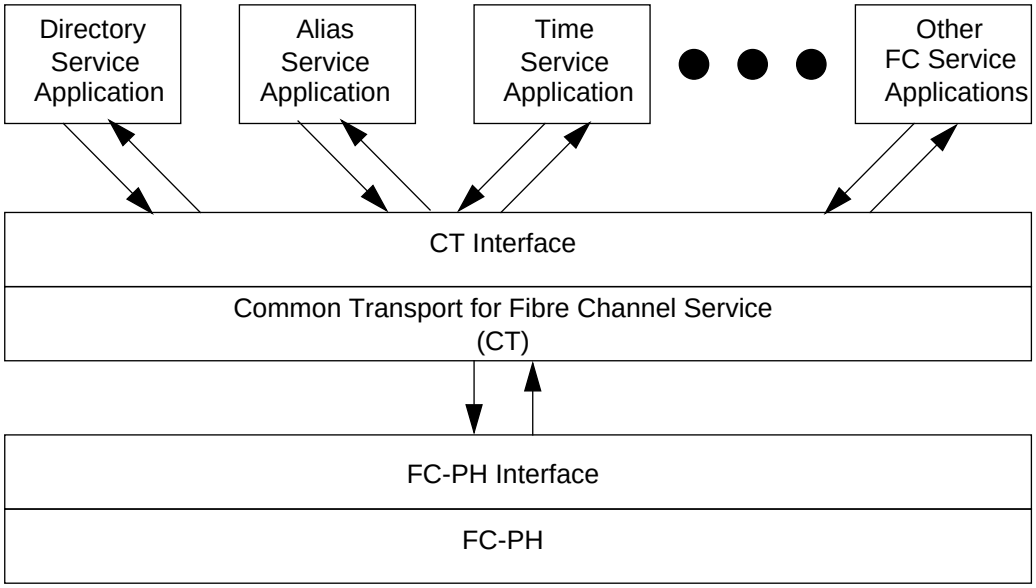


Figure 1 – Relationship of CT with its ULP and FC-PH

4.2 General concepts

The following parameters describe the service that the CT provides to the applications:

– type of Fibre Channel service;

– type of transaction;

– mode of transaction;

– class of service preference;

– maximum size of an IU.

There are currently three types of Fibre Channel services that are mapped to CT:

– Directory service;

– Alias Service;

– Time service.

There are three types of transactions:

– Request: where one entity (client) transmits an IU to another peer (server) to request a Fibre Channel Service;

– Response: where the server transmits an IU to the client responding to its earlier request for a service;

– Unsolicited: where one entity transmits an IU to another entity about an event.

There are two modes of transaction:

– Asynchronous: in which a client may transmit multiple requests without having to wait for the responses; an unsolicited IU is transmitted under this mode since there is no required response to an unsolicited IU;

– Synchronous: in which a client shall not transmit another request until the corresponding response has been received or an indication of non-response.

The class of service preference is an indication of the quality of service that an application expects from the underlying transport. FC-PH defines three classes of service available to an N_Port:

– Class 1;

– Class 2;

– Class 3.

Class F is defined for services within a fabric. This class of service may be used for server-to-server communication where both servers are embedded in the fabric. Since not all classes are necessarily available to an application in order to communicate with its peer, this parameter describes a list of classes of services in a descending order of preference. It is used to express the desire of the service in the order of availability. A sample list of choices are as follows:

– dependable;

– monopolistic;

– tolerant.

NOTE – The class of service preference is specified according to the local service interface (see Annex A). Since the class of service is only meaningful to the local node, this indication is not transported as part of the CT header information.

An application may wish to restrict the size of IUs that it wants to receive from another entity. A destination CT shall observe and reinforce this restriction on behalf of the application. It may do so by setting the abort sequence condition parameter in the ACK frames, or use the abort exchange link service request.

4.3 CT protocol

For each common transport IU (CT_IU) to be transmitted, a source CT shall provide a header (CT_HDR) to a destination CT for each IU as shown in table 1. The source CT sends a request CT_IU to the destination CT. When a response is expected, the destination CT sends a corresponding response CT_IU to the source CT.

The resulting CT IU shall be mapped into FC-PH constructs (see 4.4).

4.3.1 CT_HDR description

The CT_HDR consists of sixteen bytes and is defined as follows.

4.3.1.1 Revision Field

This field denotes the revision of the protocol. The first revision has the value of 1.

Table 1 – CT IU

Word Bits	3322 2222 1098 7654	2222 1111 3210 9876	1111 1100 5432 1098	0000 0000 7654 3210
0	Revision	IN_ID		
1	FCS_Type	FCS_Subtype	Options	Reserved
2	Command/Response code		Reserved	
3	Reserved	Reason code	Reason Code Explanation	Vendor Unique
4	Application			
...	Information			
n	Unit			

4.3.1.2 IN_ID Field

This field is transparent to the source CT representing the original requestor. An entry server shall use this field to store the N_Port address identifier of the original requestor if the request is forwarded to another server.

4.3.1.3 FCS_Type Field

This field is used to indicate the type of Fibre Channel service. The values are defined in table 2.

4.3.1.4 FCS_Subtype Field

This field indicates the specific service behind the well-known N_Port. Values in this field are provided by the individual service.

NOTE – The FCS_Subtype field is used to indicate second level routing behind the N_ port. For example, if more than one directory service is provided at the well-known address X'FFFFFC, then the FCS_Subtype field is used to distinguish these different directory services.

Table 2 – FCS_Type values

Values in hex	Description
00-1F	Vendor Unique (16)
20-F7	Reserved for future services
F8	Alias Server Application
F9	Reserved for future services
FA	Management Service Application
FB	Time Service Application
FC	Directory Service Application
FD	Reserved - Fabric Controller Service
FE-FF	Reserved

This field denotes various options used by the source CT:

Bit Position	7	6	5	4	3	2	1	0
Bit Name	X	Reserved						

– X_Bit: Exchange mapping;

  – O =>> single bidirectional exchange;

  – 1 =>> multiple exchanges;

– the other bits are reserved.

4.3.1.5 Command/Response code field

This field indicates whether the CT_IU is an FS_ IU. When an FS_IU is designated, this field shall either specify a command code, or a response code. Table 3 depicts the valid command/response code values.

Table 3– Command/Response codes

Encoded Value	Description
X'0000'	Non-FS_IU
X'0001'-X'7FFF'	FS_REQ IU
X'8001'	FS_REJ IU
X'8002'	FS_ACC IU
other values	Reserved

4.3.1.6 Reason code field

The reason code field shall designate the reason code associated with an FS_RJT IU (see 4.6.3). When the command /response code field contains a value of X'0000', this field shall not be used.

4.3.1.7 Reason code explanation field

The reason code field designates a reason code explanation associated with an FS_RJT IU (see 4.6.3). When the command /response code field contains a value of X'0000', this field shall not be used.

4.3.1.8 Vendor Unique field

The vendor unique field designates a vendor unique reason code associated with an FS_RJT IU (see 4.6.3). When the command /response code field contains a value of X'0000', this field shall not be used.

4.4 FC-2 mapping and management

Given a service request from a Fibre Channel service application, the CT shall map that into appropriate Fibre Channel constructs.

4.4.1 Fabric login and N_Port login

CT assumes that the Fibre Channel port shall handle the fabric login and N_Port login in the manner that is specified in FC- PH.

4.4.2 Class of service

Based on the class of service preference and the availability of the classes with respect to a destination CT, the source CT shall determine which class of service is to be used for an IU transmission request. The availability of the classes of service is determined from the FC-PH.

NOTE – For simplification, a destination CT entity should use the same class of service in its subsequent communication with the initiating CT.

4.4.3 Exchange and sequence management

The interchange of CT_ IUs between a pair of N_ Ports is coordinated via one or more exchanges, based on the transaction mode selected by the respective application.

In asynchronous mode, separate exchanges in each direction will be used. That is, each source CT shall originate an exchange and hold the Sequence Initiative (SI). In this mode, the source CT shall set the X_Bit in CT_HDR to 1. Transfer of the SI to the destination CT shall be considered a protocol error and the destination CT shall terminate the exchange.

In synchronous mode, a single bidirectional exchange shall be used, and the SI is transferred at the end of an IU transmission. If the destination

CT does not have the SI at the end of an IU reception, it shall consider this to be a protocol error and shall terminate the exchange. In this mode, the source CT shall set the X_Bit in the respective CT_HDR to O.

An exchange created by an CT is to be used only for a specific application instance, and shall not be shared with another application instance.

Each CT_IU shall be mapped into a sequence. The CT_IU tables for asynchronous and synchronous transactions are shown in tables 4 and 5 respectively.

4.4.4 Routing bits

The routing bits shall be set to “FC-4 device data” (binary 0000).

4.4.5 Information category

The source CT shall set this parameter according to the following mapping:

Type of Transaction	Information Category
Request	Unsolicited Control
Response	Solicited Control
Unsolicited	Unsolicited Control

See 4.3.1 for a description of the request and response CT_IUs.

Table 4 – IU table for asynchronous transmission

IU Name	Operations Phase	Information_Set_1		F M L	S I	M or O
		Cat	Payload			
Request	-	2	One request CT information unit	F,M,L	H	M
Response	-	3	One response CT information unit	F,M,L	H	M
Unsolicited	-	2	One request CT information unit	F,M,L	T	M

Table 5 – IU table for synchronous transaction

IU Name	Operation Phase	Information_Set_1		F M L	S I	M or O
		Cat	Payload			
Request	-	2	One request CT information unit	F,M,L	T	M
Response	-	3	One response CT information unit	M,L	T	M

4.4.6 D_ID

The D_ID shall identify the destination Fibre Channel address identifier of the IU. This parameter shall be provided by an application to its CT.

4.4.7 S_ID

The S_ID shall identify the source Fibre Channel address identifier of the IU. The source CT shall specify an address identifier that the source N_Port is allowed to use.

4.4.8 Type

The source CT shall set this parameter to either “Fibre Channel services” (binary 0010 0000).

4.4.9 First sequence

The source CT shall set this parameter to originate a new exchange in order to transmit a IU on behalf of its application.

4.4.10 Last sequence

The source CT may set this parameter to terminate an exchange at the end of a transaction.

4.4.11 Sequence initiative

The source CT shall set this parameter according to the exchange and sequence mapping described in 4.4.3.

4.4.12 Chained sequence

The source CT shall set the chained_sequence bit to zero. This indicates that no reply sequence is expected within a dedicated connection. This parameter shall be passed to the destination CT. If the chained_sequence bit is set to one, the destination CT shall treat this as an error and terminate the dedicated connection.

4.4.13 Continue sequence condition

The source CT may set this parameter according to the size of its IU output queue. This is implementation specific.

4.4.14 Exchange reassembly

CT shall not use exchange reassembly and thus shall set this parameter to 0.

4.4.15 Relative offset

Relative offset may be used by CT if the underlying FC-PH supports it. Each CT IU shall be treated as a continuous data block by the FC-PH and the initial relative offset of each IU shall be set to 0.

4.4.16 Optional headers

The use of any optional header is both implementation and system dependent, and is beyond the scope of CT.

4.5 Error handling

There are two levels of error that may be detected by CT:

- invalid CT_HDR, CT protocol;
  - invalid/undefined FCS_Type;
  - invalid revision level;
  - invalid options;
  - sequence payload exceeds the maximum size of IU at a destination FC_CT.
- FC-PH protocol errors;
  - Sequence errors;
  - Exchange errors.

When a CT protocol error, or invalid CT_HDR error is recognized, the responder indicates the error condition to the requester using a response CT_IU.

When an FC-PH protocol error is detected, the exchange shall be terminated. If the error is detected by the exchange originator, it shall send the no operation link service sequence with the last sequence bit set to the exchange responder. If the error is detected by the exchange responder, there are two methods for the responder to terminate the exchange:

- if the exchange responder has the sequence initiative, it shall send the no operation link service as the last sequence of the exchange;
- if the exchange responder does not have the SI, it shall transmit the abort exchange link service in another exchange to the destination N_Port.

Each error condition shall also be indicated to its application.Request and response information units

4.6 FS information units

A set of Fibre Channel Service request and response information units (FS_IUs) are defined by CT for use by the Fibre Channel services. One FS request information unit is defined:

– Request (FS_REQ)

Two FS response information units are defined:

– Accept (FS_ACC);

– Reject (FS_RJT).

4.6.1 FS_REQ information unit

A CT_IU is designated a FS_REQ IU when the command/response code field contains a command code value of X'0001'-X'7FFF'.

The Command Code shall define the particular request that is to be executed by the Server. The Command Codes shall be defined independently by each Server.

The application information unit contains the associated command specific data. The associated command specific data shall be defined independently by each Server, based on the command code.

4.6.2 FS_ACC information unit

A CT_IU is designated a FS_ACC IU when the command/response code field contains a value of X'8002'. The FS_ACC shall notify the Initiator of a Server request that the request has been successfully completed.

The application information unit contains the associated response specific data. The associated response specific data shall be defined independently by each Server, based on the command code.

4.6.3 FS_RJT information unit

A CT_IU is designated a FS_RJT IU when the command/response code field contains a value of X'8001'. The FS_RJT shall notify the Initiator of a Server request that the request has been unsuccessfully completed.

The Reason code indicates the general reason why the request was rejected. Table 3 indicates the defined FS_RJT reason codes.

The Reason code explanation further defines the indicated Reason Code. These are unique to the particular Server and are defined by each Server.

The vendor unique field may be used by Vendors to specify additional reason codes.

Table 6– FS_RJT Reason Codes

Encoded Value	Description
00000001	Invalid command code
00000010	Invalid version level
0000 0011	Logical error
0000 0100	Invalid IU Size
0000 0101	Logical busy
0000 0111	Protocol error
0000 1001	Unable to perform command request
0000 1011	Command not supported
others	Reserved
1111 1111	Vendor Unique Error (see Vendor Unique field)

Invalid command code: The command code passed in the FS_REQ is not defined for the addressed Server.

Invalid version level: The specified version level is not supported for the addressed server.

Logical error: The request identified by the FS_REQ command code and Payload content is invalid or logically inconsistent for the conditions present.

Invalid IU size: The IU size is invalid for the addressed server.

Logical busy: The Server is logically busy and unable to process the request at this time.

Protocol error: This indicates that an error has been detected which violates the rules of the Server protocol which are not specified by other error codes.

Unable to perform command request: The Recipient of the FS_REQ is unable to perform the request.

Command not supported: The Recipient of the FS_REQ does not support the command requested.

Vendor Unique Error: The Vendor Unique Field may be used by Vendors to specify additional reason codes.

4.7 Correlation of requests and responses

The correlation of requests and responses shall be managed by the specific service application. Therefore, CT provides no mechanism for this management.

5 Overview of directory services

The Fibre Channel directory services structure is modeled after the CCITT data communication networks directory recommendations X.500 - X.521 (ISO 9594-1). Due to the lower complexity and different requirements of the Fibre Channel environment, Fibre Channel directory services is a non-compatible subset of the CCITT specification. Terminology used in this document follows that of the CCITT directory services whenever possible.

In addition to the X.500 based directory service, an individual FC-4 may provide its own specific directory access protocol. FC-4 based directory access service payloads and protocols are defined by the specific FC-4.

NOTE – The common transport (see clause 4) allows additional directory service functions to be defined in addition to the X.500 based directory service.

5.1 Directory services functional model

The Directory services functional model consists primarily of two entities:

– Directory Agent (DA): The DA is a process which represents the user in accessing the directory;

– Directory Server (DS): The Directory is composed of one or more DSs, which may work together or individually to service requests from DAs.

5.2 Scope of directory

The scope of a directory is a single region within a fabric address space. A given directory service has no information on entities outside of this region.

5.3 Directory information base

The Directory Information Base (DIB) is a conceptual model of the information stored by the directory. The DIB may be located in a single DS, or distributed or replicated among multiple DSs. Entries in the DIB may be either object entries or directory-alias entries. An object entry contains information on a specific object. A directory-alias entry is a pointer to an object entry. In effect, a directory-alias provides another name for the same object. An object (such as an N_Port) shall have only one object entry in the DIB. The uniqueness of an object is enforced through its distinguished name in the DIT. However an object may have more than one directory-alias entry.

5.4 Directory information tree

DIB entries are organized in a hierarchical tree structure, with each vertex, other than the root, representing an entry. This tree structure is called the directory information tree (DIT).

A DIB entry consists of a set of attributes which describe a given object. An attribute consists of an attribute descriptor and the corresponding attribute value(s) (e.g., N_Port ID, ‘012345’H). Each DIB entry is given a relative distinguished name (RDN), comprised of a single attribute of that entry. The attribute which is assigned as part of the RDN is called the distinguished attribute. In addition to the RDN, each entry also has a distinguished name (DN), formed by concatenating the RDNs of all vertices traversed on the tree while tracing a path from the root to the entry.

A DA modifies or queries the directory by requesting operations on one or several entries in the DIT. The DA identifies an entry by its distinguished name, which may be a directory-alias.

5.5 Object class

Each object entry in the DIB contains a mandatory attribute called “object class”. The object class defines the allowed characteristics of an entry by describing the set of mandatory and optional attributes with which it is associated. (For example, the N_Port object class contains the attributes of N_Port Name, N_Port ID, etc...).

Object classes may be related to each other using the concept of inheritance. Class inheritance allows one object class to be defined as a “subclass” of another, previously defined, object class. The sub-class contains all of the attributes of the previously defined class, as well as additional refinements, specific to the subclass.

Each attribute descriptor must belong to at least one object class.

When a DA adds an entry to the DIB, it specifies the object class of the entry, thus allowing the DS to enforce any rules associated with that object class (e.g., disallow addition of attributes not defined for that object class). Once an entry has

been added, the DA may not modify the object class attribute.

5.5.1 Top object class

Enforcing the rule that every object entry in the DIB contains an object class attribute is achieved by use of the Top object class. This object class consists of the single attribute “object class”. Every other object class is required to be a sub-class of the Top object class. Thereby, every object class is forced to have an “object class” attribute.

5.5.2 Directory-alias object class

In a manner similar to that of the Top object class, the directory-alias object class is used to enforce the rule that all directory-alias entries contain an “aliased object name” attribute.

The directory-alias object class (which is a subclass of Top) consists of the single attribute “aliased object name”. This attribute serves as a pointer to the object which the directory-alias entry represents. All directory alias type classes are required to be sub-classes of the directory-alias object class.

5.6 Transportation of directory service payloads

Directory service payloads shall be transported between the DA and the DS using the common transport (CT) mechanism (see clause 4).

5.6.1 Directory service mapping to CT

For a directory request, the directory payload shall be transported from the DA to the DS using a request CT_IU. The corresponding directory response is transported from the DS to the DA using a response CT_IU. FS information units shall not be used to transport directory service requests and responses (i.e. command code value of X’0000’ is specified in the CT_HDR).

5.6.2 CT_HDR Values

The following values are provided in the CT_HDR:

Revision: X’01’;

IN_ID: N_Port identifier of the DA;

FCS_Type: X’FC’;

FCS_Subtype: See table 7.

Options (Xbit set to B’1’);

Command code: X’0000’.

Table 7 – Directory service subtype values

Values in hex	Description
01	X.500 based directory service
02	Name service
80-EF	FC-4 specific service
other values	Reserved

5.7 FC-PH constructs

Before performing any directory operation, the N_Port associated with the DA shall perform F_Port login followed by N_Port login with the well-known destination address hex ‘FFFFFC’. It is recommended that the DA perform N_Port Logout with the directory when no further operations are pending. If a DS becomes resource constrained, it may perform N_Port Logout with a DA, using an implementation-dependent guideline such as a least recently used algorithm.

5.7.1 Exchanges

Directory services exchanges are utilized in a unidirectional manner. That is, DS information units are sent in only one direction on a given exchange. The Invoke ID of a given request (and possibly the source N_Port ID and IPA) is used to correlate requests and responses. A DA may maintain an exchange to the DS for as long as it has requests to be sent. Sequence Initiative shall not be passed except in some error recovery scenarios such as with the use of the abort sequence condition.

Either a DA or a DS may originate an exchange. For request transmission, the DA shall be the exchange originator. For response transmission, the DS shall be the exchange originator.

5.7.2 Information units

The Information Unit construct defines the information placed in the payload of one or more Fi-

bre Channel frames for transmission between a DS and a DA. A single Information Unit contains either a DA request or a DS response. All communication occurs through the exchange of Information Units. This clause describes both the data which are transparent to Fibre Channel and those control parameters which are required by Fibre Channel.

5.7.3 Common required FC parameters

5.7.3.1 Class of service

The DS shall support Classes 1, 2 and 3, where they are available. The DA may communicate with the DS using any desired class.

If a DA or DS uses Class 3 to communicate, it shall provide the Sequence_Tag on the FC-PH service interface. Any Sequence_Tag provided on a given IU shall not be reused on a subsequent IU associated with the same Exchange until either of the following conditions exists:

– R_A_TOV has expired since the DS or DA has determined that the IU has been transmitted by the FC-2.

– The DS or DA receives a response to the transmitted IU from the receiver of the IU.

5.7.3.2 R_CTL routing bits

Routing bits of the R_CTL field shall indicate FC-4 Device Data.

5.7.3.3 Information category

All request IUs shall specify unsolicited control. All response IUs shall specify solicited control.

5.7.3.4 Sequence initiative

Sequence Initiative shall not be transferred during normal operation. Sequence Initiative shall only be transferred in the handling of abort sequence, as described by FC-PH. The response to abort sequence shall transfer initiative back to the exchange originator.

5.7.3.5 Destination ID (D_ID)

This parameter shall be set to the well known destination address hex ‘FFFFFC’ in a directory agent request. In a directory server response, the destination id is set to the value of the source id in the associated directory agent request.

5.7.3.6 Source ID (S_ID)

This parameter shall identify the source address identifier of the IU.

5.7.3.7 Type

All DS IUs shall specify the Fibre Channel services type.

5.7.3.8 Error policy

All error policies with the exception of Process Policy are permitted.

5.7.4 Common optional FC parameters

5.7.4.1 Expiration/security header

The use of this parameter is beyond the scope of this document and is both implementation and system dependent.

5.7.4.2 Network header

The use of this parameter is beyond the scope of this document and is both implementation and system dependent

5.7.4.3 Association header

The use of this parameter is beyond the scope of this document and is both implementation and system dependent.

5.7.4.4 Device header

The Device Header shall not be used.

6 Fibre channel directory schema

The Directory schema provides a set of rules which apply to DIT structure, object classes, attribute descriptors and attribute syntaxes. These rules and restrictions ensure that the relations described by the DIT are consistent and logical.

6.1 DIT structure definition

The DIT structure for Fibre Channel Directory Services is shown in figure 2.

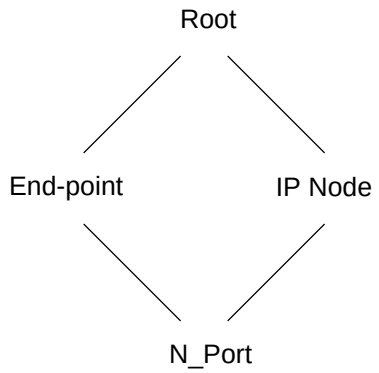


Figure 2 – Fibre channel DIT structure

In addition to the Top and Directory-Alias object classes, four Fibre Channel specific object classes are defined: End-point, N_Port, IP Node, and N_Port Directory-Alias. These object classes and attribute descriptions are described in the following sections. In describing the attributes, the following definitions apply:

– Mandatory indicates whether the attribute is required in all entries of the specified object class. A Directory Server is required to support registration of all mandatory attributes. Support for registration of all other attributes is optional however it is recommended. If a DS does not support registration of the C-Name attribute, it shall still support filters with the C-Name attribute. General handling of unsupported attributes is described in 8.2.5. The Query Capabilities function allows a DA to determine which optional attributes are supported by a DS;

– Distinguished attribute indicates whether this attribute is defined as the entry's Relative Distinguished Name. For Fibre Channel Directory services, each entry may have only a single attribute as its Relative Distinguished Name;

– Single value versus "multi-valued" indicates whether the attribute shall have only one value or whether multiple values may be associated with the attribute.

6.2 Object class definitions

NOTE – When the attributes of a given object class are listed, attributes of super-classes are not included. In order to obtain a complete list of attributes associated with an object class, all superclass attributes must be considered.

6.2.1 Top object class

The Top object class is a superclass of all other object classes. It provides the structure which is required for all other object classes, in order to allow the Directory to effectively enforce the schema.

6.2.1.1 Top attributes

The attributes associated with the Top object class are as follows:

– Object Class (mandatory; multi-valued).

6.2.2 Directory-alias object class

The Directory-Alias object class is a superclass of all other object classes which serve as Directory-Aliases. It provides the structure which is required to enforce the schema on Directory-Alias entries. The Directory-Alias object class is subordinate only to the Top object class.

6.2.2.1 Directory-alias attributes

The attributes associated with the Directory-Alias object class are as follows:

– Aliased Object Name (mandatory, single value). The Aliased Object Name contains the Distinguished Name of the object to which the Directory-Alias refers.

6.2.3 End-point object class

The End-point object class contains those attributes associated with either a Fibre Channel Node, an FC-LE entity within a Node (identified by an IEEE Name Identifier), or the End-point within a Node (identified by an Initial Process Associator combined with a Name Identifier). Note that, while there may be a relationship among a given set of Nodes, End-points and

FC-LE entities (e.g., End-point X is within Node Y), this information is not necessarily reflected in the Directory. SNMP is recommended for gathering such configuration information. The End-point object class is subordinate only to the Top object class.

6.2.3.1 End-point attributes

The attributes associated with the End-point object class are as follows:

– End-point Name Identifier (mandatory, distinguished attribute, single value);

– Initial Process Associator (single value);

– Symbolic Name (single value);

– ULPSpecificInfo (multiple values).

6.2.4 N_Port object class

The N_Port object class contains those attributes associated with a Fibre Channel N_Port. This object class is subordinate only to the Top object class.

6.2.4.1 N_Port attributes

The attributes associated with the N_Port object class are as follows:

– N_Port Name (mandatory, distinguished attribute, single value);

– N_Port ID (mandatory, single value);

– Alias ID (multi-valued);

– ULPSpecificInfo (multiple values);

– Communicating Name (multi-valued): This attribute contains the remote (i.e., on the remote side of the Fibre Channel link/Fabric) End-point Names with which this N_Port is configured to communicate. If an N_Port is capable of communicating with every End-point, this attribute is not present;

– FC4DataType (mandatory, multi-valued);

– IEEE Multicast Group (multi-valued);

– Symbolic Name (single value).

6.2.5 N_Port Directory-alias object class

The N_Port Directory-Alias object class is used to describe an alias for an N_Port object. The N_Port Directory-Alias provides a pointer to the N_Port object which it represents. That is, there shall exist only one N_Port object under a single End-point or IP address; any other End-points or IP addresses which share this N_Port shall have a subordinate N_Port Directory-Alias for that N_Port. This object class is subordinate to the Directory-Alias object class.

The Directory Server is not required to validate the existence of an object pointed to by a Directory-Alias at the time the Directory-Alias entry is added. This dereferencing is done only at the time a search is executed and the Directory-Alias is encountered.

6.2.5.1 N_Port directory-alias attributes

The attributes associated with the N_Port Directory-Alias object class are as follows:

– N_Port Name (mandatory, distinguished attribute, single value).

6.2.6 IP node object class

The IP Node object class contains those attributes associated with a given IP address. This object class is subordinate only to the Top object class.

6.2.6.1 IP node attributes

The attributes associated with the IP Node object class are as follows:

– IP Address (mandatory, distinguished attribute, single value);

– Initial Process Associator (single value).

6.3 Syntax definitions

The syntax for the defined attribute descriptors and other constructs is described in the following sections.

6.3.1 ASN.1 and BER overview

ASN.1 (Abstract Syntax Notation One) is an abstract syntax language used to describe the functions and attributes associated with the Directory. It is the same language used to describe the Simple Network Management Protocol (SNMP) and associated Management Information Base (MIB) definitions, which are utilized in management of TCP/IP based networks. ASN.1 is also used in the OSI X.500 directory standard and the OSI systems management standards.

The purpose of an abstract syntax language is to describe data types in a machine-independent manner and to allow complex data types to be constructed from simpler data types. The characteristic of machine-independence allows the transfer of data from one machine to another without concern for how data types are represented in a given machine architecture (e.g., representing an integer as a 16 versus 32 bit quantity). This is achieved through the tagging of data. If, for example, a value is tagged as an integer, a receiving machine may represent the integer according to its local architecture.

In an ASN.1 description, capitalization provides information on what is represented by a given word. If a word consists of all upper-case letters, it is an ASN.1 macro or keyword. If the first letter of a word is capitalized, the word represents a type. If a word consists of all lower-case letters, it represents a specific instance of a type

ASN.1 defines four types of descriptive tags, three of which are utilized by Fibre Channel Directory Services:

– Universal: These tags are defined by the ASN.1 document. Examples are INTEGER, SEQUENCE, OCTET STRING);

– Application-wide: These tags are unique within a given module, such as Directory Services. Examples are N_Port ID, Node Name);

– Context-specific: These tags are unique within a SEQUENCE or SET, for example. Each item in a SEQUENCE may be tagged with a context-specific tag, which identifies it within the SEQUENCE.

In order for the abstract syntax notation to be meaningful, it must be mapped to a transfer syntax for transmission. The transfer syntax describes the actual Fibre Channel payload. The Basic Encoding Rules (BER) are the type of transfer syntax used with ASN.1.

The BER are applied to transform information described by ASN.1 into a form suitable for unambiguous transmission on a link and parsing by a recipient. The BER use a structure based on a “TLV” concept to concretely represent ASN.1 constructs. Each instance of an ASN.1 construct is mapped into a Tag, a Length and a Value (making a received stream of data entirely self- descriptive). The Tag describes the data type (e.g., N_Port ID or INTEGER), the Length provides the length of the data, and the Value contains the actual data.

6.3.2 Definitions

The following sections describe the ASN.1 types defined for Fibre Channel Directory Services.

Several encoding restrictions are defined:

– Indefinite length encodings are prohibited;

– All simple types shall be encoded in primitive form. This includes INTEGER, OCTET STRING and BIT STRING.

6.3.2.1 Alias ID

This attribute contains the 3 byte alias address identifier as defined in FC-PH

This attribute may be multi-valued, with each value corresponding to a unique alias address identifier:

```
AliasID ::=
[APPLICATION 5]
IMPLICIT OCTET STRING (SIZE (3))
```

6.3.2.2 Aliased object name

This attribute contains the Distinguished Name of the entry pointed to by the Directory-alias.

```
AOname ::=
[APPLICATION 9]
IMPLICIT DistinguishedName
```

6.3.2.3 Attribute descriptor

The attribute descriptor specifies the attributes which are accessible from the Directory by a DA. It is used when an attribute type is specified without a value, for example in the EntryInformationSelection (see 7.2.2).

The following attribute types are defined:

```
AttributeDescriptor ::=
[APPLICATION 17]
IMPLICIT INTEGER{
endPointName (1),
initialPas (2),
ipAddress (3),
nPortID (4),
aliasID (5),
nPortName (6),
cName (7),
fc4DataType (8),
aoName (9),
symbolicName (10)
objectClass (18)}
ulpSpecificInfo (28),
ieeeMulticastGroup (29)}
```

6.3.2.4 Boolean

This attribute is used to specify a true/false condition:

```
Boolean ::= INTEGER{
true (1),
false (2)}
```

6.3.2.5 Communicating name

This attribute contains the remote End-point Names with which the N_Port object is configured to communicate.

The C_Name is needed because certain system architectures require that specific local N_Ports be capable of accepting frames from only a subset of the remote End-points attached to the Fabric. The remote End-points which may communicate with a given N_Ports are specified by the C_Name and registered with the Directory. When a query is received which requests available N_Ports associated with a given End-point, the requester provides its C-Name (i.e., its End-point Name), in effect identifying itself and requesting only those N_Ports which are capable of communicating with that End-point,

An End-point which does not utilize the C-Name concept does not register a C-Name value.

The C_Name attribute may be multi-valued. Each instance contains a single End-point Name Identifier, as defined by FC-PH:

```
Cname ::=
[APPLICATION 7]
IMPLICIT OCTET STRING (SIZE (8))
```

6.3.2.6 Distinguished name

```
DistinguishedName ::= SEQUENCE OF Rdn
```

The sequence of RDNs which comprises a Distinguished Name must be fully qualified. That is, it must traverse a path starting from the root of the tree. However, an RDN for Root shall be implied when any other RDNs are present in the Distinguished Name. It is only explicitly specified when Root itself is specified, and no other RDNs are present. This attribute is single-valued.

6.3.2.7 EndPoint name

The EndPoint Name contains the Name Identifier (described in FC-PH) of an End-point. (see 3.1.19).

This attribute is single-valued:

```
EndPointName ::=
[APPLICATION 1]
IMPLICIT OCTET STRING (SIZE (8))
```

6.3.2.8 Fibre channel attribute

This attribute contains a list of all attributes which are accessible from the Directory by a DA. It is used when an attribute type and value are specified, for example in the AddEntry request (see 7.6.1).

This attribute is single-valued:

```
FCattribute ::= CHOICE(
DistinguishedAttribute,
NonDistinguishedAttribute}

DistinguishedAttribute ::= CHOICE{
endPointName EndPointName,
nPortName NportName,
ipAddress IPaddress,
root Root}

NonDistinguishedAttribute ::= CHOICE{

initialPas InitialPas,
nPortID NportID,
aliasID AliasID,
cName Cname,
fc4DataType FC4DataType,
symbolicName SymbolicName,
aoName AOname,
objectClass ObjectClass,
ulpSpecificInfo ULPspecificInfo,
ieeeMulticastGroup IEEEmulticastGroup}
```

6.3.2.9 FC4DataType

```
FC4DataType ::=
[APPLICATION 8]
IMPLICIT INTEGER (0..255)
--as per FC-PH Rev 4.2 Table 36
```

This attribute may be multi-valued.

Each integer contains a value from the table entitled “Type Codes - FC-4 (Device_Data and Link_Data)” i n FC-PH.

6.3.2.10 Initial process associator

The Initial Process Associator is described in FC-PH.

This attribute is single-valued:

```
InitialPas ::=
[APPLICATION 2]
IMPLICIT OCTET STRING (SIZE (8))
```

6.3.2.11 Invoke ID

The Invoke ID is a unique identifier for a Directory operation, provided by the DA.

This attribute is single-valued:

```
InvokeID ::=
INTEGER (0..'ffffffff'h)
```

6.3.2.12 IP address

This attribute conforms to the standard definition of IP addresses in the TCP/IP protocol.

This attribute is single-valued:

```
IPaddress ::=
[APPLICATION 3]
IMPLICIT OCTET STRING (SIZE (4))
```

6.3.2.13 IEEE multicast group

```
IEEEmulticastGroup ::=
[APPLICATION 29]
IMPLICIT OCTET STRING (SIZE(6))
```

The IEEE Multicast Group attribute contains one or more 48 bit IEEE group address(es) recognized by the associated N_Port.

This attribute may be multi-valued.

6.3.2.14 Nport ID

The N_Port ID is described in FC-PH.

```
NportID ::=
[APPLICATION 4]
IMPLICIT OCTET STRING (SIZE (3))
```

6.3.2.15 Nport name

The N_Port Name is described in FC-PH.

This attribute is single-valued.

```
NportName ::=
[APPLICATION 6]
IMPLICIT OCTET STRING (SIZE (8))
```

6.3.2.16 Object class

```
ObjectClass ::=
[APPLICATION 18]
IMPLICIT INTEGER{
nPort (1),
endPoint (2),
ipNode (3),
nPortDirectoryAlias (4),
directoryAlias (5)}
```

The Object Class attribute may not be modified by a DA. The DA shall specify the value of the entry’s object class in the AddEntry request, if that entry does not exist in the DIB. If the request is accepted by the DS, the DS shall include its superclass values, except the Top, in the entry. The DA may not subsequently modify the values of the Object Class attribute of that entry.

This attribute may be multi-valued.There shall be one value of the attribute for its object class and one for each of its superclasses. No value for the “Top” object class is necessary because all object classes are subclasses of “Top”.

6.3.2.17 Relative distinguished name

RDN is defined in 3.1.26.

DistingushedAttribute is defined in 6.3.2.8.

This attribute is single-valued.

```
Rdn ::= DistinguishedAttribute
```

6.3.2.18 Root

Root is defined as a Null and does not constitute a DIB entry. Root is used when the DA is specifying the root of the DIT as the starting point for a search operation. Unless explicitly stated, root shall not be used in the operations where the object entry is to be specified. Specifically, the root is not part of a distinguished name when RDNs other that the root itself are specified.

```
Root ::= NULL
```

6.3.2.19 Symbolic name

Symbolic Name is defined in 3.1.27.

This attribute is single-valued:

```
SymbolicName ::=
[APPLICATION 10]
IMPLICIT OCTET STRING (SIZE(1..255))
```

6.3.2.20 ULP specific information

```
ULPspecificInfo ::=
[APPLICATION 28]
IMPLICIT OCTET STRING (SIZE(1..16))}
```

ULP Specific Information is a generic attribute which is made available to ULPs for storing information defined by the FC-4 associated with the ULP. It is intended that the Octet String will define one or more characteristics of the ULP. The format of the information contained in ULP Specific Information is beyond the scope of this document.

Although not enforced by the Directory, it is recommended that the first byte of the Octet String should contain the FC-4 Type code. This allows a Directory Agent to use a substring filter, matching the desired type on the initial byte of the octet string and any other information on subsequent bytes of the string.

This attribute may be multi-valued. In this case, each Octet String could apply to a different ULP.

7 Directory access service

The directory access service describes the protocols used by a DA to communicate with the directory.

7.1 Directory service message

A directory request or response is transported between the DA and the server using a directory service message. The directory service message contains the version number of the directory message unit, and the operation/result data. A directory service message shall be contained in one application information unit (see 4.3).

The directory service message is defined as:

```
Message ::= SEQUENCE{
            version INTEGER{
                  version-1(0)
                   },
            data   OperationsAndResults}
```

7.1.1 Operation/Result data

The operation/result data consists of either the directory operation request, or the result of a directory operation request. The operation/result data is defined as:

```
OperationsAndResults ::= CHOICE{
   compareRequest  [1]IMPLICIT CompareRequest,
   compareResult   [2]IMPLICIT CompareResult,
   abandonRequest  [3]IMPLICIT InvokeID,
   abandonResult   [4]IMPLICIT InvokeID,
   searchRequest    [5]IMPLICIT SearchRequest,
   searchResult    [6]IMPLICIT SearchResult,
   addEntryRequest  [7]IMPLICIT AddEntryRequest,
   addEntryResult  [8]IMPLICIT InvokeID,
   removeEntryRequest [9] IMPLICIT RemoveEntryRe-
                          quest,
   removeEntryResult[10] IMPLICIT InvokeID,
   modifyEntryRequest  [11]  IMPLICIT  ModifyEn-
                               tryRequest,
   modifyEntryResult[12] IMPLICIT InvokeID,
   queryCapRequest  [13] IMPLICIT InvokeID,
   queryCapResult  [14] IMPLICIT QueryCapResult,
   error           [15] IMPLICIT Error}
```

7.2 Directory access service - common parameters

The following sections describe the parameters which are common to more than one directory operation. These parameters are:

– CommonArguments;

– EntryInformationSelection;

– Filter;

– PagedResultsRequest;

– CommonResults;

– EntryInformation.

7.2.1 Common arguments

```
CommonArguments ::= SEQUENCE{
   controls   [30]IMPLICIT ServiceControls
                OPTIONAL}
```

The commonArguments, described below, shall be provided with each directory request.

7.2.1.1 Service controls

```
ServiceControls ::= SEQUENCE{
 useCopy           [0]IMPLICIT Boolean
                     DEFAULT true,
 dereferenceAliases[1]IMPLICIT Boolean
                     DEFAULT true,
 priority          [2]IMPLICIT INTEGER{
             low(0),
             medium (1),
             high (2)} DEFAULT medium,
 timeLimit         [3]IMPLICIT INTEGER OPTIONAL,
 sizeLimit         [4]IMPLICIT INTEGER OPTIONAL}
```

The Service Controls parameter defines the constraints to be placed on the execution of the current request by the directory:

– Use copy;

  – If the usecopy parameter is true, it indicates that copied (cached) information may be used to provide the service;

  – If the usecopy parameter is false, it indicates that copied (cached) information shall not be used to provide the service;

– Dereference aliases;

  – If the dereferenceAliases parameter is true, it indicates that any alias used to identify the first or base object entry affected by an operation shall be dereferenced;

  – If the dereferenceAliases parameter is false, it indicates that any alias used to identify the first or base object entry affected by an operation shall not be dereferenced;

– Priority: The priority parameter provides a request that the current operation be treated as high, medium or low priority. It does not guarantee such action by the directory;

– Time limit: The timelimit parameter de-

fines the maximum amount of elapsed time, in seconds, within which the service shall be provided. If the time constraint cannot be met, an error is reported;

– Size limit: The sizelimit parameter defines the maximum number of entries to be returned to the requestor. This parameter only applies to Search operations.

7.2.2 EntryInformationSelection

```
 EntryInformationSelection ::= CHOICE{
allAttributes NULL,
select SEQUENCE OF AttributeDescriptor
 --empty sequence implies no attributes--
 --are requested--}
```

The EntryInformationSelection parameter describes what information is being requested from an entry. This parameter is only applicable to search operations. Two options are available:

– AllAttributes: Information is requested about all attributes associated with an entry;

– Select: A list is provided which contains those attributes which are of interest to the DA. If the list is empty, then no attributes shall be returned.

7.2.3 Filter

The Filter parameter applies a test which is either satisfied or not by a particular entry. The filter is expressed in terms of assertions about the presence or value of certain attributes of the entry and is satisfied if and only if it evaluates to true. Note: a filter may be TRUE, FALSE or undefined.

In effect, the filter specifies to the DS those attribute values that the DA knows and wants to match in the DS. For example, the filter may specify that the DA is interested in all entries which have an FC4datatype of IP.

```
Filter ::= CHOICE{
  item          [0] FilterItem,
  and           [1] IMPLICIT SEQUENCE OF
                    FilterItem,
  or            [2] IMPLICIT SEQUENCE OF
                    FilterItem,
  not           [3] Filter}

 FilterItem ::= CHOICE{
  equality          [0] FCattribute,
  substrings        [1] SEQUENCE{
        type     AttributeDescriptor
        strings  SEQUENCE OF CHOICE{
                    initial [0] String,
                    any     [1] String,
                    final   [2] String}},
  greaterOrEqual   [2] FCattribute,
  lessOrEqual      [3] FCattribute,
  present          [4] AtributeDescriptor,
  approximateMatch [5] FCattribute}

String ::= CHOICE{
  bit              BIT STRING,
  byte             OCTET STRING}
```

A filter is either a FilterItem (described below) or an expression using simpler filters composed together using the logical operators **and**, **or** and **not**. Where the filter is:

– Item: A filter which is a filteritem takes on the value of the filteritem (i.e., TRUE, FALSE or undefined);

– And: A filter which is the and of a set of filters is TRUE if the set is empty or if each filter is TRUE. It is FALSE if at least one filter is FALSE. Otherwise it is undefined (i.e., if at least one filter is undefined and no filters are FALSE);

– Or: A Filter which is the or of a set of filters is FALSE if the set is empty or if each filter is FALSE. It is TRUE if at least one filter is TRUE. Otherwise it is undefined (i.e., if at least one filter is undefined and no filters are TRUE);

– Not: A Filter which is the not of a filter is TRUE if the filter is FALSE, FALSE if the filter is TRUE, and undefined if the filter is undefined.

Table 8 summarizes the evaluation of a filter based on which logical operators are applied.

Table 8 - Evaluation of filters

A	B	OR	AND	NOT A
t	t	t	t	f
t	f	t	f	f
t	u	t	u	f
f	f	f	f	t
f	u	u	f	t
u	u	u	u	u
empty	empty	f	t	na

Key:

A=filter B=filter t=true f=false u=undefined

na=not applicable

If a FilterItem attribute is not supported by a DS, it shall be treated as false in the evaluation of the filter.

7.2.3.1 FilterItem

A FilterItem is an assertion about the presence or value(s) of an attribute of a particular type in the entry under test. Each such assertion is TRUE, FALSE, or undefined. Every FilterItem includes an FCattribute which identifies the particular attribute concerned. Any assertion about the value of such an attribute is only defined if the attribute is known and the attribute syntax is appropriate for the attribute descriptor. When these conditions are not met, the FilterItem is undefined.

Assertions about the value of an attribute are evaluated using the matching rules associated with the attribute syntax defined for that attribute descriptor. A matching rule not defined for a particular attribute syntax cannot be used to make assertions about that attribute. Where this condition is not met, the FilterItem is undefined.

A FilterItem may be undefined. Otherwise, where the FilterItem asserts:

– Equality;

  – Equality is TRUE if and only if there is any value of the attribute which is equal to that asserted;

  – This filter item may be applied to all attribute types;

– Substrings;

  – Substrings are TRUE if and only if there exists a value of the attribute specified by the type parameter in which the specified substrings appear in the given order. The substrings shall be non-overlapping, and may (but need not) be separated from the ends of the attribute value and from one another by zero or more string elements;

NOTE – Non-overlapping means that there is no intersection of the bytes contained in the separate sets of substrings.

  – If “initial” is indicated, the substring shall match the initial substring of the attribute value; if “final” is indicated, the substring shall match the final substring of the attribute value; if “any” is indicated, the substring may match any substring in the attribute value;

  – This filter item may be applied only to attributes which are of type BIT STRING or OCTET STRING;

– greaterOrEqual;

  – GreaterOrEqual is TRUE if and only if the relative ordering (as defined by the appropriate ordering algorithm) places the supplied value before or equal to any value of the attribute;

  – This filter item may be applied only to attributes which are of type INTEGER;

– lessOrEqual;

  – LessOrEqual is TRUE if and only if the relative ordering (as defined by the appropriate ordering algorithm) places the supplied value after or equal to any value of the attribute;

  – This filter item may be applied only to attributes which are of type INTEGER;

– present;

  – Present is TRUE if and only if such an attribute is present in the entry;

  – This filter item may be applied to all attribute types;

– approximateMatch;

  – ApproximateMatch is TRUE if and only if there is a value of the attribute which matches that which is asserted by a specific approximate matching algorithm;

  – Presently, only the CName approximate matching algorithm is defined;

    – CName: If an approximate match is attempted against this attribute, the match shall be TRUE if one of the following two conditions exists;

    – The entry contains a CName attribute which has a value equal to that

asserted;

– The entry does not contain a CName attribute descriptor;

– The intent of the CName approximate matching algorithm is to allow the CName attribute to be supported for those End-points which require it and a minimal impact for those End-points which do not require it. The second approximate match condition above allows the N_Port which does not restrict communication to be considered capable of communicating with any C-Name provided in the Search Filter;

– If a DS does not support registration of the CName attribute, it shall still allow a filter to contain the CName attribute. The above approximate matching algorithm will still have the desired effect of allowing the entry which does not contain the CName to pass through the filter;

– This filter item may be applied only to attributes which are of type Communicating Name.

7.2.3.2 Default environment filter restrictions

In a default environment, certain restrictions are placed on the Filter parameters. These restrictions represent the minimal set of functions which a DS is required to support. The Query Capabilities command is used to determine whether the Directory is capable of advanced function (see 7.7.1)

The following restrictions apply in a default environment:

– A maximum of 1 filter item may be specified in a single query;

– The only kinds of filter items allowed are equality and present.

7.2.4 PagedResultsRequest

```
PagedResultsRequest ::= CHOICE{
    pageSize       INTEGER,
    queryReference OCTET STRING}
```

A PagedResultsRequest parameter is used by the DA to request that the results of a Search operation be returned to it “page-by-page”. It requests the DS to return only a subset - a page - of the results of the operation, in particular the next pageSize subordinates or entries, and to return a queryReference which can be used to request the next set of results on a follow-up query. Although a DA may request paged results, a DS is permitted to ignore the request and return its results in the normal manner.

For a new Search operation, the PagedResultsRequest is set to pageSize, which specifies the maximum number of entries to return in the results. The DS shall return up to but not more than the requested number of entries. The sizeLimit, if any, is ignored.

For a follow-up request, i.e., to request the next set of paged results, the DA makes the same Search request as before but sets pagedResultsRequest to queryReference, with the value of this parameter the same as that returned in the previous search results. The DA has no understanding of the queryReference, which is available to a DS to use as it wishes to record context information for the query. The DS uses this information to determine which results to return next.

NOTE – If the DIB changes between search requests, the DA may not see the effects of these changes. This is implementation dependent. A query reference remains valid even if a DA begins a new Search operation. A DA may request paged results with several queries and then return to an earlier query and request the next page of results using the query reference supplied for it. The number of “active” query references to which a DA can return is a local DS implementation option, as is the lifetime of those query references.

7.2.5 CommonResults

```
CommonResults ::= SEQUENCE{
directoryAliasDereferenced [30] IMPLICIT
                                  Boolean
                                  DEFAULT false}
```

The commonResults described below shall be provided with each Directory response to a retrieval request.

– DirectoryAliasDereferenced: The DirectoryAliasDereferenced component is set to TRUE whenever the directory server dereferences any alias during a search operation. This is true regardless of whether the dereferenced alias object passes the specified filter.

7.2.6 EntryInformation

```
EntryInformation ::= SEQUENCE{
  name        [27] IMPLICIT DistinguishedName,
  information[28]  IMPLICIT SEQUENCE OF
                            FCattribute
                             OPTIONAL}
```

The entryInformation parameter described below shall be contained in successful retrieval responses provided by the Directory.

– name: The Distinguished Name of the entry is always included;

– information: A set of FCattributes is returned, based on the requirements of the EntryInformationSelection. The RDN is not returned as part of information because it is returned in the name portion of the EntryInformation. If the only attribute requested is the RDN, then no information parameter is returned.

NOTE – Root is not considered a qualified entry.

7.3 Compare operation

```
CompareRequest ::=
    SEQUENCE{
      invokeID  [0] IMPLICIT InvokeID,
      object    [1] IMPLICIT
                    DistinguishedName,
      purported [2] FCattribute,
      COMPONENTS OF CommonArguments}

CompareResult ::=
    SEQUENCE{
      invokeID  [0] IMPLICIT InvokeID,
      name      [1] IMPLICIT DistinguishedName
                                 OPTIONAL,
      matched   [2] IMPLICIT Boolean
                    DEFAULT true,
      COMPONENTS OF CommonResults}
```

A Compare operation is used to compare a value (which is supplied as an argument of the request) with the value(s) of a particular attribute descriptor in a particular object entry.

7.3.1 Compare request

The following components comprise the Compare Request Argument:

– invokeID: The InvokeID component is a unique identifier for the operation;

– object: The object component provides the Distinguished Name of the object entry with which the compare is associated. Any directory-aliases contained in the name are dereferenced unless this is prohibited by the relevant Service Controls;

– purported: The purported component identifies the attribute descriptor and value to be compared with that in the entry;

– CommonArguments: CommonArgumentsare included with the Compare request (see 7.2.1).

7.3.2 Compare result

If the Compare request is successful (i.e., the compare is actually carried out), the Compare Result is returned, containing the following parameters:

– invokeID: The InvokeID component is a unique identifier for the operation;

– name: The DistinguishedName component is present if a directory-alias was dereferenced and represents the distinguished name of the object itself;

– matched: The matched parameter holds the result of the comparison. The parameter takes the value TRUE if the values were compared and matched, and FALSE if they did not;

– CommonResults: CommonResults, are included with the Compare response (see 7.2.5).

7.3.3 Errors

Should the request fail, an error shall be reported (see clause 8).

7.4 Abandon operation

The Compare and Search operations may be abandoned using the Abandon operation if the user is no longer interested in the result.

Note that, if an error is found in the Abandon Request by the DS, the only allowed error response is Abandon Failed.

7.4.1 Abandon request

The Abandon Request consists of a single argument, the InvokeID, which identifies the operation that is to be abandoned. The value of the InvokeID is the same as that of the InvokeID which was used to invoke the operation which is to be abandoned.

7.4.2 Abandon result

Should the request succeed, a result shall be returned, with the InvokeID as the single argument.

7.5 Search operation

```
  SearchRequest ::=
     SEQUENCE{
        invokeID  [0] IMPLICIT InvokeID,
        baseobject [1] IMPLICIT
                     DistinguishedName,
        subset    [2] IMPLICIT INTEGER{
                        baseObject(0),
                        oneLevel(1),
                        wholeSubtree(2)}
                        DEFAULT wholeSubtree,
        filter    [3] Filter,
        passBaseObject [4] IMPLICIT Boolean
                                DEFAULT true,
        searchDirectoryAliases [5] IMPLICIT
                                 Boolean
                                 DEFAULT true,
        selection [6] EntryInformationSelection,
        pagedResults [7] PagedResultsRequest
                        OPTIONAL,
        COMPONENTS OF CommonArguments}

  SearchResult ::=
     SEQUENCE{
      invokeID  [0] IMPLICIT InvokeID,
      object    [1] IMPLICIT DistinguishedName
                                OPTIONAL,
      entries   [2] IMPLICIT SEQUENCE OF
                                EntryInforma-
tion,
      limitProblem [3] IMPLICIT LimitProblem
                     OPTIONAL,
      queryReference [4] OCTET STRING OPTIONAL,
      COMPONENTS OF CommonResults}

LimitProblem ::= INTEGER{
     timeLimitExceeded (0),
     sizeLimitExceeded (1),
     administrativeLimitExceeded (2)}
```

A Search operation is used to search a portion of the DIT for entries of interest and to return selected information from those entries. Under some circumstances, the list returned may be incomplete.

7.5.1 Search request

The following components comprise the Search Request:

– invokeID: The InvokeID component is a unique identifier for the operation;

– baseObject: The baseObject component provides the Distinguished Name of the object entry (or possibly the root) relative to which the search is to take place. Any directory-aliases contained in the name are dereferenced unless this is prohibited by the relevant Service Controls;

– subset: The subset argument indicates whether the search is to be applied to;

  – the base object only;

  – the immediate subordinates of the base object only (unless inclusion of the base object is specified by passBaseObject=true);

  – the base object and all its subordinates;

– filter: The filter argument is used to eliminate arguments from the search space which are not of interest. Information shall only be returned on entries which satisfy the filter (see 7.2.3);

– passBaseObject: The passBaseObject argument is only meaningful when a Filter is present. It indicates that the baseObject entry shall be passed through the Filter and considered among other entries which pass through the Filter when satisfying the specifications of the EntryInformationSelection parameter (see 7.2.2);

– searchDirectoryAliases: The searchDirectoryAliases argument specifies whether subordinates of the base object shall be dereferenced during the search. If searchDirectoryAliases is set to TRUE, directory-aliases shall be dereferenced and the search shall continue in the subtree of the aliased object. In this case, serves merely as a pointer and is not included in the search space. If searchDirectoryAliases is FALSE, the Directory-alias entry is not dereferenced.

NOTE – This parameter is not affected by the value of the dereferenceAliases parameter in the Service controls. The dereferenceAliases parameter refers to the entry at which the search is started, while the searchDirectoryAliases parameter refers to the subordinates of the starting point of the search.

– selection: The selection argument indicates what information from the entries is requested. It is specified in terms of the EntryInformationSelection parameter (see 7.2.2);

– pagedResults: The pagedResults argument is used to request that results of the op-

eration be returned page-by-page (see 7.2.4);

– commonArguments: CommonArguments are included with the Search request (see 7.2.1).

7.5.2 Search result

The Search operation is considered to be successful if the specified object is located regardless of whether there is any subordinate information to be returned. If the Search request is successful, the Search Result is returned, containing the following parameters:

– invokeID: The invokeID component is a unique identifier which matches the invokeID provided in the Search request;

– object: The object parameter (a DistinguishedName) is present if the baseObject entry was dereferenced. It represents the distinguished name of the aliased baseObject;

NOTE – A DA can use the distinguished name returned in the object parameter to access the object directly on subsequent directory requests.

– entries;

– The entries parameter contains the requested information from each entry (zero or more) which satisfied the filter. The information is specified in terms of the EntryInformation parameter (see 7.2.6);

– If an entry is reached more than once in the search (e.g., directly and via a directory-alias), the Directory Server shall not return duplicate information to the Directory Agent. Only one set of information per unique entry shall be returned;

– limitProblem: The limitProblem parameter indicates whether the time limit, the size limit or an administrative limit has been exceeded. The results being returned are those which were available when the limit was reached;

– queryReference: The queryReference parameter shall be present when the DA has requested paged results and the DS has not returned all of the available results;

– CommonResults: CommonResults are included with the Search response (see 7.2.5).

7.5.3 Errors

Should the request fail, an error shall be reported (see clause 8).

7.5.4 Recommended search formats

In order to obtain meaningful information from the Directory, certain restrictions are placed of the format of Directory Search requests, as described in the sections below.

When considering the Search parameters, it is helpful to think of the Filter and EntryInformationSelection parameters as follows:

– Filter: The Filter specifies the attribute(s) that the DA provides to the DS for the DS to match against existing entries. Essentially, the DA is only interested in those entries which have the characteristics specified by the Filter;

– EntryInformationSelection: After applying the Filter and discarding any entries which do not match the Filter, the DS uses the EntryInformationSelection parameter provided by the DA to determine which of the entry's attributes the DA wants returned;

For example, if a DA is interested in those N_Ports which support IP, the Filter would specify IP and the EntryInformationSelection would specify N-PortID.

7.5.4.1 Search for N_Ports associated with an End-Point

In order to properly identify the N_Port(s) associated with an End-Point, the Search parameters described below shall be set as specified. The specifications are minimum requirements. Other filter items or selection items (including allAttributes) may be specified in addition to those described below.

– filter: The filter shall specify the c-name (end-point name identifier) of the requester, as an approximate match. Whenever c-name is specified in a filter it shall be as an approximate match;

– passBaseObject: passBaseObject shall be specified as true if the IPA or any other End-Point attributes are being requested in the EntryInformationSelection;

– searchDirectoryAliases: searchDirectoryAliases shall be set to true;

– selection - If the End-Point uses the IPA, selection shall specify Initial Process Associator (in addition to N_Port ID and/or N_Port Name).

7.6 Directory modify operations

Three operations exist for modifying the Directory:

– Add Entry;

– Remove Entry;

– Modify Entry.

In each modification request, the target entry is identified by means of its distinguished name.

7.6.1 Add entry

```
AddEntryRequest ::=
     SEQUENCE{
        invokeID         [0] IMPLICIT InvokeID,
        primary          [1] IMPLICIT
                             PrimaryEntry,
        subordinateList  [2] IMPLICIT SEQUENCE OF
                        SubordinateEntry OPTIONAL,
        COMPONENTS OF CommonArguments}

PrimaryEntry ::= SEQUENCE{
  object[3] IMPLICIT DistinguishedName,
  entry [4] IMPLICIT SEQUENCE OF FCattribute
            DEFAULT{}}

SubordinateEntry ::= SEQUENCE{
  object [5] Rdn,
  entry  [6] IMPLICIT SEQUENCE OF FCattribute
            DEFAULT{}}
```

The Add Entry operation is used to add one or more entries to the DIT.

The Add Entry operation is atomic; that is, either all specified entries or no entries are added to the DIB. Each entry may be an object or a directory- alias, with the condition that a directory-alias entry shall be a leaf node.

Entries are added by specifying a “group” of entries to be added. A group consists of a primary entry and zero or more immediate subordinates to that entry. The primary entry is specified by a DistinguishedName, which indicates where on the existing DIT the group is to be added. The primary entry may represent one of two options:

– A Distinguished Name which already exists on the DIT. If the primary entry represents a Distinguished Name which already exists, then the intent of the AddEntry operation is to add one or more immediate subordinates to the specified entry (primary entry);

– A Distinguished Name which does not exist on the DIT;

– If the primary entry represents a Distinguished Name which does not exist, then the intent of the Addentry operation is to add one primary entry and zero or more immediate subordinates of the primary entry;

– In this case, if the last RDN of the primary entry’s Distinguished Name is removed, the remaining Distinguished Name specifies an existing entry (which must exist for the operation to succeed) on the DIT to which the primary entry is added as an immediate subordinate.

A subordinate entry is specified by its RDN and is added to the DIT as an immediate subordinate of the primary entry.

For example, a group may consist of an EndpointName primary entry and several subordinate N_PortIDs. Because the Distinguished Name of the primary entry is a single RDN, it becomes a subordinate of root. If the End-pointName does not exist in the DIT, it is added, along with the subordinate N_PortIDs. If the End-pointName exists, only the subordinate N_PortIDs are added to the DIT.

Aliases are never dereferenced by this operation.

The Directory shall ensure that all entries conform to the Directory schema. Where the entry being created is a directory-alias, no check is made to ensure that the Aliased Object Name attribute points to a valid entry.

If an attempt is made to add an entry with an empty subordinateList and the entry already exists, the DS shall recognize an Update Error with the reason of entryAlreadyExists. The same error shall be recognized if the primary entry already exists and one or more subordinate entries in the subordinateList already exist.

7.6.1.1 Add entry argument

The following components comprise the Add Entry Argument, provided with the Add Entry request.

– invokeID: The InvokeID parameter is a unique identifier for the operation;

– primary: The primary parameter identifies the primary entry which either exists or is to be added to the DIT. It is specified by two parameters;

– object: The object argument identifies the Distinguished Name of the primary entry;

– entry: The entry argument contains the attribute information which, together with that from the RDN of the primary Distinguished Name, constitutes the entry to be created. If the primary entry already exists in the DIT, the entry parameter should not be present. If present, it shall be ignored by the DS;

– subordinateList: The subordinateList contains the list of entries to be added as immediate subordinates of the primary entry. Each subordinate entry is specified by two parameters (in SubordinateEntry);

– object: The object argument identifies the Relative Distinguished Name which, when concatenated to the Distinguished Name of the primary entry, provides the Distinguished Name of the subordinate entry to be added;

– entry: The entry argument contains the attribute information which, together with that from the RDN, constitutes the entry to be created;

– CommonArguments: CommonArguments are described in 7.2.1.

7.6.1.2 AddEntryResult

Should the request succeed, a result shall be returned, with the InvokeID as the single argument.

7.6.1.3 Errors

Should the request fail, an error shall be reported. (see clause 8).

7.6.2 Remove Entry

```
RemoveEntryRequest ::=
     SEQUENCE{
        invokeID   [0] IMPLICIT InvokeID,
        object     [1] IMPLICIT
                       DistinguishedName,
        COMPONENTS OF CommonArguments}
```

The Remove Entry operation is used to remove an entry from the DIT. Multiple entries may be removed by specifying a non-leaf entry for removal, which also causes all subordinate entries to be removed. The Remove Entry operation is atomic; that is, either all entries or no entries are removed from the DIB. The entry specified may be an object or a directory-alias.

Aliases are never dereferenced by this operation.

7.6.2.1 Remove entry argument

The following components comprise the Remove Entry argument.

– invokeID: The InvokeID component is a unique identifier for the operation;

– object: The object argument identifies the Distinguished Name of the entry to be removed. All subordinates are also removed;

– CommonArguments: CommonArguments are described in 7.2.1.

7.6.2.2 Remove entry result

Should the request succeed, a result shall be returned, with the InvokeID as the single argument.

7.6.2.3 Errors

Should the request fail, an error shall be reported, as described in 7.3.3.

7.6.3 Modify entry

```
ModifyEntryRequest ::=
     SEQUENCE{
        invokeID   [0] IMPLICIT InvokeID,
        object     [1] IMPLICIT
                       DistinguishedName,
        changes    [2] IMPLICIT SEQUENCE OF
                        EntryModification,
        COMPONENTS OF CommonArguments}

EntryModification ::= CHOICE{
  addAttribute     [0] FCattribute,
  removeAttribute  [1] FCattribute}
```

The Modify Entry operation is used to perform a series of one or more of the following modifications to a single entry:

– add a new attribute;

– remove an attribute;

– replace attribute values;

– modify a directory-alias.

Aliases are never dereferenced by this operation.

7.6.3.1 Modify entry argument

The Modify Entry argument contains the following:

– invokeID: The invokeID component is a unique identifier for the operation;

– object: The object argument identifies the Distinguished Name of the entry to be modified;

– changes: The changes argument defines a sequence of modifications, which are applied in the order specified. If any of the individual modifications fails, then an Attribute Error shall be generated and the entry left in the state it was prior to the operation. That is, the operation is atomic. If an attempt is made to modify the object class attribute, an update error is returned. The end result of the sequence of modifications shall not violate the Directory schema. The following types of modification may occur;

– Add Attribute: This identifies a new attribute to be added to the entry. An attempt to add an already existing attribute value results in an error;

– Remove Attribute: The argument identifies (by its descriptor) an attribute to be removed from the entry. Any attempt to remove a non-existing attribute results in an Attribute Error;

– Attribute values may be replaced using a combination of Remove Attributes and Add Attributes in a single Modify Entry operation. If the ModifyEntry operation attempts to modify an entry's RDN, the DS ensures that the RDN is left in a valid state. Otherwise (if the RDN is removed, for example) the operation is not performed and the DS returns an error response;

– CommonArguments: CommonArguments are described in 7.2.1.

7.6.3.2 Modify entry result

Should the request succeed, a result shall be returned, with the invokeID as the single argument.

7.6.3.3 Errors

Should the request fail, an error shall be reported, as described in 7.3.3.

7.7 Query Capabilities Operation

```
QueryCapResult ::=
     SEQUENCE{
        invokeID       [0] IMPLICIT InvokeID,
        versionNumber  [1] IMPLICIT INTEGER,
        numFilterItems [2] IMPLICIT INTEGER,
        matchTypes     [3] IMPLICIT
                           MatchSelection,
        entryLimit [5] IMPLICIT
                                INTEGER,
        pagedResults [6] IMPLICIT Boolean,
        unsupportedAttributes [7]
                  IMPLICIT SEQUENCE OF
                  AttributeDescriptor,
       queryReferenceLife [8) IMPLICIT INTEGER,
        timeLimitUsage [9] IMPLICIT INTEGER}

MatchSelection ::= BIT STRING{
     equality                (0),
     substrings              (1),
     greaterOrEqual          (2),
     lessOrEqual             (3),
     present                 (4),
     approximateMatch        (5)}
```

The Query Capabilities operation allows a DA to determine the set of parameters being used by the DS which describes its operating environment.

7.7.1 Query capabilities result

The Query Capabilities result is transported in a response IU and contains the following:

– invokeID: The InvokeID component is a unique identifier for the operation;

– versionNumber: The versionNumber parameter specifies the version of Directory Services architecture which is implemented. The current value of this field is 1;

– numFilterItems: The numFilterItems parameter specifies the maximum number of filter items which a DA may provide on a Search request;

– matchTypes: The matchTypes parameter specifies which of the CHOICE of FilterItems is allowed (see 7.2.3);

– entryLimit: The administrativeLimit parameter specifies the maximum number of entries which the Directory will return on a Directory search or accept on a Directory modification;

– pagedResults: The pagedResults parameter specifies whether the DS supports paged results. The TRUE setting indicates that paged results are supported;

– unsupportedAttributes: The unsupported Attributes parameter specifies those attributes of which the Directory does not support registration. Only those attributes which are not mandatory in any object classes may be unsupported by a Directory Server.

7.8 Usage of directory aliases

It is recommended that a Directory Agent only add a directory-alias for an object which it owns. That is, the object to which the directory-alias points must have been added by the same DA which is adding the directory-alias. Similarly, when the object to which the directory-alias points is removed, the DA should appropriately update any affected directory- aliases.

8 Directory service error

8.1 Error recovery at the directory agent

The directory agent performs two types of error detection.

8.1.1 Response error

When a response received from a directory server contains any kind of error, the directory agent may retry the operation.

8.1.2 Operation timeout error

If a directory agent does not receive a response IU from a directory server within the operation timeout duration following the most recent transmission of a request IU, the directory agent shall detect an operation timeout error.

The duration of the operation timer is implementation-dependent but shall be larger than R_A_TOV, a timer which is described in FC-PH.

8.2 Error recovery at the directory server

8.2.1 Overview

When a DS detects an error, it returns one of the responses defined in the following CHOICE:

```
Error ::=
     IMPLICIT SEQUENCE{
        invokeID   [0]   IMPLICIT InvokeID,
        error      [1]   ErrorChoice}

ErrorChoice ::= CHOICE{
        abandoned        [0] IMPLICIT Abandoned,
        abandonFailed    [1] IMPLICIT
                             AbandonFailed,
        nameError        [2] IMPLICIT NameError,
        updateError      [3] IMPLICIT
                             UpdateError,
        attributeError   [4] IMPLICIT
                             AttributeError,
        protocolError    [5] IMPLICIT
                             ProtocolError,
        serviceError     [6] IMPLICIT
                             ServiceError}
```

Descriptions of these errors are provided in the following sections.

8.2.2 Error precedence

The Directory does not continue to perform an operation beyond the point at which it determines that an error is to be reported.

Note that an implication of this rule is that the first error encountered can differ for repeated instances of the same query, as there is not a specific logical order in which to process a given query. For example, DSs may be searched in different orders

Should the Directory simultaneously detect more than one error, the following list determines which error is reported. An error higher in the list has a higher logical precedence than the one below it, and is the error which is reported:

– NameError;

– UpdateError;

– AttributeError;

– ServiceError;

– ProtocolError.

The following errors do not present any precedence conflicts:

– AbandonFailed: This does not present any precedence conflicts because it is specific to one operation, Abandon, which can encounter no other error;

– Abandoned: Abandoned is not reported if an Abandon operation is received simultaneously with the detection of an error. In this case and AbandonFailed error, reporting the problem noSuchOperation is reported along with the report of the actual error encountered.

8.2.3 Abandoned

```
Abandoned ::= NULL
```

The Abandoned error is reported in response to an outstanding Directory enquiry operation (i.e., Compare, Search) if the Abandon operation is performed for that operation.

Note that there are two responses associated with an Abandon operation. The first response indicates success or failure of the Abandon request. The second response (described here) is to the operation which has been abandoned, indicating that the Abandon was performed.

The Abandoned error is not literally an “error”. There are no parameters associated with this error.

8.2.4 Abandon failed

```
AbandonFailed ::=
     SEQUENCE{
        operation  [0]   InvokeID,
        problem    [1]   IMPLICIT
                         AbandonProblem}

AbandonProblem ::=
     INTEGER{
        noSuchOperation  (1),
        cannotAbandon    (2)}
```

The AbandonFailed error reports a problem encountered during an attempt to abandon an operation. This is the only error indication allowed in response to an Abandon operation.

The various parameters have the following meanings:

– operation: The operation parameter contains the InvokeID of the particular operation to be abandoned;

– problem: Any of the following problems may be indicated;

– noSuchOperation: This problem is indicated when the Directory has no knowledge of the operation which is to be abandoned. (This could be because no such invoke took place or the operation has already been responded to);

– cannotAbandon: This problem is indicated when an attempt has been made to abandon an operation for which this is prohibited (e.g., modify), or the abandon could not be performed.

8.2.5 Attribute error

```
AttributeError::=
     SEQUENCE{
        object           [0] DistinguishedName,
        attribute        [1] AttributeSpecifier,
        description       [2] AttributeProblem}

AttributeSpecifier ::=
     CHOICE{
        attributeDescriptor AttributeDescriptor,
        fcAttribute              FCAttribute}

AttributeProblem ::=
     INTEGER{
        noSuchAttribute          (1),
        invalidSyntax            (2),
        undefinedAttribDescriptor(3),
        inappropriateMatching    (4),
        constraintViolation      (5),
        attributeOrValueExists   (6),
        attributeNotSupported    (7)}
```

The various parameters have the following meanings:

– object: The object parameter identifies the entry to which the operation was being applied when the error occurred;

– attribute: The attribute parameter specifies the attribute which contains the error. If the error is associated with an attribute descriptor, then only the attribute descriptor shall be returned. If the error is associated with an attribute value, then the associated FCAttribute (which contains both descriptor and value) shall be returned;

– description: The description parameter provides details on the type of error detected;

– noSuchAttribute: This error indicates that the named entry lacks one of the attributes specified as an argument of the operation;

– invalidSyntax: This error indicates that the value of the attribute which was specified as an argument of the operation does not conform to the attribute syntax for that attribute;

– undefinedAttribDescriptor: This error indicates that the attribute descriptor provided as an argument for this operation is undefined;

– inappropriateMatching: This error indicates that an attempt was made, e.g., in a filter, to use a matching rule not defined for the attribute type concerned;

– constraintViolation: This error indicates that an attribute value supplied in the argument of an operation does not conform to constraints imposed by the attribute definition (e.g., the value exceeds the maximum size allowed);

– attributeOrValueExists: This error indicates that an attempt was made to add an attribute or value which already existed in the entry;

– attributeNotSupported: This error indicates that an attempt was made to specify an attribute which is optional, and is not supported by this Directory Server. This response is only used if the requested opera-

tion involves only unsupported attributes. If other attributes are present in the request, the operation is partially performed for all supported attributes specified.

8.2.6 Name Error

```
NameError::=
     SEQUENCE{
        problem    [0]    NameProblem,
        matched    [1]    DistinguishedName}

NameProblem ::=
     INTEGER{
        noSuchObject             (1),
        aliasProblem             (2),
        invalidAttributeSyntax   (3),
        aliasDereferencingProblem(4)}
```

A NameError reports a problem related to the name provided as an argument to an operation. The various parameters have the following meanings:

– problem: The problem parameter describes the particular problem encountered. Any of the following may be indicated;

  – noSuchObject: This error indicates that the name supplied does not match the name of any object;

  – aliasProblem: This error indicates that an alias has been dereferenced which names no object;

  – invalidAttributeSyntax: This error indicates that an attribute descriptor and its associated attribute value in the name are incompatible;

  – aliasDereferencingProblem: This error indicates that an alias was encountered in a situation where it was not allowed;

– matched: The matched parameter contains the name of the lowest entry (object or alias) in the DIT that was matched, and is a truncated form of the name provided or, if an alias has been dereferenced, of the resulting name.

8.2.7 ProtocolError

```
ProtocolError::= NULL
```

A ProtocolError response is used when a general error is detected which is not covered by any other error categories. This includes errors such as un undefined Operation in OperationsAndResults and an invalid subset under Search Request.

8.2.8 Service Error

```
ServiceError::=
     INTEGER{
      busy                          (1),
      unavailable                   (2),
      timeLimitExceeded             (3),
      administrativeLimitExceeded(4),
      ditError                      (5),
      invalidQueryReference(6)}
```

A ServiceError reports a problem related to the Directory's ability to provide the requested service. The error has a single parameter which reports the particular problem encountered. The following problems may be indicated:

– busy: This error indicates that the Directory, or some part of it, is presently too busy to perform the requested operation, but may be able to do so after a short while;

– unavailable: This error indicates that the Directory, or some part of it, is currently unavailable;

– timeLimitExceeded: This error indicates that the Directory has reached the limit of time set by the user in a service control. No partial results are available to return to the user;

– administrativeLimitExceeded: This error indicates that the Directory has reached some administrative limit and no partial results are available to return to the user;

– ditError: This error indicates that the Directory is unable to accomplish this request due to a DIT consistency problem;

– invalidQueryReference: This error indicates that the queryReference in paged results is invalid.

8.2.9 Update Error

```
UpdateError ::=
     INTEGER{
      namingViolation            (1),
      objectClassViolation       (2),
      notAllowedOnRdn            (3),
      entryAlreadyExists         (4),
      objectClassModification    (5),
      notAllowedOnMandatoryAttribute(6)}
```

An UpdateError reports a problem related to attempts to add, delete or modify information in

the DIB. The error has a single parameter, which reports the particular problem encountered. The following problems may be indicated:

– namingViolation: This error indicates that the attempted addition or modification would violate the structure rules of the DIT as defined in the Directory schema (e.g., an attempt to place an entry as the subordinate of an alias entry);

– objectClassViolation: This error indicates that the attempted update would produce an entry inconsistent with the definition provided by its object class;

– notAllowedOnRDN: This error indicates that the attempted update would affect the RDN (e.g., removal of the RDN attribute);

– entryAlreadyExists: This error indicates that an attempted AddEntry operation names an entry which already exists;

– objectClassModification: This error indicates that an operation attempted to modify the object class attribute;

– notAllowedOnMandatoryAttribute: This error indicates that the attempted update would affect a mandatory attribute (e.g., removal of a mandatory attribute).

NOTE – The UpdateError is not used to report problems with attribute types, values, or constraint violations encountered in an AddEntry, RemoveEntry or ModifyEntry operation. Such problems are reported via an AttributeError.

8.3 Recovery Actions following FC-PH recovery

8.3.1 Abort exchange (ABTX)

The occurrence of an abort exchange at the FC-PH level has no direct effect on operations at the directory services level. Operations are identified by S_ID and Invoke ID. Therefore, loss of an exchange does not necessarily impact the operation

8.3.1.1 ABTX effect on directory agent

If a directory agent detects that an ABTX has been executed (sent or received), outstanding operations are not affected, unless the ABTX caused the transmission or reception of an IU to be abnormally terminated. If no IUs are affected by the ABTX, the DA simply continues the timer(s) for any outstanding operation(s). If transmission of an IU is abnormally terminated as a result of the ABTX, the DA may retry the operation.

If reception of an IU is abnormally terminated as a result of the ABTX, the DA continues the operation timer. If the IU is successfully received before the operation timer expires, no action is taken. Otherwise, recovery for an operation timeout shall be invoked.

8.3.1.2 ABTX effect on directory server

If a directory server detects that an ABTX has been executed (sent or received), outstanding operations are not affected, unless the ABTX caused the transmission or reception of an IU to be abnormally terminated.

If transmission of an IU is abnormally terminated as a result of the ABTX, the DS shall retry transmission of the IU a model-dependent number of times. When the retries are exhausted (due to any errors), the DS shall terminate the operation. If the operation requested any modification of the DIB, the modification is not made.

If reception of an IU is abnormally terminated as a result of the ABTX, no action is taken.

8.3.2 Abort sequence (ABTS)

The occurrence of an Abort Sequence at the FC-PH level during transmission or reception of an IU causes the following actions by directory agents and servers.

8.3.2.1 ABTS effect on directory agent

If a directory agent detects that an ABTS has been executed (sent or received) during the transmission of an IU, the DA may retry the operation.

If reception of an IU is abnormally terminated by an ABTS, the DA continues the operation timer. If the IU is successfully received before the operation timer expires, no action is taken. Otherwise, recovery for an operation timeout shall be invoked.

8.3.2.2 ABTS effect on directory server

If a Directory Server detects that an ABTS has been executed (sent or received) during the transmission of an IU, the DS shall retry transmission of the IU a model-dependent number of

times. When the retries are exhausted (due to any errors), the DS shall terminate the operation. If the operation requested any modification of the DIB, the modification is not made.

If reception of an IU is abnormally terminated as a result of ABTS, no action is taken.

8.3.3 Stop sequence

The occurrence of stop sequence is treated similarly to abort sequence. When any IU is terminated by stop sequence, its contents are discarded by the recipient.

8.3.3.1 Stop sequence effect on directory agent

If a directory agent detects that a stop sequence has been received during the transmission of an IU, the DA may retry the operation.

If a directory agent detects that a stop sequence has been transmitted during the reception of an IU, the DA continues the operation timer. If the IU is successfully received before the operation timer expires, no action is taken. Otherwise, recovery for an operation timeout shall be invoked.

8.3.3.2 Stop sequence effect on directory server

If a directory server detects that a stop sequence has occurred during the transmission of an IU, the DS may retry the transmission a model-dependent number of times. When the retries are exhausted (due to any errors), the DS shall terminate the operation. If the operation requested any modification of the DIB, the modification is not made.

If a directory server detects that a stop sequence has occurred during the reception of an IU, the IU is discarded and no further action is taken.

9 SNMP based management service

The SNMP based management service is optionally provided within the Fibre Channel system and its sub-systems. Management service covers the following areas:

– configuration management;

– performance management;

– fault management;

– security management;

– accounting management.

9.1 Configuration management

Configuration management deals with initiating and terminating the operation of a certain components or subsystems. It also deals with changing the characteristics of a component or subsystem, and mapping of the topology of the Fabric.

9.2 Performance management

Performance management covers two major categories: monitoring and controlling. The monitoring function collects performance statistics. The controlling function enables performance management to make adjustments to improve the network performance through reconfiguration or capacity planning.

9.3 Fault management

Fault management provides the detection, isolation and correction of abnormal or faulty operation of a Fibre Channel component or subsystem.

9.4 Security management

Security management addresses the security aspects of the Fibre Channel environment. It includes the maintenance of passwords, encryption, authentication, access control, detection and tracking of security violation.

9.5 Accounting management

Accounting management concerns with the usage of resources, and possibly its associated costs by its users.

9.6 SNMP model

The subclause describes how Fibre Channel Management Service is supported using the Simple Network Management Protocol (SNMP). SNMP-based network management systems are widely deployed in the Internet. It is adopted to provide a level of Fibre Channel management service. In this context, the term network is used to mean the Fibre Channel system that includes the Fabric, the N_Ports, and the Nodes.

9.6.1 Overview

Within the SNMP model, management service is provided by a collection of entities: one or

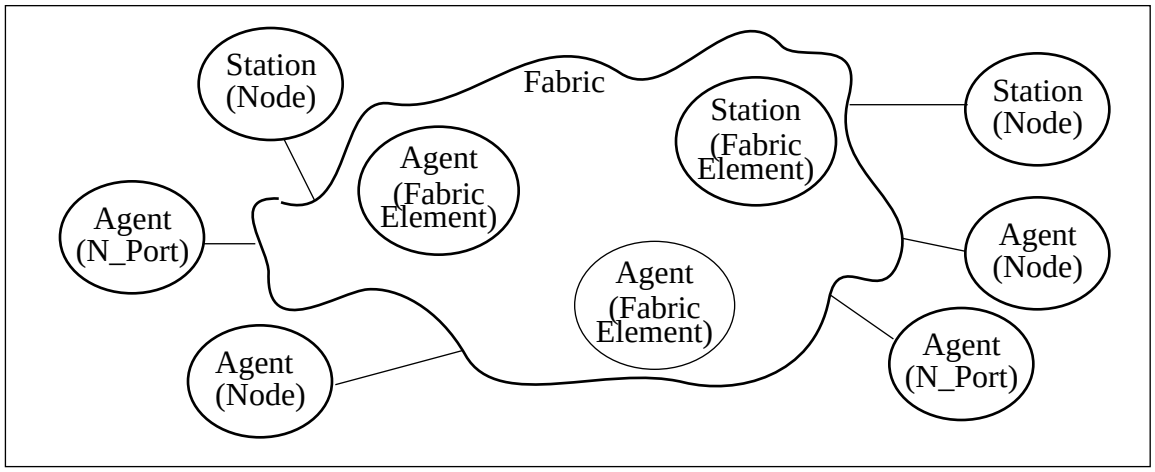


Figure 3 — Functional model of SNMP-based Management system

more management stations, and one or more management agents. The model is illustrated in figure 3. A management station, or simply manager, monitors and control network elements. A management agent, or simply agent, is an entity that represents each network element. A network element may be an N_Port, a Fabric Element or a Node. An agent is responsible for performing the management functions requested by the management stations. The SNMP is used to communicate management information between the management stations and the agents.

In version 1 of SNMP, five operations are defined:

– GetRequest: initiated by a manager to retrieve the value of objects at an agent;

– GetNextRequest: initiated by a manager to retrieve the value of the lexicographical successor to each named object at an agent;

– SetRequest: initiated by a manager to configure the value of objects at an agent;

– GetResponse: generated an agent to respond to a prior GetRequest, GetNextRequest, or SetRequest, from a manager;

– Trap: initiated by an agent to inform a manager of a significant event.

In SNMP version 2, the operation GetResponse is renamed as Response; and two more operations are defined:

– GetBulkRequest: initiated by a manager to retrieve the values of the multiple lexicographical successors of each named object at an agent; it is a more powerful enhancement of GetNextRequest; the corresponding response from the agent is a Response;

– InformRequest: initiated by a manager to request management information to another manager; the responding manager shall transmit a Response.

The flow of the SNMP messages are illustrated in figures 4 and 5.

The resources within the network are represented as objects, each being a simple data variable that is associated with one aspect of the managed agent. The set of objects is referred to as a management information base (MIB).

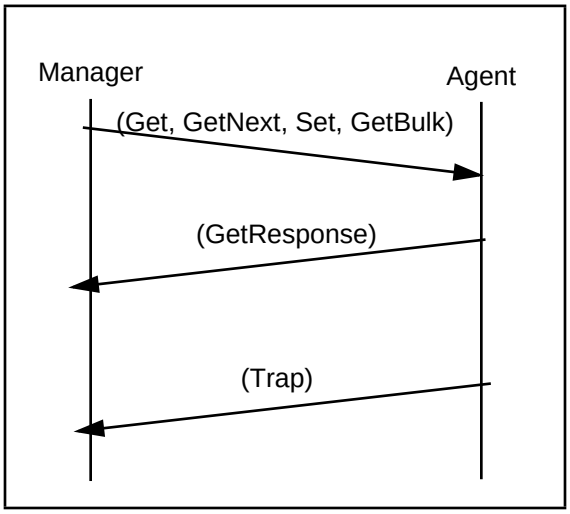


Figure 4 — Message flow between a manager and an agent

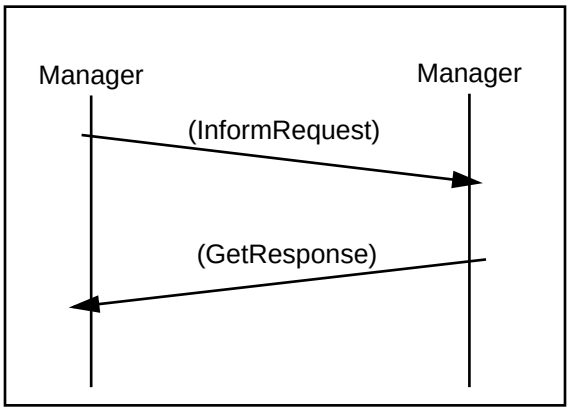


Figure 5 — Message flow between a manager and a manager

The SNMP is defined to be independent of the transport. Within the Internet, it is commonly mapped on top of the User Datagram Protocol (UDP). Refer to RFCs1157 and 1441 for the specification of SNMP, and RFC 768 for the specification of UDP.

For Fibre Channel management service, the UDP mapping is endorsed. However, in order for SNMP to be supported by relatively low cost devices such as disk controllers, a native Fibre Channel mapping, denoted as FC-SNMP, is defined. The protocol mappings are illustrated in figure 6.

9.6.2 UDP mapping

The UDP is the preferred transport mapping in the Internet or TCP/IP environment. This map-

ping is endorsed for early deployment of Fibre Channel in the Internet and legacy internets.

9.7 Native SNMP Mapping

The native mapping of SNMP over FC-PH constructs is defined in this subclause. Each SNMP message is transported as an Information Unit. SNMP information units are not mapped to CT.

9.7.1 Login/Logout

Before any SNMP operation takes place, the N_Ports associated with SNMP entities must have performed the N_Port Login with each other successfully.

9.7.2 Exchanges

Exchanges are used in a unidirectional manner. That is, FC-SNMP information units are sent in only one direction on a given Exchange. For a given pair of SNMP entities, a request and its corresponding response are correlated with the request ID in the request message. The Sequence Initiative of an Exchange shall not be passed except in some error recovery scenarios such as with the use of the Abort Sequence condition. Exchange Reassembly is not allowed.

9.7.3 Information units

An SNMP message is transmitted between a manager and an agent as an Information Unit. The Information Unit construct allows the information to be placed in the Payload of one or more frames within a Sequence.

9.7.4 Class of service

All classes of service (1, 2, 3, 4 and F) are allowed where each is available and applicable.

9.7.5 R_CTL routing bits

Routing bits of the R_CTL field shall indicate FC-4 Device Data.

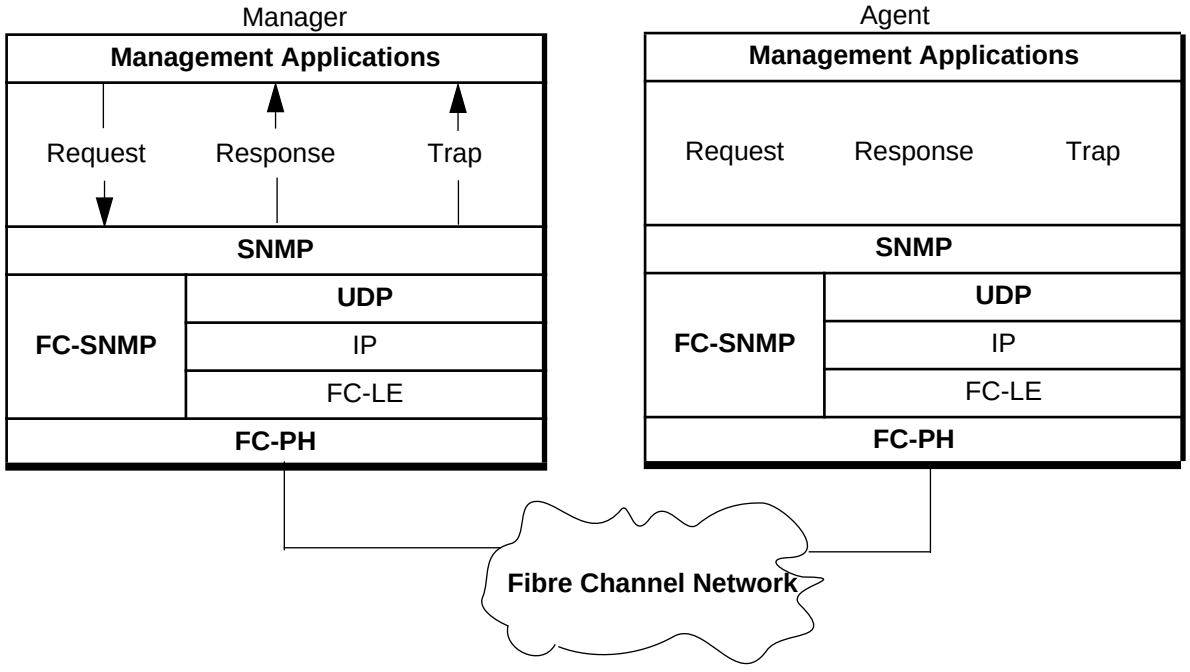


Figure 6 — The SNMP transport mappings

9.7.6 Information category

Table 9 defines the information category according to each SNMP operation message.

Table 9 – Mapping of information categories

SNMP operations	Information Categories
GetRequest	Unsolicited Control
GetNextRequest	Unsolicited Control
GetBulkRequest	Unsolicited Control
InformRequest	Unsolicited Control
SetRequest	Unsolicited Control
GetResponse/Response	Solicited Control
Trap	Unsolicited Data

9.7.7 Sequence initiative

Sequence Initiative shall not be transferred during normal operation.

9.7.8 Destination ID

The destination ID shall be set to that of the Exchange responder.

9.7.9 Source ID

The source ID shall be set to that of the Exchange Originator.

9.7.10 Type

This parameter shall be set to binary 0010 0100 (SNMP).

9.7.11 Relative offset

The relative offset shall not be used.

9.7.12 Error policy

All error policies except Process Policy are allowed.

9.7.13 Expiration/Security header

The use of this optional header is outside the scope of this document.

Note that SNMP version 2 defines security parameters within a message and these security parameters are not related to the Expiration/Security Header.

9.7.14 Network header

The use of this header is beyond the scope of this document and is both implementation and system dependent.

9.7.15 Association header

The use of this header is beyond the scope of this document and is both implementation and system dependent.

9.7.16 Device header

The Device Header shall not be used.

9.7.17 Information unit descriptions

IUs transferred between two SNMP entities are summarized in table 10.

9.8 Management information base

The management information base (MIB) is a virtual database of managed objects, accessible to an agent and manipulated via SNMP to achieve a level of network management. Two sets of Fibre Channel MIB are defined in two Internet drafts:

– *Definitions of Managed Objects for the Fabric Element in Fibre Channel Standard*;

– *Definitions of Managed Objects for the Node in Fibre Channel Standard using SMIv2*.

Note also that there is a set of Internet Standard MIB and a multitude of vendor-specific MIBs.

Table 10 – FC-SNMP Information Units

IU Name	M/O	SI	F/M/L	Sent By
GetRequest	M	H	F/M/L	Manager
GetNextRequest	M	H	F/M/L	Manager
GetBulkRequest	M	H	F/M/L	Manager
InformRequest	M	H	F/M/L	Manager
SetRequest	M	H	F/M/L	Manager
GetResponse/Response	M	H	F/M/L	Agent, Manager
Trap	M	H	F/M/L	Agent

9.9 Agent addressing

The agent is a logical entity that is associated with an N_Port or fabric element. As such, an agent entity shall be addressed by a specific N_Port identifier associated with the fabric element, or the N_Port itself. The means by which the associated specific address is determined is currently beyond the scope of this document.

9.10 Other management models

Another model of management service is based on the well-known address identifier, hex ‘FFFFFA’. The definition of this model is at present not considered.

10 Time service

The time service (TS) is optionally provided. The time server has the well-known address identifier, hex ‘FFFFFB’. Upon a request, the time service shall provide the time information that is sufficient for managing expiration time.

10.1 Functional model

The functional model of time service consists of primarily two entities:

– Time service client: the entity representing a user in accessing the time service;

– Time server: the entity that provides the time information.

There may be more than one physical time server within the Fibre Channel network. However, from a client’s perspective, the time service appears to come from the entity that is accessible at the well-known time service address identifier. if the time service is distributed, it shall be transparent to the clients. Figure 7 illustrates the functional model.

10.2 Basic TS protocol interaction

The basic TS protocol interaction is initiated when the TS client requests time information from the time server by sending the Get_Time request to the time server. The time server then responds with a Get_Time Response.

10.2.1 TS information units

Three different information unit types are associated with the basic TS protocol interaction:

– Get_time request (FS_REQ);

– Get_time response-accept (FS_ACC);

– Get_time response-reject (FS_RJT).

10.3 TS information unit mapping to CT

The information units associated with the basic TS protocol interaction are mapped to the CT for transportation between the TS client and the TS server. FS_IUs are used for time service requests and responses.

10.3.1 CT_HDR

The following values are provided in the CT_HDR:

Revision: X’01’;

IN_ID: N_Port identifier of the TS client;

FCS_Type: X’FB’;

FCS_Subtype: X’01’;

Options (Xbit set to B’0’).

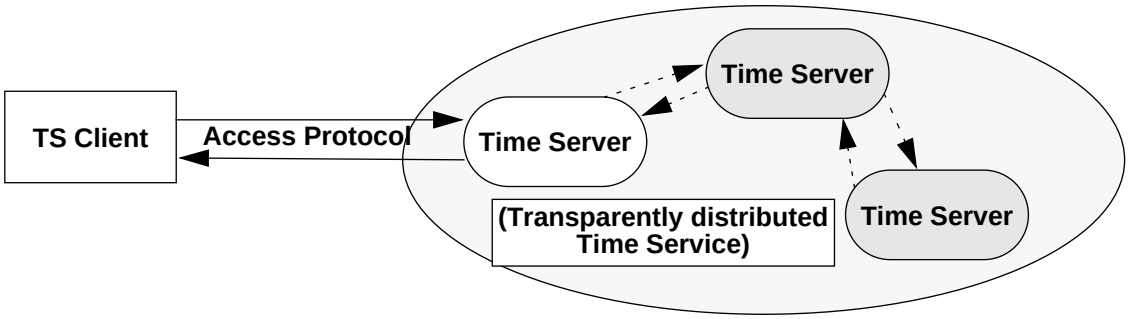


Figure 7 — Functional model of time service

10.3.2 Class of service

All classes of service (1, 2, 3, 4 and F) are allowed where each is available and applicable. However, the same class of service must be used for a pair of related request and reply.

10.3.3 Get_Time request

An FS_REQ IU shall be used by a time service client to request the time information from the time server. An FS_REQ IU is sent from the client to the time server. A command code of X'00B1' designates the get time request. No application information unit is provided for the get time request.

10.3.4 Get_Time response - accept

If the time server was successful in providing the requested time information, then that information is transported to the requesting client using an FS_ACC IU. The application information unit returned in the response is depicted in table 11.

Table 11 – Get_Time response - accept AIU

Item	Size - Bytes
Integer part of the time value	4
Fractional time value	4

The time information consists of two parts:

– The integer part of the time value in seconds relative to the epoch, 0000 universal time (UT) on 1 January 1990;

– and the fractional part of the time value.

The fractional part is optional. If it is not supported, it shall contain the value 0.

10.3.5 Get_Time response - reject

If the time server was not successful in providing the requested time information, or was not able to accept and service the Get_Time request, then the time server sends an FS_RJT IU to the requesting client.

The reason codes are described in 4.6.3.

10.4 Distributed time service

If multiple time servers exists, they shall be synchronized to an accuracy of ±2 seconds, or optionally better. The mechanism and protocol for time distribution and synchronization among multiple time servers are currently not defined.

11 Alias Server

11.1 Alias server

The Alias Server manages the registration and deregistration of Alias IDs for both Hunt Groups and Multicast Groups. The Alias Server is not involved in the routing of frames for any Group.

The Alias Server may be internal or external to the Fabric, but, in either case, it is addressed by means of the well-known address identifier, hex'FFFFF8'. The following sections describe the registration/ de-registration process in more detail.

Authorization for Alias Server operations is provided.

11.1.1 Alias service protocol

Alias registration and de-registration are managed through protocols containing a set of request/reply IUs supported by the Alias Server. These requests and replies use FC-PH constructs as defined in the following sections.

11.1.2 Use of FC-PH constructs

11.1.2.1 Login/Logout

Before performing any Alias Server operation, an N_Port shall perform F_Port Login followed by N_Port Login with the well-known destination address hex 'FFFFF8'. When the N_Port has no further operations pending, it shall perform N_Port Logout with the Alias Server.

11.1.2.2 Exchanges

Alias Services Exchanges shall be used in a bidirectional manner. That is, Alias Server request IUs and reply IUs shall be transferred on the same Exchange, via the passing of Sequence initiative.

11.1.2.3 Information units

The Information Unit construct defines the information transferred as a single Fibre Channel Sequence for Alias Server requests and replies. A single Information Unit contains either an Alias Server request or a reply. All communication occurs through the Exchange of Information Units. This clause describes both the data which are transparent to FC-PH-2 and those control parameters which are required by FC-PH-2.

11.1.2.4 Common required FC parameters

Class of service

The Alias Server shall support all Classes of Service supported by the Fabric Region to which it is attached. See FC-SW for a discussion of Regions. An N_port may communicate with the Alias Server using any desired Class.

If the N_Port or the Alias Server uses Class 3 to communicate, it shall provide the Sequence_Tag on the FC_PH service interface. Any Sequence_Tag provided on a given IU shall not be reused on a subsequent IU associated with the same Exchange until either of the following conditions exists:

– R_A_TOV has expired since the N_Port or the Alias Server has determined that the IU has been transmitted by the FC-2;

– The N_Port or the Alias Server receives a response to the transmitted IU from the receiver of the IU.

R_CTL routing bits

Routing bits of the R_CTL field shall indicate FC-4 Device Data.

Information category

All Request IUs shall specify Unsolicited Control. All Reply IUs shall specify Solicited Control.

Sequence initiative

Sequence Initiative shall be transferred after the transmission of the Request IU, to allow the return of the associated Reply IU (FS_ACC or FS_RJT) on the same Exchange. If Sequence Initiative is not passed on the Request IU, The Recipient shall abort the Exchange.

Destination ID (D_ID)

This parameter shall be set to the well known destination address hex 'FFFFF8'.

Source ID (S_ID)

This parameter shall identify the source address identifier of the IU.

Type

All Alias Server IUs shall specify the Fibre Channel Services TYPE (b'0010 0000').

Error policy

All error policies, with the exception of Process Policy, are permitted.

11.1.2.5 Common optional FC parameters

Expiration/Security header

The use of this parameter is beyond the scope of this document and is both implementation and system dependent.

Network header

The use of this parameter is beyond the scope of this document and is both implementation and system dependent.

Association header

The use of this parameter is beyond the scope of this document and is both implementation and system dependent.

Device header

The Device Header shall not be used.

11.1.2.6 CT_HDR

The CT_HDR, as defined in 4.3.1, shall be supported by the Alias Server. That is, all requests and replies shall contain the CT_HDR. Following is a description of the usage of the various CT_HDR fields.

Revision

This field shall be set to hex ‘01’.

IN_ID

This field shall be ignored by the Alias Server.

FCS_Type

This field shall be set to hex ‘F8’.

FCS_Subtype

This field shall be set to hex’01’.

Options

This field shall be set to hex ‘00’.

Application information unit

This field shall contain the payload of the a single request or reply.

11.1.3 Alias service requests

A Sequence Initiator shall transmit an Alias Server Request to solicit the Alias Server to perform an Alias management function. If an Alias Server Request is transmitted without the transfer of Sequence Initiative, the Alias Server shall abort the Exchange and not perform the Request. The Alias Server Protocol is composed of an Alias Server Request, followed by an Alias Server Reply, on the same Exchange. The following Alias Server Requests are defined:

1. Join Alias Group
2. Remove From Alias Group
3. Listen
4. Stop Listen
5. Read Alias Group

For any of the above Requests, if an FS_RJT is generated, it shall specify a Reason Code of “Unable to perform command request”, unless otherwise indicated. The Reason Code Explanation shall indicate the specific reason for the FS_RJT.

11.1.3.1 Join alias group (JNA)

This request is sent to the Alias Server to cause it to add the passed list of candidate N_Ports to the Alias Group specified by the Alias_Token. The payload of the request contains, among other parameters, the Service Parameters to be used for the Alias Group and a list of the N_Ports to be formed into the new Alias Group. The Originator N_Port may or may not be a member of this list.

If the Alias Group does not exist, it shall be created.

If the Alias Group already exists, it shall be modified as indicated. The request shall be rejected with a Reason Code Explanation of “Unauthorized Request (Invalid Authorization Control)” if the Authorization_Control for the specified Alias Group does not indicate that the Initiator N_Port may add the passed N_Ports to the Alias Group.

A command code of hex’0001’ in the CT_HDR indicates the join alias group request.

The format of the payload is shown in table 12.

Table 12– Join alias group payload

Item	Size (Bytes)
Authorization_PW	12
Authorization_Control	4
Alias_Token	12
Alias_SP	80
NP_List_Length	4
NP_List(1)	4
NP_List (2 to n-1)	(n-2) x 4
NP_List(n)	4

Authorization Password (Authorization_PW): The Authorization_PW provides password protection for modifications to an Alias Group. In conjunction with the Authorization_Control, it shall be used to validate subsequent requests that modify the Alias Group.

When an Alias Group is being created as a result of this request, the Authorization_PW shall be attached to the Alias Group being created.

When an Alias Group with a non-zero Authorization_PW is being modified by this request, the request shall be rejected unless the Authorization_PW matches the Authorization_PW of the Alias Group. The FS_RJT reason code explanation indicates “Unauthorized Request (Invalid Password)”.

An Authorization_PW of all zeroes shall be defined as the universal password. That is, an Alias Group created with an Authorization_PW of all zeroes shall not be password protected. It may be modified, as allowed by the Authorization_Control, without regard to the passed Authorization_PW. The contents of a non-zero Authorization_PW are not defined.

Authorization Control (Authorization_Control): In conjunction with the Authorization_PW, the Authorization_Control determines which N_Ports are authorized to modify the Alias Group. If the passed Authorization_Control is not valid, the request shall be rejected with a Reason Code Explanation of “Invalid Authorization_Control”.

When an Alias Group is being created as a result of this request, the Authorization_Control shall be attached to the Alias Group being created. In conjunction with the Authorization_PW, it is used to validate subsequent requests that modify the Alias Group.

When an Alias Group is being modified by this request, this field shall be ignored.

Authorization_Control has the format defined in table 23.

Alias Group Token (Alias_Token): The Alias_Token defines the Alias Group being created. The request shall be rejected if the Alias_Token is invalid (Reason Code Explanation of “Invalid Alias_Token”), or the Alias Group specified in the Flags is not supported (Reason Code Explanation of “Unsupported Alias_Token”). The format of the Alias_Token is described in Table 21.

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 in the format defined in FC-PH, although only the Common Service Parameters and the appropriate Class Service Parameters are actually used.

NOTE – These Service Parameters may differ from those passed during Login.

If a Multicast Group is being created, only the Class 3 Service Parameters are applicable and the Class 3 Validity bit shall be set. Otherwise, the request shall be rejected with a Reason Code Explanation of “Alias Group cannot be formed (Invalid Class)”.

If a Hunt Group is being created, the Service Class Parameters may indicate any Class that is supported by all members of the Hunt Group.

These Service Parameters are used to perform an implicit Login among the members of the Group.

If an attempt is made to Join an existing Alias group and the passed Service Parameters are in conflict with the Service Parameters of the existing Alias Group, then this request shall be rejected with a Reason Code Explanation of “Alias Group cannot be joined (Service Parameter conflict)”.

N_Port List Length (NP_List_Length): The NP_List_Length specifies the number of entries in the following NP_List.

N_Port List (NP_List): The NP_List contains one entry for each N_Port address identifier to be included in the Alias Group. The N_Port address identifier shall be right-aligned within the NP_List entry. That is, the high-order byte of the entry shall be ignored and the low-order 3 bytes shall contain the N_Port address identifier. The N_Port address identifier shall not be an Alias_ID.

When an Alias Group is being created as a result of this request, an FS_ACC shall be returned indicating that the Alias Group has been formed and an alias address identifier has been assigned, by the Fabric, for the Alias Group identified by the Alias_Token.

When an Alias Group is being modified as a result of this request, an FS_ACC shall be returned indicating that a valid Alias Group is being modified and that the Service Parameters are compatible.

In either case, the FS_ACC does not necessarily indicate that any of the listed N_Ports were actually formed into an Alias Group. A “Read Alias Group” request may be issued, or the Directory Server may be queried to determine which N_Ports were actually formed into the Alias Group. The format of the FS_ACC payload is shown in table 13.

Table 13– Join alias group accept payload

Field	Size (Bytes)
Alias_Token	12
Alias_ID	4
Alias_SP	80

Alias Group Token (Alias_Token): The Alias_Token defines the Alias Group which was just created. It is the same Alias_Token as was passed in the request. The format is described in table 21.

Alias Group Identifier (Alias_ID): This is the alias address identifier that is associated with the Alias Group, as assigned by the Fabric. The alias address identifier shall be right-aligned within the Alias_ID field. That is, the high-order byte of the entry shall be ignored and the low-order 3 bytes shall contain the alias address identifier.

Alias Group Service Parameters (Alias_SP):

The Alias_SP returns the Service Parameters in effect for the Alias Group indicated by the Alias_Token.

An FS_RJT shall also be returned if an Alias_ID could not be assigned by the Fabric. The FS_RJT Reason Code Explanation indicates the appropriate Reason Code Explanation from table 22.

11.1.3.2 Remove from alias group (RMA)

This request is sent to the Alias Server to cause it to delete the N_Ports in the passed list from the existing Alias Group defined by the passed Alias_Token. Only N_Ports that joined an Alias Group via a Join Alias Group shall be removed. N_Ports that are listening shall not be removed unless the Alias group is disbanded. The payload of the request contains, among other parameters, the Alias_Token of the Alias Group from which the N_Ports are to be deleted, and a list of the N_Ports to be deleted from the Alias Group. If, at the conclusion of this operation, all N_Ports have been removed from the Alias Group, the Alias Group is disbanded. The Originator N_Port may or may not be a member of this list.

This request shall be rejected with a Reason Code Explanation of “Unauthorized Request (Invalid Authorization Control)” if the Authorization_Control for the specified Alias Group does not indicate that the Initiator N_Port may delete the passed N_Ports from the Alias Group.

A command code of hex’0002’ in the CT_HDR indicates the remove from alias group request.

The format of the payload is shown in table 14.

Table 14– Remove from alias group payload

Item	Size (Bytes)
Authorization_PW	12
Reserved	4
Alias_Token	12
NP_List_Length	4
NP_List(1)	4
NP_List (2 to n-1)	(n-2) x 4
NP_List(n)	4

Authorization Password (Authorization_PW): This field contains the Authorization_PW for this Alias Group. The request shall be rejected with a Reason Code Explanation of "Unauthorized Request (Invalid Password)" if this Authorization_PW does not match the Authorization_PW used to create the Alias Group.

Alias Group Token (Alias_Token): The Alias_Token defines the Alias Group from which the N_Ports are to be deleted. The request shall be rejected if the Alias_Token is invalid (Reason Code Explanation of "Invalid Alias_Token"), or the Alias_Token does not exist (Reason Code Explanation of "Alias_Token does not exist"). The format of the Alias_Token is described in table 21.

N_Port List Length (NP_List_Length): The NP_List_Length specifies the number of entries in the following NP_List.

N_Port List (NP_List): The NP_List contains one entry for each N_Port address identifier to be deleted from the Alias Group. The N_Port address identifier shall be right-aligned within the NP_List entry. That is, the high-order byte of the entry shall be ignored and the low-order 3 bytes shall contain the N_Port address identifier.

An FS_ACC is returned indicating that the N_Ports in the passed list have been deleted from the indicated Alias Group. No payload is associated with this response.

An FS_RJT shall also be returned if an Alias_ID could not be de-assigned by the Fabric. The FS_RJT Reason Code Explanation indicates the appropriate Reason Code Explanation from table 22.

11.1.3.3 Listen (LSN)

This request is sent to the Alias Server to cause it to implicitly add the passed N_Port to every Alias Group whose Alias_Class matches the passed Alias_Class. If the Alias Group was created with an Authorization_Control indicating that no N_Ports may Listen in on this Alias Group, then the request shall be rejected with a Reason Code Explanation of "Unauthorized Request (Invalid Authorization_Control). If the Alias group was created with an Authorization_Control indicating that all N_Ports may Listen in, then the passed N_Port is added to all current and subsequent Alias Groups with the same Alias_Class.

For Multicast Groups, this provides the capability for an N_Port to "listen in" on all the traffic for all Multicast Groups of a given Alias_Class.

For Hunt Groups, this request causes no actions to be taken by the Alias Server.

The payload of the request contains, among other parameters, the N_Port ID of the "listening" N_Port, and the Alias_Class to which it wishes to listen. The "listening" N_Port may or may not be the Originator of the request.

A command code of hex'0003' in the CT_HDR indicates the listen request.

The format of the payload is shown in table 15 .

Table 15– Listen payload

Item	Size (Bytes)
Alias_Token	12
Listening N_Port ID	4

Alias group token (Alias_Token): The Alias_Token defines the Alias Group to which the N_Ports are to be added. The format of the Alias_Token is described in table 21.

When the Flags field indicates a Hunt Group, no further processing is performed and an FS_ACC is returned.

When the Flags field indicates a Multicast Group, only the Alias_Class field is referenced by this request. The Alias_Qualifier field shall be ignored. The request shall be rejected with a

Reason Code Explanation of “Alias_Token does not exist” if the Alias_Token does not specify an existing Multicast Group.

Listening N_Port ID (L_N_Port ID): The L_N_Port ID identifies the N_port which wishes to listen to all traffic for the associated Alias Group defined in the Alias_Token. The N_Port address identifier shall be right-aligned within the L_N_Port ID field. That is, the high-order byte of the entry shall be ignored and the low-order 3 bytes shall contain the N_Port address identifier.

An FS_ACC shall be returned indicating that the L_N_Port has been implicitly added to all Alias Groups of the same Alias_Class. The format of the FS_ACC payload is shown in table 16 .

Table 16– Listen accept payload

Item	Size (Bytes)
Alias_Token	12

Alias Group Token (Alias_Token): The Alias_Token defines the Alias Group(s) to which the N_Port is listening. It is the same Alias_Token as was passed in the request. The format is described in table 21.

An FS_RJT shall be returned if the passed Alias_Token did not contain a valid Flags field. The FS_RJT Reason Code Explanation indicates “Invalid Alias_Token”.

11.1.3.4 Stop listen (SLSN)

This request is sent to the Alias Server to cause it to implicitly delete the passed listening N_Port from every Alias Group whose Alias_Class matches the passed Alias_Class. For Multicast Groups, this provides the capability for an N_Port to “stop listening” on all the traffic for all Multicast Groups of a given Alias_Class. For Hunt Groups, this request causes no actions to be taken by the Alias Server. The payload of the request contains, among other parameters, the N_Port ID of the N_Port which is to stop listening, and the Alias_Class to which it wishes to stop listening. The “listening” N_Port may or may not be the Originator of the request.

A command code of hex’0004’ in the CT_HDR indicates the stop listen request.

The format of the payload is shown in table 17.

Table 17– Stop listen payload

Item	Size (Bytes)
Alias_Token	12
Listening N_Port ID	4

Alias Group Token (Alias_Token): The Alias_Token defines the Alias Group for which the passed N_Port wishes to stop listening. The format is described in table 21.

When the Flags field indicates a Hunt Group, no further processing shall be performed and an FS_ACC shall be returned.

When the Flags field indicates a Multicast Group, only the Alias_Class field is referenced by this request. The Alias_Qualifier field shall be ignored. The request shall be rejected with a Reason Code Explanation of “Alias_Token does not exist” if the Alias_Token does not specify an existing Multicast Group.

Listening N_Port ID (L_N_Port ID): The L_N_Port ID identifies the N-port which wishes to stop listening to all traffic for the associated Alias Group defined in the Alias Token. The N_Port address identifier shall be right-aligned within the L_N_Port ID field. That is, the high-order byte of the entry shall be ignored and the low-order 3 bytes shall contain the N_Port address identifier.

An FS_ACC shall be returned indicating that the L_N_Port has been implicitly deleted from all Alias Groups of the same Alias_Class. No payload is associated with the FS_ACC.

An FS_RJT shall be returned if the passed Alias_Token did not contain a valid Flags field. The FS_RJT Reason Code Explanation indicates “Invalid Alias_Token”.

11.1.3.5 Read alias group (RAG)

This request is sent to the Alias Server to cause it to return a list of the N_Port IDs that have been formed into the Alias Group specified by the passed Alias_Token. If the Alias Group does not exist, no N_Ports are returned. The payload of the request contains the Alias_Token for the Alias Group of interest.

A command code of hex’0005’ in the CT_HDR indicates theread alias group request.

The format of the payload is shown in table 18.

Table 18– Read alias group payload

Item	Size (Bytes)
Alias_Token	12

Alias Group Token (Alias_Token): The Alias_Token defines the Alias Group for which the member N_Port IDs are to be returned. The format is described in table 21. The request shall be rejected with a Reason Code Explanation of "Invalid Alias_Token" if the Alias_Token is invalid.An FS_ACC shall be returned containing a list of all the N_Port IDs in the associated Alias Group, if any. The format of the FS_ACC payload is shown in table 19.

Table 19– Read alias group accept payload

Item	Size (Bytes)
Alias_Token	12
Alias_SP	80
NP_List_Length	4
NP_List(1)	4
NP_List (2 to n-1)	(n-2) x 4
NP_List(n)	4

Alias Group Token (Alias_Token): The Alias_Token defines the Alias Group(s) to which the N_Port is listening. It is the same Alias_Token as was passed in the request. The format is described in table 21.

Alias Group Service Parameters (Alias_SP):

The Alias_SP returns the Service Parameters in effect for the Alias Group indicated by the Alias_Token.

N_Port List Length (NP_List_Length): The NP_List_Length specifies the number of entries in the following NP_List.

N_Port List (NP_List): The NP_List contains one entry for each N_Port address identifier to be deleted from the Alias Group. The format of the NP_List entry is shown in table 20.

Table 20– NP_List entry format

Item	Size (Bytes)
Membership	1
N_Port ID	3

Membership: The Membership indicates the type of membership the N_Port has in the Alias Group. The following membership types are defined:

x'0' = Grouped, i.e. via Join Alias Group.

x'1' = Listening, i.e., via Listen.

Others = Not used.

N_Port ID: The N_Port ID of the N_Port.

An FS_RJT shall only be returned for an Invalid Alias_Token, as described above. If the Alias_Token does not define an existing Alias Group, the FS_ACC shall indicate an NP_List_Length of 0.

11.1.4 Alias server replies

An Alias Server reply shall signify that the Alias Server request is completed. The reply IU may contain data following the FS_Command code word. The Alias Server uses the generic Accept (FS_ACC) and Reject (FS_RJT) replies defined in 4.6.

11.1.4.1 Accept (FS_ACC)

The FS_ACC shall notify the Initiator of an Alias Server request that the request has been successfully completed. The Initiator of the FS_ACC shall terminate the Exchange by setting the Last Sequence bit (bit 20) in F_CTL on the last Data frame of the FS_ACC. The Payload is unique to the Alias Server requests and is defined by those requests.

11.1.4.2 Reject (FS_RJT)

The FS_RJT (see 4.6) shall notify the Initiator of an Alias Server request that the request has been unsuccessfully completed. The Initiator of the FS_RJT shall terminate the Exchange by setting the Last Sequence bit (bit 20) in F_CTL on the last Data frame of the FS_RJT. The first

error condition encountered shall be the error reported.

When a Reason Code of “Unable to perform command request” is generated, table 22 defines and explains the various FS_RJT Reason Code Explanations. Reason Code Explanations x’30’ through x’38’x are identical to those defined for the Extended Link Services necessary to support the Alias Server function.

If a valid Alias Server request is not received, the request is rejected with a Reason Code of “Invalid Command code” and a Reason Code Explanation of “No additional explanation”.

A valid Alias Server request shall not be rejected with a Reason Code of “Command not supported”.

11.1.4.3 Alias_Token

Table 21 defines the format of the Alias_Token field .

Table 21– Alias_Token

Item	Size (Bytes)
Flags	1
Alias_Class	3
Alias_Qualifier	8

Flags: The Flags field specifies the type of Alias Group being defined and also specifies some options for the Alias Group.

Bits 7-4: Alias Group Type

These bits are an encoded value defining the supported Alias Group Types.

x’0’ = Reserved

x’1’ = Multicast Group

x’2’ = Hunt Group

Others = Reserved

Bits 3-0: Alias Options

These bits define the supported Alias options.

Bit 3: Send to Initiator. When set to one, this bit indicates that the Initiator is also eligible to receive the frame that was sent to the Alias Group. For Multicast Groups, the transmitted frame may also be routed to the Initiator of the frame. For Hunt Groups, the Initiator is also considered a member, for route selection purposes. When set to zero, this bit indicates that the Initiator shall not be considered a member of the Alias Group, for routing purposes.

Bits 2-1: Reserved.

Bit 0: MG_IPA. Refer to 11.3 for details.

Alias_Class: This field is used to identify the class of Alias.

For Multicast Groups, it is further defined as follows:

Bits 23-16: TYPE

Bits 15-12: Routing Bits

The TYPE and Routing Bits fields provide a means to define and identify Multicast Groups based on the FC-PH TYPE and Routing Bits fields. For example, a Multicast Group can be created for all SCSI-FCP TYPEs and a different Multicast Group may be created for all SBCCS TYPEs. Routing Bits are included to handle the Video_Data specification. The values that can be assigned for these fields are identical to the assigned TYPE and Routing Bits values specified in FC-PH. Additionally, the value of all ones is defined as meaning a Multicast Group of all TYPEs and Routing Bits.

Bits 11-0: Reserved

For Hunt Groups, this field is available for use by the Common Controlling Entity to uniquely identify multiple Hunt Groups for that Common Controlling Entity.

Table 22 – FS_RJT reason code explanation

Encoded Value (Bits 15-8)	Description	Applicable commands
0000 0000	No additional information	Invalid commands
0011 0000	No Alias IDs available for this Alias Type	Join Alias Group
0011 0001	Alias ID cannot be activated at Fabric (no resource available)	Join Alias Group
0011 0010	Alias ID cannot be activated at Fabric (Invalid Alias ID)	Join Alias Group
0011 0011	Alias ID cannot be deactivated at Fabric (doesn't exist)	Remove From Alias Group
0011 0101	Alias Group cannot be joined (Service Parameter conflict)	Join Alias Group
0011 0100	Alias ID cannot be deactivated at Fabric (resource problem)	Remove From Alias Group
0011 0110	Invalid Alias_Token	Join Alias Group, Remove From Alias Group, Listen, Stop Listen, Read Alias Group
0011 0111	Unsupported Alias_Token	Join Alias Group
0011 1000	Alias Group cannot be formed (Invalid N_Port List)	Join Alias Group
0100 0000	Alias Group cannot be formed (Invalid Class)	Join Alias Group
0100 0001	Alias_Token does not exist	Remove From Alias Group, Listen, Stop Listen
0100 0010	Unauthorized Request (Invalid Password)	Join Alias Group, Remove From Alias Group
0100 0011	Unauthorized Request (Invalid Authorization_Control)	Join Alias Group, Remove From Alias Group, Listen
0100 0100	Invalid Authorization_Control	Join Alias Group

Alias_Qualifier:

For Multicast Groups, the Alias_Qualifier field provides the means to define and identify different Multicast Groups within a particular TYPE/Routing Bits. For example, an SBCCS Multicast Group may be created for channels and a different SBCCS Multicast Group may be created for control units.

The Alias_Qualifier shall be defined as follows (all values are hex):

– All zeroes**:** Unknown. A unique value will be assigned by the Server;

– All ones: Multicast Group for all N_Ports with the associated TYPE/Routing Bits combination;

– All others: Assigned by the particular FC-4 defined by TYPE. The N_Ports have an intrinsic knowledge, defined by the associated FC-4, as to the meaning of these values.

For Hunt Groups, the Alias_Qualifier contains the Node_Name for the Common Controlling Entity forming the Hunt Group.

11.1.4.4 Authorization_Control

Table 23 defines the format of the Authorization_Control field .

Table 23– Authorization_Control

Item	Size (Bytes)
Add_Authorization	1
Delete_Authorization	1
Listen_Authorization	1
Reserved	1

Add_Authorization: This field determines which N_Ports are allowed to add N_Ports to the Alias Group specified in the request, under control of the Authorization_PW.

If the Authorization_PW of the Alias Group being modified is non-zero, then a subsequent Join Alias Group shall be rejected if the passed Authorization_PW does not match the Authorization_PW of the Alias Group. The Reason Code Explanation shall indicate “Unauthorized Request (Invalid Password)”.

If the Authorization_PW of the Alias Group being modified is all zeroes, or matches the passed Authorization_PW, then the Authorization_Control is checked.

The values for this field are defined as (hex):

00: Any N_Port may issue a subsequent Join Alias Group to add itself or any other N_Port(s) to the Alias Group defined by the passed Alias_Token.

01: An N_Port may issue a subsequent Join Alias Group to add only itself to the Alias Group defined by the passed Alias_Token. An attempt to add any N_Port(s) but the Initiator N_Port shall be rejected with a Reason Code Explanation of “Unauthorized Request (Invalid Authorization_Control)”.

02: The N_Port that initiated the Join Alias Group that created the Alias Group is the only N_Port allowed to add to the Alias Group. A Join Alias Group initiated by any other N_Port shall be rejected with a Reason Code Explanation of “Unauthorized Request (Invalid Authorization_Control)”.

03-FF: Reserved. A Join Alias Group specifying an Add_Authorization equal to one of these values shall be rejected with a Reason Code Explanation of “Invalid Authorization_Control”.

Delete_Authorization: This field determines which N_Ports are allowed to delete N_Ports from the Alias Group specified in the request.

If the Authorization_PW of the Alias Group being modified is non-zero, then a subsequent Remove From Alias Group shall be rejected if the passed Authorization_PW does not match the Authorization_PW of the Alias Group. The Reason Code Explanation shall indicate “Unauthorized Request (Invalid Password)”.

If the Authorization_PW of the Alias Group being modified is all zeroes, or matches the passed Authorization_PW, then the Authorization_Control is checked.

The values for this field are defined as (hex):

00: Any N_Port may issue a subsequent Remove From Alias Group to delete itself or any other N_Port(s) from the Alias Group defined by the passed Alias_Token.

01: An N_Port may issue a subsequent Remove From Alias Group to delete only itself from the Alias Group defined by the passed Alias_Token. An attempt to delete any N_Port(s) but the Initiator N_Port shall be rejected with a Reason Code Explanation of “Unauthorized Request (Invalid Authorization_Control)”.

02: The N_Port that initiated the Join Alias

Group that created the Alias Group is the only N_Port allowed to delete from the Alias Group. A Remove From Alias Group initiated by any other N_Port shall be rejected with a Reason Code Explanation of “Unauthorized Request (Invalid Authorization_Control)”.

03-FF: Reserved. A Remove From Alias Group specifying a Delete_Authorization equal to one of these values shall be rejected with a Reason Code Explanation of “Invalid Authorization_Control”.

Listen_Authorization: This field determines whether N_Ports are allowed to Listen in on this Alias Group.

The values for this field are defined as (hex):

00: Any N_Port may issue a subsequent Listen to start listening to the Alias Group(s) defined by the passed Alias_Token.

01: No N_Port may Listen to the Alias Group(s) defined by the passed Alias_Token. A subsequent Listen request for these Alias Group(s) shall be rejected with a Reason Code Explanation of “Unauthorized Request (Invalid Authorization_Control)”.

02-FF: Reserved. A Join Alias Group specifying a Listen_Authorization equal to one of these values shall be rejected with a Reason Code Explanation of “Invalid Authorization_Control”.

11.1.5 Function flow

Figure 8 illustrates the flow among the Originator N_Port, participating N_Ports, Alias Server, Directory Server, and Fabric Controller to create an Alias Group.

11.1.6 Alias server functions

The following sections describe the functions performed by the Alias Server for each of the supported requests.

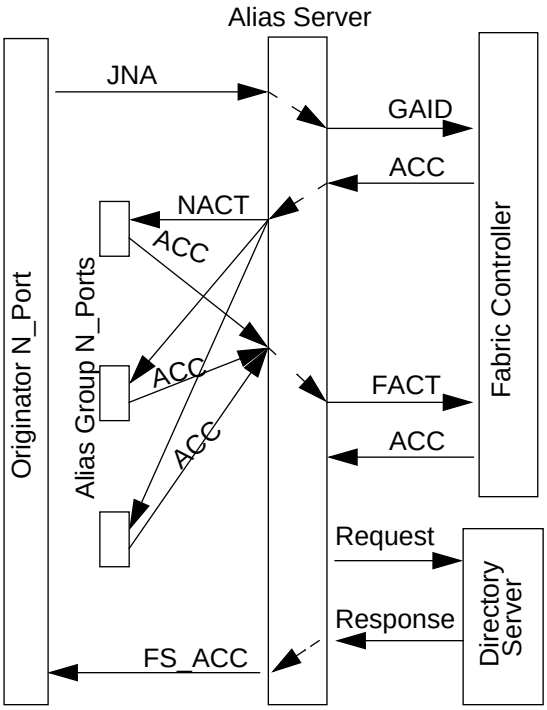


Figure 8 –Function flow

11.1.6.1 Join alias group

Upon reception of a Join Alias Group request, the Alias Server shall perform the following functions, in the specified order:

a) The Alias Server shall reject the request with a Reason Code Explanation of “Invalid Alias_Token” if the passed Alias_Token is not valid;

b) The Alias Server shall reject the request with a Reason Code Explanation of “Unsupported Alias_Token” if the Flags in the passed Alias_Token indicate an Alias Group that is not supported;

c) The Alias Server shall determine whether or not the specified Alias Group has already been created;

d) If the Alias Group has not already been created, an attempt is made to create a new

Alias Group;

The request shall be rejected with a Reason Code Explanation of “Invalid Authorization_ Control” if the passed Authorization_Control is not valid;

If a Multicast Group is being formed and the Alias_SP do not contain valid Class 3 Service Parameters, the request shall be rejected with a Reason Code Explanation of “Alias Group cannot be formed (Invalid Class)”;

The Alias Server shall send a Fabric Controller Request, Get Alias Group ID (GAID) to the Fabric Controller to obtain a unique alias address identifier for this Alias Group. The Alias_Token is passed in the payload of the request and the reply returns the assigned alias address identifier. If the Fabric returns an LS_RJT, the Join Alias Group is rejected with the same Reason Code Explanation as was contained in the LS_RJT to the GAID;

The passed Authorization_PW and Authorization_Control are attached to the defined Alias Group;

e) If the Alias Group has already been created, an attempt is made to modify the Alias Group;

The request shall be rejected with a Reason Code Explanation of “Unauthorized Request (Invalid Password)” if the Authorization_PW of the indicated Alias Group is non-zero and does not match the passed Authorization_ PW.;

The request shall be rejected with a Reason Code Explanation of “Unauthorized Request (Invalid Authorization_Control)” if the Authorization_Control attached to the passed Alias Group does not permit the initiating N_ Port to add the N_Ports in the passed NP_ List;

f) ;The Alias Server shall send an Extended Link Services request, NACT, to each of the N_Ports in the passed list. Refer to FC-PH-2 for details of this request;

If the Alias Group is being created, and other N_Ports are allowed to Listen, then the Alias Server shall also send a NACT to all N_Ports that have registered to listen to the Alias Class matching the Alias Class of the Alias Group being created (if any);

Upon reception of this request, if the destination N_Port can perform all of the following functions, it shall respond with an LS_ACC which indicates that it:

– Is capable of supporting the Alias_Class in the Alias_Token;

– Is capable of supporting the Alias Group Service Parameters;

– Has assigned the passed Alias Group address identifier as an alias for this N_Port;

– If the N_Port cannot perform all of the above functions, it shall send an LS_RJT as a reply;

g) When all of the N_Ports in the passed N_Port list and all of the Listening N_Ports have responded with either LS_ACC or LS_ RJT, or 2*R_A_TOV has expired, the Alias Server shall send a Fabric Controller request, FACT, to the Fabric Controller to activate the alias address identifier at the Fabric. The Alias_ID and a list of the N_Ports that responded with LS_ACC to the NACT are passed in the payload. Refer to FC-PH-2 for more details of this request;

NOTE – The Alias_ID shall not be activated at the Fabric for those N_Ports that did not respond within 2*R_A_TOV.

h) Upon reception of this request, if the Fabric Controller can assign this alias for all the N_Ports in the list, it shall return an LS_ ACC as a reply. If it cannot assign this alias for all the N_Ports in the list, it shall return LS_RJT;

i) When the Fabric has responded, and if there is a Directory Server accessible on the Fabric, the Alias Server shall inform the Directory Server of the existence or modification of this Alias Group. Refer to the Directory Server for details of this operation;

j) Finally, the Alias Server shall respond to the original Join Alias Group Request from the N_Port. An FS_ACC shall be returned to indicate that the Alias_ID has been assigned, even if none of the N_Ports in the original list were formed into the Alias Group. A Read Alias group request or a Directory Services request is necessary to determine the members of the created Alias Group. If the Fabric

returns an LS_RJT indicating that it was unable to assign an Alias_ID, the Join Alias Group is rejected with the same Reason Code Explanation as was contained in the LS_RJT to the FACT.

11.1.6.2 Remove from alias group

Upon reception of a Remove From Alias Group request, the Alias Server shall perform the following functions, in the specified order:

a) The Alias Server shall reject the request with a Reason Code Explanation of “Invalid Alias_Token” if the passed Alias_Token is not valid;

b) The Alias Server shall reject the request with a Reason Code Explanation of “Alias_Token does not exist” if the passed Alias_Token does not indicate an existing Alias Group. The request shall be rejected with a Reason Code Explanation of “Unauthorized Request (Invalid Password)” if the Authorization_PW of the indicated Alias Group is non-zero and does not match the passed Authorization_PW;

c) The request shall be rejected with a Reason Code Explanation of “Unauthorized Request (Invalid Authorization_Control)” if the Authorization_Control attached to the passed Alias Group does not permit the initiating N_Port to delete the N_Ports in the passed NP_List;

d) For each grouped N_Port in the passed NP_List, the Alias Server shall send a Fabric Controller Request, FDACT, to the Fabric Controller to deactivate the Alias_ID at the Fabric. The Alias_Token and Alias_ID are passed in the payload of the request, along with a list of the N_Ports to be removed. Refer to FC-PH-2 for more details of this request. The Alias Server shall not send an FDACT for an N_Port that is only listening;

e) Upon reception of this request, if the Fabric Controller cannot de-assign this alias for all the N_Ports in the list, it shall return LS_RJT. The Alias Server rejects the original request with the same Reason Code Explanation as was contained in the LS_RJT to the FDACT;

f) if the Fabric Controller can de-assign this alias for all the N_Ports in the list, it shall return an LS_ACC as a reply;

The Alias Server shall then send an Extended Link Services request, NDACT, to each N_Port in the passed list. Refer to FC-PH-2 for more details of this request;

Upon reception of this request, the destination N_Port attempts to deactivate the Alias_ID as an alias identifier. If successful, it shall return an LS_ACC. If it is unable to deactivate the Alias_ID as an alias identifier, it shall return an LS_RJT;

g) When the last member N_Port has been removed from the Alias Group, the Alias Server shall delete the Alias group;

If there are any N_Ports that were listening to this Alias Group, the Alias Server shall send an FDACT to the Fabric to deactivate the Alias_ID for each listening N_Port. When the Fabric has responded, the Alias Server shall then send an NDACT to each listening N_Port to deactivate the Alias_ID at the N_Port;

h) When all of the N_Ports in the passed N_Port list have responded with either LS_ACC or LS_RJT, or 2*R_A_TOV has expired, the Alias Server shall respond with an FS_ACC to indicate that the indicated N_Ports have been removed;

i) If there is a Directory Server accessible on the Fabric, the Alias Server shall inform the Directory Server of the modification or deactivation of this Alias Group. Refer to the Directory Server for details of this operation.

11.1.6.3 Listen

Upon reception of a Listen request, the Alias Server shall perform the following functions, in the specified order:

a) If the Alias_Token indicates a Hunt Group, The Alias Server shall perform no functions other than returning an FS_ACC indicating that this request has been completed. If the Alias_Token indicates a Multicast Group, the remaining functions shall be performed;

b) The Alias Server shall reject the request with a Reason Code Explanation of “Invalid Alias_Token” if the Flags in the passed Alias_Token are not valid;

c) The Alias Server shall reject the request with a Reason Code Explanation of “Alias_ Token does not exist” if the passed Alias_Token does not indicate an existing Alias Group;

d) The request shall be rejected with a Reason Code Explanation of “Unauthorized Request (Invalid Authorization_Control)” if the Authorization_Control attached to the passed Alias Group does not permit the initiating N_ Port to listen to the Alias Group(s) indicated by the passed Alias_Token;

e) The Alias Server shall determine the Alias IDs for all Alias Groups having the same Alias_Class as the passed Alias_Token. The Alias_Qualifier shall be ignored;

f) The Alias Server shall send an Extended Link Services request, NACT, for each Alias ID that was found, to the Listening N_Port ID. Refer to FC-PH-2 for details of this request;

g) Upon reception of each request, if the destination N_Port can perform all of the following functions, it shall respond with an LS_ ACC which indicates that it:

– Is capable of supporting the Alias_Class in the Alias_Token;

– Is capable of supporting the Alias Group Service Parameters;

– Has assigned the passed Alias Group address identifier as an alias for this N_Port;

– If the N_Port cannot perform all of the above functions, it shall send an LS_RJT as a reply;

h) For each LS_ACC from the Listening N_ Port, the Alias Server shall send a Fabric Controller request, FACT, to activate the alias address identifier at the Fabric, for the Listening N_Port. The Alias_ID and the Listening N_Port ID shall be passed in the payload;

i) Upon reception of each request, if the Fabric Controller can assign this alias for the Listening N_Port ID, it shall return an LS_ ACC as a reply. If it cannot assign this alias for all the N_Ports in the list, it shall return an LS_RJT;

j) When the Fabric has responded, and if there is a Directory Server accessible on the Fabric, the Alias Server shall inform the Directory Server of the modification of this Alias Group. Refer to the Directory Server for details of this operation;

k) When the Fabric has responded to all of the FACT requests, the Alias Server shall respond to the original Listen request. An FS_ ACC shall be returned to indicate that the Listening N_Port has been added to the specified Alias Groups. Whether or not the necessary Alias IDs have been activated at either the Listening N_Port or the Fabric is not indicated. A Read Alias group request or a Directory Services request is necessary to determine which Alias Groups have been activated for the listening N_Port;

l) Subsequently, if the Alias Server receives a Join Alias Group request to create a new Alias Group for the same Alias Class, it shall enable Listening to the new Alias Group for all N_Ports currently Listening to that Alias Class.

Listening N_Ports shall only be removed from an Alias Group by a Stop Listen request. They shall not be removed by a Remove From Alias Group request.

11.1.6.4 Stop listen

Upon reception of a Stop Listen request, the Alias Server shall perform the following functions, in the specified order:

a) If the Alias_Token indicates a Hunt Group, The Alias Server shall perform no functions other than returning an FS_ACC indicating that this request has been completed. If the Alias_Token indicates a Multicast Group, the remaining functions shall be performed;

b) The Alias Server shall reject the request with a Reason Code Explanation of “Invalid Alias_Token” if the Flags in the passed Alias_Token are not valid;

c) The Alias Server shall reject the request with a Reason Code Explanation of “Alias_ Token does not exist” if the passed Alias_Token does not indicate an existing Alias Group;

d) The Alias Server shall determine the Alias IDs for all Alias Groups having the same Alias_Class as the passed Alias_Token. The Alias_Qualifier is ignored;

e) If there are no Alias Groups having the same Alias_Class, an FS_ACC shall be returned indicating that the passed N_Port is no longer listening. Otherwise, processing continues;

f) The Alias Server shall send a Fabric Controller Request, FDACT, to the Fabric Controller to deactivate each Alias_ID at the Fabric, for the Listening N_Port ID. The Alias_Token and Alias_ID are passed in the payload of the request, along with the Listening N_Port ID. Refer to FC-PH-2 for more details of this request;

g) For each LS_ACC from the Fabric Controller, the Alias Server shall send an Extended Link Services request, NDACT, to the Listening N_Port ID. Refer to FC-PH-2 for more details of this request;

Upon reception of this request, the destination N_Port attempts to deactivate the Alias_ID as an alias identifier. If successful, it returns an LS_ACC. If it is unable to deactivate the Alias_ID as an alias identifier, it returns an LS_RJT;

h) If there is a Directory Server accessible on the Fabric, the Alias Server shall inform the Directory Server of the modification of the various Alias Groups. Refer to the Directory Server for details of this operation;

i) After an attempt has been made to deactivate all Alias IDs for the Listening N_Port ID, the Alias Server responds to the original Stop Listen request. An FS_ACC is returned to indicate that the Listening N_Port is no longer listening to the specified Alias Groups. Whether or not the necessary Alias IDs have been deactivated at either the Listening N_Port or the Fabric is not indicated. A Read Alias group request or a Directory Services request is necessary to determine which Alias Groups have been deactivated for the listening N_Port.

11.1.6.5 Read alias group

Upon reception of a Read Alias Group request, the Alias Server shall perform the following functions, in the specified order:

a) The Alias Server shall reject the request with a Reason Code Explanation of “Invalid Alias_Token” if the passed Alias_Token is not valid;

b) If the passed Alias Group does not exist, an FS_ACC shall be returned indicating that there are no N_Ports in the Alias Group (i.e., NP_List_Length is zero);

c) If the passed Alias Group does exist, an FS_ACC shall be returned specifying the Alias Group Service Parameters and a list of all the N_Port IDs that comprise the Alias Group, along with an indication of whether the N_Port is grouped or listening.

11.2 Alias routing

All routing of frames is done by the Fabric, based on a recognition that the D_ID of the transmitted frame is an Alias_ID. For Multicast groups, the exact frame that entered the Fabric is replicated to every destination N_Port in the Multicast Group associated with the Alias_ID of the frame. The Fabric shall not alter the frame header or the frame contents in any manner during this replication. For Hunt Groups, the exact frame that entered the Fabric is routed to a single destination N_Port in the Hunt Group.

NOTE – The Fabric may assign Alias_IDs to easily partition Multicast Group Alias_IDs from Hunt Group Alias_IDs.

The Sequence Initiator performs no special function to transmit a frame other than to use a D_ID indicating the Alias_ID and to use the Alias Group Service Parameters, rather than the Log-in Service Parameters. For example, the Receive Data Field Size for a multicast frame may be different than the Receive Data Field Size for a unicast frame.

If the Sequence Recipient is a member of an Alias Group, it shall recognize the Alias_ID as an alias address identifier and accept the frame.

For Multicast Groups, Class 1 and Class 2 frames with a D_ID equal to an Alias_ID shall be rejected by the Fabric.

11.3 IPA Considerations

11.3.1 Hunt groups

For Hunt Groups, there are no IPA considerations, since there is a requirement that all members of a Hunt Group be within a single Common Controlling Entity, which cannot be spread across images. Therefore, the same IPA may be used no matter which N_Port receives the frame.

11.3.2 Multicast groups

For Multicast Groups, the considerations for multicasting to N_Ports that require an Initial Process Associator are minimal as any multicasting behind the N_Port is handled internally. It is not necessary to know the IPA of each image behind an N_Port that belongs to a Multicast Group. Instead, if the Multicast Group requires an IPA, then a Multicast Group IPA (MG_IPA) is used by the Sequence Initiator. This MG_IPA is common for all images that belong to the same Multicast Group. An MG_IPA is set in the Association Header as follows:

The Responder Process Associator is set equal to the Alias_Qualifier for the Multicast Group.

Bit 24 of the Association_Header Validity bits is set to binary ‘1’. This indicates an MG_IPA.

Internally, images shall register with their N_Ports to join Multicast Groups, they do not register with the Alias Server. The MG_IPA becomes an alias for the actual IPA of the image and is recognized as such by the N_Port as a result of the internal registration. When a frame is received, the N_Port “multicasts” the payload to all images in the internal Multicast Group, based on the MG_IPA.

11.3.3 Broadcast

For Broadcast, there are no IPA considerations. Since the intent of Broadcast is to deliver to all possible recipients, it is the responsibility of the N_Port receiving a broadcast frame to broadcast the payload internally to all its images. Therefore, an IPA shall not be included in a frame being Broadcast.

Annex A
(Informative)

A. Service Interface Provided by FC-CT

This annex specifies the services provided by FC-CT and the services required by FC-CT. The services provided by FC-CT are categories as:

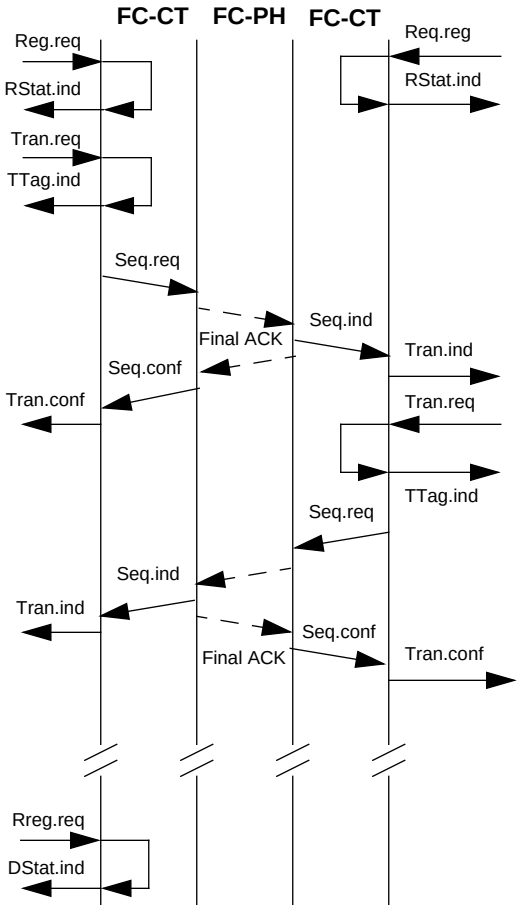
a) Session and Transaction services provided by FC-CT to its local users (a Fibre Channel Service application), denoted by the prefix FC_CT_;

b) Data services required by FC-CT from its local Fibre Channel layer entity, denoted by the prefix FC_PH_.

The definition of these services is for reference purposes only. It is not intended to imply any implementation.

NOTE - Throughout this service interface, confirmation primitives may not be indicated when Class 3 service is used. At present, the specification of FC-CT does not support end-to-end protocol over Class 3 service is used. At present, the specification of FC-CT does not support end-to-end protocol over Class 3 service.

Figure A.1 shows a sample interchange of request and response transactions between two FC-CT entities.



Abbreviations:

Dreg	FC_CT_DREG
DSta	FC_CT_DREG_STATUS
Reg	FC_CT_REG
RStat	FC_CT_REG_STATUS
Seq	FC_CT_PH_SEQUENCE
Tran	FC_CT_TRANSACTION
TTag	FC_CT_TRANSACTION_TAG
conf	confirmation
ind	indication
req	request

Figure A.1 – A sample transaction exchange

A.1 FC-CT Session Services

A.1.1 FC_CT_REG.request

This primitive defines a registration of an FC-CT session from a local FC-CT user entity. It allows the user to specify some FC-CT service parameters during the session such that subsequent transaction requests shall be supported with the same service parameters.

A.1.1.1 Semantics of the primitive

```
FC_CT_REG.request          {
        FcsType,
        CosPreference,
        MaxIUsize,
        TransactionMode,
}
```

The FcsType indicates the type of Fibre Channel Service provided by the local FC-CT user entity.

The cosPreference specifies the classes of service preference by the local FC-CT user entity.

The user entity also specifies the maximum size of an information unit that it is expecting to receive from any remote peer entity with MaxIUsize.

The TransactionMode is used to indicate the mode of transaction.

Optionally, the Source specifies the address identification of the user entity. This may be in the form of an N_Port Address ID (S_ID) or any other form that is implementation specific. This may be necessary since the N_Port may also support multiple aliases other than its native address ID.

A.1.1.2 When Generated

This primitive is generated by an FC-CT user entity to establish a service session during which certain FC-CT service parameters shall be provided by the local FC-CT to the user entity.

A.1.1.3 Effect of Receipt

Upon receipt of this primitive, the local FC-CT shall allocate the necessary resources and verify if the requested service (such as class of service) can be provided. If the session can be supported, the FC-CT shall establish the session resources.

A.1.2 FC_CT_REG_STATUS.indication

This primitive defines the response by FC-CT to the FC_CT_REG.request primitive, indicating the success or failure of the request.

A.1.2.1 Semantics of the primitive

```
FC_CT_REG_STATUS.indication {
        RegStatus,
        SessionTag,
        FailureReason
}
```

The RegStatus shall be used by FC-CT to inform the local user entity about the success or failure of the previous registration request. If the parameter indicates a success, the SessionTag shall provide a local identification for the session and shall be used to relate to subsequence transaction primitives. If the RegStatus parameter indicates a failure, the reason shall be provided in the FailureReason parameter and the SessionTag is meaningless.

A.1.2.2 When Generated

This primitive is generated in response to the FC_CT_REG.request primitive.

A.1.2.3 Effect of Receipt

Upon receipt of this primitive, the user entity shall determine whether a FC-CT session has been established based on the RegStatus parameter<M%2>. If established, the user entity shall be able to generate FC_CT_TRANSACTION.request or receive FC_CT_TRANSACTION.indication primitives.

A.1.3 FC_CT_DEREG.request

This primitive defines the deregistration of an established FC-CT session from a local FC-CT user entity. It allows the user to terminate the session, identified by the parameter, SessionTag, with the FC-CT.

A.1.3.1 Semantics of the primitive

```
FC_CT_DEREG.request {
        SessionTag
}
```

A.1.3.2 When Generated

This primitive is generated by an FC-CT user entity after it has successfully established a session via the primitive, FC-CT_REG.request.

A.1.3.3 Effect of Receipt

Upon receipt of this primitive, the local FC-CT shall relinquish all resources associated with the session. The FC-CT may also relinquish the associated FC-PH resources through means unspecified here. No further transaction shall be supported for the user entity.

A.1.4 FC-CT_DEREG_STATUS.indication

This primitive defines the response by FC-CT to the FC-CT_DEREG.request primitive, indicating the success or failure of the request.

A.1.4.1 Semantics of the primitive

```
FC_CT_REG_STATUS.indication {
        DeRegStatus,
        FailureReason
}
```

The DeRegStatus shall be used by FC-CT to inform the local user entity about the success or failure of the previous deregistration request. If the parameter indicates success, the session has been terminated. If the parameter indicates failure, the FailureReason shall contain the reason (e.g. invalid SessionTag).

A.2 FC-CT Transaction Services

A.2.1 FC_CT_TRANSACTION.request

This primitive defines the transfer of an Information Unit from a local user entity to FC-CT for delivery to a remote peer entity.

A.2.1.1 Semantics of the primitive

```
FC_CT_TRANSACTION.request {
        RequestTag,
        Destination,
        TransactionType,
}
```

The RequestTag may optionally be provided with this primitive. If provided, the RequestTag shall uniquely identify this transaction request. The RequestTag may be assigned by the user entity and included in this primitive or assigned by FC-CT and indicated in the FC_CT_TRANSACTION_TAG.indication.

The Destination contains the address of the remote user to which this transaction information unit is to be delivered. It may be an N_Port Address ID or an implementation specific address.

The type of transaction is indicated in the parameter, TransactionType.

The parameter, Iu, specifies the information unit to be transmitted as the payload of the request.

A.2.1.2 When Generated

This primitive is generated by an FC-CT user entity to request a transaction information unit transfer by FC-CT. It can only be generated after the user entity has established a session with FC-CT.

A.2.1.3 Effect of Receipt

Upon receipt of this primitive, the source FC-CT performs the following actions:

a) receives the indicated unique RequestTag or may indicate to the user entity a unique RequestTag using the FC_CT_TRANSACTION_TAG.indication

b) validates the parameters and verifies that the requested operation is possible, e.g. check the accessibility of the Destination.

c) encapsulates the user information unit with the necessary CT_HDR.

d) issues FC_PH_SEQUENCE.request to the local FC-PH to transfer the resulting information unit to the destination.

e) uses FC_CT_TRANSACTION.confirmation to notify the user entity as to whether the transaction is successfully completed.

A.2.2 FC_CT_TRANSACTION_TAG.indication

This primitive defines an indication of the RequestTag for the FC_CT_TRANSACTION.request.

A.2.2.1 Semantics of the primitive

```
FC_CT_TRANSACTION_TAG.indication {
        RequestTag
}
```

The RequestTag shall provide the local unique identifier for a previous transaction request.

A.2.2.2 When Generated

This primitive is atomically generated in response to the transmit the transaction by the FC_CT_TRANSACTION.request.

A.2.2.3 Effect of Receipt

The effect of receipt of this primitive by the FC-CT use entity is unspecified.

A.2.3 FC_CT_TRANSACTION.confirmation

This primitive defines the response to a FC_CT_TRANSACTION.request, signifying the success or failure of the transaction. This primi-

tive is issued when Class 1 or 2 service is used. In Class 3, this primitive may not be issued.

A.2.3.1 Semantics of the primitive

```
FC_CT_TRANSACTION.confirmation {
        RequestTag,
        TransactionStatus,
        FailureReason
}
```

The RequestTag shall provide the local unique identifier for a previous transaction request.

The TransactionStatus provides the status information to the local user entity about the success or failure of the request identified by RequestTag.

The FailureReason shall contain the reason code when the TransactionStatus indicates a failure.

A.2.3.2 When Generated

This primitive is generated upon the completion of the attempt to transmit the transaction by the source FC-CT.

A.2.3.3 Effect of Receipt

The effect of receipt of this primitive by the FC-CT user entity is unspecified.

A.2.4 FC_CT_TRANSACTION.indication

A.2.4.1 Semantics of the primitive

```
FC_CT_TRANSACTION.indication {
        Source,
        TransactionType,
}
```

This primitive defines the transfer of information unit from FC-CT to the local FC-CT user entity.

The Source contains the address of the remote user entity that transmitted the information unit.

The type of transaction is indicated by TransactionType.

The parameter, Iu, specifies the information unit received.

A.2.4.2 When Generated

This primitive is generated upon the successful completion of a transaction reception by the destination FC-CT to its relevant user entity.

A.2.4.3 Effect of Receipt

The effect of receipt of this primitive by the FC-CT user entity is unspecified.

Annex B
(Informative)

B. FC-DS ASN.1 Module

```
FC-DS DEFINITIONS ::= BEGIN

 --top level message


 Message ::= SEQUENCE{

        version      INTEGER{
                        version-1(0)
                             },

        data          OperationsAndResults}


--Operations and Results

OperationsAndResults ::= CHOICE{
                        compareRequest       [1]      IMPLICIT CompareRequest,
                        compareResult        [2]      IMPLICIT CompareResult,
                        abandonRequest       [3]  IMPLICIT InvokeID,
                        abandonResult        [4]  IMPLICIT InvokeID,
                        searchRequest        [5]  IMPLICIT SearchRequest,
                        searchResult         [6]  IMPLICIT SearchResult,
                        addEntryRequest      [7]  IMPLICIT AddEntryRequest,
                        addEntryResult       [8]  IMPLICIT InvokeID,
                        removeEntryRequest   [9]   IMPLICIT RemoveEntryRequest,
                        removeEntryResult    [10] IMPLICIT InvokeID,
                        modifyEntryRequest   [11] IMPLICIT ModifyEntryRequest,
                        modifyEntryResult    [12] IMPLICIT InvokeID,
                        queryCapRequest      [13] IMPLICIT InvokeID,
                        queryCapResult       [14] IMPLICIT QueryCapResult,
                        error                [15] IMPLICIT Error}



--Operations, arguments and results--


        --Compare Request
                CompareRequest ::=
                    SEQUENCE{
                        invokeID    [0] IMPLICIT InvokeID,
                        object      [1] IMPLICIT DistinguishedName,
                        purported   [2] FCattribute,
                        COMPONENTS OF CommonArguments}
```

```
        --Compare Result
                CompareResult ::=
                    SEQUENCE{
                        invokeID    [0] IMPLICIT InvokeID,
                        name        [1] IMPLICIT DistinguishedName OPTIONAL,
                        matched     [2] IMPLICIT Boolean DEFAULT true,
                        COMPONENTS OF CommonResults}


        --Search Request
                SearchRequest ::=
                    SEQUENCE{
                        invokeID                [0] IMPLICIT InvokeID,
                        baseobject              [1] IMPLICIT DistinguishedName,
                        subset                  [2] IMPLICIT INTEGER{
                                                        baseObject(0),
                                                        oneLevel(1),
                                                        wholeSubtree(2)}      DEFAULT
wholeSubtree,
                        filter                  [3] Filter,
                        passBaseObject          [4] IMPLICIT Boolean DEFAULT true,
                        searchDirectoryAliases  [5] IMPLICIT Boolean DEFAULT true,
                        selection               [6] EntryInformationSelection,
                        pagedResults            [7] PagedResultsRequest OPTIONAL,
                        COMPONENTS OF CommonArguments}


        --Search Result
                SearchResult ::=
                    SEQUENCE{
                        invokeID        [0] IMPLICIT InvokeID,
                        object          [1] IMPLICIT DistinguishedName OPTIONAL,
                        entries         [2] IMPLICIT SEQUENCE OF EntryInformation,
                        limitProblem    [3] IMPLICIT LimitProblem OPTIONAL,
                        queryReference  [4] OCTET STRING OPTIONAL,
                        COMPONENTS OF CommonResults}


        --Limit Problem
            LimitProblem ::= INTEGER{
                timeLimitExceeded (0),
                sizeLimitExceeded (1),
                administrativeLimitExceeded (2)}


        --Add Entry Request
                AddEntryRequest ::=
                    SEQUENCE{
                        invokeID        [0] IMPLICIT InvokeID,
```

```
                primary             [1] IMPLICIT PrimaryEntry,
                subordinateList     [2] IMPLICIT SEQUENCE OF SubordinateEntry
OPTIONAL,
                COMPONENTS OF CommonArguments}


    --Primary Entry
        PrimaryEntry ::= SEQUENCE{
            object  [3] IMPLICIT DistinguishedName,
             entry  [4] IMPLICIT SEQUENCE OF FCattribute DEFAULT{}}

    --Subordinate Entry
        SubordinateEntry ::= SEQUENCE{
            object  [5] Rdn,
            entry   [6] IMPLICIT SEQUENCE OF FCattribute DEFAULT{}}


    --Remove Entry Request
            RemoveEntryRequest ::=
                SEQUENCE{
                    invokeID    [0] IMPLICIT InvokeID,
                    object      [1] IMPLICIT DistinguishedName,
                    COMPONENTS OF CommonArguments}



    --Modify Entry Request
            ModifyEntryRequest ::=
                SEQUENCE{
                    invokeID    [0] IMPLICIT InvokeID,
                    object      [1] IMPLICIT DistinguishedName,
                    changes     [2] IMPLICIT SEQUENCE OF EntryModification,
                    COMPONENTS OF CommonArguments}

    --Entry Modification
        EntryModification ::= CHOICE{
            addAttribute        [0] FCattribute,
            removeAttribute     [1] FCattribute}



    --Query Capabilities Result
        QueryCapResult ::=
                SEQUENCE{
                    invokeID                [0] IMPLICIT InvokeID,
                    versionNumber           [1] IMPLICIT INTEGER,
                    numFilterItems          [2] IMPLICIT INTEGER,
                    matchTypes              [3] IMPLICIT MatchSelection,
                    entryLimit              [5] IMPLICIT INTEGER,
                    pagedResults            [6] IMPLICIT Boolean,
                    unsupportedAttributes [7] IMPLICIT SEQUENCE OF
```

```
                                                    AttributeDescriptor,
                        queryReferenceLife [8] IMPLICIT INTEGER,
                        timeLimitUsage [9] IMPLICIT INTEGER}




        --Match Selection
            MatchSelection ::= BIT STRING{

                        equality (0),
                        substrings (1),
                        greaterOrEqual (2),
                        lessOrEqual (3),
                        present (4),
                        approximateMatch (5)}

--Errors and Parameters--



Error ::=
        SEQUENCE{
                invokeID    [0] IMPLICIT InvokeID,
                errorChoice     [1] ErrorChoice}

ErrorChoice ::= CHOICE{
                abandoned[0] IMPLICIT Abandoned,
                abandonFailed[1] IMPLICIT AbandonFailed,
                nameError[2] IMPLICIT NameError,
                updateError[3] IMPLICIT UpdateError,
                attributeError[4] IMPLICIT AttributeError,
                protocolError [5] IMPLICIT ProtocolError,
                serviceError[6] IMPLICIT ServiceError}

Abandoned ::=
                    NULL
AbandonFailed ::=
            SEQUENCE{
                operation[0]InvokeID,
                problem[1]IMPLICIT AbandonProblem}

AbandonProblem ::=
            INTEGER{
                noSuchOperation(1),
                cannotAbandon(2)}
```

```
AttributeError::=
        SEQUENCE{
            object  [0] DistinguishedName,
            attribute    [1] AttributeSpecifier,
            description [2] AttributeProblem}

AttributeSpecifier ::=
        CHOICE{
            attributeDescriptor      AttributeDescriptor,
            fcAttribute              FCattribute}



AttributeProblem ::=
        INTEGER{
            noSuchAttribute(1),
            invalidSyntax(2),
            undefinedAttribDescriptor(3),
            inappropriateMatching(4),
            constraintViolation(5),
            attributeOrValueExists(6),
            attributeNotSupported(7)}


NameError::=
        SEQUENCE{
            problem    [0]      NameProblem,
            matched    [1]      DistinguishedName}

NameProblem ::=
        INTEGER{
            noSuchObject (1),
            aliasProblem(2),
            invalidAttributeSyntax(3),
            aliasDereferencingProblem(4)}

ProtocolError::=NULL

ServiceError::=
        INTEGER{
            busy    (1),
            unavailable     (2),
            timeLimitExceeded     (3),
            administrativeLimitExceeded  (4),
            ditError     (5),
            invalidQueryReference     (6)}

UpdateError ::=
        INTEGER{
            namingViolation    (1),
            objectClassViolation   (2),
```

```
                notAllowedOnRdn       (3),
                entryAlreadyExists    (4),
                objectClassModification   (5)}



--Type definitions--


        --Alias N_Port Identifier
            AliasID ::=
                [APPLICATION 5]
                    IMPLICIT OCTET STRING (SIZE (3))




        --AliasedObjectName
            AOname ::=
                [APPLICATION 9]
                    IMPLICIT DistinguishedName

        --Attribute Descriptor
            AttributeDescriptor ::=
                [APPLICATION 17]
                    IMPLICIT INTEGER{
                                endPointName (1),
                                initialPas (2),
                                ipAddress (3),
                                nPortID (4),
                                aliasID (5),
                                nPortName (6),
                                cName (7),
                                fc4DataType (8),
                                aoName (9),
                                symbolicName (10),
                                objectClass (18),
                                ulpSpecificInfo (28),
                                ieeeMulticastGroup (29)}

        --Boolean
            Boolean ::= INTEGER{
                                true (1),
                                false (2)}

        --Communicating Name
            Cname ::=
                [APPLICATION 7]
                    IMPLICIT OCTET STRING (SIZE (8))

        --Distinguished Name
```

```
    DistinguishedName ::= SEQUENCE OF Rdn



--Fibre Channel Attribute
    FCattribute ::= CHOICE{
                DistinguishedAttribute,
                 NonDistinguishedAttribute}


    DistinguishedAttribute ::= CHOICE{
                endPointName     EndPointName,
                nPortName        NportName,
                ipAddress        IPaddress,
                root             Root}




    NonDistinguishedAttribute ::= CHOICE{
                initialPas          InitialPas,
                nPortID              NportID,
                aliasID              AliasID,
                cName                Cname,
                fc4DataType          FC4DataType,
                symbolicName         SymbolicName,
                aoName               AOname,
                objectClass          ObjectClass,
                ulpSpecificInfo      ULPspecificInfo,
                ieeeMulticastGroup IEEEmulticastGroup}

--FC4DataType (e.g., SCSI FCP)
    FC4DataType ::=
        [APPLICATION 8]
            IMPLICIT INTEGER (0..255)
                                --as per FC-PH Rev 4.2 Table 36

--Initial Process Associator
    InitialPas ::=
        [APPLICATION 2]
            IMPLICIT OCTET STRING (SIZE (8))

--Invoke ID
    InvokeID ::=
        INTEGER (0..'ffffffff'H)

--IP Address
    IPaddress ::=
        [APPLICATION 3]
            IMPLICIT OCTET STRING (SIZE (4))
```

Annex C
(Informative)

C. Sample Directory Transactions

This annex provides three sample directory transactions:
1. Registration of an IP node with a single N_Port;
2. Query for an N_Port Address ID; and
3. De-registration of an N_Port that has been re-assigned a different Port ID.

Each transaction is illustrated with the ASN.1 value notation on the related message information unit. An ASN.1 encoding based on BER is provided for the first example. The others are left as an exercise for the enthusiatic reader. Note that there is more than one possible BER encoding and the sample encoding is provided only as a guide. Figure C.1 illustrates a sample Directory Information Tree (DIT) maintained by a Directory Server and the sample transactions are made with respect to it. For the specification of ASN.1 and the BER, refer to International standards ISO 8824 and ISO 8825.

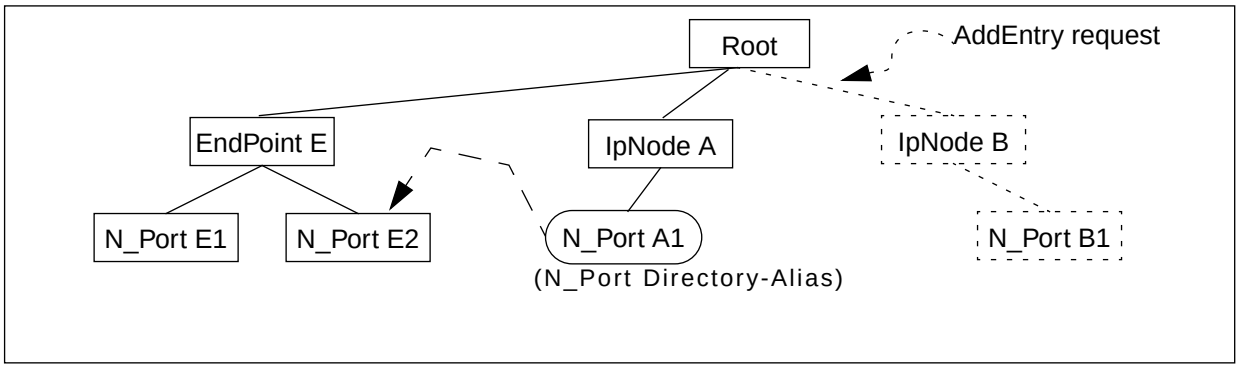


Figure C.1 - A sample Directory Information Tree

C.1 Registration of an IP Node with a single N_Port

Assumptions:
- IP Node B object entry does not exist in the DIT;
- IP Node B has the IP address 18.3.4.5;
- N_Port B1 has the Portname whose value in IEEE address format is 08:00:09:18:DE:25;
- the N_Port Address ID has been assigned as 012C03h;
- the N_Port supports IS8802-2 LLC/SNAP.

Using the ASN.1 value notation, an instance of the request message looks like:

```
requestMsg SEQUENCE {
   version version-1,
   addEntryRequest {-- IMPLICIT SEQUENCE
      invokeID ‘0120BEFE’H,-- some arbitrary value
      -- first object entry
      primary {-- IMPLICIT PrimaryEntry
         object[3]{-- IMPLICIT DistinguishedName (SEQUENCE OF)
            ‘12030405’H,-- Rdn of Ip Node B
```

```
            }-- end of DistinguishedName
            entry[4] { -- IMPLICIT SEQUENCE OF FCattribute
               objectClass 3
            } -- end of entry[4]
         } -- end of primary
         subordinateList{-- IMPLICIT SEQUENCE OF SubordinateEntry
            {  -- SubordinateEntry
               object nPortName '100008000918DE25'H, -- RDN
               entry {  -- IMPLICIT SEQUENCE OF FCAttribute
                  nPortID '012C03'H,-- mandatory attribute
                  fc4DataType 5, -- IS8802-2 LLC/SNAP
                  objectClass 1
               } -- end of attribute entry
            }  -- end of one SubordinateEntry
      } -- end of addEntryRequest
   } -- end of requestMsg
```

The encoding is shown in , with the second last column being the actual byte value in hex format.

Table C.1 - A sample ASN.1 BER encoding of the request message.

Description	Hex Value	Byte
reqMsg: Tag (SEQUENCE, Constructed)	30	1
reqMsg: Length (55)	37	2
reqMsg > value of reqMsg > version: Tag (INTEGER, Primitive)	02	3
reqMsg > value of reqMsg > version: Length (1)	01	4
reqMsg > value of reqMsg > version: Value (0)	00	5
reqMsg > value of reqMsg > addEntryRequest: Tag (CONTEXT 7, Constructed)	A7	6
reqMsg > value of reqMsg > addEntryRequest: Length (50)	32	7
reqMsg > value of reqMsg > addEntryRequest > value of AddEntryRequest > invokeID[0]: Tag (CONTEXT 0, Primitive)	80	8
reqMsg > value of reqMsg > addEntryRequest > value of AddEntryRequest > invokeID[0]: Length (4)	04	9
reqMsg > value of reqMsg > addEntryRequest > value of AddEntryRequest > invokeID[0]: Value (0120BEFEh)	01	10
	20	11
	BE	12
	FE	13
reqMsg > value of reqMsg > addEntryRequest > value of AddEntryRequest > primary[1]: Tag (CONTEXT 1, Constructed)	A1	14
reqMsg > value of reqMsg > addEntryRequest > value of AddEntryRequest > primary[1]: Length (13)	0D	15
reqMsg > value of reqMsg > addEntryRequest > value of AddEntryRequest > primary[1] > value > object[3] > value > Ip Node > value: Tag (CONTEXT 3, Constructed)	A3	16
reqMsg > value of reqMsg > addEntryRequest > value of AddEntryRequest > primary[1] > value > object[3] > value > Ip Node > value: Length (6)	06	17
reqMsg > value of reqMsg > addEntryRequest > value of AddEntryRequest > primary[1] > value > object[3] > value > Ip Node > value: Tag (APPLICATION 3, Primitive)	43	18
reqMsg > value of reqMsg > addEntryRequest > value of AddEntryRequest > primary[1] > value > object[3] > value > Ip Node > value: Length (4)	04	19
reqMsg > value of reqMsg > addEntryRequest > value of AddEntryRequest > primary[1] > value > object[3] > value > Ip Node > value: Value (IP address 18.3.4.5)	12	20
	03	21
	04	22
	05	23
reqMsg > value of reqMsg > addEntryRequest > value of AddEntryRequest > primary[1] > value > entry: Tag (Context 4, Constructed)	A4	24
reqMsg > value of reqMsg > addEntryRequest > value of AddEntryRequest > primary[1] > value > entry: Length (3)	03	25

Table C.1 - A sample ASN.1 BER encoding of the request message.(Concluded)

Description																					Hex Value	Byte
... reqMsg ...	... value (of reqMsg)	... AddEntry ...	... value (of AddEntry) ...	subbordinateList	primary																52	26
																				Length(1)	01	27
																				Value (ipNode)	03	28
					value															Tag (CONTEXT 2, Constructed)	A2	29
																				Length (24)	18	30
					Value of subordinateList															Tag(SEQUENCE OF, Constructed)	30	31
																				Length (25)	19	32
																				Tag (CONTEXT 5, Constructed)	A5	33
																				Length (10)	0A	34
																				Tag (APPLICATION 6, Primitive)	46	35
																				Length (8)	08	36
																				Value (IEEE 08:00:09:18:de:25) in IEEE 802.1A bit order	10	37
																					00	38
																					10	39
																					00	40
																					90	41
																					18	42
																					7B	43
																					A4	44
																				Tag (CONTEXT 6, Constructed)	A6	45
																				Length (11)	0B	46
																				Tag (APPLICATION 4, Primitive)	44	47
																				Length (3)	03	48
																				Value ($012C03_{16}$)	01	49
																					2C	50
																					03	51
																				Tag (APPLICATION 8, Primitive)	48	52
																				Length (1)	01	53
																				Value (IS8802-2 LLC/SNAP)	05	54
																				Tag (APPLICATION 18, Primitive)	52	55
																				Length (1)	01	56
																				Value (nPort)	01	57

Assuming that the AddEntry request is successful, the corresponding response in ASN.1 value notation shall look like:

```
responseMsg SEQUENCE {
   version version-1,
   AddEntryResult -- IMPLICIT INTEGER
      '0120BEFE'H -- the invokeID in the requestMsg
```

An encoding of the responseMsg is shown in the table below:

Table C.2 – Response Message

Description				Hex Value	Byte
responseMsg			Tag (SEQUENCE, Constructed)	30	1
			Length (9)	09	2
	value of responseMsg	version	Tag (INTEGER, Primitive)	02	3
			Length (1)	01	4
			Value (0)	00	5
		addEntryResult	Tag (CONTEXT 8, Constructed)	A8	6
			Length (4)	04	7
			Value (0120BEFEh)	01	8
				20	9
				BE	10
				FE	11

C.2 Query for An N_Port Address ID given its IP address

Assumptions:

- IpNode A and B entries now exist in the DIT;
- IpNode A has a subordinate N_Port directory-alias entry which *points to* N_Port E2 entry;
- IpNode A has an IP address 18.3.4.2;
- N_Port E2 also supports ISO/IEC 8802-2 LLC/SNAP, its PortName in IEEE address format is 08:00:09:18:fa:ce, and its N_Port Address ID has been assigned to be 012C08h;
- the EndPointName of EndPoint E has an IEEE address, 08:00:09:18:11:01.

Using the ASN.1 value notation, an instance of such a search message looks like:

```
searchMsg SEQUENCE {
   version version-1,
   Search { -- IMPLICIT SEQUENCE
      invokeID ‘0130000E’H,-- some arbitrary value
      -- first object entry
      baseobject { -- IMPLICIT DistinguishedName
         Root, -- start from the root
      }
      subset wholeSubtree,
      filter {
         FilterItem {
            equality ipAddress ‘12030402’H -- 18.3.4.2
         }
      },
      passBaseObject true,
      searchDirectoryAliases true,
      selection {
         select { -- SEQUENCE OF AttributeDescriptor
            nPortID,
            nPortName
```

```
            }
         }
         -- no pagedResults
         -- no COMPONENTS of CommonArguments
      } -- end of Search SEQUENCE
   } -- end of searchMsg
```

The corresponding sucessful response in ASN.1 value notation shall look like:

```
   searchResultMsg SEQUENCE {
      version version-1,
      SearchResult { -- IMPLICIT SEQUENCE
         invokeID '0130000E'H, -- as in the searchMsg
         entries {   -- IMPLICIT SEQUENCE OF EntryInformation
            { -- SEQUENCE (entryInfo 1)
               name { -- IMPLICIT DistinguishedName
                  endPointName '1000080009181101'H, -- EndPoint E
                  nPortName '100008000918FACE'H -- NportName E2
               },
               information { -- IMPLICIT SEQUENCE OF FCAttribute
                  nPortID '012C08'H,
                  nPortName '100008000918FACE'H
               }
            }        }
         -- no limitProblem
         -- no queryReference
         -- no COMPONENTS OF CommonResults
   }
```

C.3 Re-registration of an N_Port that has been re-assigned a different Port ID

Assumptions:

- N-Port B1 went off-line and came on-line;
- its port address ID has been re-assigned to 012CADh
- the old entry has not been deleted.

The registration message in ASN.1 value notation looks like:

```
   registrationMsg SEQUENCE {
      version version-1,
      AddEntry {-- IMPLICIT SEQUENCE
         invokeID 'DEADBEEF'H,-- some arbitrary value
         -- first object entry
         primary {-- IMPLICIT PrimaryEntry
            object[3] {-- IMPLICIT DistinguishedName (SEQUENCE OF)
               '12030405'H,-- RDN of IpNode B
               n-portName '10000800091DE25'H -- RDN of N_Port B1
            }-- end of DistinguishedName
            entry { -- IMPLICIT SEQUENCE OF FCAttribute
                  n-PortID '012CAD'H,
                  fc4DataType 5, -- IS8802-2 LLC/SNAP
                  fc4DataType 8, -- SCSI-FCP
                  objectClass 1
```

```
            }
         }, -- end of primary
      }
      -- no common arguments
   } -- end of registrationMsg
```

The Directory Server detects that the N_Port entry exists and responds with an error message. In ASN.1 value notation, it looks like:

```
   errorMsg SEQUENCE {
      version version-1,
      Error {  -- IMPLICIT SEQUENCE
         invokeID ‘DEADBEEF’H,-- as in registrationMsg
         updateError { -- IMPLICIT UpdateError
            entryAlreadyExists
      }
   } -- end of errorMsg
```

Annex D

(Informative)

D. Bibliography

ISO/IEC 8824 Information Technology - Open Systems Interconnection- Specification of Abstract Syntax Notation One (ASN.1).

ISO/IEC 8825 Information Technology - Open Systems Interconnection- Specification of the Basic Encoding Rules of Abstract Syntax Notation One (ASN.1).

Internet Standard (RFC 768), User Datagram Protocol, August1980.

Internet Standard 15 (RFC1157), A Simple Network Management Protocol (SNMP), May 1990.

Internet Standard 16 (RFC1155), Structure and Identification of Management Information for TCP/IP-based Internets, May 1990.

Internet Request For Comments 1441 (RFC1441), Introduction To Version 2 of the Internet-Standard Network Management Framework, April 1993.

Internet Request For Comments 1442 (RFC1442), Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2), April 1993.

Internet Request For Comments 1448 (RFC1448), Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2), April 1993.

Internet Request For Comments 1449 (RFC1449), Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2), April 1993.

Internet Request For Comments 1446 (RFC1446), Security Protocol for version 2 of the Simple Network Management Protocol (SNMPv2), April 1993.

Internet Draft, IP and ARP on Fibre Channel (FC), July 1994.

Internet Draft, Definitions of Managed Objects for the Node in Fibre Channel Standard using SMIv2, August 1994.

Internet Draft, Definitions of Managed Objects for the Fabric Element in Fibre Channel Standard, June 1995.

Index

W