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

X3.288-199x
T11/Project 1134-D/Rev 5.3

FIBRE CHANNEL

GENERIC SERVICES - 2

(FC-GS-2)

REV 5.3

NCITS working draft proposed
American National Standard
for Information Technology

November 25, 1998

Secretariat: Information Technology Industry Council

NOTE:
This is a working draft American National Standard of Accredited Standards Committee NCITS. As such this is not a completed standard. The T11 Technical Committee or anyone else may modify this document as a result of comments received anytime, or during a future public review and its eventual approval as a Standard. Use of the information contained herein is at your own risk.

POINTS OF CONTACT:

Roger Cummings (T11 Chairman)
Distributed Processing Technology
140 Candace Drive
Maitland, FL 32751
Voice: (407) 830-5522 x348
Fax: (407) 260-5366
EMail: cummings_roger@dpt.com

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

Jeffrey Stai (Technical Editor & Facillitator)
Brocade Communications Systems, Inc.
15707 Rockfield Boulevard, Suite 215
Irvine, CA 92618
Voice: (949) 455-2908
Fax: (949) 455-9287
E-mail: stai@brocade.com

Release Notes for version 5.3: Below is a summary of the work done to the prior version (5.2) to create this version.

- Edits made in response to the T11 “letter” ballot. See 98-559v0 and 98-560v0.

Release Notes for version 5.2: Below is a summary of the work done to the prior version (5.1) to create this version.

- Update to name server to create new IP address object relative to Port Identifier object. This includes the appropriate Query and Register commands to support it. The name of the Node-relative object was changed; nothing else about this existing object was changed.
- Other assorted minor edits to prepare for a T11 “letter” ballot.

Release Notes for version 5.1: Below is a summary of the work done to the prior version (5.0) to create this version.

- Assorted minor edits to prepare for a T11 “letter” ballot.

Release Notes for version 5.0: Below is a summary of the work done to the prior version (4.0) to create this version.

- The version is bumped to 5.0 because the directory server was removed by unanimous decision of T11, and because the top level clause numbering relative to FC-GS changed.
- A few clarifications were made to the Alias Server.
- Otherwise, assorted minor edits.

Release Notes for version 4.0: Below is a summary of the work done to the prior version (0.1) to create this version.

- The version is bumped to 4.0 to follow the convention of using the next higher whole rev (FC-GS ended on version 3.3). FC-GS clause numbering was retained whenever possible (which was nearly all of the time).
- The maximum/residual size field was added to the CT_HDR.
- The Overview for directory services was re-organized to include the Name Server. Some sections got their numbering changed as a result.
- The latest version of the Name Server was incorporated. Also, the FS_RJT rules were grouped mostly into a single location for ease of access and to save trees. The information presented in some of the tables (like the command list) was expanded for ease of use; also, many of the command mnemonics were made more, well, mnemonic (a cross reference between the old and new has been added as an annex). The intent was to improve the readability of this section, make it a little less dense.
- The Security-Key Server had numerous internal inconsistencies which I attempted to clean up without breaking it. The services were described in a mixture of ELS and FCS descriptions; since this is an FCS document, I normalized them to be FCS Sequences. I also made several notations consistent, or tried to. Someone should verify that I didn’t break it!
- Every chapter of this document was in its own file, with its own style, design, and paragraph format definitions. I have attempted to unify all chapters to the same format, mostly for my own sanity. I have also attempted to unify the documentation style somewhat, with consistent nomenclature and appearance. Again, I hope that nothing was broken in the process. Better read it to be sure...
- I cleaned up some inconsistencies in the FC-CT clause; for example, sometimes “FS” is used, other times “FCS”. I normalized on “FS”.
- Annex B had about two pages or so of information hidden in the file, such that it was never printed! I have exposed that information in this rev. Did anyone read this stuff when it was FC-GS?

ANSI ®
dp ANS X3.288-199x

draft proposed American National Standard
for Information Technology

Fibre Channel —
Generic Services - 2 (FC-GS-2)

Secretariat

Information Technology Industry Council

Approved,199x

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. Services described include name services, time services, alias services, SNMP services, and security key distribution.

This standard replaces X3.288-1996.

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.

PATENT STATEMENT

The developers of this Standard have requested that holder's of patents that may be required for the implementation of the Standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this Standard.

As of the date of publication of this Standard and following calls for the identification of patents that may be required for the implementation of the Standard, no such claims have been made. No further patent search is conducted by the developer or the publisher in respect to any Standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this Standard.

Published by

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

Printed in the United States of America

INSERT CODE HERE

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

The Fibre Channel Generic Services (FC-GS-2) 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 T11 of Accredited Standards Committee NCITS during 1997-1998. The standards approval process started in 1998. This document includes annexes which are informative and are not considered part of the standard.

Requests for interpretation, suggestions for improvement or addenda, or defect reports are welcome. They should be sent to the NCITS 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, NCITS. Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, NCITS had the following members:

Karen Higginbottom, Chair
David Michael, Vice Chair
Monica Vago, Secretary

Organization Represented Name of Representative

(Membership list to be added)

Subcommittee T11 on Computer Input/Output Interfaces, which reviewed this standard, had the following members:

xxx, Chair
Edward Grivna, Vice-Chair

Organization Represented Name of Representative

(Membership list to be added)

Introduction

FC-GS-2 is one of the Fibre Channel family of standards. This family includes ANSI X3.230, FC-PH, which specifies the Physical and SIgnalling Interface. ANSI X3.297, FC-PH-2, and ANSI X3.303, FC-PH-3, specify enhanced functions added to FC-PH. ANSI X3.289, FC-FG, and ANSI X3.321, FC-SW, are documents related to Fabric requirements. ANSI X3.272, FC-AL, specifies the arbitrated loop topology.

FC-GS-2 provides generic services that may be utilized by any upper layer protocol that makes use of Fibre Channel as a transport. These upper layer protocols are usually called “FC-4s”. Some FC-4s include ANSI X3.269, SCSI-FCP, and ANSI X3.271, FC-SB. Other FC-4s are under development to support data storage, server clusters, and audio-visual applications.

Acknowledgements

The technical editor would like to thank the following individuals for their special contributions to this standard:

– R. Bryan Cook

– Edward Jacques

– Bent Stoevhase

– Kha Sin Teow

Contents Page

Contents Page

Contents Page

Contents Page

Figures Page

Tables Page

Tables Page

draft proposed American National Standard
for Information Technology —

Fibre Channel —
Generic Services - 2 (FC-GS-2)

1 Scope

FC-GS-2 describes in detail the basic Fibre Channel services introduced in ANSI X3.230, FC-PH.

The Fibre Channel services described in this document are:

– Name Service

– SNMP based Management Service

– Time Service

– Alias Service

– Security-key Service

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 the 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), and approved foreign Standards (including BSI, JIS, and DIN). For further information, contact ANSI Customer Service Department at 212-642-4900 (phone), 212-302-1286 (fax) or via the World Wide Web at http://www.ansi.org.

Additional availability contact information is provided below as needed.

2.1 Approved references

[1] ANSI X3.230:1994, *Information Technology - Fibre Channel Physical and Signaling Interface (FC-PH)*.

[2] ANSI X3.297:1997, *Information Technology - Fibre Channel - Physical and Signalling Interface-2 (FC-PH-2)*

[3] ANSI X3.272:1996, *Information Technology - Fibre Channel - Arbitrated Loop (FC-AL).*

[4] ANSI X3.289:1996, *Information Technology - Fibre Channel - Fabric Generic (FC-FG)*.

[5] ANSI X3.303:1998, *Fibre Channel - Physical and Signalling Interface-3 (FC-PH-3)*.

[6] ANSI X3.321-1998, *Fibre Channel - Switch Fabric (FC-SW)*.

[7] ANSI NCITS TR-20-1998, *Fibre Channel - Fabric Loop Attachment (FC-FLA)*.

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 document, or regarding availability, contact the relevant Standards body or other organization as indicated.

NOTE – For more information on the current status of a document, contact the NCITS Secretariat at the address listed in the front matter. To obtain copies of this document, contact Global Engineering at the address listed in the front matter, or the NCITS Secretariat.

[8] ANSI X3.xxx-199x, *Fibre Channel - Arbitrated Loop (FC-AL-2),* T11/Project 1133D/Rev 6.2

[9] ANSI X3.xxx-199x, *Fibre Channel - Backbone (FC-BB),* T11/Project 1238D/Rev 1.0

[10] ANSI X3.xxx-199x, *Fibre Channel - Switch Fabric -2 (FC-SW-2),* T11/Project 1305D/Rev 4.0

[11] ANSI X3.xxx-199x, *Fibre Channel - Virtual Interface (FC-VI),* T11/Project 1332D/Rev 0.3

2.3 Other references

The following document is available from the Fibre Channel Association (FCA), 2570 West El Camino Real, Ste. 304, Mountain View, CA 94040-1313; (800) 272-4618 or (650) 949-6730 (phone); or via e-mail, fca@fibrechannel.com.

[12] FCSI-101*, FCSI Common FC-PH Feature Sets Used in Multiple Profiles*, Rev 3.1

The following documents are available from the RFC Editor, Information Sciences Institute, University of Southern California, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292-6695; (310) 822-1511 or (310) 823-6714 (fax); http://www.isi.edu/.

[13] RFC 768, *User Datagram Protocol*, August1980.

[14] RFC 1157, *A Simple Network Management Protocol (SNMP)*, May 1990.

[15] RFC 1155, *Structure and Identification of Management Information for TCP/IP-based Internets*, May 1990.

[16] RFC 1901, *Introduction to Community-based SNMPv2*, January 1996.

[17] RFC 1902, *Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)*, January 1996.

[18] RFC 1903, *Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)*, January 1996.

[19] RFC 1905, *Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)*, January 1996.

[20] RFC 1906, *Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2)*, January 1996.

[21] RFC 2373*, IP Version 6 Addressing Architecture*, July 1998.

3 Definitions and conventions

For FC-GS-2, 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 directory: a repository of information about objects which provides directory services to its users, thereby allowing access to the information.

3.1.4 Link Service Facilitator: The entity at well-known address hex 'FFFFFE'; known as the "Fabric F_Port" in FC-PH, the address identifier to which Fabric Login is directed.

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

3.1.6 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.

3.1.7 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.

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

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

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

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

3.1.12 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-2, a number of conditions, mechanisms, sequences, parameters, events, states, or similar terms are printed with the first letter of each word in uppercase and the rest lowercase (e.g., Exchange, Class). Any lowercase uses of these words have the normal technical English meanings.

Numbered items 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.

3.2.1 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 hex '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 clause 3.1 "Definitions".

AIU	Application Information Unit
AS	Alias Service
CT	Common Transport
CT_IU	Common Transport Information Unit
CT_HDR	Common Transport Header
D_ID	Destination address identifier
FS	Fibre Channel Service
FS_ACC	Fibre Channel Service Accept Information Unit
FS_IU	Fibre Channel Service Information Unit
FS_REJ	Fibre Channel Service Reject Information Unit
FS_REQ	Fibre Channel Service Request Information Unit
IN_ID	Initial Identifier
IP	Internet Protocol
IPA	Initial Process Associator
IU	Information Unit
NAA	Network Address Authority
NS	Name Server
NS_DU	Name Server Data Unit
NSM	Native SNMP Mapping
SI	Sequence Initiative
S_ID	Source address identifier
SNMP	Simple Network Management Protocol
TS	Time Service
UDP	User Datagram Protocol
ULP	Upper Level Protocols

4 Common transport for FC services (CT)

4.1 Overview

Fibre Channel services share a common transport at the FC-4 level. The common transport provides a Fibre Channel service application (e.g. name 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 units (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 the 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.

NOTE – Not all Fibre Channel Generic Services are mapped to CT.

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.

The types of Fibre Channel services described by this document that are mapped to CT are:

– Name service;

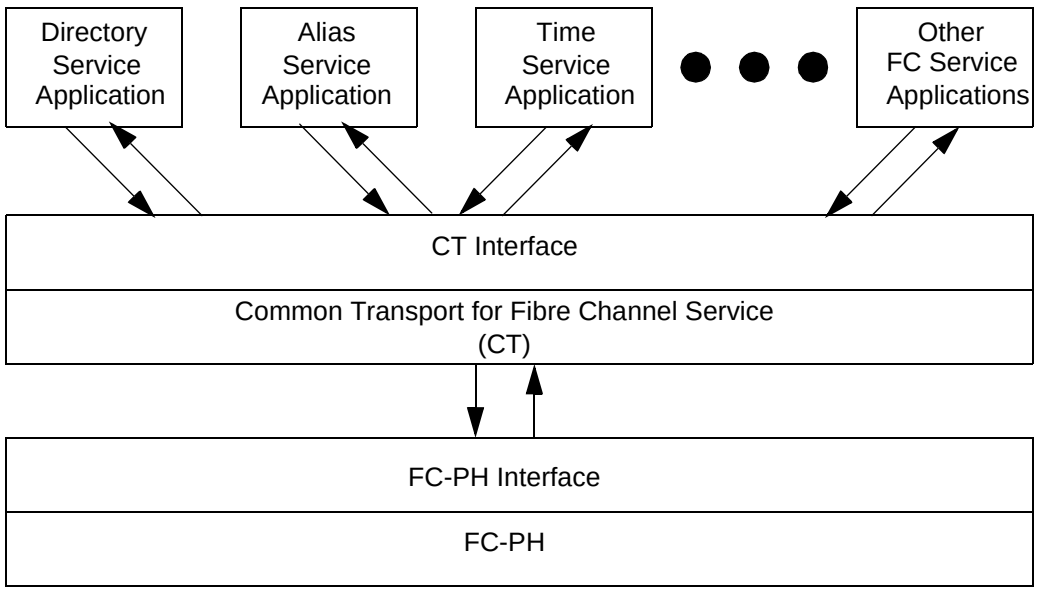


Figure 1 – Relationship of CT with its ULP and FC-PH

– Alias service;

– Time service;

– Security-key service.

The types of Fibre Channel services described by this document that are not mapped to CT are:

– SNMP based management 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 (see 4.7);

– 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.

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 enforce this restriction on behalf of the application. It may do so as described in 4.5.

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. The CT_HDR is the first sixteen bytes of the CT_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.

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	FS_Type	FS_Subtype	Options	Reserved
2	Command/Response code		Maximum/Residual Size	
3	Reserved	Reason code	Reason Code Explanation	Vendor Unique
4	Application Information Unit			
...				
n				

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.

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.

NOTE – The IN_ID field is provided to allow distributed servers to know the identity of the original requestor. This field is not intended to enable third-party responses by distributed servers.

4.3.1.3 FS_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 FS_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 FS_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 hex 'FFFFFC, then the FS_Subtype field is used to distinguish these different directory services.

Table 2 – FS_Type values

Encoded value (hex)	Description
00-1F	Vendor Unique
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

4.3.1.5 Options Field

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;

– zero indicates a single bidirectional Exchange;

– one indicates multiple Exchanges;

– bits 6-0 are reserved.

4.3.1.6 Command/response code field

This field indicates whether the CT_IU is an FS_IU (see 4.6). 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 (hex)	Description
0000	Non-FS_IU
0001-7FFF	FS_REQ IU
8001	FS_RJT IU
8002	FS_ACC IU
other values	Reserved

4.3.1.7 Maximum/Residual Size field

This field manages the size of the information returned in an FS_ACC IU. The sender of an FS_REQ may specify the maximum size of the FS_ACC it is able to receive. If the recipient of the FS_REQ has more available information to send than allowed by the maximum size, it shall indicate the excess residual size in the FS_ACC. The values for Maximum/Residual Size are interpreted as follows:

– In the FS_REQ IU:

– hex ‘0000’: No Maximum Size indicated; the FS_ACC may be any size.

– hex ‘FFFF’: This value is reserved for the FS_REQ IU.

– Any other value: The Encoded Value indicates the maximum number of words that shall be sent in the FS_ACC IU, not inclusive of the CT_HDR.

– In the FS_ACC IU:

– hex ‘0000’: All available information was returned in the FS_ACC IU.

– hex ‘FFFF’: The Encoded Value indicates greater than 65534 available words of information were not sent in the FS_ACC IU, in excess of the Maximum Size specified in the FS_REQ.

– Any other value: The Encoded Value indicates the number of available words of information that were not sent in the FS_ACC IU, in excess of the Maximum Size specified in the FS_REQ.

– For the FS_RJT IU and any other IU, this field is reserved.

4.3.1.8 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 hex ’0000’, this field shall not be used.

4.3.1.9 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 hex ’0000’, this field shall not be used.

4.3.1.10 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 hex '0000', this field shall not be used.

4.3.2 Application Information Unit

The Application Information Unit contains information specific to the FS_Type, FS_Subtype, and Command/Response code.

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 shall 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 one. Transfer of the SI to the destination CT shall be considered a protocol error and the destination CT shall terminate the Exchange.

NOTE – Asynchronous transaction mode is not used by any service application defined in this standard. This mode may be removed in a future version of FC-GS.

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 zero.

An Exchange created by a CT is to be used only for a specific service application, and shall not be shared with another service application. For example, a single Exchange cannot be used for both Name Server and Alias Server requests.

Each CT_IU shall be mapped into a Sequence. The CT_IU tables for asynchronous and synchronous transactions are shown in table 4 and table 5 respectively. The “F/M/L” column indicates whether the Sequence may be the First, Middle, or Last Sequence of the Exchange. The “SI” column indicates whether Sequence Initiative is Held or Transferred. The “M/O” column indicates whether support for the Sequence is Mandatory or Optional.

Table 4 – IU table for asynchronous transmission

IU Name	Information Category	Payload	F/M/L	SI	M/O
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	H	M

Table 5 – IU table for synchronous transaction

IU Name	Information Category	Payload	F/M/L	SI	M/O
Request	2	One request CT information unit	F,M	T	M
Response	3	One response CT information unit	M,L	T	M

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 for a description of the request and response CT_IUs.

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 “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 Continue Sequence condition

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

4.4.13 Exchange reassembly

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

4.4.14 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.

4.4.15 Optional headers

The use of any Optional Header is not defined by this standard.

4.5 Error handling

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

- invalid CT_HDR, CT protocol;
  - invalid/undefined FS_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. Synchronous mode transactions shall be terminated by the Exchange Originator or the Exchange Responder using the Abort Sequence Link Service with the LS_bit set to terminate the Exchange.

Asynchronous mode transactions shall be terminated as follows:

- 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.

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

An FS_REQ IU is a CT_IU in which the command/response code field contains a command code value of hex ’0001’-hex ’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

An FS_ACC IU is a CT_IU in which the command/response code field contains a value of hex ’8002’. The FS_ACC shall notify the Originator 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

An FS_RJT IU is a CT_IU in which the command/response code field contains a value of hex '8001'. The FS_RJT shall notify the Originator 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
0000 0001	Invalid command code
0000 0010	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.

NOTE – Service applications that make use of synchronous transaction mode will typically correlate a request to a response through the use of Exchange IDs. Service applications that make use of asynchronous transaction mode will typically include a tag value within the AIU to correlate requests with responses.

5 Overview of directory (name) service

This clause introduces the standard directory service.

NOTE – The X.500 directory server that was defined in the first publication of this standard has been removed.

5.1 Introduction to the directory service model

The directory service facility is provided as a means to discover information about Nodes and Ports attached to a Fabric. This service is provided through well-known address hex 'FFFFFC'.

This document defines a standard model for requests and responses to access the directory service data base. This standard does not define the structure of the data base.

NOTE – The Name Service is a Required feature of FC-FLA compliant Fabrics.

5.1.1 Directory service subtypes

Table 7 defines the FS_Subtype codes for the directory service models.

Table 7 – Directory service subtype values

Values in hex	Description
01	Reserved
02	Name Service
80-EF	FC-4 specific service
other values	Reserved

NOTE – Value hex '01' indicated the X.500 directory service defined in the first publication of this standard.

5.1.2 Name Service

The Name Service provides a simple fixed request and response model. A key object is provided in the request, and the requested object is provided in the response.

5.1.3 Other models

In addition to the standard directory service models, 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.

6 Name Server

The Name Server provides a way for N_Ports and NL_Ports to register and discover Fibre Channel attributes. Registrations may be performed by a third party. However, the Name Server may refuse such third party registration for unspecified reasons. Once registered, the attributes are made available to requestors.

Requests for the Name Server are carried over the common transport FC–CT service (see clause 4). Three types of requests are defined for the Name Server, as shown in table 8.

Table 8 – Name Server – Request types

Code (hex)	Description
01xx	Get Object(s) (Query)
02xx	Register Object
03xx	Deregister Object(s)

Table 9 lists the different Fibre Channel objects defined for the Name Server.

Table 9 – Name Server – Objects

Object number (hex)	Object Mnemonic	Object Name	Description
0	A	All Name Server objects	Contains all objects listed below
1	ID	Port Identifier	3-byte address identifier
2	PN	Port Name	8-byte Name_Identifier
3	NN	Node Name	8-byte Name_Identifier
4	CS	Class of Service	4-byte bit field, one bit per Class supported
5	IP	IP Address (Node)	32-bit or 128-bit Internet Protocol address
6	IPA	Initial Process Associator	8-byte Process_Associator
7	FT	FC–4 TYPEs	32-byte bit field, one bit per TYPE supported
8	SPN	Symbolic Port Name	variable length (0 to 255-byte) field
9	SNN	Symbolic Node Name	variable length (0 to 255-byte) field
A	PT	Port Type	1-byte encoded Port Type
B	IPP	IP Address (Port)	32-bit or 128-bit Internet Protocol address
C	FPN	Fabric Port Name	8-byte Name_Identifier
D	HA	Hard Address	3-byte address identifier

NOTE – The numbers assigned to the Name Server objects are used in the formation of the request command codes. The mnemonics assigned are used to form the command mnemonics.

The Name Server is intended to be distributed among Fabric Elements, making Name Service immediately available to N_Ports and NL_Ports once they have successfully completed Fabric Login. However, the Name Service is not restricted or required to be part of a Fabric, and may be located in any N_Port or NL_Port. The Name Server may be made available on any Fibre Channel topology.

6.1 Name Service protocol

Name Service registration, deregistration and queries are managed through protocols containing a set of request and reply IUs supported by the Name Server.

6.2 Transportation of Name Server IUs

Name Service requests and responses shall be transported between the requestor and the Name Server using the common transport protocol (see clause 4).

6.2.1 Name Server mapping to CT

For a Name Server request, the Name Server payload shall be transported from the requestor to the Name Server using a synchronous CT_IU request. The corresponding Name Server response is transported from the Name Server to the requestor, in the Exchange established by the requestor, using a response CT_IU. CT_IUs, when used for Name Service are known as NS_IUs.

The Application Information Unit (see 4.3.2) of the CT_IU is known as the NS_DU (Name Server Data Unit).

6.2.1.1 Operation timeout

If the requestor does not receive a response NS_DU from the Name Server within a time of 120 seconds, then it shall initiate error recovery for the affected Exchange.

6.2.2 CT_HDR values

The following values shall be set in the CT_HDR both for Name Server request and their responses:

– Revision field: hex ‘01’;

– IN_ID: reserved (hex ‘00 00 00’), may be non-zero in Name Server responses;

– FS_Type: hex ‘FC’ (Directory Service application);

– FS_Subtype: hex ‘02’ (Name Service, see table 7);

– Options: hex ‘00’ (single bidirectional Exchange);

– Command Code: see table 10 for request command codes.

Table 10 – Name Server – Requests

Code (hex)	Mnemonic (see note 1)	Description	Object(s) in Request payload:	Object(s) in Accept payload:
0100	GA_NXT	Get all next	Port Identifier	All
0101	GI_A	Get identifiers - scope	none (see note 3)	none (see note 3)
0112	GPN_ID	Get Port Name	Port Identifier	Port Name
0113	GNN_ID	Get Node Name - Port Identifier	Port Identifier	Node Name
0114	GCS_ID	Get Class of Service	Port Identifier	Class of Service
0117	GFT_ID	Get FC-4 TYPEs	Port Identifier	FC-4 TYPEs
0118	GSPN_ID	Get Symbolic Port Name	Port Identifier	Symbolic Port Name
011A	GPT_ID	Get Port Type	Port Identifier	Port Type
011B	GIPP_ID	Get IP Address (Port) - Port Identifier	Port Identifier	IP Address (Port)
011C	GFPN_ID	Get Fabric Port Name - Port Identifier	Port Identifier	Fabric Port Name

Table 10 – Name Server – Requests

Code (hex)	Mnemonic (see note 1)	Description	Object(s) in Request payload:	Object(s) in Accept payload:
011D	GHA_ID	Get Hard Address - Port Identifier	Port Identifier	Hard Address
0121	GID_PN	Get Port Identifier - Port Name	Port Name	Port Identifier
012B	GIPP_PN	Get IP Address (Port) - Port Name	Port Name	IP Address (Port)
0131	GID_NN	Get Port Identifiers - Node Name	Node Name	List of Port Identifiers
0135	GIP_NN	Get IP Address (Node)	Node Name	IP Address (Node)
0136	GIPA_NN	Get Initial Process Associator - Node Name	Node Name	Initial Process Associator
0139	GSNN_NN	Get Symbolic Node Name	Node Name	Symbolic Node Name
0153	GNN_IP	Get Node Name - IP Address (Node)	IP Address (Node)	Node Name
0156	GIPA_IP	Get Initial Process Associator - IP Address (Node)	IP Address (Node)	Initial Process Associator
0171	GID_FT	Get Port Identifiers - FC-4 TYPE	none (see note 2)	List of Port Identifiers
0173	GNN_FT	Get Node Names - FC-4 TYPE	none (see note 2)	List of Port Identifiers and Node Names
01A1	GID_PT	Get Port Identifiers - Port Type	Port Type	List of Port Identifiers
01B1	GID_IPP	Get Port Identifiers - IP Address (Port)	IP Address (Port)	List of Port Identifiers
01B2	GPN_IPP	Get Port Name - IP Address (Port)	IP Address (Port)	Port Name
0212	RPN_ID	Register Port Name	Port Identifier, Port Name	none
0213	RNN_ID	Register Node Name	Port Identifier, Node Name	none
0214	RCS_ID	Register Class of Service	Port Identifier, Class of Service	none
0217	RFT_ID	Register FC–4 TYPEs	Port Identifier, FC-4 TYPEs	none
0218	RSPN_ID	Register Symbolic Port Name	Port Identifier, Symbolic Port Name	none
021A	RPT_ID	Register Port Type	Port Identifier, Port Type	none
021B	RIPP_ID	Register IP Address (Port) - Port Identifier	Port Identifier, IP Address (Port)	none

Table 10 – Name Server – Requests

Code (hex)	Mnemonic (see note 1)	Description	Object(s) in Request payload:	Object(s) in Accept payload:
021C	RFPN_ID	Register Fabric Port Name - Port Identifier	Port Identifier, Fabric Port Name	none
021D	RHA_ID	Register Hard Address - Port Identifier	Port Identifier, Hard Address	none
0235	RIP_NN	Register IP Address (Node)	Node Name, IP Address (Node)	none
0236	RIPA_NN	Register Initial Process Associator	Node Name, Initial Process Associator	none
0239	RSNN_NN	Register Symbolic Node Name	Node Name, Symbolic Node Name	none
0300	DA_ID	De-register all	Port Identifier	none

Notes:

1 These mnemonics are based on the following system: a leading “G” indicates a “Get” request, “R” indicates a “Reqister”, and “D” indicates a “De-register”. The letters between the leading character and the underscore (“_”) indicate the object that the requested operation is performed upon, as defined in table 9. The letters after the underscore indicate the object that the request is based upon; for example, “RPN_ID” is a “Register Port Name based on the Port Identifier”. These mnemonics are changed from earlier versions of the draft of this standard. See Annex C for a cross reference between the original mnemonics and this system.

2 The FC-4 TYPE is specified as an encoded value, not as an object.

3 The GI_A request specifies a scope in the request, and the response contains a list of Domain_IDs or Domain_ID/Area_IDs.

6.3 FC–PH constructs

Before performing any Name Server operation, an N_Port shall perform N_Port Login with the Name Server, at the well-known address hex ‘FF FF FC’. N_Ports or NL_Ports which only use the Name Server to register themselves should perform explicit N_Port Logout once registration has been completed. However, N_Ports and NL_Ports which often query the Name Server may wish to retain the Login to speed future queries or may elect to Logout with the Name Server. The Name Server may perform N_Port Logout (LOGO) if it becomes resource constrained. The Name Server may use a least recently used algorithm in determining which entity to Logout.

6.3.1 Common required FC–2 parameters

6.3.1.1 Class of Service

The Name Server may use any Class of Service.

The Name Server shall send responses in the Class of Service used by the requesting N_Port in the initial Name Service request.

6.3.1.2 Routing control bits

The R_CTL field shall indicate:

– FC–4 Device_Data (hex ‘0’);

– Unsolicited Control (hex ‘2”) in Name Server requests;

– Solicited Control (hex ‘3’) in Name Server responses.

6.3.1.3 Exchange control

Each Name Service request shall be the first Sequence of an Exchange, and its associated response shall be the last Sequence of the same Exchange.

6.3.1.4 Destination ID (D_ID)

This parameter shall be set to the well-known address hex ‘FF FF FC’ in all Name Server requests. In a Name Server response the D_ID shall be the address identifier of the requesting N_Port or NL_Port.

6.3.1.5 Source ID (S_ID)

This parameter shall identify the requesting N_Port or NL_Port in all Name Server requests. In a Name Server response the S_ID shall be the well-known address identifier hex ‘FF FF FC’.

6.3.1.6 TYPE

All NS_IUs shall specify the Fibre Channel Services TYPE (hex ‘20’).

6.3.1.7 Error policy

Only the ‘Abort, Discard a single Sequence’ error policy shall be used.

6.3.2 Common optional FC parameters

6.3.2.1 Expiration_Security header

The Expiration_Security header shall not be used.

6.3.2.2 Network header

The Network header shall not be used.

6.3.2.3 Association header

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

6.3.2.4 Device header

The Device Header shall not be used.

6.4 Name Server objects – Formats

The format of the Name Server objects summarized in table 9 are described below. A null value is defined for each Name Server object. This value is used when the Name Server needs to return an FS_ACC NS_DU but no value has been registered for the requested Name Server object.

Table 11 shows the relationship between the different Fibre Channel objects within the Name Server database (note that this only represents the model and does not seek to dictate specific implementation details). The “primary” key for all objects in a Name Server record is the Port Identifier. All objects are ultimately related back to this object. Because a Node may have more than one Port, it is appropriate to have the Node Name act as a “secondary” key for some objects.

6.4.1 Port Identifier – Format

The Port Identifier is a Fibre Channel address identifier, assigned to an N_Port or NL_Port during implicit or explicit Fabric Login. The format for the Port Identifier object, as used by the Name Server, shall be identical to the address identifier format defined in FC–PH.

The Port Identifier serves as the unique data base key for the Name Server.

The null value for the Port Identifier is hex ‘00 00 00’.

6.4.2 Port Name – Format

The format of the Port Name object, as used by the Name Server, shall be identical to the Name_Identifier format defined in FC–PH.

The null value for the Port Name object is hex ‘00 00 00 00 00 00 00 00’.

6.4.3 Node Name – Format

The format of the Node Name object, as used by the Name Server, shall be identical to the Name_Identifier format defined in FC–PH.

The null value for the Node Name object is hex ‘00 00 00 00 00 00 00 00’.

Table 11 – Name Server Database Organization

Primary Key	Indexed Fields	Secondary Key	Indexed Fields
Port Identifier	Port Name		
	Node Name		IP Address (Node)
	Class of Service		Initial Process Associator
	FC-4 TYPEs		Symbolic Node Name
	Symbolic Port Name		
	Port Type		
	IP Address (Port)		
	Hard Address		

6.4.4 Class of Service – Format

The format of the Class of Service object shall be bit mapped as shown below:

Bit 0 – Class F
0 = Class F is not supported by the Port Identifier.
1 = Class F is supported by the Port Identifier.

Bit 1 – Class 1
0 = Class 1 is not supported by the Port Identifier.
1 = Class 1 is supported by the Port Identifier.

Bit 2 – Class 2
0 = Class 2 is not supported by the Port Identifier.
1 = Class 2 is supported by the Port Identifier.

Bit 3 – Class 3
0 = Class 3 is not supported by the Port Identifier.
1 = Class 3 is supported by the Port Identifier.

Bit 4– Class 4
0 = Class 4 is not supported by the Port Identifier.
1 = Class 4 is supported by the Port Identifier.

Bit 5 – Class 5
0 = Class 5 is not supported by the Port Identifier.
1 = Class 5 is supported by the Port Identifier.

Bit 6 – Class 6
0 = Class 6 is not supported by the Port Identifier.
1 = Class 6 is supported by the Port Identifier.

Bits 7 to 31: reserved

The null value for the Class of Service Name Server object is hex ‘00 00 00 00’.

6.4.5 IP Address – Format

Both 32 bit (IPv4) and 128 bit (IPv6) Internet Protocol (IP) address formats may be supported by the Name Server. This object description shall apply to both the IP Address (Port) and the IP Address (Node) objects; however, a Registration for IP Address (Port) shall not affect a value registered for IP Address (Node); and, a Registration for IP Address (Node) shall not affect a value registered for IP Address (Port).

The format of the 32 bit (IPv4) IP address, as used by the Name Server, shall use big endian bit and byte order, within a word, and shall be preceded by a hex ‘00 00 00 00 00 00 00 00 00 00 FF FF’ prefix. This is the format specified in RFC 2373 (see reference [21]). For example, the Name Server format for the 32 bit (IPv4) IP address 198.53.144.31 is hex ‘00 00 00 00 00 00 00 00 00 00 FF FF C6 35 90 1F’. This is broken into words as shown below:

– Word 0 shall contain the most significant word of the 128 bit IP address (hex ‘00 00 00 00’);

– Word 1 shall contain the second most significant word of the 128 bit IP address (hex ‘00 00 00 00’);

– Word 2 shall contain the second least significant word of the 128 bit IP address (hex ‘00 00 FF FF’);

– Word 3 shall contain the least significant word of the 128 bit IP address (hex ‘C6 35 90 1F’).

The format of the 128 bit (IPv6) IP address, as used by the Name Server, shall use big endian bit and byte order, within a word. This is the format specified in RFC 2373. For example, the Name Server format for the 128 bit (IPv6) IP address 1080:0:0:0:8:800:200C:417A is hex ‘10 80 00 00 00 00 00 00 08 08 00 20 0C 41 7A’. This is broken into words as shown below:

– Word 0 shall contain the most significant word of the 128 bit IP address (hex ‘10 80 00 00’);

– Word 1 shall contain the second most significant word of the 128 bit IP address (hex ‘00 00 00 00’);

– Word 2 shall contain the second least significant word of the 128 bit IP address (hex ‘00 08 08 00’);

– Word 3 shall contain the least significant word of the 128 bit IP address (hex ‘20 0C 41 7A’).

The null value for the IP Address (Port) and IP Address (Node) Name Server object types is hex ‘00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00’.

6.4.6 Initial Process Associator – Format

The format of the Initial Process Associator, as used by the Name Server, shall be identical to the Process_Associator format defined in FC–PH.

The null value for the Initial Process Associator object is hex ‘FF FF FF FF FF FF FF FF’.

6.4.7 FC–4 TYPEs – Format

The format of the FC–4 TYPEs object, as defined in FC–PH, shall be bit mapped as shown below:

– the 3 most significant bits of the FC–4 TYPE field shall be used to identify the word for the FC–4 TYPEs object;

  – Word 0 contains information related to FC–4 TYPE code hex ‘00’ through hex ‘1F’;

  – Word 1 contains information related to FC–4 TYPE code hex ‘20’ through hex ‘3F’;

  – Word 2 contains information related to FC–4 TYPE code hex ‘40’ through hex ‘5F’;

  – Word 3 contains information related to FC–4 TYPE code hex ‘60’ through hex ‘7F’;

  – Word 4 contains information related to FC–4 TYPE code hex ‘80’ through hex ‘9F’;

  – Word 5 contains information related to FC–4 TYPE code hex ‘A0’ through hex ‘BF’;

  – Word 6 contains information related to FC–4 TYPE code hex ‘C0’ through hex ‘DF’;

  – Word 7 contains information related to FC–4 TYPE code hex ‘E0’ through hex ‘FF’.

– the 5 least significant bits of the FC–4 TYPE field shall be used to identify the bit position within the word for the FC–4 TYPE (see table 12).

– To mark an FC–4 TYPE as supported, the bit identified using the above procedure shall be set = 1; a value of = 0 shall indicate that the identified FC–4 TYPE is unsupported.

A Port Identifier supporting SCSI FCP (TYPE is hex ’08’), ISO/IEC 8802-2 LLC/SNAP (In-order)

Table 12 – FC–4 TYPEs mapping

FC–4 TYPE Bit 4 3210	FC–4 TYPE Bit 7 6 5	FC–4 TYPE Bit 7 6 5	FC–4 TYPE Bit 7 6 5	FC–4 TYPE Bit 7 6 5	FC–4 TYPE Bit 7 6 5	FC–4 TYPE Bit 7 6 5	FC–4 TYPE Bit 7 6 5	FC–4 TYPE Bit 7 6 5
	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0 0000	wd 0 [0]	wd 1 [0]	wd 2 [0]	wd 3 [0]	wd 4 [0]	wd 5 [0]	wd 6 [0]	wd 7 [0]
0 0001	wd 0 [1]	wd 1 [1]	wd 2 [1]	wd 3 [1]	wd 4 [1]	wd 5 [1]	wd 6 [1]	wd 7 [1]
0 0010	wd 0 [2]	wd 1 [2]	wd 2 [2]	wd 3 [2]	wd 4 [2]	wd 5 [2]	wd 6 [2]	wd 7 [2]
0 0011	wd 0 [3]	wd 1 [3]	wd 2 [3]	wd 3 [3]	wd 4 [3]	wd 5 [3]	wd 6 [3]	wd 7 [3]
0 0100	wd 0 [4]	wd 1 [4]	wd 2 [4]	wd 3 [4]	wd 4 [4]	wd 5 [4]	wd 6 [4]	wd 7 [4]
0 0101	wd 0 [5]	wd 1 [5]	wd 2 [5]	wd 3 [5]	wd 4 [5]	wd 5 [5]	wd 6 [5]	wd 7 [5]
0 0110	wd 0 [6]	wd 1 [6]	wd 2 [6]	wd 3 [6]	wd 4 [6]	wd 5 [6]	wd 6 [6]	wd 7 [6]
0 0111	wd 0 [7]	wd 1 [7]	wd 2 [7]	wd 3 [7]	wd 4 [7]	wd 5 [7]	wd 6 [7]	wd 7 [7]
0 1000	wd 0 [8]	wd 1 [8]	wd 2 [8]	wd 3 [8]	wd 4 [8]	wd 5 [8]	wd 6 [8]	wd 7 [8]
0 1001	wd 0 [9]	wd 1 [9]	wd 2 [9]	wd 3 [9]	wd 4 [9]	wd 5 [9]	wd 6 [9]	wd 7 [9]
0 1010	wd 0 [10]	wd 1 [10]	wd 2 [10]	wd 3 [10]	wd 4 [10]	wd 5 [10]	wd 6 [10]	wd 7 [10]
0 1011	wd 0 [11]	wd 1 [11]	wd 2 [11]	wd 3 [11]	wd 4 [11]	wd 5 [11]	wd 6 [11]	wd 7 [11]
0 1100	wd 0 [12]	wd 1 [12]	wd 2 [12]	wd 3 [12]	wd 4 [12]	wd 5 [12]	wd 6 [12]	wd 7 [12]
0 1101	wd 0 [13]	wd 1 [13]	wd 2 [13]	wd 3 [13]	wd 4 [13]	wd 5 [13]	wd 6 [13]	wd 7 [13]
0 1110	wd 0 [14]	wd 1 [14]	wd 2 [14]	wd 3 [14]	wd 4 [14]	wd 5 [14]	wd 6 [14]	wd 7 [14]
0 1111	wd 0 [15]	wd 1 [15]	wd 2 [15]	wd 3 [15]	wd 4 [15]	wd 5 [15]	wd 6 [15]	wd 7 [15]
1 0000	wd 0 [16]	wd 1 [16]	wd 2 [16]	wd 3 [16]	wd 4 [16]	wd 5 [16]	wd 6 [16]	wd 7 [16]
1 0001	wd 0 [17]	wd 1 [17]	wd 2 [17]	wd 3 [17]	wd 4 [17]	wd 5 [17]	wd 6 [17]	wd 7 [17]
1 0010	wd 0 [18]	wd 1 [18]	wd 2 [18]	wd 3 [18]	wd 4 [18]	wd 5 [18]	wd 6 [18]	wd 7 [18]
1 0011	wd 0 [19]	wd 1 [19]	wd 2 [19]	wd 3 [19]	wd 4 [19]	wd 5 [19]	wd 6 [19]	wd 7 [19]
1 0100	wd 0 [20]	wd 1 [20]	wd 2 [20]	wd 3 [20]	wd 4 [20]	wd 5 [20]	wd 6 [20]	wd 7 [20]
1 0101	wd 0 [21]	wd 1 [21]	wd 2 [21]	wd 3 [21]	wd 4 [21]	wd 5 [21]	wd 6 [21]	wd 7 [21]
1 0110	wd 0 [22]	wd 1 [22]	wd 2 [22]	wd 3 [22]	wd 4 [22]	wd 5 [22]	wd 6 [22]	wd 7 [22]
1 0111	wd 0 [23]	wd 1 [23]	wd 2 [23]	wd 3 [23]	wd 4 [23]	wd 5 [23]	wd 6 [23]	wd 7 [23]
1 1000	wd 0 [24]	wd 1 [24]	wd 2 [24]	wd 3 [24]	wd 4 [24]	wd 5 [24]	wd 6 [24]	wd 7 [24]
1 1001	wd 0 [25]	wd 1 [25]	wd 2 [25]	wd 3 [25]	wd 4 [25]	wd 5 [25]	wd 6 [25]	wd 7 [25]
1 1010	wd 0 [26]	wd 1 [26]	wd 2 [26]	wd 3 [26]	wd 4 [26]	wd 5 [26]	wd 6 [26]	wd 7 [26]
1 1011	wd 0 [27]	wd 1 [27]	wd 2 [27]	wd 3 [27]	wd 4 [27]	wd 5 [27]	wd 6 [27]	wd 7 [27]
1 1100	wd 0 [28]	wd 1 [28]	wd 2 [28]	wd 3 [28]	wd 4 [28]	wd 5 [28]	wd 6 [28]	wd 7 [28]
1 1101	wd 0 [29]	wd 1 [29]	wd 2 [29]	wd 3 [29]	wd 4 [29]	wd 5 [29]	wd 6 [29]	wd 7 [29]
1 1110	wd 0 [30]	wd 1 [30]	wd 2 [30]	wd 3 [30]	wd 4 [30]	wd 5 [30]	wd 6 [30]	wd 7 [30]
1 1111	wd 0 [31]	wd 1 [31]	wd 2 [31]	wd 3 [31]	wd 4 [31]	wd 5 [31]	wd 6 [31]	wd 7 [31]

(TYPE is hex '04') and Fibre Channel Services (TYPE is hex '20')would register hex '00 00 01 10 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ' as it's FC–4 TYPEs object.

The null FC–4 TYPEs object value is set to hex '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'.

6.4.8 Symbolic Port Name – Format

The Symbolic Port Name object is of variable length, with a minimum of 0 and a maximum of 255 bytes. The contents of these bytes are not defined and shall not be restricted by the Name Server.

If a Symbolic Port Name is not registered then the Symbolic Port Name defaults to a 0 byte length object.

6.4.9 Symbolic Node Name – Format

The Symbolic Node Name object is of variable length, with a minimum length of 0 and a maximum length of 255 bytes. The contents of these bytes are not defined and shall not be restricted by the Name Server.

If a Symbolic Node Name is not registered then the Symbolic Node Name defaults to 0 byte length object.

6.4.10 Port Type – Format

The format of the Port Type object, shall be as shown in table 13.

Port Type 'Nx_Port' is provided as a means to request all Port Types less than hex '80'. Port Type Nx_Port may only be specified in a GID_PT request, and shall never be specified in the response to a GA_NXT or GPT_ID request, or in an RPT_ID request.

The null Port Type object value is set to an 'Unidentified' type.

Table 13 – Port TYPE – encoding

Encoded value (hex)	Description
00	Unidentified
01	N_Port
02	NL_Port
03	F/NL_Port
7F	Nx_Port
03-80	Reserved
81	F_Port
82	FL_Port
83	Reserved
84	E_Port
85-FF	Reserved

6.4.11 Fabric Port Name – Format

The format of the Fabric Port Name object, as used by the Name Server, shall be identical to the Name_Identifier format defined in FC-PH. The Fabric Port Name for a given Port Identifier is the Port_Name for the F_Port to which the Port is attached.

The null value for the Fabric Port Name object is hex'00 00 00 00 00 00 00 00'.

6.4.12 Hard Address – Format

The format of the Hard Address object, as used by the Name Server, shall be identical to the format for Hard Address defined in the Discover Address (ADISC) Extended Link Service (see FC-PH-2).

The null value for the Hard Address object is hex'00 00 00'.

6.5 FS_RJT reason code explanations

If a Name Server request is rejected with a reason code of 'Unable to perform command re-

quest', then one of the reason code explanations, shown in table 14, are returned.

Table 14 – FS_RJT Reason code explanations

Encoded value (hex)	Description
00	No additional explanation
01	Port Identifier not registered
02	Port Name not registered
03	Node Name not registered
04	Class of Service not registered
05	IP Address (Node) not registered
06	Initial Process Associator not registered
07	FC–4 TYPEs not registered
08	Symbolic Port Name not registered
09	Symbolic Node Name not registered
0A	Port Type not registered
0B	IP Address (Port) not registered
0C	Fabric Port Name not registered
0D	Hard Address not registered
10	Access denied
11	Unacceptable Port Identifier
12	Data base empty
13	No object registered in the specified scope
Others	Reserved

The use of these codes is further defined as follows:

– If a Name Server request is rejected by the Name Server because of the identity of the requestor, then the FS_RJT reason code shall be 'Unable to perform command request', with a reason code explanation of 'Access denied'.

– If a Name Server Query request is rejected by the Name Server because no Name Server entries exist, then the FS_RJT reason code shall be 'Unable to perform command request', with a reason code explanation of 'Data base empty'.

– If a Name Server Query request other than GA_NXT, GID_FT, and GNN_FT is rejected by the Name Server because the object specified in the request is not found in the Name Server data base (within the specified scope, in the case of GID_PT), then the FS_RJT reason code shall be 'Unable to perform command request', with a reason code explanation that indicates the specified object is not registered.

– If a Name Server GID_FT or GNN_FT request is rejected by the Name Server because no FC-4 TYPEs object is found in the Name Server data base within the specified scope, then the FS_RJT reason code shall be 'Unable to perform command request', with a reason code explanation of 'FC-4 TYPEs not registered'.

– If a Name Server GI_A Query request is rejected by the Name Server because the requested information is not found within the specified Domain_ID Scope, then the FS_RJT reason code shall be 'Unable to perform command request', with a reason code explanation of 'No object registered within the specified scope'.

– If a Name Server Registration request is rejected by the Name Server because the Port Identifier cannot be registered, then the FS_RJT reason code shall be 'Unable to perform command request', with a reason code explanation of 'Unacceptable Port Identifier'.

– Additional uses may be defined for specific Name Server requests.

6.6 Queries

The Name Server may reject any query requests for reasons not specified in this document.

The queries defined for the Name Service are summarized in table 10.

6.6.1 Query – Get all next (GA_NXT)

The GA_NXT shall be used by a requestor to obtain all Name Server objects associated with a specific Port Identifier. The Name Server shall return all Name Server objects, not for the supplied Fibre Channel address identifier, but for the next higher valued Port Identifier, registered with the Name Server. If there are no registered Port Identifier higher valued than the value in the GA_NXT request, then the Name Server shall return the Name Server objects for the lowest registered Port Identifier. If there are no registered Name Server objects, then the Name Server shall reject the GA_NXT request. Fibre Channel address identifiers are treated as 24 bit unsigned entities for the purposes of comparison.

NOTE – No information is returned for well-known addresses or Alias addresses.

A requestor wishing to obtain all information on a specific Port Identifier may set the value of the Port Identifier in the request to be one less than the Port Identifier for which information is sought.

The GA_NXT request may be used to find all registered Port Identifiers in the Fabric, by reissuing the GA_NXT request, using the Port Identifier obtained from the FS_ACC NS_DU, stopping when the initially used Port Identifier threshold is recrossed.

NOTE – This function cannot be used to find all N_Ports or NL_Ports in a Fabric, unless the registration recommendations for the Fabric are followed (see 6.7), because N_Ports and NL_Ports are not required to register with the Name Server.

The format of the GA_NXT request is shown in table 15. The requestor supplies a Port Identifier using the format in 6.4.1, and the Name Server returns all Name Server objects for the next higher valued Port Identifier.

The format of the reply FS_ACC NS_DU to a GA_NXT request is shown in table 16. The format of the various objects returned is defined in 6.4.

Table 15 – NS_DU GA_NXT Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3

The Port Type field returns the registered value for the Port Type, or the null value if no Port Type is registered for the Port Identifier.

The Port Identifier field indicates the Name Server entry for which association and other objects are returned.

The Port Name field returns the registered value for the Port Name, or the null value if no Port Name is registered for the Port Identifier.

The Length of Symbolic Port Name field shall contain a single byte unsigned value indicating the size of the variable length Symbolic Port Name.

The Symbolic Port Name field returns the registered value for the Symbolic Port Name, or the null value if no Symbolic Port Name is registered for the Port Identifier.

The Node Name field returns the registered value for the Node Name, or the null value if no Node Name is registered for the Port Identifier.

The Length of Symbolic Node Name field shall contain a single byte unsigned value indicating the size of the variable length Symbolic Node Name.

The Symbolic Node Name field returns the registered value for the Symbolic Node Name, or the null value if no Symbolic Node Name is registered for the Port Identifier.

The Initial Process Associator field returns the registered value for the Initial Process Associator, or the null value if no Initial Process Associator is registered for the Port Identifier.

The IP Address (Node) field returns the registered value for the IP Address (Node), or the null value if no IP Address (Node) is registered for the Node related to the Port Identifier.

The Class of Service field returns the registered value for the Class of Service object, or the null value if no Class of Service object is registered for the Port Identifier.

The FC–4 TYPEs object field returns the registered value for the FC–4 TYPEs object, or the null value if no FC–4 TYPEs object is registered for the Port Identifier.

The IP Address (Port) field returns the registered value for the IP Address (Port), or the null value if no IP Address (Port) is registered for the Port Identifier.

The Fabric Port Name field returns the registered value for the Fabric Port Name, or the null value if no Fabric Port Name is registered for the Port Identifier.

The Hard Address field returns the registered value for the Hard Address, or the null value if no Hard Address is registered for the Port Identifier.

Table 16 – FS_ACC NS_DU to GA_NXT Request

Item	Size (Bytes)
Port Type	1
Port Identifier	3
Port Name	8
Length of Symbolic Port Name (m)	1
Symbolic Port Name	m
Reserved	255-m
Node Name	8
Length of Symbolic Node Name (n)	1
Symbolic Node Name	n
Reserved	255-n
Initial Process Associator	8
IP Address (Node)	16
Class of Service	4
FC–4 TYPEs	32
IP Address (Port)	16
Fabric Port Name	8
Reserved	1
Hard Address	3

6.6.2 Query – Get identifiers (GI_A)

The Name Server shall, when it receives a GI_A request, return identifiers for the specified scope. The format of the GI_A request is shown in table 17. The requestor supplies a Domain_ID Scope which defines the scope for which identifiers are sought.

NOTE – The identifiers returned by this request are not Port Identifier objects. The intended purpose of

this command is to allow the Name Service user to determine which Domains and Areas are available for use in the Scope field of other Queries.

The Domain_ID Scope field specifies the scope of the request. If the Domain_ID Scope field is zero, the Name Server shall return a list of Domain_IDs corresponding to registered Port Identifiers. If the Domain_ID Scope field is non-zero, the Name Server shall return a list of Domain_IDs and Area_IDs within the Domain specified by the Domain_ID Scope corresponding to registered Port Identifiers.

Table 17 – NS_DU GI_A Request Payload

Item	Size (Bytes)
Reserved	1
Domain_ID Scope	1
Reserved	2

The formats of the reply FS_ACC NS_DU to a GI_A request are shown in table 18 and table 19.

One or more identifiers are returned. Each returned identifier is preceded by an 8 bit Control field. The format of the Control field is:

– Bit 7 is set to zero if the identifier following the Control field is not the last identifier to be returned by the FS_ACC; the bit is set to one if the identifier following the Control field is the last identifier returned by the FS_ACC.

– Bits 6-0 are reserved.

The identifiers may be returned in any order. Furthermore, the order may be different for every request even if the same identifiers are returned and the requestor is the same.

6.6.3 Query – Get Port Name (GPN_ID)

The Name Server shall, when it receives a GPN_ID request, return the registered Port Name object for the specified Port Identifier. The format of the GPN_ID request is shown in table 20. The requestor supplies the Port Identifier for which the Port Name is sought.

Table 18 – FS_ACC NS_DU to GI_A Request, Domain_ID Scope is zero

Item	Size (Bytes)
Control (0 r r r r r r r)	1
Domain_ID #1	1
Reserved	2
...	
Control (1 r r r r r r r)	1
Domain_ID #n	1
Reserved	2

Table 19 – FS_ACC NS_DU to GI_A Request, Domain_ID Scope is non-zero

Item	Size (Bytes)
Control (0 r r r r r r r)	1
Domain_ID #1	1
Area_ID #1	1
Reserved	1
...	
Control (1 r r r r r r r)	1
Domain_ID #n	1
Area_ID #n	1
Reserved	1

Table 20 – NS_DU GPN_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3

The format of the reply FS_ACC NS_DU to a GPN_ID request is shown in table 21.

The Port Name field returns the registered value for the Port Name.

Table 21 – FS_ACC NS_DU to GPN_ID Request

Item	Size (Bytes)
Port Name	8

6.6.4 Query – Get Node Name (GNN_ID)

The Name Server shall, when it receives a GNN_ID request, return the registered Node Name object for the specified Port Identifier. The format of the GNN_ID request is shown in table 22. The requestor supplies the Port Identifier for which the Port Name is sought.

Table 22 – NS_DU GNN_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3

The format of the reply FS_ACC NS_DU to a GNN_ID request is shown in table 23.

The Node Name field returns the registered value for the Node Name.

Table 23 – FS_ACC NS_DU to GNN_ID Request

Item	Size (Bytes)
Node Name	8

6.6.5 Query – Get Class of Service (GCS_ID)

The Name Server shall, when it receives a GCS_ID request, return the registered Class of Service object for the specified Port Identifier. The format of the GCS_ID request is shown in table 24. The requestor supplies the Port Identifier

Table 24 – NS_DU GCS_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3

The format of the reply FS_ACC NS_DU to a GCS_ID request is shown in table 25.

The Class of Service field returns the registered value for the Class of Service object.

Table 25 – FS_ACC NS_DU to GCS_ID Request

Item	Size (Bytes)
Class of Service	4

6.6.6 Query – Get FC–4 TYPEs (GFT_ID)

The Name Server shall, when it receives a GFT_ID request, return the registered FC–4 TYPEs object for the specified Port Identifier. The format of the GFT_ID request is shown in table 26. The requestor supplies the Port Identifier for which the FC–4 TYPEs object is sought.

Table 26 – NS_DU GFT_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3

The format of the reply FS_ACC NS_DU to a GFT_ID request is shown in table 27.

The FC–4 TYPEs field (see 6.4.7) returns the registered value for the FC–4 TYPEs.

Table 27 – FS_ACC NS_DU to GFT_ID Request

Item	Size (Bytes)
FC–4 TYPEs	32

6.6.7 Query – Get Symbolic Port Name (GSPN_ID)

The Name Server shall, when it receives a GSPN_ID request, return the registered Symbolic Port Name for the specified Port Identifier. The format of the GSPN_ID request is shown in table 28. The requestor supplies the Port Identifier for which the Symbolic Port Name is sought.

Table 28 – NS_DU GSPN_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3

The format of the 256 byte reply FS_ACC NS_DU to a GSPN_ID request is shown in table 29.

The String length field shall contain a single byte unsigned value indicating the size of the variable length Symbolic Port Name.

The Symbolic Port Name field returns the registered Symbolic Port Name for the specified Port Name.

Table 29 – FS_ACC NS_DU to GSPN_ID Request

Item	Size (Bytes)
String length (m)	1
Symbolic Port Name (Octet string)	m
Reserved	255-m

6.6.8 Query – Get Port Type (GPT_ID)

The Name Server shall, when it receives a GPT_ID request, return the registered Port Type for the specified Port Identifier. The format of the GPT_ID request is shown in table 30. The requestor supplies the Port Identifier for which the Port Type is sought.

The format of the reply FS_ACC NS_DU to a GPT_ID request is shown in table 31.

Table 30 – NS_DU GPT_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3

The Port Type field returns the registered Port Type for the specified Port Identifier.

Table 31 – FS_ACC NS_DU to GPT_ID Request

Item	Size (Bytes)
Port Type	1
Reserved	3

6.6.9 Query – Get IP Address (Port) (GIPP_ID)

The Name Server shall, when it receives a GIPP_ID request, return the registered IP Address (Port) for the specified Port Identifier. The format of the GIPP_ID request is shown in table 32. The requestor supplies the Port Identifier for which the IP Address (Port) is sought.

Table 32 – NS_DU GIPP_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3

The format of the reply FS_ACC NS_DU to a GIPP_ID request is shown in table 33.

The IP Address (Port) field returns the registered value for the IP Address (Port).

Table 33 – FS_ACC NS_DU to GIPP_ID Request

Item	Size (Bytes)
IP Address (Port)	16

6.6.10 Query – Get Fabric Port Name (GFPN_ID)

The Name Server shall, when it receives a GFPN_ID request, return the registered Fabric Port Name object for the specified Port Identifier. The format of the GFPN_ID request is shown in table 34. The requestor supplies the Port Identifier for which the Fabric Port Name is sought.

Table 34 – NS_DU GFPN_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3

The format of the reply FS_ACC NS_DU to a GFPN_ID request is shown in table 35.

The Fabric Port Name field returns the registered value for the Fabric Port Name.

Table 35 – FS_ACC NS_DU to GFPN_ID Request

Item	Size (Bytes)
Fabric Port Name	8

6.6.11 Query – Get Hard Address (GHA_ID)

The Name Server shall, when it receives a GHA_ID request, return the registered Hard Address object for the specified Port Identifier. The format of the GHA_ID request is shown in table 36. The requestor supplies the Port Identifier for which the Hard Address is sought.

Table 36 – NS_DU GHA_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3

The format of the reply FS_ACC NS_DU to a GHA_ID request is shown in table 37.

The Hard Address field returns the registered value for the Hard Address.

Table 37 – FS_ACC NS_DU to GHA_ID Request

Item	Size (Bytes)
Reserved	1
Hard Address	3

6.6.12 Query – Get Port Identifier (GID_PN)

The Name Server shall, when it receives a GID_PN request, return the Port Identifier associated with the specified Port Name. The format of the GID_PN request is shown in table 38. The requestor supplies the Port Name for which the Port Identifier is sought.

Table 38 – NS_DU GID_PN Request Payload

Item	Size (Bytes)
Port Name	8

The format of the reply FS_ACC NS_DU to a GID_PN request is shown in table 39.

The Port Identifier field returns the registered Port Identifier value for the specified Port Name.

Table 39 – FS_ACC NS_DU to GID_PN Request

Item	Size (Bytes)
Reserved	1
Port Identifier	3

6.6.13 Query – Get IP Address (Port) (GIPP_PN)

The Name Server shall, when it receives a GIPP_PN request, return the registered IP Address (Port) for the specified Port Name. The format of the GIPP_PN request is shown in table 40. The requestor supplies the Port Name for which the IP Address (Port) is sought.

Table 40 – NS_DU GIPP_PN Request Payload

Item	Size (Bytes)
Port Name	8

The format of the reply FS_ACC NS_DU to a GIPP_PN request is shown in table 41.

The IP Address (Port) field returns the registered value for the IP Address (Port).

Table 41 – FS_ACC NS_DU to GIPP_PN Request

Item	Size (Bytes)
IP Address (Port)	16

6.6.14 Query – Get Port Identifiers (GID_NN)

The Name Server shall, when it receives a GID_NN request, return all Port Identifiers registered for the specified Node Name. The format of the GID_NN request is shown in table 42. The requestor supplies the Node Name for which associated Port Identifiers are sought.

Table 42 – NS_DU GID_NN Request Payload

Item	Size (Bytes)
Node Name	8

The format of the reply FS_ACC NS_DU to a GID_NN request is shown in table 43.

One or more Port Identifiers are returned. Each returned Port Identifier is preceded by an 8 bit Control field. The format of the Control field is:

– Bit 7 is set to zero if the Port Identifier following the Control field is not the last Port Identifier to be returned by the FS_ACC; the bit is set to one if the Port Identifier following the Control field is the last Port Identifier returned by the FS_ACC.

– Bits 6-0 are reserved.

Table 43 – FS_ACC NS_DU to GID_NN Request

Item	Size (Bytes)
Control (0 r r r r r r r)	1
Port Identifier #1	3
...	
Control (1 r r r r r r r)	1
Port Identifier #n	3

The Port Identifiers may be returned in any order. Furthermore, the order may be different for every request even if the same Port Identifiers are returned and the requestor is the same.

6.6.15 Query – Get IP Address (Node) (GIP_NN)

The Name Server shall, when it receives a GIP_NN request, return the registered IP Address (Node) for the specified Node Name. The format of the GIP_NN request is shown in table 44. The requestor supplies the Node Name for which the IP Address (Node) is sought.

Table 44 – NS_DU GIP_NN Request Payload

Item	Size (Bytes)
Node Name	8

The format of the reply FS_ACC NS_DU to a GIP_NN request is shown in table 45.

The IP Address (Node) field returns the registered value for the IP Address (Node).

Table 45 – FS_ACC NS_DU to GIP_NN Request

Item	Size (Bytes)
IP Address (Node)	16

6.6.16 Query – Get Initial Process Associator (GIPA_NN)

The Name Server shall when it receives a GIPA_NN request, return the registered Initial Process Associator object for the specified Node Name. The format of the GIPA_NN request is shown in table 46. The requestor supplies the Node Name for which the Initial Process Associator is sought.

Table 46 – NS_DU GIPA_NN Request Payload

Item	Size (Bytes)
Node Name	8

The format of the reply FS_ACC NS_DU to a GIPA_NN request is shown in table 47.

The Initial Process Associator field returns the registered Initial Process Associator object for the specified Node Name.

Table 47 – FS_ACC NS_DU to GIPA_NN Request

Item	Size (Bytes)
Initial Process Associator	8

6.6.17 Query – Get Symbolic Node Name (GSNN_NN)

The Name Server shall, when it receives a GSNN_NN request, return the registered Symbolic Node Name object for the specified Node Name. The format of the GSNN_NN request is shown in table 48. The requestor supplies the Node Name for which the Symbolic Node Name is sought.

Table 48 – NS_DU GSNN_NN Request Payload

Item	Size (Bytes)
Node Name	8

The format of reply FS_ACC NS_DU to a GSNN_NN request is shown in table 49.

The String length field shall contain a single byte unsigned value indicating the size of the variable length Symbolic Node Name.

The Symbolic Node Name field returns the registered Symbolic Node Name object for the specified Node Name.

Table 49 – FS_ACC NS_DU to GSNN_NN Request

Item	Size (Bytes)
String length (n)	1
Symbolic Node Name (Octet string)	n
Reserved	255-n

6.6.18 Query – Get Node Name (GNN_IP)

The Name Server shall, when it receives a GNN_IP request, return the registered Node Name object for the specified IP Address (Node). The format of the GNN_IP request is shown in table 50. The requestor supplies the IP Address (Node) for which the Port Name is sought.

Table 50 – NS_DU GNN_IP Request Payload

Item	Size (Bytes)
IP Address (Node)	16

The format of the reply FS_ACC NS_DU to a GNN_IP request is shown in table 51.

The Node Name field returns the registered Node Name.

Table 51 – FS_ACC NS_DU to GNN_IP Request

Item	Size (Bytes)
Node Name	8

6.6.19 Query – Get Initial Process Associator (GIPA_IP)

The Name Server shall, when it receives a GIPA_IP request, return the registered Initial Process Associator object for the specified IP Address (Node). The format of the GIPA_IP request is shown in table 52. The requestor supplies the IP Address (Node) for which the Initial Process Associator is sought.

Table 52 – NS_DU GIPA_IP Request Payload

Item	Size (Bytes)
IP Address (Node)	16

The format of the reply FS_ACC NS_DU to a GIPA_IP request is shown in table 53.

The Initial Process Associator field returns the registered value for the Initial Process Associator.

Table 53 – FS_ACC NS_DU to GIPA_IP Request

Item	Size (Bytes)
Initial Process Associator	8

6.6.20 Query – Get Port Identifiers (GID_FT)

The Name Server shall, when it receives a GID_FT request, return all Port Identifiers having registered support for the specified FC–4 TYPE. The format of the GID_FT request is shown in table 54. The requestor supplies the FC–4 TYPE code (as defined in FC–PH) for which supporting Port Identifiers are sought.

NOTE – The TYPE is specified as an 8-bit encoded FC-PH value, not as an FC-4 TYPE object.

The Domain_ID Scope and Area_ID Scope fields specify the scope of the request. If both the Domain_ID Scope and the Area_ID Scope fields are zero, the Name Server shall return all Port Identifiers having registered support for the specified FC–4 TYPE code. If the Domain_ID Scope field is non-zero and the Area_ID Scope field is zero, the Name Server shall return Port Identifiers within the specified Domain having registered support for the specified FC–4 TYPE code. If the Area_ID Scope field is non-zero, the Name Server shall return Port Identifiers within the specified Domain and Area having registered support for the specified FC–4 TYPE code.

NOTE – Suitable values for the Domain_ID Scope and Area_ID Scope fields may be discovered using the GI_A query.

Table 54 – NS_DU GID_FT Request Payload

Item	Size (Bytes)
Reserved	1
Domain_ID Scope	1
Area_ID Scope	1
FC–4 TYPE Code	1

The format of the reply FS_ACC NS_DU to a GID_FT request is shown in table 55.

One or more Port Identifiers, having registered support for the specified FC–4 TYPE, are returned. Each returned Port Identifier is preceded by an 8 bit Control field. The format of the Control field is:

– Bit 7 is set to zero if the Port Identifier following the Control field is not the last Port Identifier to be returned by the FS_ACC; the bit is set to one if the Port Identifier following the Control field is the last Port Identifier returned by the FS_ACC.

– Bits 6-0 are reserved.

The Port Identifiers may be returned in any order. Furthermore, the order may be different for every request even if the same Port Identifiers are returned and the requestor is the same.

6.6.21 Query – Get Node Names (GNN_FT)

The Name Server shall, when it receives a GNN_FT request, return a list of Port Identifiers and Node Names having registered support for the specified FC–4 TYPE. The format of the GNN_FT request is shown in table 56. The re-

Table 55 – FS_ACC NS_DU to GID_FT Request

Item	Size (Bytes)
Control (0 r r r r r r r r)	1
Port Identifier #1	3
...	
Control (1 r r r r r r r r)	1
Port Identifier #n	3

questor supplies the FC–4 TYPE code (as defined in FC–PH) for which supporting Port Identifiers and Node Names are sought.

NOTE – The TYPE is specified as an 8-bit encoded FC-PH value, not as an FC-4 TYPE object.

The Domain_ID Scope and Area_ID Scope fields specify the scope of the request. If both the Domain_ID Scope and the Area_ID Scope fields are zero, the Name Server shall return all Port Identifiers and Node Names having registered support for the specified FC–4 TYPE code. If the Domain_ID Scope field is non-zero and the Area_ID Scope field is zero, the Name Server shall return Port Identifiers and Node Names within the specified Domain having registered support for the specified FC–4 TYPE code. If the Area_ID Scope field is non-zero, the Name Server shall return Port Identifiers and Node Names within the specified Domain and Area having registered support for the specified FC–4 TYPE code.

NOTE – Suitable values for the Domain_ID Scope and Area_ID Scope fields may be discovered using the GI_A query.

The format of the reply FS_ACC NS_DU to a GNN_FT request is shown in table 57.

One or more Port Identifiers and Node Names, having registered support for the specified FC–4 TYPE, are returned. Each returned Port Identifier

Table 56 – NS_DU GNN_FT Request Payload

Item	Size (Bytes)
Reserved	1
Domain_ID Scope	1
Area_ID Scope	1
FC–4 TYPE	1

and Node Name is preceded by an 8 bit Control field. The format of the Control field is:

– Bit 7 is set to zero if the Port Identifier and Node Name following the Control field is not the last Port Identifier and Node Name to be returned by the FS_ACC; the bit is set to one if the Port Identifier and Node Name following the Control field is the last Port Identifier and Node Name returned by the FS_ACC.

– Bits 6-0 are reserved.

The Port Identifiers and Node Names may be returned in any order. Furthermore, the order may be different for every request even if the same Port Identifiers and Node Names are returned and the requestor is the same.

6.6.22 Query – Get Port Identifiers (GID_PT)

The Name Server shall, when it receives a GID_PT request, return all Port Identifiers having registered support for the specified Port Type. If the specified Port Type is equal to ‘Nx_Port’, then the Name Server shall return all Port Identifiers that have registered Port Types with an unsigned value of less than hex ‘80’, i.e. Port Identifiers for all registered Unidentified ports, N_Ports, NL_Ports, F/NL_Ports, etc. The format of the GID_PT request is shown in table 58. The requestor supplies the Port Type for which supporting Port Identifiers are sought.

The Domain_ID Scope and Area_ID Scope fields specify the scope of the request. If both the Domain_ID Scopeand the Area_ID Scope fields are zero, the Name Server shall return all Port Identifiers having registered support for the specified Port Type. If the Domain_ID Scope field is non-zero and the Area_ID Scope field is

Table 57 – FS_ACC NS_DU to GNN_FT Request

Item	Size (Bytes)
Control (0 r r r r r r r)	1
Port Identifier #1	3
Reserved	4
Node Name #1	8
...	
Control (1 r r r r r r r)	1
Port Identifier #n	3
Reserved	4
Node Name #n	8

zero, the Name Server shall return Port Identifiers within the specified Domain having registered support for the specified Port Type. If the Area_ID Scope field is non-zero, the Name Server shall return Port Identifiers within the specified Domain and Area having registered support for the specified Port Type.

NOTE – Suitable values for the Domain_ID Scope and Area_ID Scope fields may be discovered using the GI_A query.

Table 58 – NS_DU GID_PT Request Payload

Item	Size (Bytes)
Port Type	1
Domain_ID Scope	1
Area_ID Scope	1
Reserved	1

The format of the reply FS_ACC NS_DU to a GID_PT request is shown in table 59.

One or more Port Identifiers, having registered as the specified Port Type, are returned. Each returned Port Identifier is preceded by an 8 bit Control field. The format of the Control field is:

– Bit 7 is set to zero if the Port Identifier following the Control field is not the last Port Identifier to be returned by the FS_ACC; the bit is set to one if the Port Identifier following the Control field is the last Port Identifier returned by the FS_ACC.

– Bits 6-0 are reserved.

The Port Identifiers may be returned in any order. Furthermore, the order may be different for every request even if the same Port Identifiers are returned and the requestor is the same.

Table 59 – FS_ACC NS_DU to GID_PT Request

Item	Size (Bytes)
Control (0 r r r r r r r)	1
Port Identifier #1	3
...	
Control (1 r r r r r r r)	1
Port Identifier #n	3

6.6.23 Query – Get Port Identifier (GID_IPP)

The Name Server shall, when it receives a GID_IPP request, return return all Port Identifiers having registered the specified IP Address (Port). The format of the GID_IPP request is shown in table 60. The requestor supplies the IP Address (Port) for which Port Identifiers are sought.

Table 60 – NS_DU GID_IPP Request Payload

Item	Size (Bytes)
IP Address (Port)	16

The format of the reply FS_ACC NS_DU to a GID_IPP request is shown in table 61.

One or more Port Identifiers, having registered as the specified Port Type, are returned. Each returned Port Identifier is preceded by an 8 bit Control field. The format of the Control field is:

– Bit 7 is set to zero if the Port Identifier following the Control field is not the last Port Identifier to be returned by the FS_ACC; the bit is set to one if the Port Identifier following the Control field is the last Port Identifier returned by the FS_ACC.

– Bits 6-0 are reserved.

The Port Identifiers may be returned in any order. Furthermore, the order may be different for every request even if the same Port Identifiers are returned and the requestor is the same.

Table 61 – FS_ACC NS_DU to GID_IPP Request

Item	Size (Bytes)
Control (0 r r r r r r r)	1
Port Identifier #1	3
...	
Control (1 r r r r r r r)	1
Port Identifier #n	3

6.6.24 Query – Get Port Name (GPN_IPP)

The Name Server shall, when it receives a GPN_IPP request, return the registered Port Name object for the specified IP Address (Port). The format of the GPN_IPP request is shown in table 62. The requestor supplies the IP Address (Port) for which the Port Name is sought.

The format of the reply FS_ACC NS_DU to a GPN_IPP request is shown in table 63.

The Port Name field returns the registered Port Name.

Table 62 – NS_DU GPN_IPP Request Payload

Item	Size (Bytes)
IP Address (Port)	16

Table 63 – FS_ACC NS_DU to GPN_IPP Request

Item	Size (Bytes)
Port Name	8

6.7 Registration

Registrations are limited to a single Name Server object at a time. A registrant submits a tuple, consisting of a primary or secondary key object along with an object to be associated with the key object. The Port Identifier is the primary key object and the Node Name the secondary key object. The secondary key shall not be used as a key object until it has been registered and associated with the primary key.

The registration requests defined for the Name Service are summarized in table 10.

The Name Server may reject registrations:

– due to Name Server resource limitations;

– of Name Server objects associated with Alias addresses;

– of Name Server objects associated with unassigned or unused Port Identifiers.

However, the Name Server shall support registration of all Name Server object types, once registration of a single object has been accepted for a given Port Identifier.

The Name Server shall reject registrations of Name Server objects associated with:

– known Alias addresses such as:

  – Hunt group identifiers;

  – Multicast group identifiers.

However, the Name Server shall not be required to know all Alias addresses nor be required to validate registration requests with the Alias Server.

The Name Server shall reject all registrations of Name Server objects associated with:

– the address identifier hex ‘00 00 00’;

– well-known address identifiers.

The Name Server may reject all registrations of Name Server objects associated with Fibre Channel addresses not used or not usable as Port Identifiers in the Fabric.

The Name Server may reject any registration requests for reasons not specified in this document.

The Fabric may register the following objects once Fabric Login, implicit or explicit, has been successfully completed:

– Port Type;

– Port Identifier;

– Port Name;

– Node Name;

– Class of Service.

The Fabric may also cause the registered value of other Objects to change following a successful Fabric Login. If a Port becomes logged out with the Fabric, implicitly or explicitly, the Fabric may de-register all Objects associated with that Port.

If overlapping registrations for the same object is performed, then the Name Server shall, when all registrations have completed, leave the object as one of the registered object values. However, it is indeterminate which of the overlapping registration requests will have won.

A time lag may exist between successful completion of the registration and the time that the registered object can be returned in a query. This time lag is implementation and system dependent but shall not exceed 60 seconds.

6.7.1 Register Port Name (RPN_ID)

The RPN_ID Name Server request shall be used to associate a Port Name with a given Port Identifier.

The Name Server shall accept RPN_ID requests received from the Port with its address identifier equal to the Port Identifier in the NS_DU payload, from the Link Service Facilitator or from the Fabric Controller. Name Server may reject registration of the Port Name from any other source. The Fabric may register the Port Name for a Port Identifier before explicit Fabric Login (FLOGI) has completed.

The Name Server shall not attempt validation of the Port Name object. This means that any 64 bit Port Name value shall be accepted.

Deregistration may be accomplished by registering a null Port Name (see 6.4.2).

The format of the NS_DU payload for the RPN_ID request is shown in table 64.

There is no NS_DU payload in the FS_ACC to an RPN_ID Name Server request.

Table 64 – NS_DU RPN_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3
Port Name	8

6.7.2 Register Node Name (RNN_ID)

The RNN_ID Name Server request shall be used to associate a Node Name with a given Port Identifier.

The Name Server shall accept RNN_ID requests received from the Port with its address identifier equal to the Port Identifier in the NS_DU payload, from the Link Service Facilitator or from the Fabric Controller. Name Server may reject registration of the Port Name from any other source. The Fabric may register the Port Name for a Port

Identifier before explicit Fabric Login (FLOGI) has completed.

The Name Server shall not attempt validation of the Node Name object. This means that any 64 bit Node Name value shall be accepted.

Deregistration may be accomplished by registering a null Node Name (see 6.4.3).

The format of the NS_DU payload for the RNN_ID request is shown in table 65.

There is no NS_DU payload in the FS_ACC to an RNN_ID Name Server request.

Table 65 – NS_DU RNN_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3
Node Name	8

6.7.3 Register Class of Service (RCS_ID)

The RCS_ID Name Server request shall be used to record which Classes of Service are supported by a given Port Identifier.

The Name Server shall accept RCS_ID requests received from the Port with its address identifier equal to the Port Identifier in the NS_DU payload, from the Link Service Facilitator or from the Fabric Controller. Name Server may reject registration of the Port Name from any other source. The Fabric may register the Port Name for a Port Identifier before explicit Fabric Login (FLOGI) has completed.

The Name Server shall not attempt validation of the Class of Service object. This means that any 32 bit Class of Service object value shall be accepted.

Deregistration may be accomplished by registering a null Class of Service object (see 6.4.4).

The format of the NS_DU payload for the RCS_ID request is shown in table 66.

There is no NS_DU payload in the FS_ACC to an RCS_ID Name Server request.

Table 66 – NS_DU RCS_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3
Class of Service	4

6.7.4 Register FC–4 TYPEs (RFT_ID)

The RFT_ID Name Server request shall be used to record which FC–4 TYPEs are supported by a given Port Identifier.

The Name Server shall accept RFT_ID requests received from the Port with its address identifier equal to the Port Identifier in the NS_DU payload, from the Link Service Facilitator or from the Fabric Controller. Name Server may reject registration of the Port Name from any other source.

The Name Server shall not attempt validation of the FC–4 TYPEs object. This means that any 32 byte FC–4 TYPEs object value shall be accepted.

Deregistration may be accomplished by registering a null FC–4 TYPEs object (see 6.4.7).

The format of the NS_DU payload for the RFT_ID request is shown in table 67.

There is no NS_DU payload in the FS_ACC to a RFT_ID Name Server request.

Table 67 – NS_DU RFT_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3
FC–4 TYPEs	32

6.7.5 Register Symbolic Port Name (RSPN_ID)

The RSPN_ID Name Server request shall be used to associate a Symbolic Port Name with a given Port Identifier.

The Name Server may reject registration of the Symbolic Port Name unless the registration is attempted by the Port with its address identifier equal to the Port Identifier in the NS_DU payload.

The Name Server shall not attempt validation of the Symbolic Port Name object. This means that any variable length Symbolic Port Name value less than 256 bytes long, including a 0 length, shall be accepted.

Deregistration may be accomplished by registering a null Symbolic Port Name object (see 6.4.8).

The format of the NS_DU payload for the RSPN_ID request is shown in table 68. The String length field shall contain a single byte unsigned value indicating the size of the variable length Symbolic Port Name.

There is no NS_DU payload in the FS_ACC to an RSPN_ID Name Server request.

If the RSPN_ID Name Server request is rejected by the Name Server because the String length field value does not match the size of the Symbolic Port Name in the NS_DU, then the FS_RJT reason code shall be 'Invalid IU Size', with a reason explanation code of 'No additional explanation'.

Table 68 – NS_DU RSPN_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3
String length (n)	1
Symbolic Port Name (Octet string)	n

6.7.6 Register Port Type (RPT_ID)

The RPT_ID Name Server request shall be used to associate a Port Type with a given Port Identifier.

The Name Server shall accept RPT_ID requests received from the Port with its address identifier equal to the Port Identifier in the NS_DU payload, from the Link Service Facilitator or from the Fabric Controller. Name Server may reject registration of the Port Type from any other source. The Fabric may register the Port Type for a Port Identifier before explicit Fabric Login (FLOGI) has completed.

The Name Server shall not attempt validation of the Port Type object. This means that any 8 bit value, shall be accepted. Although not precluded by the Name Server, a Port Identifier shall not register its Port Type as an Nx_Port.

Deregistration may be accomplished by registering a null Port Type object (see 6.4.10).

The format of the NS_DU payload for the RPT_ID request is shown in table 69.

There is no NS_DU payload in the FS_ACC to an RPT_ID Name Server request.

Table 69 – NS_DU RPT_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3
Port Type	1
Reserved	3

6.7.7 Register IP Address (Port) (RIPP_ID)

The RIPP_ID Name Server request shall be used to associate an IP Address (Port) with a given Port Identifier.

The Name Server shall accept RIPP_ID requests received from the Port with its address identifier equal to the Port Identifier in the NS_DU payload, from the Link Service Facilitator or from the Fabric Controller. Name Server

may reject registration of the IP Address (Port) from any other source.

The Name Server shall not attempt validation of the IP Address (Port) object. This means that any 128 bit IP Address (Port) value shall be accepted.

Deregistration may be accomplished by registering a null IP Address (Port) object (see 6.4.5).

The format of the NS_DU payload for the RIPP_ID request is shown in table 70.

There is no NS_DU payload in the FS_ACC to an RIPP_ID Name Server request.

Table 70 – NS_DU RIPP_ID Request Payload)

Item	Size (Bytes)
Reserved	1
Port Identifier	3
IP Address (Port)	16

6.7.8 Register Fabric Port Name (RFPN_ID)

The RFPN_ID Name Server request shall be used to associate a Fabric Port Name with a given Port Identifier.

The Fabric may register the Fabric Port Name for a Port Identifier before explicit Fabric Login (FLOGI) has completed.

The Name Server shall not attempt validation of the Fabric Port Name object. This means that any 64 bit Fabric Port Name value shall be accepted.

Deregistration may be accomplished by registering a null Fabric Port Name (see 6.4.11).

The format of the NS_DU payload for the RFPN_ID request is shown in table 71.

There is no NS_DU payload in the FS_ACC to an RFPN_ID Name Server request.

Table 71 – NS_DU RFPN_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3
Fabric Port Name	8

6.7.9 Register Hard Address (RHA_ID)

The RHA_ID Name Server request shall be used to associate a Hard Address with a given Port Identifier.

The Name Server shall accept RHA_ID requests received from the Port with its address identifier equal to the Port Identifier in the NS_DU payload, from the Link Service Facilitator or from the Fabric Controller. Name Server may reject registration of the Hard Address from any other source. The Fabric may register the Hard Address for a Port Identifier before explicit Fabric Login (FLOGI) has completed.

The Name Server shall not attempt validation of the Hard Address object. This means that any 64 bit Fabric Port Name value shall be accepted.

Deregistration may be accomplished by registering a null Hard Address (see 6.4.12).

The format of the NS_DU payload for the RHA_ID request is shown in table 72.

There is no NS_DU payload in the FS_ACC to an RHA_ID Name Server request.

Table 72 – NS_DU RHA_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3
Reserved	1
Hard Address	3

6.7.10 Register IP Address (Node) (RIP_NN)

The RIP_NN Name Server request shall be used to associate an IP Address (Node) with a given Node Name.

Attempts at registration of an IP Address (Node) shall be rejected by the Name Server unless Node Name registration has been successfully completed (see 6.7.2). The Name Server may reject registration of the IP Address (Node) unless the registration is attempted by one of the Port Identifiers associated with the Node Name in the NS_DU payload.

The Name Server shall not attempt validation of the IP Address (Node) object. This means that any 128 bit IP Address (Node) value shall be accepted.

Deregistration may be accomplished by registering a null IP Address (Node) object (see 6.4.5).

The format of the NS_DU payload for the RIP_NN request is shown in table 73.

There is no NS_DU payload in the FS_ACC to an RIP_NN Name Server request.

If the RIP_NN Name Server request is rejected by the Name Server because the Node Name is not registered with the Name Server, then the FS_RJT reason code shall be 'Unable to perform command request', with a reason code explanation of 'Node Name not registered'.

Table 73 – NS_DU RIP_NN Request Payload)

Item	Size (Bytes)
Node Name	8
IP Address (Node)	16

6.7.11 Register Initial Process Associator (RIPA_NN)

The RIPA_NN Name Server request shall be used to associate an Initial Process Associator with a given Node Name.

Attempts at registration of an Initial Process Associator shall be rejected by the Name Server unless Node Name registration has been successfully completed (see 6.7.2). The Name Server may reject registration of the Initial Process Associator unless the registration is attempted by one of the Port Identifier associated with the Node Name in the NS_DU payload.

The Name Server shall not attempt validation of the Initial Process Associator object. This means that any 8 byte Initial Process Associator value shall be accepted.

Deregistration may be accomplished by registering a null Initial Process Associator object (see 6.4.6).

The format of the NS_DU payload for the RIPA_NN request is shown in table 74.

There is no NS_DU payload in the FS_ACC to an RIPA_NN Name Server request.

If the RIPA_NN Name Server request is rejected by the Name Server because the Node Name is not registered with the Name Server, then the FS_RJT reason code shall be 'Unable to perform command request', with a reason code explanation of 'Node Name not registered'.

Table 74 – NS_DU RIPA_NN Request Payload

Item	Size (Bytes)
Node Name	8
Initial Process Associator	8

6.7.12 Register Symbolic Node Name (RSNN_NN)

The RSNN_NN Name Server request shall be used to associate a Symbolic Node Name with a given Node Name.

Attempts at registration of a Symbolic Node Name shall be rejected by the Name Server unless Node Name registration has been successfully completed (see 6.7.2). The Name Server may reject registration of the Symbolic Node Name unless the registration is attempted by one of the Port Identifier associated with the Node Name in the NS_DU payload.

The Name Server shall not attempt validation of the Symbolic Node Name object. This means that any variable length Symbolic Node Name value less than 256 bytes long, including a 0 length, shall be accepted.

Deregistration may be accomplished by registering a null Symbolic Node Name object (see 6.4.9).

The format of the NS_DU payload for the RSNN_NN request is shown in table 75. The String length field shall contain a single byte unsigned value indicating the size of the variable length Symbolic Node Name.

There is no NS_DU payload in the FS_ACC to a RSNN_NN Name Server request.

If the RSNN_NN Name Server request is rejected by the Name Server because the Node Name is not registered with the Name Server, then the FS_RJT reason code shall be 'Unable to perform command request', with a reason code explanation of 'Node Name not registered';

If the RSNN_NN Name Server request is rejected by the Name Server because the String length field value does not match the size of the Symbolic Node Name in the NS_DU, then the FS_RJT reason code shall be 'Invalid IU Size', with a reason code explanation of 'No additional explanation'.

Table 75 – NS_DU RSNN_NN Request Payload

Item	Size (Bytes)
Node Name	8
String length (n)	1
Symbolic Port Name (Octet string)	n

6.8 Deregistration

Only one global deregistration request is defined for the Name Server. The requests defined for the Name Service are all summarized in table 10.

The Name Server may reject any deregistration requests for reasons not specified in this document.

6.8.1 Remove all (DA_ID)

The DA_ID shall be used to delete all entries and associations for a given Port Identifier in the Name Server's data base.

The Name Server shall accept DA_ID requests received from the Port with its address identifier equal to the Port Identifier in the NS_DU payload, from the Link Service Facilitator or from the Fabric Controller. Name Server may reject registration of the Port Name from any other source.

The Fabric should not issue the DA_ID Name Server request, unless the address identifier is removed as a Port Identifier, has disappeared from the Fabric or if the address identifier has been reused.

The format of the NS_DU payload for the DA_ID request is shown in table 76. The Port Identifier format shall be as defined in 6.4.1.

There is no NS_DU payload in the FS_ACC to a DA_ID Name Server request.

If the DA_ID Name Server request is rejected by the Name Server because the Port Identifier is not registered with the Name Server, then the FS_RJT reason code shall be 'Unable to perform command request', with a reason code explanation of 'Port Identifier not registered'.

Table 76 – NS_DU DA_ID Request Payload

Item	Size (Bytes)
Reserved	1
Port Identifier	3

7 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.

7.1 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.

7.2 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.

7.3 Fault management

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

7.4 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.

7.5 Accounting management

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

7.6 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.

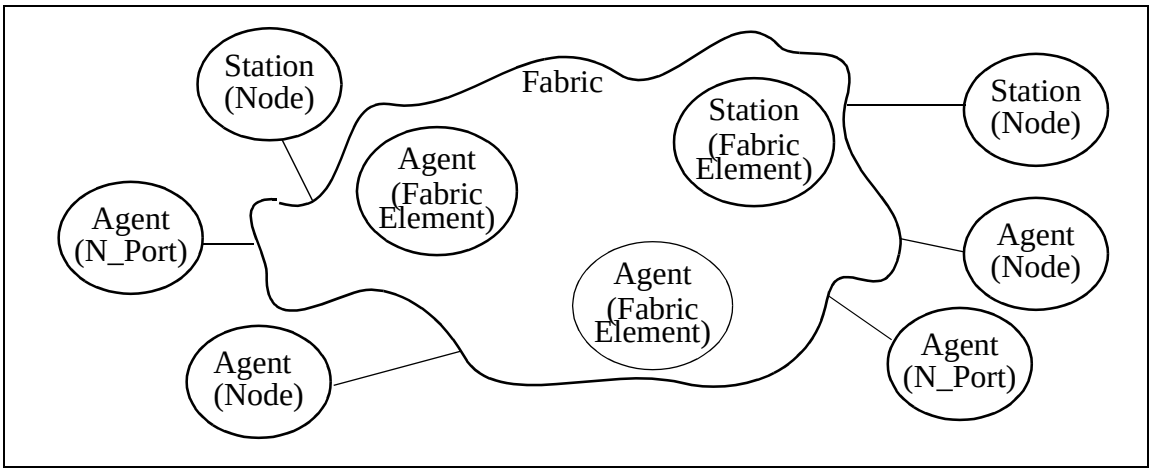


Figure 1 – Functional model of SNMP-based Management system

7.6.1 Overview

Within the SNMP model, management service is provided by a collection of entities: one or more management stations, and one or more management agents. The model is illustrated in figure 1. 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 2 and 3.

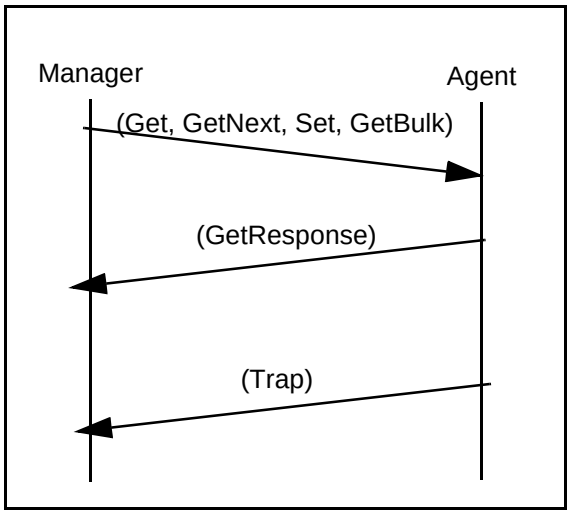


Figure 2 – Message flow between a manager and an agent

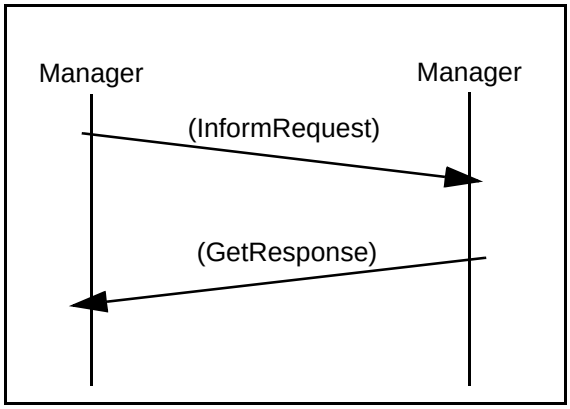


Figure 3 – Message flow between a manager and a manager

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).

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 4.

7.6.2 UDP mapping

The UDP is the preferred transport mapping in the Internet or TCP/IP environment. This mapping is endorsed for early deployment of Fibre Channel in the Internet and legacy internets.

7.7 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.

7.7.1 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.

7.7.2 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. These Exchanges may be long-lived (i.e., the Exchange is not terminated in the absence of error) to reduce Exchange origination overhead.

7.7.3 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.

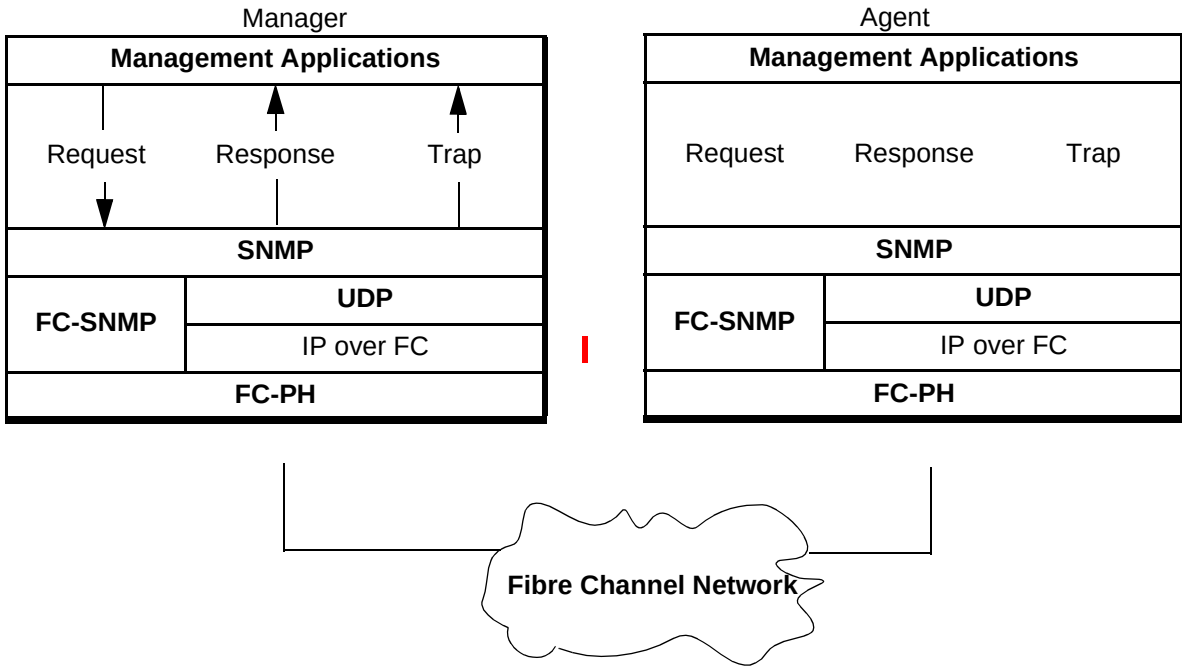


Figure 4 – The SNMP transport mappings

7.7.4 Class of service

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

7.7.5 R_CTL routing bits

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

7.7.6 Information category

Table 77 defines the information category according to each SNMP operation message.

Table 77 – 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

7.7.7 Sequence initiative

Sequence Initiative shall not be transferred during normal operation.

7.7.8 Destination ID

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

7.7.9 Source ID

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

7.7.10 Type

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

7.7.11 Relative offset

The relative offset shall not be used.

7.7.12 Error policy

All error policies except Process Policy are allowed.

7.7.13 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.

7.7.14 Network header

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

7.7.15 Association header

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

7.7.16 Device header

The Device Header shall not be used.

7.7.17 Information unit descriptions

IUs transferred between two SNMP entities are summarized in table 78.

7.8 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.

NOTE – As of publication of this standard, there are Internet Standard MIBs in development for Fibre Channel. See clause 2 for more information. Several vendor-specific MIBs also exist.

Table 78 – 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

7.9 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.

7.10 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.

8 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.

8.1 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 5 illustrates the functional model.

8.2 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.

8.2.1 TS information units

For a Time Server request, the Time Server payload shall be transported from the requestor to the Time Server using a synchronous CT_IU request. The corresponding Time Server response is transported from the Time Server to the requestor, in the Exchange established by the requestor, using a response CT_IU.

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).

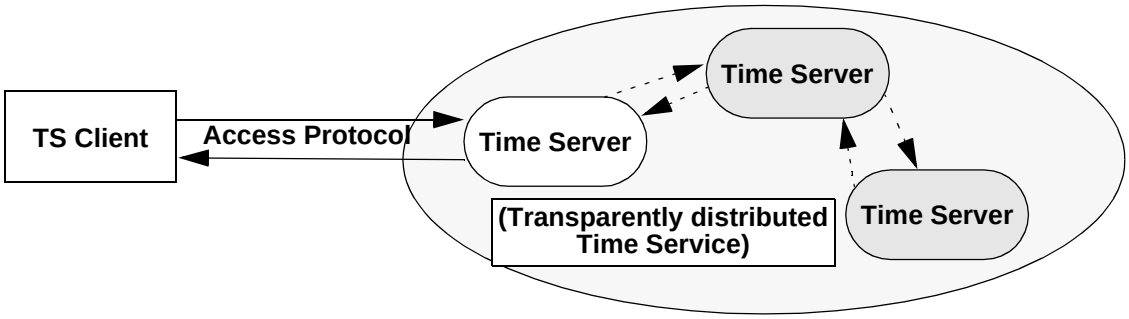


Figure 5 – Functional model of time service

8.2.2 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.

8.2.3 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').

8.2.4 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.

8.2.5 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.

8.2.6 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 79.

Table 79 – 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, 00:00:00 1 January 1990 UTC;

– and the fractional part of the time value.

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

8.2.7 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.

8.3 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.

9 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.

9.1 Alias service protocol

For a Alias Server request, the Alias Server payload shall be transported from the requestor to the Alias Server using a synchronous CT_IU request. The corresponding Alias Server response is transported from the Alias Server to the requestor, in the Exchange established by the requestor, using a response CT_IU.

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.

9.2 Use of FC-PH constructs

9.2.1 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.

9.2.2 Exchanges

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

9.3 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.

9.3.1 Common required FC parameters

9.3.1.1 Class of service

The Alias Server shall support all Classes of Service supported by the Fabric Region to which it is attached. See FC-FG 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.

9.3.1.2 R_CTL routing bits

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

9.3.1.3 Information category

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

9.3.1.4 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.

9.3.1.5 Destination ID (D_ID)

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

9.3.1.6 Source ID (S_ID)

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

9.3.1.7 Type

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

9.3.1.8 Error policy

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

9.3.2 Common optional FC parameters

9.3.2.1 Expiration/Security header

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

9.3.2.2 Network header

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

9.3.2.3 Association header

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

9.3.2.4 Device header

The Device Header shall not be used.

9.3.3 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.

9.3.3.1 Revision

This field shall be set to hex ‘01’.

9.3.3.2 IN_ID

This field shall be ignored by the Alias Server.

9.3.3.3 FCS_Type

This field shall be set to hex ‘F8’.

9.3.3.4 FCS_Subtype

This field shall be set to hex’01’.

9.3.3.5 Options

This field shall be set to hex ‘00’.

9.3.3.6 Application information unit

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

9.4 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 Alias Server Requests are shown in table 80.

Table 80 – Alias Server Requests

Code (hex)	Mnem.	Description
0001	JNA	Join Alias Group
0002	RMA	Remove from Alias Group
0003	LSN	Listen
0004	SLSN	Stop Listen
0005	RAG	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.

9.4.1 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 81.

Table 81 – 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 92.

Alias Group Token (Alias_Token): The Alias_Token defines the Alias Group being cre-

ated. 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 91.

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 82.

Table 82 – 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 91.

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 90.

9.4.2 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 83.

Table 83 – Remove from alias group payload

Item	Size (Bytes)
Authorization_PW	12
Reserved	4
Alias_Token	12
NP_List_Length	4

Table 83 – Remove from alias group payload

Item	Size (Bytes)
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 91.

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 90.

9.4.3 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 84 .

Table 84 – 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 91.

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 85 .

Table 85 – 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 91.

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".

9.4.4 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 86.

Table 86 – 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 91.

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”.

9.4.5 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 the read alias group request.

The format of the payload is shown in table 86.

Table 87 – 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 91. 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 88.

Table 88 – 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 91.

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 89.

Table 89 – 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:

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

9.5 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.

9.5.1 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.

9.5.2 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 90 defines and explains the various FS_RJT Reason Code Explanations. Reason Code Explanations hex '30' through hex '38' 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".

Table 90 – 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 0100	Alias ID cannot be deactivated at Fabric (resource problem)	Remove From Alias Group
0011 0101	Alias Group cannot be joined (Service Parameter conflict)	Join 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

9.5.3 Alias_Token

Table 91 defines the format of the Alias_Token field .

Table 91 – 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
- 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, if the Initiator is in 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 9.9 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: For Multicast Groups, this field is 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.

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, as specified in the Alias_Class field. For example, an SBCCS Multicast Group may be created for channels (TYPE = hex '19') and a different SBCCS Multicast Group may be created for control units (TYPE = hex '1A').

The Alias_Qualifier shall be defined as follows:

- All zeroes: Alias_Qualifier is unknown. A unique value may be assigned by the Server. If the Server is unable to assign the Alias_Qualifier, it shall reject the request;
- All ones: for Multicast, this value indicates that all Alias_Qualifiers for the assocuiated Alias_Class may be processed. If the Server is unable to process the request, it shall reject the request. This value is reserved for Hunt Groups;
- All others: The Alias_Qualifier shall be assigned by the particular FC-4 defined by the Alias_Class value. The N_Ports have an intrinsic knowledge, defined by the as-

sociated 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.

9.5.4 Authorization_Control

Table 92 defines the format of the Authorization_Control field .

Table 92 – 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”.

9.6 Function flow

Figure 6 illustrates the flow among the Originator N_Port, participating N_Ports, Alias Server, and Fabric Controller to create an Alias Group.

9.7 Alias server functions

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

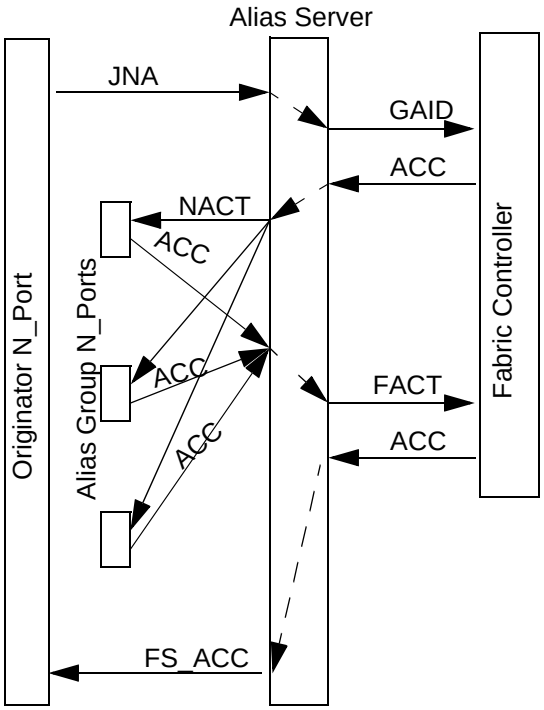


Figure 6 –Function flow

9.7.1 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 obtain a unique alias address identifier for this Alias Group from the Fabric Controller either with a Fabric Controller Request, Get Alias Group ID (GAID), or an implicit mechanism.. 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 may send an Extended Link Services request, N_Port Activate Alias Group ID (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 may 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 request the Fabric Controller to activate the alias address identifier at the Fabric. This shall be accomplished either with a Fabric Controller request, F_Port Activate Alias Group ID (FACT), or an implicit mechanism. 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) 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.

9.7.2 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 request the Fabric Controller to deactivate from the Alias_ID at the Fabric. This shall be accomplished either with a Fabric Controller Request, Fabric Deactivate Alias Group ID (FDACT), or an implicit mechanism. 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 may then send an Extended Link Services request, N_Port Deactivate Alias Group ID (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;

9.7.3 Listen

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

a) If the Alias Server does not support Listen request, it shall reject the request with a Reason Code Explanation of "No additional information".

b) 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;

c) 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;

d) 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;

e) 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;

f) 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;

g) The Alias Server may 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;

h) 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;

i) For each LS_ACC from the Listening N_Port, the Alias Server shall request the Fabric Controller to activate the alias address identifier at the Fabric, for the Listening N_Port. It shall do so either with the Fabric Controller request, FACT, or an implicit mechanism. The Alias_ID and the Listening N_Port ID shall be passed in the payload;

j) 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;

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 be explicitly removed from an Alias Group by a Stop Listen request. Listening N_Ports shall not be explicitly removed by a Remove From Alias Group request. Listening N_Ports may be implicitly removed by a Remove From Alias Group request if the request removes all Grouped N_Ports (see 9.4.5, Membership, and 9.7.2g).

9.7.4 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 Server does not support Stop Listen request, it shall reject the request with a Reason Code Explanation of "No additional information".

b) 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;

c) 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;

d) 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;

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 is ignored;

f) 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;

g) The Alias Server may 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;

h) For each LS_ACC from the Listening N_Port, the Alias Server shall request the Fabric Controller to activate the alias address identifier at the Fabric, for the Listening N_Port. It shall do so either with the Fabric Controller request, NDACT, or an implicit mechanism. 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;

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 is necessary to determine which Alias Groups have been deactivated for the listening N_Port.

9.7.5 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.

9.8 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 Login Service Parameters. For example, the Receive Data Field Size for a multicast frame may be different than the Receive Data Field Size for a unicast frame.

If the Sequence Recipient is a member of an Alias Group, it shall recognize the Alias_ID as an alias address identifier and accept the frame.

For Multicast Groups, Class 1 and Class 2 frames with a D_ID equal to an Alias_ID shall be rejected by the Fabric.

9.9 IPA Considerations

9.9.1 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.

9.9.2 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.

9.9.3 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.

10 Security-key distribution service

This clause describes and formally defines the methods and formats for the distribution of secret keys.

10.1 Introduction

The security-key distribution service offers a mechanism for the secure distribution of secret encryption keys. Secure distribution is accomplished through the use of a Security-Key Server which utilizes data encryption techniques and verification protocols. The underlying assumption of this service is that the client contains no encryption keys at start-up other than a distribution key which is unique to the client.

NOTE – Other than for encryption keys, the distribution of encrypted data is beyond the scope of this document. Data encryption is an optional FC-2 function which utilizes the secret keys distributed by this service.

10.1.1 Function

The security-key distribution service is modeled after the IEEE 802.10 Standard for Interoperable LAN/MAN Security. IEEE 802.10 is a center-based (Security-Key Server) Key Exchange Technique with authentication.

The Security-Key Server contains a copy of each client's unique distribution key. The Security-Key Server authorizes secure connections between client pairs (Originator and Responder) and generates a secret key for bulk data encryption. The secret key is encrypted by the Security-Key Server using the Originator and Responder distribution keys. The encrypted secret key is returned to the Originator client then forwarded to the Responder client.

The security offered by this service depends on the quality of the encryption algorithm, the length of the encryption key, and the protection of the client's unique distribution key. Only the Security-Key Server shall have a copy of the client's unique distribution key. The mechanism by which the distribution keys are distributed to the security-key clients and server is beyond the scope of this document.

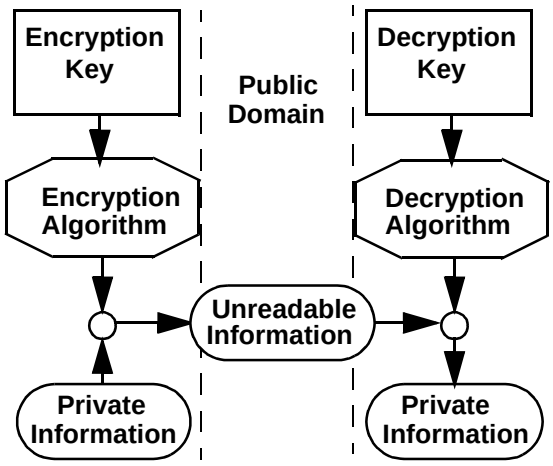
This service employs authentication techniques to assure the validity of all server and client responses. Authentication is accomplished through the use of unique identifiers that are sent in all requests. By returning the unique identifier in an encrypted form, the response to a request is authenticated.

10.1.2 Terms

10.1.2.1 Encryption algorithm

An encryption algorithm (see figure 7) transforms private information into a form that is unreadable during public transport by anyone without a decryption key.

Figure 7 – Encryption Algorithm



10.1.2.2 Key encryption

Key encryption is used by the Security-Key Server to securely distribute secret keys to clients. A unique distribution key is required by the client to decrypt all secret keys originated by the server. The size of the distribution key (see Annex B) and its algorithm can be optimized for use over many years. The size of the distribution key can range from 75-bits to 90-bits. Secret key encryption is not practical for high-speed encryption of bulk data.

10.1.2.3 Data encryption

Data encryption is used by a security-key client to securely transmit bulk data to another client using a previously exchanged secret key. The transmitting client uses the secret key to encrypt the message and the receiving client uses

the same secret key to decrypt the message. The size (see Annex B) of a high-speed secret key encryption can range from 40-bits to 75-bits depending on its use.

10.1.2.4 Notations and conventions

The following notation is defined for use within this clause.

{some_data} encryption_key
- means *some_data* has been encrypted using *encryption_key*.

The following table format for data structures is used within this clause.

Item	Size (bytes)
unencrypted_item X	p
unencrypted_item Y	q
encrypted with specified key T	
encrypted_item A	i
encrypted_item B	j
encrypted_item C	k
encrypted with specified key U	
encrypted_item D	i

In the table:

– items X and Y are not encrypted;

– items A, B, and C are encrypted using key T;

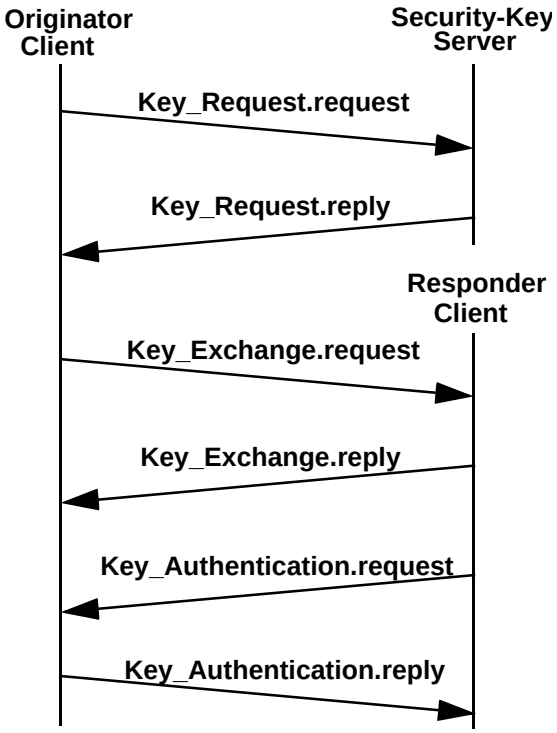
– item D is encrypted using key U;

– neither key T nor key U are included themselves in the data, rather they are only applied to the items.

10.2 Communications model

The security-key distribution protocol involves a three phase process (see figure 8) consisting of a key request phase followed by a key exchange phase and a key authentication phase.

Figure 8 – Security-Key Model



The Security-key request phase request payload shall be transported from the requestor to the Security-key Server using a synchronous CT_IU request. The corresponding Security-key request phase reply is transported from the Security-key Server to the requestor, in the Exchange established by the requestor, using a response CT_IU.

The Security-key exchange phase request payload shall be transported from the requestor to the Responder Client using a synchronous CT_IU request. The corresponding Security-key exchange phase reply payload is transported from the Responder Client to the requestor, in the Exchange established by the requestor, using a response CT_IU.

The Security-key authentication phase request payload shall be transported from the requestor to the Responder Client using a synchronous CT_IU request. The corresponding Security-key authentication phase reply payload is transported from the Responder Client to the requestor, in the Exchange established by the requestor, using a response CT_IU.

10.2.1 Key request protocol

In order for an Originator client to transmit encrypted data to a Responder client , it must first request a secret encryption key (KEY_REQUEST.request) from the Security-Key Server. The unencrypted request contains the following information:

- originator_id, the address identifier of the requesting Originator client Port;
- responder_id, the address identifier of the desired Responder client Port;
- unique_id.

The Originator client includes a unique identifier (unique_id) in the request for the purpose of authenticating the response from the server.

When the Security-Key Server receives the key request, it validates the Originator/Responder pair regarding distribution keys for secure communications. If the pair is authorized for secure communications, the Security-Key Server generates a one-time secret encryption key (secret_key) for the distribution encryption algorithm (encryption_id). A secret_key is unique and shall not be repeated in subsequent key requests. The secret_key and the data related to the Originator is encrypted using the Originator client's distribution key. The secret_key and the data related to the Responder is encrypted using the Responder's and Originator's distribution keys. The encrypted data is returned to the Originator client (KEY_REQUEST.reply):

- encryption_id,
  {
  unique_id,
  responder_id,
  secret_key,
  {
  originator_id,
  secret_key
  } responder_distribution_key
  } originator_distribution_key.

The Originator client extracts the following information by decrypting the portion of the reply that was encrypted with the originator_distribution_key:

- unique_id;
- responder_id;
- secret_key;
- {
  originator_id,
  secret_key
  } responder_distribution_key.

The Originator client authenticates the reply by comparing the decrypted unique_id with the unique_id sent in the key request. Only the Responder client can decrypt the information contained in the final parameter.

10.2.2 Key exchange protocol

A secret_key may be exchanged between an Originator client and a Responder client using either the responder_distribution_key or the current secret_key.

10.2.2.1 Using the responder_distribution_key

When the reply for the key request is received, the Originator client has all of the information needed to distribute the secret_key to the Responder client. The Originator client distributes the secret_key by issuing a request (KEY_EXCHANGE.request) to the Responder client. The request contains the following information:

- encryption_id;
- {
  originator_id,
  secret_key
  } responder_distribution_key.

The encryption_id identifies the use of the responder_distribution_key to encrypt the remainder of the payload. Using its responder_distribution_key, the Responder decrypts the encrypted portion of the key distribution request to obtain the following information:

- originator_id;
- secret_key.

After the Responder client verifies the encryption_id and the originator_id, the Responder client conditionally accepts the unauthenticated secret_key by returning a reply.

10.2.2.2 Using the current secret_key

If the Originator and Responder clients have already distributed secret keys, a new secret_key may be distributed using the current secret_key. The Originator client may distribute the secret_key by issuing a request (KEY_EXCHANGE.request) to the Responder client. The request contains the following information:

– encryption_id;

– {
originator_id,
secret_key
} current_secret_key.

The encryption_id indentifies the use of the current_secret_key to encrypt the remainder of the payload. Using the current_secret_key, the Responder client decrypts the encrypted portion of the key exchange request to obtain the following information:

– originator_id;

– secret_key.

The secret_key and current_secret_key have the same encryption_id. After the Responder client verifies the encryption_id and the originator_id, the Responder client conditionally accepts the unauthenticated secret_key by returning a reply.

10.2.3 Key authentication protocol

Before the Responder client can be guaranteed the secret_key came from the Originator client, it must authenticate the secret_key. The Responder client generates a unique identifier (unique_id) encrypted with the unauthenticated secret_key. The encrypted unique_id is sent in a key authentication request (KEY_AUTHENTICATION.request) to the Originator client. The request contains the following information:

– {unique_id} secret_key.

Using the secret_key, the Originator client decrypts the key authentication request to obtain the following:

– unique_id.

The Originator client decrements the unique_id (unique_id - 1) then encrypts the resulting value with the secret_key. The encrypted value is included in the reply (KEY_AUTHENTICATION.reply) to the Responder client:

– {unique_id - 1} secret_key.

When the Responder client receives the key authentication reply, it uses the unauthenticated secret_key to decrypt the decremented unique identifier. The secret_key is authenticated if the expected difference between the unique identifiers exists.

Upon completion of key authentication, the Responder client can begin the secure transfer of data encrypted with the secret_key.

10.2.4 Data Bases

10.2.4.1 Distribution Key

Before the protocols defined in this service are invoked, the Security-Key Server must contain the encryption algorithms and a unique distribution_key for each client Port.

Before the protocols defined in this service are invoked, each client Port must contain the encryption algorithms and its unique distribution_key.

10.2.4.2 Secret Key

Upon completion of the security-key distribution protocol, the Originator and Responder clients will contain a secret_key for secure communications with each other.

10.3 FC-PH Constructs

10.3.1 Login/Logout

Before performing any security-key distribution operations, an Originator client shall perform F_Port Login, followed by N_Port Login with the Security-Key Server, and with the Responder client. When the Originator client has no further operations pending, it shall perform N_Port Logout with the Security-Key Server, and with the Responder client.

10.3.2 Exchanges

Security-key distributions shall be used in a bi-directional manner. That is, request and reply Information Units shall be transferred on the same Exchange, via the passing of Sequence Initiative.

10.3.3 Information Units

The FS_IU construct defines the information transferred as a single Fibre Channel Sequence for key distribution requests and replies. A single FS_IU contains either a key distribution request or a reply. . This clause describes both the data which are transparent to this standard and those control parameters which are required by this standard.

10.3.4 Common FC-2 parameters

10.3.4.1 Class of Service

The Security-Key Server N_Port shall support all Classes of Service supported by the Fabric Region to which it is attached. A client N_Port may communicate with the Security-Key Server N_Port using any desired Class.

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

– R_A_TOV has expired since the client N_Port or the Security-Key Server N_Port has determined that the FS_IU has been transmitted by the FC-2;

– The client N_Port or the Security-Key Server N_Port receives a reply from the receiver of the transmitted FS_IU.

NOTE – The FC-PH service interface is defined in Annex S of FC-PH. The Sequence_Tag is equated to the Sequence ID in Class 3.

10.3.4.2 Routing control bits

The R_CTL field shall indicate:

– FC–4 Device_Data (hex ‘0’);

– Unsolicited Control (hex ‘2”) in Security-Key Server requests;

– Solicited Control (hex ‘3’) in Security-Key Server responses.

10.3.4.3 Sequence Initiative

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

10.3.4.4 Destination ID (D_ID)

This parameter shall contain the destination address identifier of the FS_IU. The D_ID shall be hex ‘FFFFF7’ if the destination is the Security-Key Server N_Port.

10.3.4.5 Source ID (S_ID)

This parameter shall contain the source address identifier of the FS_IU. The S_ID shall be hex ‘FFFFF7’ if the source is the Security-Key Server N_Port.

10.3.4.6 Type

All Security-Key Server Server FS_IUs shall specify the Fibre Channel Services TYPE (hex ‘20’).

10.3.4.7 Error Policy

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

10.3.4.8 Exchange ID (X_ID)

The Originator Exchange_ID (OX_ID) can optionally be used to identify more than one encrypted Exchange between N_Port pairs. Each Exchange shall use the same secret encryption algorithm and key assigned to the N_Port pair.

10.3.5 Common optional FC parameters

10.3.5.1 Expiration/Security Header

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

10.3.5.2 Network Header

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

10.3.5.3 Association Header

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

10.3.5.4 Device Header

The Device Header shall not be used.

10.3.6 CT_HDR values

The following is a description of the usage of the various CT_HDR fields.

- Revision field: hex ‘01’;
- IN_ID: reserved (hex ‘00 00 00’), may be non-zero in Security-Key Server responses;
- FS_Type: hex ‘F7’ (Security-Key Service application);
- FS_Subtype: hex ‘00’;
- Options: hex ‘00’ (single bidirectional Exchange);
- Command Code: see table 93 for request codes.

Table 93 – Security-Key Server Requests

Code (hex)	Description	Mnemonic
1000	Request key	SKE_KEYREQ
1100	Exchange key	SKE_KEYEXC
1200	Authenticate key	SKE_KEYAUT

10.3.7 Application Information Unit

The Application Information Unit for the Security-Key Server is known as the SK_DU.

10.4 Security-key services

A Request provides the means for an N_Port to participate in the security-key distribution protocol. If a request is rejected by the Security-Key Server, then the FS_RJT Reason Code shall be “Unable to perform command request”, unless otherwise indicated. The Reason Code Explanation shall indicate the specific reason for the FS_RJT.

10.4.1 Request key (SKE_KEYREQ)

The security-key distribution request Sequence shall be used by a client to request a secret key from the Security-Key Server.

The key request is performed as follows:

- The Originator client sends the key request with a unique identifier to the Security-Key Server;
- The Security-Key Server indicates acceptance of the key request by replying with the encrypted unique identifier and the secret_key;
- The Originator client authenticates the Security-Key Server using the unique identifier.

The format of the payload is shown in table 94.

Table 94 – SK_DU SKE_KEYREQ payload

Item	Size (bytes)
Reserved	1
responder_id	3
unique_id	8

The responder_id contains the value of the address identifier for the Responder client with which secure communications is requested.

The unique_id contains the value of the unique identifier for use by the Originator client to authenticate the reply.

NOTE – In addition to the payload parameters, the security-key request includes the originator_id in the S_ID field described above.

Fibre Channel service reject (FS_RJT) signifies rejection of the SKE_KEYREQ request.

Fibre Channel service accept (FS_ACC) signifies that the Security-Key Server has transmitted the secret_key information. The format of the accept payload is shown in table 95.

Table 95 – FS_ACC SK_DU to SKE_KEYREQ

Item	Size (bytes)
encryption_id	4
encrypted with originator_distribution_key	
unique_id	8
responder_id	4
secret_key	16
reserved	4
encrypted with responder_distribution_key	
originator_id	4
secret_key	16
reserved	12

The accept payload consists of the following eight parameters:

- The first parameter is the encryption_id of the algorithm for the secret_key;
- The next four parameters (plus the encrypted form of the last three parameters) are encrypted with the originator_distribution_key:
  - The value of the unique_id;
  - the value of the responder_id;
  - the value of the secret_key;
  - reserved (fills 16-byte block);
  - the last three parameters are encrypted with the responder_distribution_key:
    - the value of the originator_id;
    - the value of the secret_key;
    - reserved (fills 16-byte block).

10.4.2 Key exchange (SKE_KEYEXC)

The key exchange Sequence shall be used by the Originator client to distribute secret_keys with the Responder client.

The key exchange is performed as follows:

- The Originator client sends the encrypted secret_key to the Responder client.

The format of the payload is shown in table 96.

Table 96 – SK_DU SKE_KEYEXC payload

Item	Size (bytes)
encryption_id	4
encrypted with responder_distribution_key or current_secret_key	
originator_id	4
secret_key	16
reserved	12

The key exchange information consists of the following four parameters:

- The value of the encryption identifier parameter (encryption_id) indicates whether the responder_distribution_key or the current_secret_key should be used to decrypt the remainder of the payload;

- If the encryption_id is equal to that of the distribution_key, the remaining parameters are encrypted with the responder_distribution_key:

  - the value of originator_id;

  - the value of secret_key;

  - reserved (fills 16-byte block).

- If the encryption_id is equal to that of the current_secret_key, the remaining parameters are encrypted with the current_secret_key:

  - the value of originator_id;

  - the value of secret_key (the new secret_key);

  - reserved (fills 16-byte block).

The secret_key and current_secret_key shall have the same encryption_id.

Fibre Channel service reject (FS_RJT) signifies rejection of the SKE_KEYEXC request.

Fibre Channel service accept (FS_ACC) signifies that the Responder client has received the key exchange information from the Originator client. There is no SK_DU payload in the FS_ACC to an SKE_KEYEXC request.

10.4.3 Key authentication (SKE_KEYAUT)

The key authentication request Sequence shall be used by Responder client to authenticate the key exchange request received from the Originator client.

The key authentication is performed as follows:

- The Responder client sends the encrypted unique identifier to the Originator client;

- The Originator client decrypts the unique identifier; decrements the unique identifier; encrypts the decremented unique identifier; then returns the encrypted and decremented unique identifier;

- The Responder client authenticates the decremented unique identifier.

The format of the payload is shown in table 97.

Table 97 – SK_DU SKE_KEYAUT payload

Item	Size (bytes)
encrypted with secret_key	
unique_id	8
reserved	8

The key authentication information consists of two parts, encrypted with the secret_key:

- the value of the unique_id;

- reserved (fills 16-byte block).

Fibre Channel service reject (FS_RJT) signifies rejection of the SKE_KEYAUT command.

Fibre Channel service accept (FS_ACC) signifies that the Originator client has received the authentication information from the Responder client. The format of the accept payload is shown in table 98.

Table 98 – FS_ACC SK_DU to SKE_KEYAUT

Item	Size (bytes)
encrypted with secret_key	
unique_id - 1	8
reserved	8

The accept consists of two parts, encrypted with the secret_key:

- the decremented value of the unique_id;

- reserved (fills 16-byte block).

10.5 Security Data Bases

The Security-Key Server shall contain a distribution data base for each client as defined in table 99.

Table 99 – Server distribution_key data base

Item	Size (bytes)
encryption_id	4
for each security-key client	
client_id	4
distribution_key	16

The data base consists of two parts:

– The value of the identifier for the encryption algorithm for the client's distribution keys;

– for each security-key client:

  – the value of the client's address identifier;

  – the value of the client's distribution_key.

Each security-key client shall contain a distribution data base as defined in table 100.

Table 100 – Client's distribution_key data base

Item	Size (bytes)
encryption_id	4
distribution_key	16

The data base consists of two parts:

– the value of the identifier for the encryption algorithm for the distribution_key;

– the value of the client N_Port's unique distribution_key.

As defined in table 101, each security-key client shall contain a secret data base for each participating client with which a secret_key was exchanged.

Table 101 – Client's secret_key data base

Item	Size (bytes)
participant_id	4
encryption_id	4
secret_key	16

The date base consists of three parts:

– The identifier of the participating client with which the secret_key was exchanged;

– the identifier of the encryption algorithm for the secret key;

– the value of the secret_key.

10.6 Encryption Algorithm

The specification of encryption algorithms for key encryption and data encryption is beyond the scope of this document (see Annex B). The Security-Key Server and all of its clients shall use the same algorithm for key encryption. Multiple algorithms may be used for data encryption.

The encryption algorithm and the encryption key shall be a characteristic of the encryption identifier. Table 102 defines encryption identifiers for several popular encryption algorithms including recommended key lengths. User

defined encryption identifiers can utilize other encryption algorithms and key lengths.

Table 102 – Encryption identifiers

Key Type	Key Size (bits)	encryption_id (hex)
RC4 Key	90	0001005A
RC4 Key	40	00010028
RC4 Data	75	0002004B
RC4 Data	40	00020028
DES Key	56	00030038
DES Data	56	00040038

Annex A: Service Interface Provided by FC-CT

(Informative)

This annex specifies the services provided by FC-CT and the services required by FC-CT. The services provided by FC-CT are categorized as:

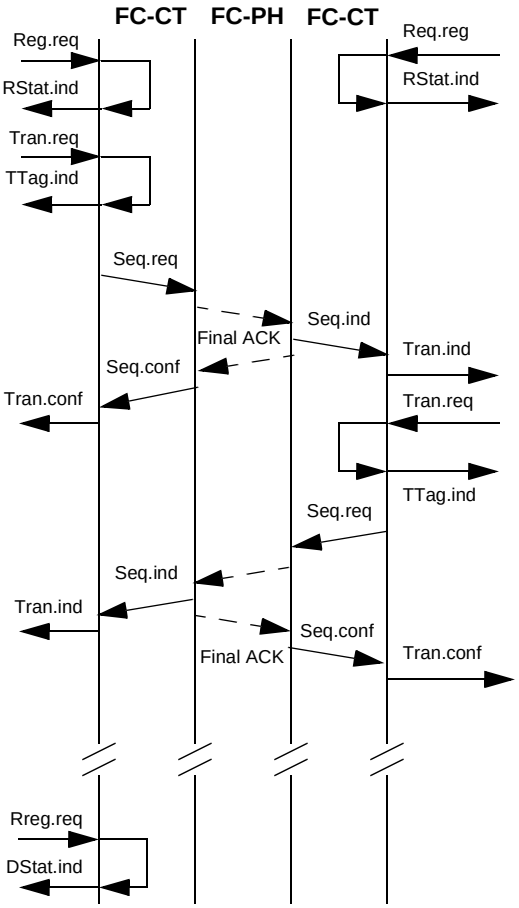
a) Session and Transaction services provided by FC-CT to its local users (a Fibre Channel Service application), denoted by the prefix FC_CT_;

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.

Figure A.1 shows a sample interchange of request and response transactions between two FC-CT entities.



Abbreviations:

Dreg	FC_CT_DEREG
DSta	FC_CT_DEREG_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 subsequent 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. 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,
Iu}

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 primitive 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,
Iu}

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: Encryption Algorithm Recommendation

(Informative)

It is recommended that RSA's RC4 encryption algorithm be used as the encryption algorithm for the security-key distribution service. In addition, it is also recommended that RC4 also be used for data encryption.

As a general rule (see subclause 2.3), it is recommended that the size of the distribution key be 90-bits in lengths. It is also recommended that the size of the secret key be 75-bits in length. However, shorter secret key lengths may be sufficient if the data is perishable or if the secret key is changed frequently.

B.1 DES

DES is the Data Encryption Standard, an encryption block cipher defined and endorsed by the U.S. government in 1977 as an official standard; the details can be found in the official FIPS publication. It was originally developed at IBM. DES has been extensively studied over the last 15 years and is the most well-known and widely used cryptosystem in the world.

NOTE – DES is almost never approved for export.

DES is a secret key, symmetric cryptosystem: when used for communication, both sender and receiver must know the same secret key, which is used both to encrypt and decrypt the message. DES can also be used for single-user encryption, such as to store files on a hard disk in encrypted form. In a multi-user environment, secure key distribution may be difficult; public-key cryptography was invented to solve this problem. DES operates on 64-bit blocks with a 56-bit key. It was designed to be implemented in hardware, and its operation is relatively fast. It works well for bulk encryption, that is, for encrypting a large set of data.

B.2 RC2 and RC4

RC2 and RC4 are variable-key-size cipher functions designed by Ron Rivest for fast bulk encryption. They are an alternative to DES and are as fast or faster than DES. They can be more secure than DES because of their ability to use long key sizes; they can also be less secure than DES if short key sizes are used. RC2 is a variable-key-size symmetric block cipher and can serve as a drop-in replacement for DES, for example in export versions of products otherwise using DES.

RC2 can be used in the same modes as DES, including triple encryption. RC2 is approximately twice as fast as DES, at least in software. RC4 is a variable-key-size symmetric stream cipher and is 10 or more times as fast as DES in software. Both RC2 and RC4 are very compact in terms of code size.

NOTE – RC2 and RC4 are proprietary algorithms of RSA Data Security, Inc.; details have not been published. An agreement between the Software Publishers Association (SPA) and the U.S. government gives RC2 and RC4 special status by means of which the export approval process is simpler and quicker than the usual cryptographic export process. However, to qualify for quick export approval a product must limit the RC2 and RC4 key sizes to 40 bits; 56 bits is allowed for foreign subsidiaries and overseas offices of U.S. companies. RC2 and RC4 have been widely used by developers who want to export their products.

B.3 Minimal Key Length

At the request of the Business Software Alliance (BSA), an ad hoc panel of seven cryptologists and computer scientists (Matt Blaze, Whitfield Diffie, Ronald L. Rivest, Bruce Schneier, Tsutomu Shimomura, Eric Thompson, and Michael Wiener) addressed the question of the minimum key length required to provide adequate security against exhaustive search in commercial applications of symmetric cryptosystems.

The following is an abstract of the BSA report dated January 1996:

> "Encryption plays an essential role in protecting the privacy of electronic information

against threats from a variety of potential attackers. In so doing, modern cryptography employs a combination of conventional or symmetric cryptographic systems for encrypting data and public-key or symmetric systems for managing the keys used by the symmetric systems. Assessing the strength required of the symmetric cryptographic systems is therefore an essential step in employing cryptography for computer and communication security.

“Technology readily available today (late 1995) makes brute-force attacks against cryptographic systems considered adequate for the past several years both fast and cheap. General purpose computers can be used, but a much more efficient approach is to employ commercially available Field Programmable Gate Array (FPGA) technology. For attackers prepared to make a higher initial investment, custom-made, special-purpose chips make such calculations much faster and significantly lower the amortized cost per solution.

“As a result, cryptosystems with 40-bit keys offer virtually no protection at this point against brute-force attacks. Even the U.S. Data Encryption Standard with 56-bit keys is increasingly inadequate. As cryptosystems often succumb to `smarter' attacks than brute-force key search, it is also important to remember that the key lengths discussed here are the minimum needed for security against the computational threats considered.

“Fortunately, the cost of very strong encryption is not significantly greater than that of weak encryption. Therefore, to provide adequate protection against the most serious threats (well funded commercial enterprises or government intelligence agencies) keys used to protect data today should be at least 75 bits long. To protect information adequately for the next 20 years in the face of expected advances in computing power, keys in newly-deployed systems should be at least 90 bits long.”

Annex C: Name Server Request Mnemonics

(Informative)

C.1 Changed Name Server Mnemonics

The original Name Server proposal submitted for FC-GS-2 had a different, and somewhat confusing set of mnemonic names for the various requests. At a later time, a different set of mnemonics was chosen to be more “user friendly”.

By the time the new mnemonics were created, other documents were created that referred to the older names. The following table is provided as a cross reference between the original and final FC-GS-2 names.

Table C.1 – Name Server Mnemonic Changes

Original Mnemonic	FC-GS-2 Mnemonic
GAN	GA_NXT
GPN	GPN_ID
GNN1	GNN_ID
GCoS	GCS_ID
GFC–4	GFT_ID
GSPN	GSPN_ID
GPT	GPT_ID
GP_ID1	GID_PN
GP_ID2	GID_NN
GIP_A	GIP_NN
GIPA1	GIPA_NN
GSNN	GSNN_NN
GNN2	GNN_IP
GIPA2	GIPA_IP
GP_ID3	GID_FT
GP_ID4	GID_PT
RPN	RPN_ID
RNN	RNN_ID
RCoS	RCS_ID
RFC–4	RFT_ID
RSPN	RSPN_ID

Table C.1 – Name Server Mnemonic Changes

Original Mnemonic	FC-GS-2 Mnemonic
RPT	RPT_ID
RIP_A	RIP_NN
RIPA	RIPA_NN
RSNN	RSNN_NN
RA	DA_ID

A

B

C

D

F

H

I

L

M

N

R

S

T

W