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

NCITS xxx-200x
T11/Project 1356-D/Rev 6.2

FIBRE CHANNEL

GENERIC SERVICES - 3

(FC-GS-3)

REV 6.2

NCITS working draft proposed
American National Standard
for Information Technology

March 21, 2000

Secretariat: Information Technology Industry Council

POINTS OF CONTACT:

Kumar Malavalli (T11 Chair)
Brocade Communications Systems, Inc.
1901 Guadalupe Parkway
San Jose, CA 95131
Voice: (408) 487-8156
Fax: (408) 487-8101
EMail: kumar@brocade.com

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

Michael E. O'Donnell (Facillitator)
McDATA Corporation
310 Interlocken Parkway
Broomfield, Co. 80021
Voice: (303) 460-4142
Fax:
E-mail: modonnell@mcdata.com

Jeffrey Stai (Technical Editor / T11.3 Chair)
QLogic Corporation
26600 Laguna Hills Drive
Aliso Viejo, CA 92656
Voice: (949) 389-6238
Fax: (949) 389-6121
E-mail: j_stai@qlc.com

Release Notes for version 6.2:

- Added Fabric Zone Server (00-017v1)
- Added Partial Response bit (00-006v0)
- Added several changes based on the Feb 2000 WG review (00-032v0)
- Added FC-4 Features object to replace the partially defined feature bits
- Unified timeout definitions to clause 4
- Added persistence rules to clause 4
- Deleted SNMP (which, er, that did not follow CT)
- Security Key Server renamed to Key Server
- Added a proposal (NOT WG approved!) for IP Address Server
- Redefined “Application” to “Client”, as a clarification
- The GS-3 Which Project: changed every “which” to “that” - at least those “which” I could stand to change...

Release Notes for version 6.1:

- Added 99-727, 99-692, 99-592, 99-697, 99-714, 99-736 - all as modified by the WG
- net changes were to DS and MS
- DS: added new commands to NS, and new FC-4 Descriptor object
- MS: added Fabric Config Server and Unzoned Name Server

Release Notes for version 6.0: Below is a summary of the work done to the prior FC-GS-2 version (5.3) to create this version.

- Create baseline draft with single column layout.
- Extensive simplifying rewrite of clause 4.
- Reorganize all Service and Server descriptions to the same structure.
- Add 99-555v0, as modified.
- Add skeleton for IP Address Server, as per WG.
- Add editor’s notes identifying potential future work items.

BSR NCITS xxx-200x

American National Standard
for Information Technology

Fibre Channel — Generic Services - 3 (FC-GS-3)

Secretariat

Information Technology Industry Council

Approved (not yet approved)

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, mangement services, discovery services, time services, alias services, SNMP services, and security key distribution services.

This standard replaces X3.288-1999.

Published by

American National Standards Institute
11 West 42nd Street, New York, NY 10036

Foreword (This Foreword is not part of American National Standard NCITS xxx-200x.)

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 1999-2000. The standards approval process started in 2000. This document includes annexes which are informative and are not considered part of the standard.

Requests for interpretation, suggestions for improvements or addenda, or defect re-ports are welcome. They should be sent to the NCITS Secretariat, Information Tech-nology Industry Council, 1250 Eye Street, NW, Suite 200, Washington, DC 20005-3922.

This standard was processed and approved for submittal to ANSI by the National Committee for Information Technology Standards (NCITS). Committee approval of the standard does not necessarily imply that all committee members voted for approval.

At the time it approved this standard, NCITS had the following members:

(to be filled in by NCITS)

Introduction

(UPDATE THIS BEFORE FORWARDING)

FC-GS-3 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:

– TBD

Contents Page

Contents Page

Figures Page

Tables Page

Tables Page

Tables Page

American National Standard
for Information Technology —

Fibre Channel — Generic Services - 3 (FC-GS-3)

1 Scope

FC-GS-3 describes in detail the basic Fibre Channel services introduced in ANSI X3.230, FC-PH.

The Fibre Channel services described in this document are:

- Directory Services
- Management Services
- Time Services
- Alias Service
- Key Service
- {OTHER SERVICES? TBD}

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 that, 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.

For electronic copies of some standards, visit ANSI's Electronic Standards Store (ESS) at www.ansi.org. or printed versions of all standards listed here, contact Global Engineering Documents, 15 Inverness Way East, Englewood, CO; 80112-5704, (800) 854-7179..

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

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

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

[9] IEEE 802.10-1998*, IEEE Standards for Local and Metropolitan Area Networks : Interoperable-LAN/MAN Security (SILS).*

2.2 References under development

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

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

[11] ANSI NCITS xxx-199x, *Fibre Channel - Switch Fabric - 2 (FC-SW-2),* T11/Project 1305D/Rev 4.1

[12] ANSI NCITS xxx-199x, *Fibre Channel - Virtual Interface (FC-VI),* T11/Project 1332D/Rev 1.2

[13] ANSI NCITS xxx-199x, *Fibre Channel - Framing and Signaling Interface (FC-FS),* T11/Project 1331D/Rev 1

[14] ANSI NCITS xxx-199x, *SCSI Fibre Channel Protocol - 2 (FCP-2),* T10/Project 1144D/Rev 4

2.3 Other references

The following document is available from the Fibre Channel Industry Association (FCIA), 2570 West El Camino Real, Ste. 304, Mountain View, CA 94040-1313; (800) 272-4618 or (650) 949-6730 (phone); or at www.fibrechannel.com.

[15] 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/`.

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

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

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

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

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

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

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

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

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

[25] RFC 2625, *IP and ARP over Fibre Channel*, June 1999.

The following documents are projects of the Internet Protocol over Fibre Channel (ipfc) working group of the Internet Engineering Task Force (IETF). Information about the ipfc and its projects is located at `http://www.ietf.org/html.charters/ipfc-charter.html`.

[26] Internet Draft, *Definitions of Managed Objects for the Fabric Element in Fibre Channel Standard*, January 31, 2000.

3 Definitions and conventions

For FC-GS-3, 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 that 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 Broadcast Zone: A Zone which {WHAT?}.

3.1.4 Client: An entity that, via its CT, makes requests of a Server.

3.1.5 Common Transport (CT): A protocol defined by this Standard that provides access to Services and their related Servers. “CT” may also refer to an instance of the Common Transport.

3.1.6 Directory: A repository of information about objects that may be accessed via the Directory Service.

3.1.7 fabric: Any interconnect between two or more N_Ports, including point-to-point, loop, and Switched Fabric.

3.1.8 Generic Services: The collection of Services defined by this Standard.

3.1.9 Hard Zone: A Zone which is enforced by the Fabric, often as a hardware function.

3.1.10 Link Service Facilitator: The entity at well-known address hex ‘FFFFFE’; known as the “Fabric F_Port” in FC-PH, the address identifier to that Fabric Login is directed.

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

3.1.12 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.

3.1.13 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.

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

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

3.1.16 Requesting_CT: A CT that is sending a request.

3.1.17 Responding_CT: A CT that is responding to a prior request.

3.1.18 Server: A Server is an entity that, via its CT, accepts requests from a Client and provides responses to the Client. A Server is accessed via a Service. For example, the Name Server is accessed using the Directory Service.

3.1.19 Service: A Service is provided by a Node, accessible via an N_Port that is addressed by a Well-Known Address. Examples of a Service include the Directory Service and the Alias Service. A Service provides access to one or more Servers.

3.1.20 Soft Zone: A Zone that is visible to devices via Name Server exposure of Zone Members.

3.1.21 Switched Fabric: A fabric comprised of one or more Switches (see FC-SW, reference [6]).

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

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

3.1.24 Well-Known Address: An address identifier defined in FC-PH to access a Service.

3.1.25 Zone: A collection of Zone Members. Zone Members in a Zone are made aware of each other, but not made aware of devices outside the Zone. A Zone can be defined to exist in one or more Zone Sets. (Note that this definition supersedes that defined in FC-FG, reference [4]).

3.1.26 Zone Definition: The parameters that define a Zone: the Zone Name, number of Zone Members, and Zone Member definition.

3.1.27 Zone Member: A device to be included in a Zone, as specified by its Zone Member Definition. Devices at well known addresses shall not be Zone Members.

3.1.28 Zone Member Definition: The parameter by which a Zone Member is specified. A Zone Member may be specified by: 1) as a port on a Switch, (specifically by Domain_ID and port number); or, 2) the device's N_Port_Name; or, 3) the device's address identifier.

3.1.29 Zone Name: The name assigned to a Zone.

3.1.30 Zone Set: One or more Zones which may be activated or deactivated as a group.

3.1.31 Zone Set Name: The name assigned to a Zone Set.

3.1.32 Zone Set State: The state of a Zone Set, which may be either activated or deactivated.

3.1.33 Active Zone Set: The Zone Set that is currently activated. Only one Zone Set may be activated at any time.

3.2 Editorial Conventions

In FC-GS-3, 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 in table 1.

Table 1 – ISO and American Conventions

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 that 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 3.1.

AS	Alias Service
CT	Common Transport
CT_IU	Common Transport Information Unit
D_ID	Destination address identifier
ELS	Extended Link Service
FCS	Fabric Configuration Server
GS	Fibre Channel Generic Service
IN_ID	Initial Identifier
IP	Internet Protocol
IPA	Initial Process Associator
IU	Information Unit
MS	Management Service
NAA	Network Address Authority
NS	Name Server
NSM	Native SNMP Mapping
RNID	Request Node Identification Data Extended Link Service
RTIN	Request Topology Information Extended Link Service

SI	Sequence Initiative
S_ID	Source address identifier
SNMP	Simple Network Management Protocol
TS	Time Service
UDP	User Datagram Protocol
ULP	Upper Level Protocols
WKA	Well-Known Address

4 Common Transport for Generic Services (CT)

4.1 Overview

Fibre Channel Generic Services share a Common Transport (CT) at the FC-4 level. The CT provides access to a Service (e.g. Directory Service) with a set of service parameters that facilitates the usage of Fibre Channel constructs. It also provides another level of multiplexing that simplifies the Server-to-Server communication for a distributed Service. It is important to note that Fibre Channel Generic Services do not require a high performance communication channel as do high performance I/O protocols such as SCSI, IP, VI, etc. The relationship of CT with respect to Generic Services and FC-PH is illustrated in figure 1.

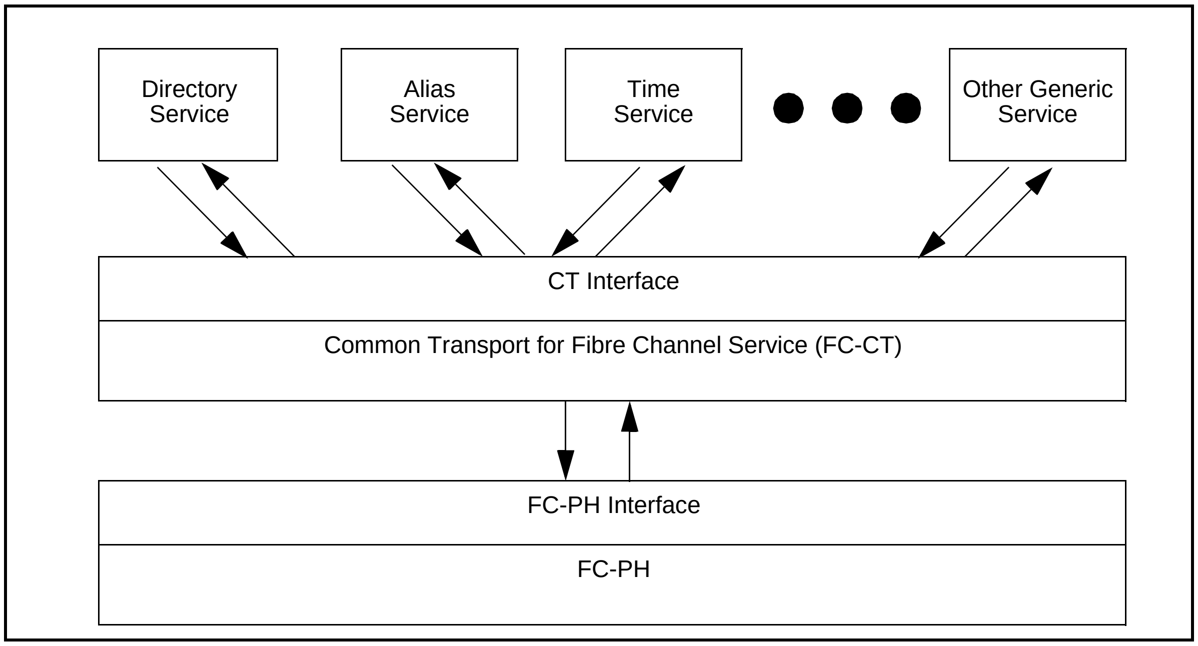


Figure 1 – Relationship of the Common Transport with Generic Services and FC-PH

The CT provides access to a Service, which may then provide access to more than one Server. Each Server may provide access to different information or controls, or each Server may provide a different access model to the same information and controls. The Service is accessed via its Well-Known Address and is referenced by its GS_Type. The Server beneath the Service is referenced by its

GS_Subtype within the GS_Type. The relationship between CT, a Service, and its Servers is illustrated in figure 2.

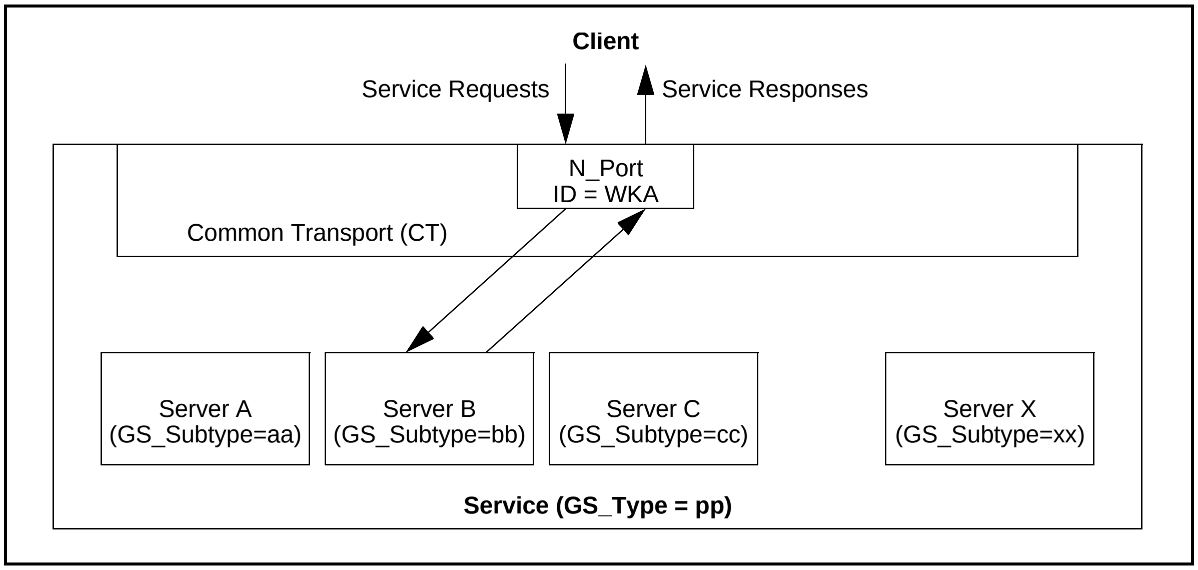


Figure 2 – Relationship between Common Transport, Service, and Servers

4.2 General concepts

The following parameters describe the information that the CT delivers to a Service:

– type of Service and Server;

– type of transaction;

– mode of transaction;

– Class of service;

– maximum size of an IU.

The types of Services described by this document that are accessible via the CT are:

– Name Service;

– Alias Service;

– Management Service;

– Time Service;

– Security-key Service.

There are three types of transactions:

– Request: where one entity (Client) transmits an IU to another entity (Server) to request a Service;

– Response: where the Server transmits an IU to the Client responding to its earlier request for a Service;

– Unsolicited: where one entity (either a Client or a Server) 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 4.5.2);

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

The Class of service is an indication of the quality of service that a Client expects from the underlying transport. The following three Classes of service may be used to request a Service:

– Class 1;

– Class 2;

– Class 3.

Since not all Classes are necessarily available, this parameter describes a list of Classes of services in a descending order of preference. It is used to express the desire of the Client in the order of availability. Class 3 service, if available in the operational environment, shall also be available to the Client.

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 preamble information.

NOTE – When the Server is distributed amongst several entities within the Fabric, Class F service may be used for communicating CT information between those entities. This usage of Class F is not defined by this standard (see FC-SW-2).

A Client may wish to restrict the size of IUs that it wants to receive from a Server. A Server shall observe and obey this restriction on behalf of the Client. It may do so as described in 4.3.1.7.

4.3 CT protocol

A Common Transport IU (CT_IU) is the common Fibre Channel Sequence used to transfer all information between a Client and a Server. The first sixteen bytes of the CT_IU contain a preamble with information common to all CT_IUs. The remainder of the CT_IU contains additional information as defined by the preamble, and may be zero or more bytes in length. The format of the CT_IU is shown in table 2.

Table 2 – 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	GS_Type	GS_Subtype	Options	Reserved
2	Command/Response code		Maximum/Residual Size	
3	Reserved	Reason code	Reason Code Explanation	Vendor Unique
4...n	additional information			

4.3.1 CT_IU preamble description

The first sixteen bytes of the CT_IU are defined as follows.

4.3.1.1 Revision Field

This field denotes the revision of the protocol. This version has the value of 1.

4.3.1.2 IN_ID Field

This field shall be set to zero by the Requesting_CT.

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

4.3.1.3 GS_Type Field

This field is used to indicate the type of Service. The values are defined in table 3.

Table 3 – GS_Type values

Encoded value (hex)	**Description**
00-1F	Vendor Unique
20	Reserved for use by FC-SW-2
F7	Key Service
F8	Alias Service
FA	Management Service
FB	Time Service
FC	Directory Service
All Others	Reserved

4.3.1.4 GS_Subtype Field

This field indicates the specific Server behind the Service. Values in this field are provided by the individual Service.

NOTE – The GS_Subtype field is used to indicate second level routing behind the N_ Port. For example, if more than one Server is provided by the Directory Service at the well-known address hex ’FFFFFC, then the GS_Subtype field is used to distinguish these different Servers.

4.3.1.5 Options Field

This field denotes options used by the Requesting_CT, as shown in table 4.

Table 4 – Options field bits

Bit Position	Description
7	Exchange mapping. A value of zero indicates a single bidirectional Exchange (synchronous mode transactions). A value of one indicates multiple Exchanges (asynchronous mode transactions).
6	Partial Response. A value of one indicates that the response is known to be incomplete. A value of zero indicates that the completeness of the response is not specified. (See note below.)
5-0	Reserved

NOTE – For example, when a Server is distributed amongst several entities (e.g., Switches), if one or more of the entities fails to respond, the Partial Response bit may be used to indicate that those entities did not participate in the answer given. Note that a zero value for this bit only indicates that the Server does not have any knowledge that the response is incomplete, or that it is unable to report that knowledge.

4.3.1.6 Command/Response code field

This field indicates whether the CT_IU is a request or a response. If the CT_IU is a request, this field then specifies the command to be performed. If the CT_IU is a response, then this field indicates whether the request was accepted or rejected. Requests and responses are further defined in 4.4. Table 5 depicts the valid Command/Response code values.

Table 5 – Command/Response codes

Encoded Value (hex)	Description
0000	Not a Request CT_IU or Response CT_IU **{THIS MAY BE RESERVED - ED}**
0001-7FFF	Request CT_IU
8001	Reject Response CT_IU
8002	Accept Response CT_IU
other values	Reserved

4.3.1.7 Maximum/Residual Size field

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

- In the Request CT_IU:
  - hex ‘0000’: No Maximum Size indicated; the Accept CT_IU may be any size.
  - hex ‘FFFF’: This value is reserved for the Request CT_IU.
  - Any other value: The Encoded Value indicates the maximum number of words that shall be sent in the Accept CT_IU, not inclusive of the CT_HDR.
- In the Accept CT_IU:
  - hex ‘0000’: All available information was returned in the Accept CT_IU.
  - hex ‘FFFF’: The Encoded Value indicates greater than 65534 available words of information were not sent in the Accept CT_IU, in excess of the Maximum Size specified in the Request CT_IU.
  - Any other value: The Encoded Value indicates the number of available words of information that were not sent in the Accept CT_IU, in excess of the Maximum Size specified in the Request CT_IU.
- For the Reject CT_IU and any other CT_IU, this field is reserved.

4.3.1.8 Reason code field

The reason code field contains the reason code associated with a Reject CT_IU (see 4.4.3). When the Command/Response code field contains a value of hex ’0000’, this field is undefined.

4.3.1.9 Reason code explanation field

This field contains a reason code explanation associated with a Reject CT_IU (see 4.4.3). When the Command/Response code field contains a value of hex ’0000’, this field is undefined.

4.3.1.10 Vendor unique field

This field contains a vendor unique reason code associated with a Reject CT_IU (see 4.4.3). When the Command/Response code field contains a value of hex ’0000’, this field is undefined.

4.3.2 CT_IU Additional information

Following the sixteen byte CT_IU preamble, the additional information bytes contain information specific to the GS_Type, GS_Subtype, and Command/Response code.

4.4 CT Information Units (CT_IU)

A set of Server request and response CT_IUs are defined by CT for use by the Generic Services. One Request CT_IU and two Response CT_IUs - Accept CT_IU and Reject CT_IU - are defined.

4.4.1 Request CT_IU

A Request CT_IU is a CT_IU in which the Command/Response code field contains a command code value.

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

4.4.2 Accept CT_IU

An Accept CT_IU is a CT_IU in which the Command/Response code field contains a value of hex ’8002’. The Accept CT_IU shall notify the Client that the request has been successfully completed.

The additional information in the CT_IU contains the associated response specific data. The associated response specific data shall be defined independently by each Server.

4.4.3 Reject CT_IU

A Reject CT_IU is a CT_IU in which the Command/Response code field contains a value of hex ’8001’. The Reject CT_IU shall notify the Client that the request has been unsuccessfully completed.

The Reason code indicates the general reason why the request was rejected. Table 5 indicates the defined Reject CT_IU 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 – Reject CT_IU Reason Codes

Encoded Value	Description
0000 0001	Invalid command code
0000 0010	Invalid version level
0000 0011	Logical error
0000 0100	Invalid CT_IU Size
0000 0101	Logical busy
0000 0111	Protocol error
0000 1001	Unable to perform command request
0000 1011	Command not supported

Table 6 – Reject CT_IU Reason Codes

Encoded Value	Description
others	Reserved
1111 1111	Vendor Unique Error (see Vendor Unique field)

Invalid command code: The command code passed in the Request CT_IU is not defined by the Server.

Invalid version level: The specified version level is not supported by the Server.

Logical error: The request identified by the Request CT_IU command code and additional information content is invalid or logically inconsistent for the conditions present.

Invalid CT_IU size: The CT_IU size is invalid for the 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 that violates the rules of the Server protocol which are not specified by other error codes.

Unable to perform command request: The Server is unable to perform the request.

Command not supported: The Server does not support the command requested.

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

4.5 FC-PH mapping

The CT maps CT_IUs into the appropriate Fibre Channel constructs.

4.5.1 Synchronous mode transactions

4.5.1.1 Fabric login and N_Port login

An N_Port shall perform Fabric Login, and shall perform N_Port Login with the WKA of the Service, in the manner that is specified in FC-PH, before making any requests of a Server provided by the Service.

NOTE – An N_Port that has completed its requests with a Server should perform explicit N_Port Logout with the Service.

A Service may perform N_Port Logout (LOGO) if it becomes resource constrained. A service may use a least recently used algorithm in determining which entity to Logout.

4.5.1.2 Class of Service

Any Class of Service may be used for the Request CT_IU.

The Class of Service of the Response CT_IU shall be the same as the Class of Service used for the corresponding Request CT_IU.

4.5.1.3 Exchange and Sequence management

For synchronous mode transactions, the Requesting_CT shall set the Exchange mapping bit in the Options field to zero.

A single bidirectional Exchange shall be used. The Request CT_IU shall be the first Sequence of an Exchange, and its associated Response CT_IU shall be the last Sequence of the same Exchange.

The SI is transferred at the end of a CT_IU transmission. If the Responding_CT does not have the SI at the end of a CT_IU reception, it shall consider this to be a protocol error and shall terminate the Exchange.

An Exchange created by a Requesting_CT is to be used only for a specific Server, and shall not be shared with another Server within the same or a different Service. 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 table for synchronous mode transactions are shown in table 7. 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 7 – CT_IU table for synchronous transaction

CT_IU Name	Information Category	Payload	F/M/L	SI	M/O
Request	2	One Request CT_IU	F,M	T	M
Response	3	One Response CT_IU	M,L	T	M

4.5.1.4 FC-2 interface

The mapping of CT_IUs to Fibre Channel Sequences and frames is described.

4.5.1.4.1 Routing bits

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

4.5.1.4.2 Information category

This parameter shall be set as indicated in table 7.

4.5.1.4.3 D_ID

The D_ID shall identify the destination Fibre Channel address identifier for the CT_IU. For Request CT_IUs, this is the WKA of the Service. For Response CT_IUs, this is address identifier of the requesting N_Port.

4.5.1.4.4 S_ID

The S_ID shall identify the source Fibre Channel address identifier for the CT_IU. For Request CT_IUs, this is address identifier of the requesting N_Port. For Response CT_IUs, this is the WKA of the Service.

4.5.1.4.5 Type

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

4.5.1.4.6 First Sequence

The Requesting_CT shall set this parameter true to originate a new Exchange in order to transmit a Request CT_IU.

4.5.1.4.7 Last Sequence

The Service shall set this parameter true in the Response CT_IU to terminate an Exchange at the end of the transaction.

4.5.1.4.8 Sequence Initiative

This parameter shall be set as described in 4.5.1.3.

4.5.1.4.9 Continue Sequence condition

The use of this parameter is implementation specific.

4.5.1.4.10 Exchange reassembly

The CT shall set this parameter false.

4.5.1.4.11 Relative offset

Relative offset may be used 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 CT_IU shall be set to 0.

4.5.1.4.12 Optional headers

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

4.5.1.5 Correlation of requests and responses

The correlation of Request CT_IUs and Response CT_IUs shall be managed by the specific Service. FC-CT provides no generic mechanism for this management.

NOTE – Services that make use of synchronous mode transactions will typically correlate a request to a response through the use of Exchange IDs.

4.5.1.6 Error detection and recoveryt

There are two levels of error that may be detected:

- invalid CT_IU or FC-CT protocol error;
  - invalid/undefined GS_Type;
  - invalid revision level;
  - invalid Options;
  - request to response timeout;
  - Sequence payload exceeds the maximum size of CT_IU at a destination FC_CT.
- FC-PH protocol errors;
  - Sequence errors;
  - Exchange errors.

When an invalid CT_IU or FC-CT protocol error is recognized, the Responding_CT indicates the error condition to the Requesting_CT using a Reject 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.

All error conditions shall also be indicated to the Client.

4.5.2 Asynchronous mode transactions

4.5.2.1 Fabric login and N_Port login

The N_Port shall perform Fabric Login, and shall perform N_Port Login with the WKA of the Service, in the manner that is specified in FC-PH.

4.5.2.2 Class of Service

For asynchronous mode transactions, the Class of Service is not defined by this Standard.

NOTE – For simplification when using asynchronous transactions, a replying CT should use the same Class of service in its subsequent communication with an initiating CT.

4.5.2.3 Exchange and Sequence management

For asynchronous mode transactions, the Requesting_CT shall set the Exchange mapping bit in the Options field to one.

Separate Exchanges in each direction shall be used. That is, each CT shall originate an Exchange and hold the Sequence Initiative (SI). Transfer of the SI to the other CT shall be considered a protocol error and the other CT shall terminate the Exchange.

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

An Exchange created by a CT is to be used only for a specific Server, and shall not be shared with another Server within the same or a different Service. 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 table for synchronous mode transactions are shown in table 8. 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 8 – CT_IU table for asynchronous transactions

CT_IU Name	Information Category	Payload	F/M/L	SI	M/O
Request	2	One Request CT_IU	F,M,L	H	M
Response	3	One Response CT_IU	F,M,L	H	M
Unsolicited	2	One Request CT_IU	F,M,L	H	M

4.5.2.4 FC-2 interface

The mapping of CT_IUs to Fibre Channel Sequences and frames is described.

4.5.2.4.1 Routing bits

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

4.5.2.4.2 Information category

This parameter shall be set as indicated in table 8.

4.5.2.4.3 D_ID

The D_ID shall identify the destination Fibre Channel address identifier for the CT_IU.

4.5.2.4.4 S_ID

The S_ID shall identify the source Fibre Channel address identifier for the CT_IU.

4.5.2.4.5 Type

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

4.5.2.4.6 First Sequence

The CT shall set this parameter true to originate a new Exchange.

4.5.2.4.7 Last Sequence

The Service shall set this parameter true to terminate an Exchange at the end of the transaction.

4.5.2.4.8 Sequence Initiative

This parameter shall be set as described in 4.5.2.3.

4.5.2.4.9 Continue Sequence condition

The use of this parameter is implementation specific.

4.5.2.4.10 Exchange reassembly

The CT shall set this parameter false.

4.5.2.4.11 Relative offset

Relative offset may be used 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 CT_IU shall be set to 0.

4.5.2.4.12 Optional headers

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

4.5.2.5 Correlation of requests and responses

The correlation of Request CT_IUs and Response CT_IUs shall be managed by the specific Service. FC-CT provides no generic mechanism for this management.

NOTE – Services that make use of asynchronous mode transactions will typically include a tag value within the additional information to correlate requests with responses.

4.5.2.6 Error detection and recoveryt

There are two levels of error that may be detected:

- invalid CT_IU or FC-CT protocol error;
  - invalid/undefined GS_Type;
  - invalid revision level;
  - invalid Options;
  - request to response timeout;
  - Sequence payload exceeds the maximum size of CT_IU at a destination FC_CT.
- FC-PH protocol errors;
  - Sequence errors;
  - Exchange errors.

When an invalid CT_IU or FC-CT protocol error is recognized, the Responding_CT indicates the error condition to the Requesting_CT using a Reject CT_IU.

When an FC-PH protocol error is detected, the asynchronous mode transaction 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.

All error conditions shall also be indicated to the Client.

4.6 Time constants

The following timeout values are defined for CT operations.

4.6.1 Request to response time

If the Requesting_CT does not receive a Response CT_IU from the Responding_CT within 30 seconds, it shall consider this to be a protocol error.

4.6.2 Database propagation delay

A time lag may exist between successful completion of a request that causes an update to a database, and the time that the updated information is returned in response to a query request. This time lag is implementation and system dependent but shall not exceed 60 seconds.

NOTE – For example, consider a large fabric with distributed Servers in each Switch. A registration request is completed at one point in the fabric. At a point distant from the first point of the fabric, a query is made related to the just-registered information. The query may return “stale” information because the new information has not yet been distributed across the fabric. Or, the local distributed Server responding to the query may not yet have received an indication that its local cache of information needs to be refreshed.

4.7 Persistence of actions after Logout

In order to conserve resources, a Client may Logout with a Server once it has completed its activity with a Server. Similarly, a Server may Logout with a Client with which it has no open Exchanges.

In either case, the Server shall preserve the results of the prior actions requested by the Client following the Logout and following any subsequent Login.

Exceptions to this rule may be defined by a specific Server.

NOTE – The idea is to allow a Client to Login with a Server, to then register information with that Server, and to Logout with that Server, and have the registered information remain registered at the Server. This allows a device to, for example, register itself with the Name Server and not have to then maintain the Login to remain registered.

5 Directory Service

The Directory Service 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 Directory Service information. This standard does not define the structure of this information.

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

Table 9 defines the GS_Subtype codes for the Directory Service.

Table 9 – Directory Service subtype values

Values in hex	Description
01	Reserved
02	Name Server
03	IP Address Server
80-EF	FC-4 specific Servers
	{OTHERS TBD?}
other values	Reserved

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

In addition to the standard Servers, an individual FC-4 may provide its own specific Server. FC-4 based Server payloads and protocols are defined by the specific FC-4.

The consumer of a Directory Service is normally a "device driver" or some other internal layer of an operating system. Directory Service information is not normally forwarded to an application level. The information provided by a Directory Service is operational, and therefore may be constrained by the operational environment (Zone) of the Node.

5.1 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 (see clause 4). Three types of requests are defined for the Name Server, as shown in table 10.

Table 10 – Name Server – Request types

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

Table 11 lists the different Fibre Channel objects defined for the Name Server.

Table 11 – 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
E	FD	FC-4 Descriptor	variable length (0 to 255-byte) field
F	FF	FC-4 Features	128-byte array, four bits per TYPE

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 the Name Server immediately available to N_Ports and NL_Ports once they have successfully completed Fabric Login. However, the Name Server 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.

5.1.1 Name Server protocol

Name Server registration, deregistration and queries are managed through protocols containing a set of Request CT_IUs and Response CT_IUs supported by the Name Server.

Synchronous transactions shall be used, as defined in 4.5.1. For a Name Server request, the Name Server payload shall be transported from the requestor to the Name Server using a Request CT_IU. 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.

If Zones exist within the fabric, the Name Server shall restrict access to information in the Name Server database based on the Zone configuration.

5.1.1.1 CT_IU preamble values

The following values shall be set in the CT_IU preamble for Name Server request and their responses; fields not specified here shall be set as defined in 4.3.1:

– Revision: hex ‘01’;

– GS_Subtype: as indicated in table 9;

– Command Code: see table 12 for Request command codes.

Table 12 – Name Server – Request Command Codes

Code (hex)	Mnemonic (see note 1)	Description	Object(s) in Request CT_IU	Object(s) in Accept CT_IU
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)

Table 12 – Name Server – Request Command Codes

Code (hex)	Mnemonic (see note 1)	Description	Object(s) in Request CT_IU	Object(s) in Accept CT_IU
011C	GFPN_ID	Get Fabric Port Name - Port Identifier	Port Identifier	Fabric Port Name
011D	GHA_ID	Get Hard Address - Port Identifier	Port Identifier	Hard Address
011E	GFD_ID	Get FC-4 Descriptors - Port Identifier	Port Identifier	List of FC-4 Descriptors
011F	GFF_ID	Get FC-4 Features - Port Identifier	Port Identifier	FC-4 Features
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
0132	GPN_NN	Get Port Names - Node Name	Node Name	List of Port Identifiers and Port Names
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
0172	GPN_FT	Get Port Names - FC-4 TYPE	none (see note 2)	List of Port Identifiers and Port Names
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

Table 12 – Name Server – Request Command Codes

Code (hex)	Mnemonic (see note 1)	Description	Object(s) in Request CT_IU	Object(s) in Accept CT_IU
01B2	GPN_IPP	Get Port Name - IP Address (Port)	IP Address (Port)	Port Name
01F1	GID_FF	Get Port Identifiers - FC-4 Features	FC-4 Features	List of Port Identifiers
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
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
021E	RFD_ID	Register FC-4 Descriptors - Port Identifier	Port Identifier, FC-4 TYPEs, FC-4 Descriptors	none
021E	RFF_ID	Register FC-4 Features - Port Identifier	Port Identifier, FC-4 Features	none
0235	RIP_NN	Register IP Address (Node)	Node Name, IP Address (Node)	none

Table 12 – Name Server – Request Command Codes

Code (hex)	Mnemonic (see note 1)	Description	Object(s) in Request CT_IU	Object(s) in Accept CT_IU
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 11. 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”.

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.

5.1.1.2 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 Server are summarized in table 12.

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 deregister 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.

5.1.1.3 Queries

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

The queries defined for the Name Server are summarized in table 12.

5.1.1.4 Deregistration

Only one global deregistration request is defined for the Name Server. The requests defined for the Name Server are all summarized in table 12.

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

5.1.2 Name Server objects – Formats

The format of the Name Server objects summarized in table 11 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 Accept CT_IU but no value has been registered for the requested Name Server object.

Table 13 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.

Table 13 – 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		
	FC-4 Descriptor(s)		

5.1.2.1 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 of 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’.

5.1.2.2 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’.

5.1.2.3 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’.

5.1.2.4 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: reserved

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’.

5.1.2.5 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 [24]). For example, the Name Server format of 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 of the 128 bit (IPv6) IP address 1080:0:0:0:8:800:200C:417A is hex ‘10 80 00 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’.

5.1.2.6 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’.

5.1.2.7 FC–4 TYPEs – Format

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

Table 14 – 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]

– 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 14).
- 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) (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’.

5.1.2.8 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.

5.1.2.9 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.

5.1.2.10 Port Type – Format

The format of the Port Type object, shall be as shown in table 15.

Table 15 – 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

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 CT_IU, and shall never be specified in the response to a GA_NXT or GPT_ID Request CT_IU, or in an RPT_ID Request CT_IU.

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

5.1.2.11 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’.

5.1.2.12 Hard Address – Format

The format of the Hard Address object, as used by the Name Server, shall be identical to the format of 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’.

5.1.2.13 FC-4 Descriptor – Format

The FC-4 Descriptor 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. More than one FC-4 Descriptor may be associated with a Port Identifier - the FC-4 Descriptor is indexed by the FC-4 TYPEs object.

NOTE – Specific FC-4s may further define the content of this object. As of this writing, FCP-2 (see reference [14]) was expected to contain such a definition.

If a FC-4 Descriptor is not registered then the FC-4 Descriptor defaults to a 0 byte length object.

5.1.2.14 FC–4 Features – Format

The format of the FC-4 Features object, as defined by the FC-4, shall be an array of 4-bit values, one for each TYPE code value, as shown in table 16.

Table 16 – FC–4 Features mapping

FC-4 TYPE Bit 76543	FC-4 TYPE Bit 2 1 0	FC-4 TYPE Bit 2 1 0	FC-4 TYPE Bit 2 1 0	FC-4 TYPE Bit 2 1 0	FC-4 TYPE Bit 2 1 0	FC-4 TYPE Bit 2 1 0	FC-4 TYPE Bit 2 1 0	FC-4 TYPE Bit 2 1 0
	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
00000	w00 [03:00]	w00 [07:04]	w00 [11:08]	w00 [15:12]	w00 [19:16]	w00 [23:20]	w00 [27:24]	w00 [31:28]
00001	w01 [03:00]	w01 [07:04]	w01 [11:08]	w01 [15:12]	w01 [19:16]	w01 [23:20]	w01 [27:24]	w01 [31:28]
00010	w02 [03:00]	w02 [07:04]	w02 [11:08]	w02 [15:12]	w02 [19:16]	w02 [23:20]	w02 [27:24]	w02 [31:28]
00011	w03 [03:00]	w03 [07:04]	w03 [11:08]	w03 [15:12]	w03 [19:16]	w03 [23:20]	w03 [27:24]	w03 [31:28]
00100	w04 [03:00]	w04 [07:04]	w04 [11:08]	w04 [15:12]	w04 [19:16]	w04 [23:20]	w04 [27:24]	w04 [31:28]
00101	w05 [03:00]	w05 [07:04]	w05 [11:08]	w05 [15:12]	w05 [19:16]	w05 [23:20]	w05 [27:24]	w05 [31:28]
00110	w06 [03:00]	w06 [07:04]	w06 [11:08]	w06 [15:12]	w06 [19:16]	w06 [23:20]	w06 [27:24]	w06 [31:28]
00111	w07 [03:00]	w07 [07:04]	w07 [11:08]	w07 [15:12]	w07 [19:16]	w07 [23:20]	w07 [27:24]	w07 [31:28]
01000	w08 [03:00]	w08 [07:04]	w08 [11:08]	w08 [15:12]	w08 [19:16]	w08 [23:20]	w08 [27:24]	w08 [31:28]
01001	w09 [03:00]	w09 [07:04]	w09 [11:08]	w09 [15:12]	w09 [19:16]	w09 [23:20]	w09 [27:24]	w09 [31:28]
01010	w10 [03:00]	w10 [07:04]	w10 [11:08]	w10 [15:12]	w10 [19:16]	w10 [23:20]	w10 [27:24]	w10 [31:28]
01011	w11 [03:00]	w11 [07:04]	w11 [11:08]	w11 [15:12]	w11 [19:16]	w11 [23:20]	w11 [27:24]	w11 [31:28]
01100	w12 [03:00]	w12 [07:04]	w12 [11:08]	w12 [15:12]	w12 [19:16]	w12 [23:20]	w12 [27:24]	w12 [31:28]
01101	w13 [03:00]	w13 [07:04]	w13 [11:08]	w13 [15:12]	w13 [19:16]	w13 [23:20]	w13 [27:24]	w13 [31:28]
01110	w14 [03:00]	w14 [07:04]	w14 [11:08]	w14 [15:12]	w14 [19:16]	w14 [23:20]	w14 [27:24]	w14 [31:28]
01111	w15 [03:00]	w15 [07:04]	w15 [11:08]	w15 [15:12]	w15 [19:16]	w15 [23:20]	w15 [27:24]	w15 [31:28]
10000	w16 [03:00]	w16 [07:04]	w16 [11:08]	w16 [15:12]	w16 [19:16]	w16 [23:20]	w16 [27:24]	w16 [31:28]
10001	w17 [03:00]	w17 [07:04]	w17 [11:08]	w17 [15:12]	w17 [19:16]	w17 [23:20]	w17 [27:24]	w17 [31:28]
10010	w18 [03:00]	w18 [07:04]	w18 [11:08]	w18 [15:12]	w18 [19:16]	w18 [23:20]	w18 [27:24]	w18 [31:28]
10011	w19 [03:00]	w19 [07:04]	w19 [11:08]	w19 [15:12]	w19 [19:16]	w19 [23:20]	w19 [27:24]	w19 [31:28]
10100	w20 [03:00]	w20 [07:04]	w20 [11:08]	w20 [15:12]	w20 [19:16]	w20 [23:20]	w20 [27:24]	w20 [31:28]
10101	w21 [03:00]	w21 [07:04]	w21 [11:08]	w21 [15:12]	w21 [19:16]	w21 [23:20]	w21 [27:24]	w21 [31:28]
10110	w22 [03:00]	w22 [07:04]	w22 [11:08]	w22 [15:12]	w22 [19:16]	w22 [23:20]	w22 [27:24]	w22 [31:28]
10111	w23 [03:00]	w23 [07:04]	w23 [11:08]	w23 [15:12]	w23 [19:16]	w23 [23:20]	w23 [27:24]	w23 [31:28]
11000	w24 [03:00]	w24 [07:04]	w24 [11:08]	w24 [15:12]	w24 [19:16]	w24 [23:20]	w24 [27:24]	w24 [31:28]
11001	w25 [03:00]	w25 [07:04]	w25 [11:08]	w25 [15:12]	w25 [19:16]	w25 [23:20]	w25 [27:24]	w25 [31:28]
11010	w26 [03:00]	w26 [07:04]	w26 [11:08]	w26 [15:12]	w26 [19:16]	w26 [23:20]	w26 [27:24]	w26 [31:28]
11011	w27 [03:00]	w27 [07:04]	w27 [11:08]	w27 [15:12]	w27 [19:16]	w27 [23:20]	w27 [27:24]	w27 [31:28]
11100	w28 [03:00]	w28 [07:04]	w28 [11:08]	w28 [15:12]	w28 [19:16]	w28 [23:20]	w28 [27:24]	w28 [31:28]
11101	w29 [03:00]	w29 [07:04]	w29 [11:08]	w29 [15:12]	w29 [19:16]	w29 [23:20]	w29 [27:24]	w29 [31:28]
11110	w30 [03:00]	w30 [07:04]	w30 [11:08]	w30 [15:12]	w30 [19:16]	w30 [23:20]	w30 [27:24]	w30 [31:28]
11111	w31 [03:00]	w31 [07:04]	w31 [11:08]	w31 [15:12]	w31 [19:16]	w31 [23:20]	w31 [27:24]	w31 [31:28]

- the 5 most significant bits of the TYPE field shall be used to identify the word for the FC–4 Features object;
  - Word 0 contains information related to TYPE code hex ‘00’ through hex ‘07’;
  - Word 1 contains information related to TYPE code hex ‘08’ through hex ‘0F’;
  - Word 2 contains information related to TYPE code hex ‘10’ through hex ‘17’;
  - and so forth to Word 31 that contains information related to TYPE code hex ‘F8’ through hex ‘FF’.
- the 3 least significant bits of the TYPE field shall be used to identify the position within the word for the 4-bit FC-4 Features value (see table 16).
- The setting and meaning of the bits within the 4-bit FC-4 Features for a specific TYPE value are not defined by this Standard.

NOTE – It is intended that the FC-4 corresponding to the TYPE value define the meaning of the 4-bit field.

The null FC–4 Features object value is all 128-bytes set to hex ‘00’.

5.1.3 Reason code explanations

A Reject CT_IU (see 4.4.3) shall notify the requestor that the request has been unsuccessfully completed. The first error condition encountered shall be the error reported by the Reject CT_IU.

If a valid Name 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 Name Server request shall not be rejected with a Reason Code of “Command not supported”

If a Name Server request is rejected with a reason code of ‘Unable to perform command request’, then one of the reason code explanations, shown in table 17, are returned.

Table 17 – Reject CT_IU 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

Table 17 – Reject CT_IU Reason code explanations

Encoded value (hex)	Description
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
0E	FC-4 Descriptor not registered
0F	FC-4 Features 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 Reject CT_IU 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 Reject CT_IU 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 Reject CT_IU 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 Reject CT_IU 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 Reject CT_IU 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 Reject CT_IU 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.

5.1.4 Commands

The commands defined for the Name Server are summarized in table 12.

5.1.4.1 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 CT_IU, 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 CT_IU 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 Accept CT_IU, 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 5.1.1.2), because N_Ports and NL_Ports are not required to register with the Name Server.

The format of the GA_NXT Request CT_IU is shown in table 18. The requestor supplies a Port Identifier using the format in 5.1.2.1, and the Name Server returns all Name Server objects for the next higher valued Port Identifier.

Table 18 – GA_NXT Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

The format of the Accept CT_IU to a GA_NXT request is shown in table 19. The format of the various objects returned is defined in 5.1.2.

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.

NOTE – FC-4 Descriptor(s) are not returned in the Accept CT_IU of the GA_NXT request. .

Table 19 – Accept CT_IU to GA_NXT Request

Item	Size (Bytes)
CT_IU preamble	16
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

5.1.4.2 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 CT_IU is shown in table 20. The requestor supplies a Domain_ID Scope that 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 Server 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 20 – GI_A Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Domain_ID Scope	1
Reserved	2

The formats of the reply Accept CT_IU to a GI_A request are shown in table 21 and table 22.

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 Accept CT_IU; the bit is set to one if the identifier following the Control field is the last identifier returned by the Accept CT_IU.

– 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.

Table 21 – Accept CT_IU to GI_A Request, Domain_ID Scope is zero

Item	Size (Bytes)
CT_IU preamble	16
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 22 – Accept CT_IU to GI_A Request, Domain_ID Scope is non-zero

Item	Size (Bytes)
CT_IU preamble	16
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

5.1.4.3 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 CT_IU is shown in table 23. The requestor supplies the Port Identifier for which the Port Name is sought.

Table 23 – GPN_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

The format of the Accept CT_IU to a GPN_ID request is shown in table 24.

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

Table 24 – Accept CT_IU to GPN_ID Request

Item	Size (Bytes)
CT_IU preamble	16
Port Name	8

5.1.4.4 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 CT_IU is shown in table 25. The requestor supplies the Port Identifier for which the Port Name is sought.

Table 25 – GNN_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

The format of the Accept CT_IU to a GNN_ID request is shown in table 26.

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

Table 26 – Accept CT_IU to GNN_ID Request

Item	Size (Bytes)
CT_IU preamble	16
Node Name	8

5.1.4.5 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 CT_IU is shown in table 27. The requestor supplies the Port Identifier for which the Class of Service object is sought.

Table 27 – GCS_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

The format of the Accept CT_IU to a GCS_ID request is shown in table 28.

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

Table 28 – Accept CT_IU to GCS_ID Request

Item	Size (Bytes)
CT_IU preamble	16
Class of Service	4

5.1.4.6 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 CT_IU is shown in table 29. The requestor supplies the Port Identifier for which the FC–4 TYPEs object is sought.

Table 29 – GFT_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

The format of the Accept CT_IU to a GFT_ID request is shown in table 30.

The FC–4 TYPEs field (see 5.1.2.7) returns the registered value for the FC–4 TYPEs.

Table 30 – Accept CT_IU to GFT_ID Request

Item	Size (Bytes)
CT_IU preamble	16
FC–4 TYPEs	32

5.1.4.7 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 CT_IU is shown in table 31. The requestor supplies the Port Identifier for which the Symbolic Port Name is sought.

Table 31 – GSPN_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

The format of the Accept CT_IU to a GSPN_ID request is shown in table 32.

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 Identifier.

Table 32 – Accept CT_IU to GSPN_ID Request

Item	Size (Bytes)
CT_IU preamble	16
String length (m)	1
Symbolic Port Name (Octet string)	m
Reserved	255-m

5.1.4.8 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 CT_IU is shown in table 33. The requestor supplies the Port Identifier for which the Port Type is sought.

Table 33 – GPT_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

The format of the Accept CT_IU to a GPT_ID request is shown in table 34.

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

Table 34 – Accept CT_IU to GPT_ID Request

Item	Size (Bytes)
CT_IU preamble	16
Port Type	1
Reserved	3

5.1.4.9 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 CT_IU is shown in table 35. The requestor supplies the Port Identifier for which the IP Address (Port) is sought.

Table 35 – GIPP_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

The format of the Accept CT_IU to a GIPP_ID request is shown in table 36.

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

Table 36 – Accept CT_IU to GIPP_ID Request

Item	Size (Bytes)
CT_IU preamble	16
IP Address (Port)	16

5.1.4.10 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 CT_IU is shown in table 37. The requestor supplies the Port Identifier for which the Fabric Port Name is sought.

Table 37 – GFPN_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

The format of the Accept CT_IU to a GFPN_ID request is shown in table 38.

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

Table 38 – Accept CT_IU to GFPN_ID Request

Item	Size (Bytes)
CT_IU preamble	16
Fabric Port Name	8

5.1.4.11 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 CT_IU is shown in table 39. The requestor supplies the Port Identifier for which the Hard Address is sought.

Table 39 – GHA_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

The format of the Accept CT_IU to a GHA_ID request is shown in table 40.

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

Table 40 – Accept CT_IU to GHA_ID Request

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Hard Address	3

5.1.4.12 Query – Get FC-4 Descriptors (GFD_ID)

The Name Server shall, when it receives a GSPN_ID request, return the registered FC-4 Descriptor(s) for the specified Port Identifier and FC-4 TYPEs. The format of the GFD_ID Request CT_IU is shown in table 41. The requestor supplies the Port Identifier and FC-4 TYPEs for which the FC-4 Descriptor(s) are sought.

Table 41 – GFD_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3
FC–4 TYPEs	32

The format of the Accept CT_IU to a GFD_ID request is shown in table 42. The Accept CT_IU shall contain the number of FC-4 Descriptors requested (as indicated by the FC-4 TYPEs field). The FC-4 Descriptors shall be returned in ascending TYPE order.

The Descriptor length field shall contain a single byte unsigned value indicating the size of the variable length FC-4 Descriptor.

The FC-4 Descriptor field returns the registered FC-4 Descriptor for the specified Port Identifier and FC-4 TYPE.

Table 42 – Accept CT_IU to GFD_ID Request

Item	Size (Bytes)
CT_IU preamble	16
Descriptor length (m) #1	1
FC-4 Descriptor #1	m
Reserved	255-m
. . .	
Descriptor length (m) #n	1
FC-4 Descriptor #n	m
Reserved	255-m

5.1.4.13 Query – Get FC–4 Features (GFF_ID)

The Name Server shall, when it receives a GFF_ID request, return the registered FC–4 Features object for the specified Port Identifier. The format of the GFF_ID Request CT_IU is shown in table 43. The requestor supplies the Port Identifier for which the FC–4 Features object is sought.

Table 43 – GFF_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

The format of the Accept CT_IU to a GFF_ID request is shown in table 44.

The FC–4 Features field (see 5.1.2.14) returns the registered value for the FC–4 Features.

Table 44 – Accept CT_IU to GFF_ID Request

Item	Size (Bytes)
CT_IU preamble	16
FC–4 Features	128

5.1.4.14 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 CT_IU is shown in table 45. The requestor supplies the Port Name for which the Port Identifier is sought.

Table 45 – GID_PN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Port Name	8

The format of the Accept CT_IU to a GID_PN request is shown in table 46.

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

Table 46 – Accept CT_IU to GID_PN Request

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

5.1.4.15 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 CT_IU is shown in table 47. The requestor supplies the Port Name for which the IP Address (Port) is sought.

Table 47 – GIPP_PN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Port Name	8

The format of the Accept CT_IU to a GIPP_PN request is shown in table 48.

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

Table 48 – Accept CT_IU to GIPP_PN Request

Item	Size (Bytes)
CT_IU preamble	16
IP Address (Port)	16

5.1.4.16 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 CT_IU is shown in table 49. The requestor supplies the Node Name for which associated Port Identifiers are sought.

Table 49 – GID_NN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Node Name	8

The format of the Accept CT_IU to a GID_NN request is shown in table 50.

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 Accept CT_IU; the bit is set to one if the Port Identifier following the Control field is the last Port Identifier returned by the Accept CT_IU.

– Bits 6-0 are reserved.

Table 50 – Accept CT_IU to GID_NN Request

Item	Size (Bytes)
CT_IU preamble	16
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.

5.1.4.17 Query – Get Port Names (GPN_NN)

The Name Server shall, when it receives a GPN_NN request, return a list of Port Identifiers and Port Names registered for the specified Node Name. The format of the GPN_NN Request CT_IU is shown in table 51. The requestor supplies the Node Name for which associated Port Identifiers and Port Names are sought.

Table 51 – GPN_NN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Node Name	8

The format of the Accept CT_IU to a GPN_NN request is shown in table 52.

One or more Port Identifiers and Port Names, registered for the specified Node Name, are returned. Each returned Port Identifier and Port 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 Port Name following the Control field is not the last Port Identifier and Port Name to be returned by the Accept CT_IU; the bit is set to one if the Port Identifier and Port Name following the Control field is the last Port Identifier and Port Name returned by the Accept CT_IU.

– Bits 6-0 are reserved.

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

Table 52 – Accept CT_IU to GPN_NN Request

Item	Size (Bytes)
CT_IU preamble	16
Control (0 r r r r r r r)	1
Port Identifier #1	3
Reserved	4
Port Name #1	8
...	
Control (1 r r r r r r r)	1
Port Identifier #n	3
Reserved	4
Port Name #n	8

5.1.4.18 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 CT_IU is shown in table 53. The requestor supplies the Node Name for which the IP Address (Node) is sought.

Table 53 – GIP_NN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Node Name	8

The format of the Accept CT_IU to a GIP_NN request is shown in table 54.

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

Table 54 – Accept CT_IU to GIP_NN Request

Item	Size (Bytes)
CT_IU preamble	16
IP Address (Node)	16

5.1.4.19 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 CT_IU is shown in table 55. The requestor supplies the Node Name for which the Initial Process Associator is sought.

Table 55 – GIPA_NN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Node Name	8

The format of the Accept CT_IU to a GIPA_NN request is shown in table 56.

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

Table 56 – Accept CT_IU to GIPA_NN Request

Item	Size (Bytes)
CT_IU preamble	16
Initial Process Associator	8

5.1.4.20 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 CT_IU is shown in table 57. The requestor supplies the Node Name for which the Symbolic Node Name is sought.

Table 57 – GSNN_NN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Node Name	8

The format of Accept CT_IU to a GSNN_NN request is shown in table 58.

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 58 – Accept CT_IU to GSNN_NN Request

Item	Size (Bytes)
CT_IU preamble	16
String length (n)	1
Symbolic Node Name (Octet string)	n
Reserved	255-n

5.1.4.21 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 CT_IU is shown in table 59. The requestor supplies the IP Address (Node) for which the Port Name is sought.

Table 59 – GNN_IP Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
IP Address (Node)	16

The format of the Accept CT_IU to a GNN_IP request is shown in table 60.

The Node Name field returns the registered Node Name.

Table 60 – Accept CT_IU to GNN_IP Request

Item	Size (Bytes)
CT_IU preamble	16
Node Name	8

5.1.4.22 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 CT_IU is shown in table 61. The requestor supplies the IP Address (Node) for which the Initial Process Associator is sought.

Table 61 – GIPA_IP Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
IP Address (Node)	16

The format of the Accept CT_IU to a GIPA_IP request is shown in table 62.

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

Table 62 – Accept CT_IU to GIPA_IP Request

Item	Size (Bytes)
CT_IU preamble	16
Initial Process Associator	8

5.1.4.23 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 CT_IU is shown in table 63. 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 63 – GID_FT Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Domain_ID Scope	1
Area_ID Scope	1
FC–4 TYPE Code	1

The format of the Accept CT_IU to a GID_FT request is shown in table 64.

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 Accept CT_IU; the bit is set to one if the Port Identifier following the Control field is the last Port Identifier returned by the Accept CT_IU.

– 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 64 – Accept CT_IU to GID_FT Request

Item	Size (Bytes)
CT_IU preamble	16
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

5.1.4.24 Query – Get Port Names (GPN_FT)

The Name Server shall, when it receives a GPN_FT request, return a list of Port Identifiers and Port Names having registered support for the specified FC–4 TYPE. The format of the GPN_FT Request CT_IU is shown in table 65. The requestor supplies the FC–4 TYPE code (as defined in FC–PH) for which supporting Port Identifiers and Port 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 Port 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 Port 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 Port 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.

Table 65 – GPN_FT Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Domain_ID Scope	1
Area_ID Scope	1
FC–4 TYPE	1

The format of the Accept CT_IU to a GPN_FT request is shown in table 66.

One or more Port Identifiers and Port Names, having registered support for the specified FC–4 TYPE, are returned. Each returned Port Identifier and Port 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 Port Name following the Control field is not the last Port Identifier and Port Name to be returned by the Accept CT_IU; the bit is set to one if the Port Identifier and Port Name following the Control field is the last Port Identifier and Port Name returned by the Accept CT_IU.
- Bits 6-0 are reserved.

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

Table 66 – Accept CT_IU to GPN_FT Request

Item	Size (Bytes)
CT_IU preamble	16
Control (0 r r r r r r r)	1
Port Identifier #1	3
Reserved	4

Table 66 – Accept CT_IU to GPN_FT Request

Port Name #1	8
...	
Control (1 r r r r r r r)	1
Port Identifier #n	3
Reserved	4
Port Name #n	8

5.1.4.25 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 CT_IU is shown in table 67. The requestor 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.

Table 67 – GNN_FT Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Domain_ID Scope	1
Area_ID Scope	1
FC–4 TYPE	1

The format of the Accept CT_IU to a GNN_FT request is shown in table 68.

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

– 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.

Table 68 – Accept CT_IU to GNN_FT Request

Item	Size (Bytes)
CT_IU preamble	16
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

5.1.4.26 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 CT_IU is shown in table 69. 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 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 69 – GID_PT Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Port Type	1
Domain_ID Scope	1
Area_ID Scope	1
Reserved	1

The format of the Accept CT_IU to a GID_PT request is shown in table 70.

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 Accept CT_IU; the bit is set to one if the Port Identifier following the Control field is the last Port Identifier returned by the Accept CT_IU.

– 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 70 – Accept CT_IU to GID_PT Request

Item	Size (Bytes)
CT_IU preamble	16
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

5.1.4.27 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 CT_IU is shown in table 71. The requestor supplies the IP Address (Port) for which Port Identifiers are sought.

Table 71 – GID_IPP Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
IP Address (Port)	16

The format of the Accept CT_IU to a GID_IPP request is shown in table 72.

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 Accept CT_IU; the bit is set to one if the Port Identifier following the Control field is the last Port Identifier returned by the Accept CT_IU.

– 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 72 – Accept CT_IU to GID_IPP Request

Item	Size (Bytes)
CT_IU preamble	16
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

5.1.4.28 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 CT_IU is shown in table 73. The requestor supplies the IP Address (Port) for which the Port Name is sought.

Table 73 – GPN_IPP Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
IP Address (Port)	16

The format of the Accept CT_IU to a GPN_IPP request is shown in table 74.

The Port Name field returns the registered Port Name.

Table 74 – Accept CT_IU to GPN_IPP Request

Item	Size (Bytes)
CT_IU preamble	16
Port Name	8

5.1.4.29 Query – Get Port Identifiers (GID_FF)

The Name Server shall, when it receives a GID_FF request, return all Port Identifiers having registered support for the specified FC–4 Features. The format of the GID_FF Request CT_IU is shown in table 75. The requestor supplies the FC–4 Features for which supporting Port Identifiers are sought.

{DOES this command also need to specify a TYPE code to qualify the features? and if so should it just contain one 4-bit features value for the TYPE? - ed}

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 the specified FC–4 Features 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 the specified FC–4 Features 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 the specified FC–4 Features code.

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

Table 75 – GID_FF Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Domain_ID Scope	1
Area_ID Scope	1
Reserved	1
FC–4 Features	128

The format of the Accept CT_IU to a GID_FF request is shown in table 76.

One or more Port Identifiers, having registered the specified FC–4 Features, 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 Accept CT_IU; the bit is set to one if the Port Identifier following the Control field is the last Port Identifier returned by the Accept CT_IU.

– 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 76 – Accept CT_IU to GID_FF Request

Item	Size (Bytes)
CT_IU preamble	16
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

5.1.4.30 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 Request CT_IU, 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 5.1.2.2).

The format of the RPN_ID Request CT_IU is shown in table 77.

Table 77 – RPN_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3
Port Name	8

The format of the RPN_ID Accept CT_IU is shown in table 78.

Table 78 – RPN_ID Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

5.1.4.31 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 Request CT_IU, 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 5.1.2.3).

The format of the RNN_ID Request CT_IU is shown in table 79.

Table 79 – RNN_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3
Node Name	8

The format of the RNN_ID Accept CT_IU is shown in table 80.

Table 80 – RNN_ID Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

5.1.4.32 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 Request CT_IU, 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 5.1.2.4).

The format of the RCS_ID Request CT_IU is shown in table 81.

Table 81 – RCS_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3
Class of Service	4

The format of the RCS_ID Accept CT_IU is shown in table 82.

Table 82 – RCS_ID Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

5.1.4.33 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 Request CT_IU, 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 5.1.2.7).

The format of the RFT_ID Request CT_IU is shown in table 83.

Table 83 – RFT_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3
FC–4 TYPEs	32

The format of the RFT_ID Accept CT_IU is shown in table 84.

Table 84 – RFT_ID Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

5.1.4.34 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 Request CT_IU.

The Name Server shall not attempt validation of the Symbolic Port Name object. This means that any variable length Symbolic Port Name value up to 255 bytes long, including a 0 length, shall be accepted.

Deregistration may be accomplished by registering a null Symbolic Port Name object (see 5.1.2.8).

The format of the RSPN_ID Request CT_IU is shown in table 85. The String length field shall contain a single byte unsigned value indicating the size of the variable length Symbolic Port Name.

Table 85 – RSPN_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3
String length (n)	1
Symbolic Port Name (Octet string)	n

The format of the RSPN_ID Accept CT_IU is shown in table 86.

Table 86 – RSPN_ID Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

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 Request CT_IU, then the Reject CT_IU reason code shall be ‘Invalid IU Size’, with a reason explanation code of ‘No additional explanation’.

5.1.4.35 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 Request CT_IU, 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 5.1.2.10).

The format of the RPT_ID Request CT_IU is shown in table 87.

Table 87 – RPT_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3
Port Type	1
Reserved	3

The format of the RPT_ID Accept CT_IU is shown in table 88.

Table 88 – RPT_ID Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

5.1.4.36 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 Request CT_IU 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 5.1.2.5).

The format of the RIPP_ID Request CT_IU is shown in table 89.

Table 89 – RIPP_ID Request CT_IU)

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

The format of the RIPP_ID Accept CT_IU is shown in table 90.

Table 90 – RIPP_ID Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

5.1.4.37 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 (FLO-GI) 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 5.1.2.11).

The format of the RFPN_ID Request CT_IU is shown in table 91.

Table 91 – RFPN_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3
Fabric Port Name	8

The format of the RFPN_ID Accept CT_IU is shown in table 92.

Table 92 – RFPN_ID Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

5.1.4.38 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 Request CT_IU 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 5.1.2.12).

The format of the RHA_ID Request CT_IU is shown in table 93.

Table 93 – RHA_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3
Reserved	1
Hard Address	3

The format of the RHA_ID Accept CT_IU is shown in table 94.

Table 94 – RHA_ID Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

5.1.4.39 Register FC-4 Descriptor (RFD_ID)

The RFD_ID Name Server request shall be used to associate one or more FC-4 Descriptors with a given Port Identifier.

The Name Server may reject registration of the FC-4 Descriptor unless the registration is attempted by the Port with its address identifier equal to the Port Identifier in the Request CT_IU.

The Name Server shall not attempt validation of an FC-4 Descriptor object. This means that any variable length FC-4 Descriptor value up to 255 bytes long, including a 0 length, shall be accepted.

Deregistration may be accomplished by registering a null FC-4 Descriptor object (see 5.1.2.13). Registration of an FC-4 Descriptor for a particular FC-4 TYPE shall not be affected by the subsequent registration of an FC-4 Descriptor for a different FC-4 TYPE.

NOTE – For example, an FC-4 Descriptor registration for FC-4 TYPE ‘A’, followed by an FC-4 Descriptor registration for FC-4 TYPE ‘B’, results in FC-4 Descriptors registered for each FC-4 TYPEs ‘A’ and ‘B’.

The format of the RFD_ID Request CT_IU is shown in table 95. The Request CT_IU shall contain the number of FC-4 Descriptors requested (as indicated by the FC-4 TYPEs field). The FC-4 Descriptors shall be supplied in ascending TYPE order. The Descriptor length field shall contain a single byte unsigned value indicating the size of the variable length FC-4 Descriptor.

Table 95 – RFD_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3
FC–4 TYPEs	32
Descriptor length (m) #1	1
FC-4 Descriptor #1	m
Reserved	255-m
. . .	
Descriptor length (m) #n	1
FC-4 Descriptor #n	m
Reserved	255-m

The format of the RFD_ID Accept CT_IU is shown in table 96.

Table 96 – RFD_ID Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

If the RFD_ID Name Server request is rejected by the Name Server because the Descriptor length field value does not match the size of the FC-4 Descriptor in the Request CT_IU, then the Reject CT_IU reason code shall be ‘Invalid IU Size’, with a reason explanation code of ‘No additional explanation’.

5.1.4.40 Register FC–4 Features (RFF_ID)

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

The Name Server shall accept RFF_ID requests received from the Port with its address identifier equal to the Port Identifier in the Request CT_IU, 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 Features object. This means that any 128-byte FC–4 Features object value shall be accepted.

Deregistration may be accomplished by registering a null FC–4 Features object (see 5.1.2.14).

The format of the RFF_ID Request CT_IU is shown in table 97.

Table 97 – RFF_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3
FC–4 Features	128

The format of the RFF_ID Accept CT_IU is shown in table 98.

Table 98 – RFF_ID Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

5.1.4.41 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 5.1.4.31). 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 Request CT_IU 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 5.1.2.5).

The format of the RIP_NN Request CT_IU is shown in table 99.

Table 99 – RIP_NN Request CT_IU)

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

The format of the RIP_NN Accept CT_IU is shown in table 100.

Table 100 – RIP_NN Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

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 Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation of ‘Node Name not registered’.

5.1.4.42 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 5.1.4.31). 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 Request CT_IU 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 5.1.2.6).

The format of the RIPA_NN Request CT_IU is shown in table 101.

Table 101 – RIPA_NN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Node Name	8
Initial Process Associator	8

The format of the RIPA_NN Accept CT_IU is shown in table 102.

Table 102 – RIPA_NN Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

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 Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation of ‘Node Name not registered’.

5.1.4.43 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 5.1.4.31). 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 Request CT_IU payload.

The Name Server shall not attempt validation of the Symbolic Node Name object. This means that any variable length Symbolic Node Name value up to 255 bytes long, including a 0 length, shall be accepted.

Deregistration may be accomplished by registering a null Symbolic Node Name object (see 5.1.2.9).

The format of the RSNN_NN Request CT_IU is shown in table 103. The String length field shall contain a single byte unsigned value indicating the size of the variable length Symbolic Node Name.

Table 103 – RSNN_NN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Node Name	8
String length (n)	1
Symbolic Node Name (Octet string)	n

The format of the RSNN_NN Accept CT_IU is shown in table 104.

Table 104 – RSNN_NN Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

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 Reject CT_IU 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 Request CT_IU, then the Reject CT_IU reason code shall be ‘Invalid IU Size’, with a reason code explanation of ‘No additional explanation’.

5.1.4.44 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 Request CT_IU 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 DA_ID Request CT_IU is shown in table 105. The Port Identifier format shall be as defined in 5.1.2.1.

Table 105 – DA_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

The format of the DA_ID Accept CT_IU is shown in table 106.

Table 106 – DA_ID Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

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 Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation of ‘Port Identifier not registered’.

5.2 IP Address Server

{NOTE - The following is a swag made by the editor for this rev of the document and in NO WAY has been approved by the working group! - ed}

The IP Address Server provides a way for N_Ports and NL_Ports to register and discover Port Identifiers associated with Internet Protocol (IP) addresses. Registrations may be performed by a third party. However, the IP Address Server may refuse such third party registration for unspecified reasons. Once registered, the Port Identifiers are made available to requestors.

5.2.1 Protocol

IP Address Server registration, deregistration and queries are managed through protocols containing a set of Request CT_IUs and Response CT_IUs.

Synchronous transactions shall be used, as defined in 4.5.1. For an IP Address Server request, the request payload shall be transported from the requestor to the IP Address Server using a Request CT_IU. The corresponding response is transported from the IP Address Server to the requestor, in the Exchange established by the requestor, using a Response CT_IU.

5.2.1.1 CT_IU preamble values

The following values shall be set in the CT_IU preamble for IP Address Server request and their responses; fields not specified here shall be set as defined in 4.3.1:

– Revision: hex ‘01’;

– GS_Subtype: as indicated in table 9;

– Command Code: see table 107 for Request command codes.

Table 107 – IP Address Server – Request Command Codes

Code (hex)	Mnemonic	Description	Object(s) in Request CT_IU	Object(s) in Accept CT_IU
0112	GIP_ID	Get IP Addresses - Port Identifier	Port Identifier	List of IP Addresses
0121	GID_IP	Get Port Identifier	IP Address	Port Identifier
0221	RID_IP	Register Port Identifier	IP Address, Port Identifier	none

5.2.1.2 Registration

Registrations are limited to a single IP Address object at a time. A registrant submits a tuple, consisting of a primary key object along with an object to be associated with the key object. The IP Address is the primary key object.

The registration requests defined for the IP Address Server are summarized in table 107.

The IP Address Server may reject registrations:

– due to IP Address Server resource limitations;

– of IP Addresses associated with unassigned or unused Port Identifiers.

The IP Address Server shall reject registrations associated with:

– well-known address identifiers.

– known Alias addresses such as:

  – Hunt group identifiers;

  – Multicast group identifiers.

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

The IP Address Server may reject all registrations associated with Fibre Channel addresses not used or not usable as Port Identifiers in the fabric.

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

5.2.1.3 Queries

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

The queries defined for the IP Address Server are summarized in table 107.

5.2.2 IP Address Server objects – Formats

The format of the IP Address Server objects summarized in table 11 are described below. A null value is defined for each IP Address Server object. This value is used when the IP Address Server needs to return an Accept CT_IU but no value has been registered for the requested IP Address Server object.

Table 108 shows the relationship between the different objects within the IP Address 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 IP Address Server record is the IP Address. All objects are ultimately related back to this object.

Table 108 – IP Address Server Database Organization

Primary Key	Indexed Field
IP Address	Port Identifier

5.2.2.1 IP Address – Format

Both 32 bit (IPv4) and 128 bit (IPv6) Internet Protocol (IP) address formats may be supported by the IP Address Server.

The format of the 32 bit (IPv4) IP address, as used by the IP Address 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 [24]). For example, the IP Address Server format of 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 IP Address Server, shall use big endian bit and byte order, within a word. This is the format specified in RFC 2373. For example, the IP Address Server format of the 128 bit (IPv6) IP address 1080:0:0:0:8:800:200C:417A is hex ‘10 80 00 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 Server object type is hex ‘00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00’.

5.2.2.2 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 of the Port Identifier object, as used by the IP Address 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 IP Address Server.

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

5.2.3 Reason code explanations

A Reject CT_IU (see 4.4.3) shall notify the requestor that the request has been unsuccessfully completed. The first error condition encountered shall be the error reported by the Reject CT_IU.

If a valid IP Address 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 IP Address Server request shall not be rejected with a Reason Code of “Command not supported”.

If an IP Address Server request is rejected with a reason code of ‘Unable to perform command request’, then one of the reason code explanations, shown in table 109, are returned.

Table 109 – Reject CT_IU Reason code explanations

Encoded value (hex)	Description
00	No additional explanation
01	IP Address not registered
02	Port Identifier not registered
10	Access Denied
11	Unacceptable Port Identifier
12	Data base empty
Others	Reserved

5.2.4 Commands

The commands defined for the IP Address Server are summarized in table 107.

5.2.4.1 Query – Get IP Addresses - Port Identifier (GIP_ID)

The IP Address Server shall, when it receives a GIP_ID request, return a list of IP Address objects for the specified Port Identifier. The format of the GIP_ID Request CT_IU is shown in table 110. The requestor supplies the Port Identifier for which the IP Address objects are sought.

Table 110 – GIP_ID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

The format of the Accept CT_IU to a GIP_ID request is shown in table 111.

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

- Bit 7 is set to zero if the IP Address following the Control field is not the last IP Address to be returned by the Accept CT_IU; the bit is set to one if the IP Address following the Control field is the last IP Address returned by the Accept CT_IU.
- Bits 6-0 are reserved.

Table 111 – Accept CT_IU to GIP_ID Request

Item	Size (Bytes)
CT_IU preamble	16
Control (0 r r r r r r r)	1
Reserved	3
IP Address #1	16
...	
Control (1 r r r r r r r)	1
Reserved	3
IP Address #n	16

5.2.4.2 Query – Get Port Identifier (GID_IP)

The IP Address Server shall, when it receives a GID_IP request, return the registered Port Identifier for the specified IP Address. The format of the GID_IP Request CT_IU is shown in table 110. The requestor supplies the IP Address for which the Port Identifier object is sought.

Table 112 – GID_IP Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
IP Address	16

The format of the Accept CT_IU to a GID_IP request is shown in table 111.

The Port Identifier field returns the registered Port Identifier value for the specified IP Address.

Table 113 – Accept CT_IU to GID_IP Request

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Port Identifier	3

5.2.4.3 Register Port Identifier (RID_IP)

The RID_IP IP Address Server request shall be used to associate a Port Identifier with a given IP Address.

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

Deregistration may be accomplished by registering a null Port Identifier (see 5.1.2.12).

The format of the RID_IP Request CT_IU is shown in table 114.

Table 114 – RID_IP Request CT_IU

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

The format of the RID_IP Accept CT_IU is shown in table 115.

Table 115 – RID_IP Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6 Management service

The Management Service provides a single management access point within the Fibre Channel fabric. Management Service covers the following areas:

– Fabric Configuration Server - provides for the configuration management of the fabric;

– Unzoned Name Server - provides access to Name Server (see 5.1) information that is not subject to Zone constraints;

– Fabric Zone Server - provides access to Zone information.

This document defines a standard model for requests and responses to access Management Service information. This standard does not define the structure of this information.

Table 116 defines the GS_Subtype codes for the Management Service.

Table 116 – Management Service subtype values

Values in hex	Description
01	Fabric Configuration Server
02	Unzoned Name Server
03	Fabric Zone Server
04	{Lock Server TBD}
E0-FF	Vendor Specific Servers
other values	Reserved

The consumer of a Management Service is normally a management application. The Management Service provides for both monitoring and control of the system by the management application. Because Directory Service information is not normally forwarded to an application level (see clause 5), the Management Service provides access to that information via its own services, for use by the management application. Note that this system view by a management application is not constrained by the operational environment (Zone) of its associated Node; therefore, some form of authentication and/or access control may be desirable.

6.1 Fabric Configuration Server

The Fabric Configuration Server provides a way for a management application to discover Fibre Channel fabric topology and attributes.

Requests for the Fabric Configuration Server are carried over the Common Transport (see clause 4).

The Fabric Configuration Server is intended to be distributed among Fabric Elements, making the Fabric Configuration Server immediately available to an N_Port once it has successfully completed Fabric Login. However, the Fabric Configuration Server is not restricted or required to be part of a Fabric, and may be located in any N_Port or NL_Port.

6.1.1 Protocol

Fabric Configuration Server registration, deregistration and queries are managed through protocols containing a set of Request CT_IUs and Response CT_IUs supported by the Fabric Configuration Server.

Synchronous transactions shall be used, as defined in 4.5.1. For a Fabric Configuration Server request, the payload shall be transported from the requestor to the Fabric Configuration Server using a Request CT_IU. The corresponding Fabric Configuration Server response is transported from the Fabric Configuration Server to the requestor, in the Exchange established by the requestor, using a Response CT_IU.

6.1.1.1 CT_IU preamble values

The following values shall be set in the CT_IU preamble for Fabric Configuration Server request and their responses; fields not specified here shall be set as defined in 4.3.1:

– Revision: hex ‘01’;

– GS_Subtype: as indicated in table 116;

– Command Code: see table 117 for Request command codes.

Table 117 – Fabric Configuration Server – Request Command Codes

Code (hex)	Mnem.	Description	Attribute(s) in Request CT_IU	Attribute(s) in Accept CT_IU
0100	GTIN	Get Topology Information	none (see note 1)	none (see note 2)
0101	GIEL	Get Interconnect Element List	none	List of Interconnect Element Names and Types
0111	GIET	Get Interconnect Element Type	Interconnect Element Name	Interconnect Element Type
0112	GDID	Get Domain Identifier	Interconnect Element Name	Domain Identifier
0113	GMID	Get Management Identifier	Interconnect Element Name	Management Identifier
0114	GFN	Get Fabric Name	Interconnect Element Name	Fabric Name
0115	GIELN	Get Interconnect Element Logical Name	Interconnect Element Name	Interconnect Element Logical Name
0116	GMAL	Get Interconnect Element Management Address List	Interconnect Element Name	Interconnect Element Management Address List
0117	GIEIL	Get Interconnect Element Information List	Interconnect Element Name	Interconnect Element Information List

Table 117 – Fabric Configuration Server – Request Command Codes

Code (hex)	Mnem.	Description	Attribute(s) in Request CT_IU	Attribute(s) in Accept CT_IU
0118	GPL	Get Port List	Interconnect Element Name	List of Port Names and Port Types
0121	GPT	Get Port Type	Port Name	Port Type
0122	GPPN	Get Physical Port Number	Port Name	Physical Port Number
0124	GAPNL	Get Attached Port Name List	Port Name	Attached Port Name List
0126	GPS	Get Port State	Port Name	Port State
0128	GATIN	Get Attached Topology Information	Port Name	Attached Topology Information
0191	GPLNL	Get Platform Node Name List	Platform Name	Platform Node Name List
0192	GPLT	Get Platform Type	Platform Name	Platform Type
0193	GPLML	Get Platform Management Address List	Platform Name	Platform Management Address List
01A1	GNPL	Get Platform Name - Node Name	Platform Node Name	Platform Name
01B1	GNID	Get Node Identification Data - Node Name	Platform Node Name	none (see note 3)
0215	RIELN	Register Interconnect Element Logical Name	Interconnect Element Name, Interconnect Element Logical Name	none
0280	RPL	Register Platform	Platform Name, Platform Type, Platform Management Address List, Platform Node Name List	none
0291	RPLN	Register Platform Node Name	Platform Name, Platform Node Name	none
0292	RPLT	Register Platform Type	Platform Name, Platform Type	none
0293	RPLM	Register Platform Management Address	Platform Name, Platform Management Address	none

Table 117 – Fabric Configuration Server – Request Command Codes

Code (hex)	Mnem.	Description	Attribute(s) in Request CT_IU	Attribute(s) in Accept CT_IU
0380	DPL	Deregister Platform	Platform Name	none
0391	DPLN	Deregister Platform Node Name	Platform Node Name	none
0393	DPLML	Deregister Platform Management Address List	Platform Name	none
Other		Reserved		

Notes:

1 The Request CT_IU for GTIN contains the request payload for the Request Topology Information Extended Link Service, not including the first word that contains the ELS opcode.

2 The Accept CT_IU for GTIN contains the ACC payload for the Request Topology Information Extended Link Service, not including the first word that contains the ACC opcode.

3 The Accept CT_IU for GNID contains the ACC payload for the Request Node Identification Data Extended Link Service, not including the first word that contains the ACC opcode.

6.1.1.2 Registration

Registrations are limited to a single Fabric Configuration Server attribute at a time. A registrant submits a tuple, consisting of an object Name_Identifier, along with an attribute to be associated with the object.

The registration requests defined for the Fabric Configuration Server are summarized in table 117. Note that some attributes do not have a corresponding registration request; this Standard does not define the registration of those attributes.

The Fabric Configuration Server may reject registrations due to Fabric Configuration Server resource limitations. However, the Fabric Configuration Server shall support registration of all attributes, once registration of a single attribute has been accepted for a given Name_Identifier.

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

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

6.1.1.3 Queries

The Fabric Configuration Server may reject any query requests for reasons not specified in this document. The queries defined for the Fabric Configuration Server are summarized in table 117.

6.1.2 Fabric Configuration Server Objects and Attributes

Figure 3 illustrates the physical fabric, consisting of one or more Interconnect Elements, that each have some number of physical Ports. These Ports are then connected either to other Ports on other Interconnect Elements, or to N_Ports outside of the physical fabric.

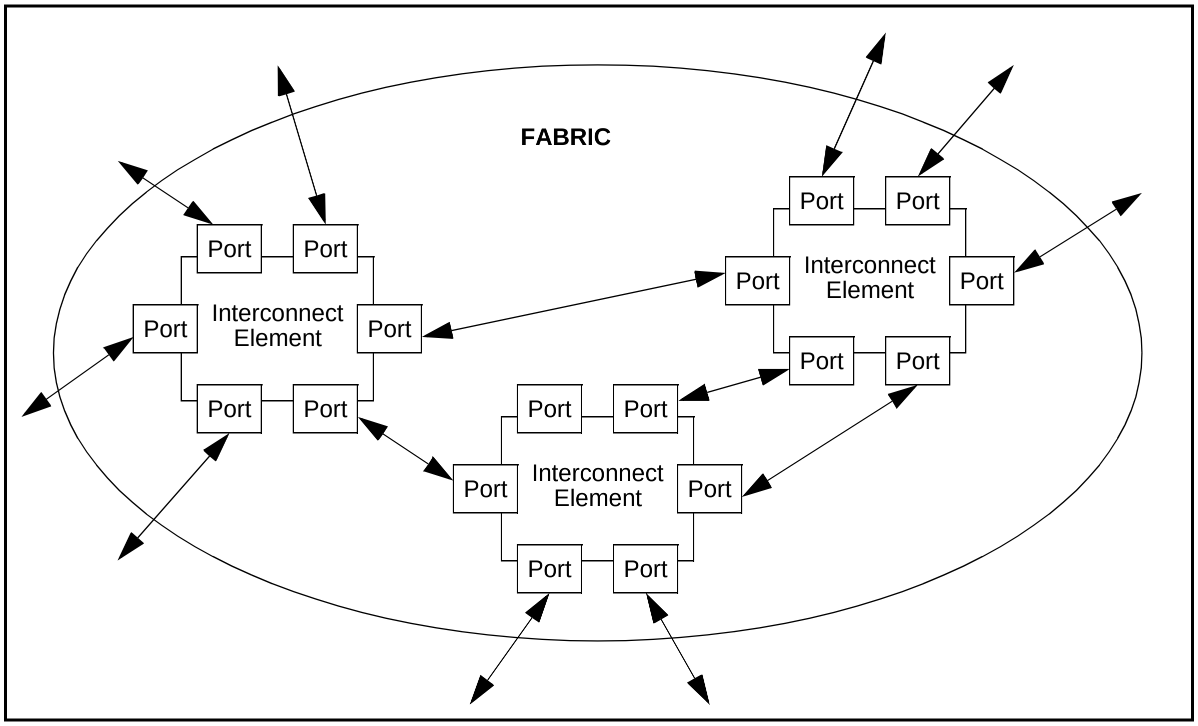


Figure 3 – Physical Fabric Illustration

The Fabric Configuration Server object model is shown in figure 4.

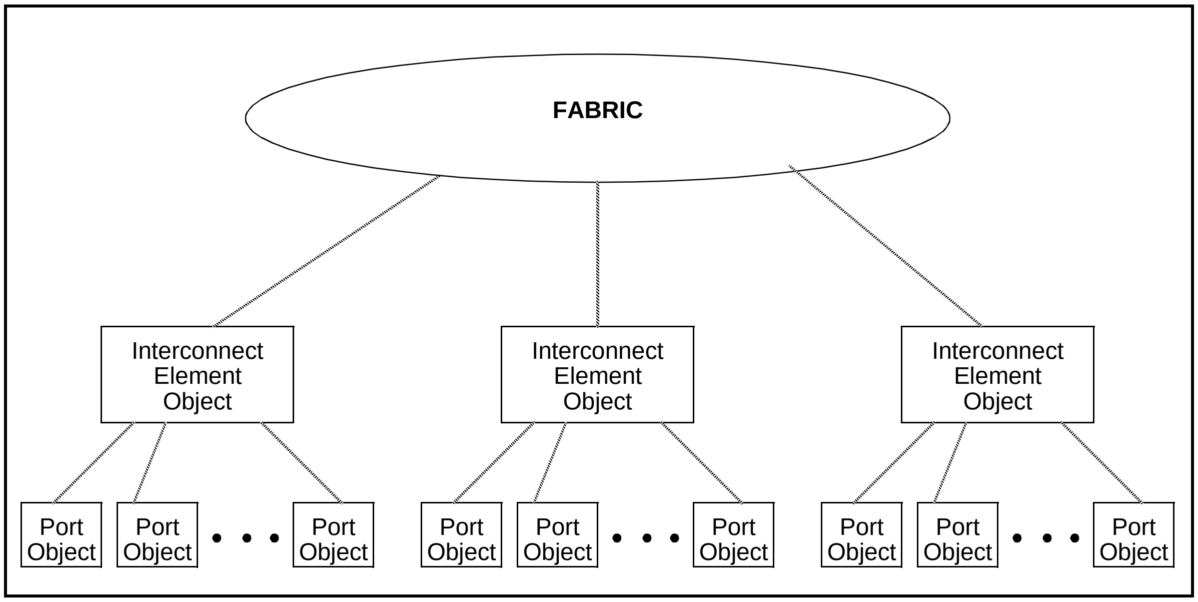

FABRIC
Interconnect Element Object
Interconnect Element Object
Interconnect Element Object
Port Object
Port Object
• • •
Port Object

Figure 4 – Fabric Configuration Server Object Model

The base object class managed by the Fabric Configuration Server is the Interconnect Element object. Interconnect Element objects have one or more associated Port objects. One or more Interconnect Element objects belong to a fabric. Interconnect Element objects and Port objects may have attributes associated with them, as shown in figure 5.

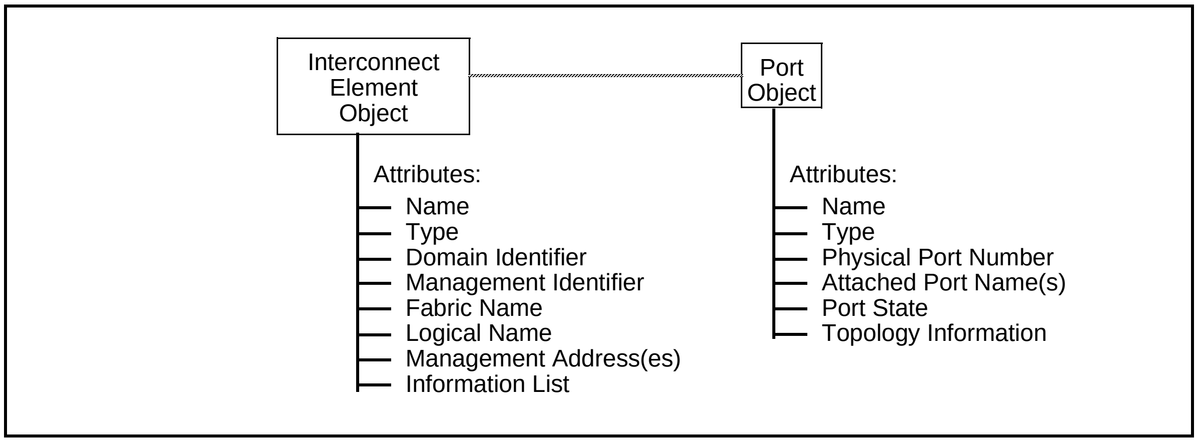


Figure 5 – Interconnect Element and Port attributes

6.1.2.1 Interconnect Element Object

Interconnect Element objects have the attributes described below.

6.1.2.1.1 Interconnect Element Name

The format of the Interconnect Element Name attribute, as used by the Fabric Configuration Server, shall be identical to the Name_Identifier format defined in FC-FS. If the Interconnect Element is a Switch (see FC-SW), the Interconnect Element Name attribute shall be the Switch_Name of the Switch.

This Standard does not define how this attribute is registered with the Fabric Configuration Server. The null value for the Interconnect Element Name attribute is hex’00 00 00 00 00 00 00 00’.

6.1.2.1.2 Interconnect Element Type

The format of the Interconnect Element Type attribute, shall be as shown in table 118.

Table 118 – Interconnect Element Type– encoding

Encoded value (hex)	Description
00	Unknown
01	Switch
02	Hub
03	Bridge
all others	Reserved

This Standard does not define how this attribute is registered with the Fabric Configuration Server. The null Interconnect Element Type attribute value is set to ‘Unknown’.

6.1.2.1.3 Interconnect Element Domain Identifier

The format of the Interconnect Element Domain Identifier attribute, as used by the Fabric Configuration Server, shall be identical to the Domain Identifier format defined in FC-FG.

This Standard does not define how this attribute is registered with the Fabric Configuration Server. The null value for the Interconnect Element Domain Identifier attribute is hex’00’.

6.1.2.1.4 Interconnect Element Management Identifier

The format of the Interconnect Element Management Identifier attribute, as used by the Fabric Configuration Server, shall be identical to the address identifier format defined in FC–FS. If the Interconnect Element is a Switch (see FC-SW), the Interconnect Element Management Identifier attribute shall be the Domain Controller identifier of the Switch.

This Standard does not define how this attribute is registered with the Fabric Configuration Server. The null value for the Interconnect Element Management Identifier attribute is hex’00 00 00’.

6.1.2.1.5 Interconnect Element Fabric Name

The format of the Interconnect Element Fabric Name attribute, as used by the Fabric Configuration Server, shall be identical to the Name_Identifier format defined in FC-FS. The value of the Interconnect Element Fabric Name shall be the same as the value Fabric_Name in the Fabric Login ELS Accept payload.

This Standard does not define how this attribute is registered with the Fabric Configuration Server. The null value for the Interconnect Element Fabric Name attribute is hex’00 00 00 00 00 00 00 00’.

6.1.2.1.6 Interconnect Element Logical Name

The format of the Interconnect Element Logical Name attribute, as used by the Fabric Configuration Server, shall be shall be as shown in table 119. The contents of these bytes are not defined and shall not be restricted by the Fabric Configuration Server.

Table 119 – Logical Name Format

Item	Size (Bytes)
Logical Name length (m)	1
Logical Name	m
Reserved	m-255

This attribute may be registered using the protocol described in 6.1.1.2. The null value for the Interconnect Element Logical Name attribute is a zero-length Interconnect Element Logical Name.

6.1.2.1.7 Interconnect Element Management Address

The format of the Interconnect Element Management Address attribute, as used by the Fabric Configuration Server, shall be as shown in table 120. Zero or more Management Address attributes may be associated with an Interconnect Element object.

Table 120 – Management Address Format

Item	Size (Bytes)
Management Address length (m)	1
Management Address value	m
Reserved	m-255

NOTE – The format of the Management Address may be based on the format of the Uniform Resource Locator (URL). The general form of this format is:

– protocol:// protocol specific address/ context specific information

Some typical management address formats that may be used are shown below:

– SNMP (Simple Network Management Protocol) - snmp://ipaddress [:port #]

– HTTP (Hypertext Transfer Protocol) - http://ipaddress [:port#]/ service

– XML (Extensible Markup Language) - xml://ipaddress [:port#]/service

This Standard does not define how this attribute is registered with the Fabric Configuration Server. The contents of the Management Address shall not be restricted by the Fabric Configuration Server. The null value for the Interconnect Element Management Address List attribute is a zero-length Interconnect Element Management Address List.

6.1.2.1.8 Interconnect Element Information List

The format of the Interconnect Element Information List attribute, as used by the Fabric Configuration Server, shall be as shown in table 121. This Standard does not define how this attribute is registered with the Fabric Configuration Server.

Table 121 – Information List Format

Item	Size (Bytes)
Vendor Specific Information Length (k)	1
Interconnect Element Type	1
Reserved	2
Type Number {?}	8
Model Number	4

Table 121 – Information List Format

Vendor Identification	16
Plant of Manufacture	4
Sequence Number	16
Reserved	12
Vendor Specific Information	4*k

Vendor Specific Information Length (n): Specifies the length of the Vendor Specific Information in words.

Interconnect Element Type: As defined in 6.1.2.1.2.

{NOTE: These definitions are still not settled and are subject to radical change in future revs. One proposal is to use the RNID format instead - ED}

Type Number: An ASCII character string that specifies the type number of the designated Interconnect Element. The character string can be up to eight bytes in length. If the character string is less than eight bytes in length, it shall be left justified and padded to the right with ASCII nulls x’00’ such that the entire field is filled. When no type number value is supplied, the type number field shall contain all ASCII nulls X’00’. {HOW IS THIS DIFFERENT THAN ELEMENT TYPE? - ED}

Model Number: An ASCII character string that specifies the model number of the designated Interconnect Element. The character string can be up to four bytes in length. If the character string is less than four bytes in length, it shall be left justified and padded to the right with ASCII nulls x’00’ such that the entire field is filled. When no model number value is supplied, the model number field shall contain all ASCII nulls X’00’. {WHY DOES THIS NEED TO BE ASCII WHEN IT WILL LIKELY RESULT IN AN INDEX LOOKUP ANYWAY? - ED}

Vendor Identification: An ASCII character string that specifies the vendor of the designated Interconnect Element. The character string can be up to sixteen bytes in length. If the character string is less than sixteen bytes in length, it shall be left justified and padded to the right with ASCII nulls x’00’ such that the entire field is filled. When no vendor identification value is supplied, the vendor identification field shall contain all ASCII nulls X’00’.

Plant of Manufacture: An ASCII character string that specifies the plant of manufacture for the designated Interconnect Element. The character string can be up to four bytes in length. If the character string is less than four bytes in length, it shall be left justified and padded to the right with ASCII nulls x’00’ such that the entire field is filled. When no plant of manufacture value is supplied, the plant of manufacture field shall contain all ASCII nulls X’00’.

Sequence Number: An ASCII character string that specifies a sequence number. The character string can be up to sixteen bytes in length. If the character string is less than sixteen bytes in length, it shall be left justified and padded to the right with ASCII nulls x’00’ such that the entire field is filled. The sequence number may be concatenated with the plant of manufacture value to designate a serial number. When no sequence number value is supplied, the sequence number field shall contain all ASCII nulls X’00’. {OPINION - THIS STD SHOULD NOT BE DEFINING SERIAL NUMBER FORMATS - ED}

Vendor Specific Information: The format of this field is undefined other than it consists of one or more character strings containing ASCII graphic codes (X’20’ - X’7E’). Each ASCII character string is terminated with an ASCII null character X’00’. If necessary, ASCII null characters X’00’ are padded to the right of the termination character associated with the last character string until the specified length is realized.

6.1.2.2 Port Object

Port objects have the attributes described below.

6.1.2.2.1 Port Name

The format of the Port Name attribute, as used by the Fabric Configuration Server, shall be identical to the Name_Identifier format defined in FC-FS. The value of the Port Name attribute shall be the same as the value Port_Name in the Fabric Login ELS Accept payload.

This Standard does not define how this attribute is registered with the Fabric Configuration Server. The null value for the Port Name attribute is hex’00 00 00 00 00 00 00 00’.

6.1.2.2.2 Port Type

The format of the Port Type attribute, shall be as shown in table 122.

Table 122 – Port Type encoding

Encoded value (hex)	Description
00	Unknown
01	N_Port
02	NL_Port
81	F_Port
82	FL_Port
84	E_Port
85	B_Port
all others	Reserved

This Standard does not define how this attribute is registered with the Fabric Configuration Server. The null Port Type attribute value is set to ‘Unknown’.

6.1.2.2.3 Physical Port Number

The format of the Physical Port Number attribute, as used by the Fabric Configuration Server, shall be as shown in table 123. The contents of this field are not defined and shall not be restricted by the Fabric Configuration Server.

Table 123 – Physical Port Number Format

Item	Size (Bytes)
Physical Port Number	4

This Standard does not define how this attribute is registered with the Fabric Configuration Server. The null value for the Interconnect Element Domain Identifier attribute is hex’00’.

6.1.2.2.4 Attached Port Name

The format of the Attached Port Name attribute, as used by the Fabric Configuration Server, shall be as shown in table 124. Zero or more Attached Port Name attributes may be associated with a Port object.

Table 124 – Attached Port Name Format

Item	Size (Bytes)
Port Name	8
Reserved	2
Port Flags	1
Port Type	1

Port Name: As defined in 6.1.2.2.1.

Port Flags: As shown in table 125. .

Table 125 – Port Flags field bits

Bit Position	Description
15-9	Reserved
8	A value of one indicates that the Port supports the Report Topology Information Extended Link Service. A value of zero indicates that the Port does not support this ELS.

Port Type: As defined in 6.1.2.2.2.

This Standard does not define how this attribute is registered with the Fabric Configuration Server. A Port object with a Port Type attribute value of “N_Port” or “NL_Port” shall have a null Attached Port

Name List. The null value for the Attached Port Name List attribute shall be a zero length Attached Port Name List.

6.1.2.2.5 Port State

The format of the Port State attribute, shall be as shown in table 126.

Table 126 – Port State encoding

Encoded value (hex)	Description
00	Unknown
01	Online - a user frame may be passed through the Port
02	Offline - a user frame can not be passed through the Port
03	Testing - port is in a test state
04	Fault - port is not operational
E0-FF	Vendor specific
all others	Reserved

This Standard does not define how this attribute is registered with the Fabric Configuration Server. The null Port State attribute value is set to ‘Unknown’.

6.1.2.2.6 Attached Topology Information

The format of the Attached Topology Information attribute, as used by the Fabric Configuration Server, shall be identical to the Request Topology Information ELS Accept format defined in FC-FS, not including the first two words {?}. The value of the Attached Topology Information attribute shall be null if the Port Type is E_Port. Otherwise, the value of the Attached Topology Information attribute shall be the same as the value of the Accept that would be returned by the Port in response to the Request Topology Information ELS.

This Standard does not define how this attribute is registered with the Fabric Configuration Server. The null value for the Attached Topology Information attribute is hex’00 00 00 00’ {?}.

6.1.2.3 Platform Object

Platform objects are defined to provide the basic ability to associate one or more Nodes with a single platform for discovery and management. Platform objects may have attributes associated with them, as shown in figure 6.

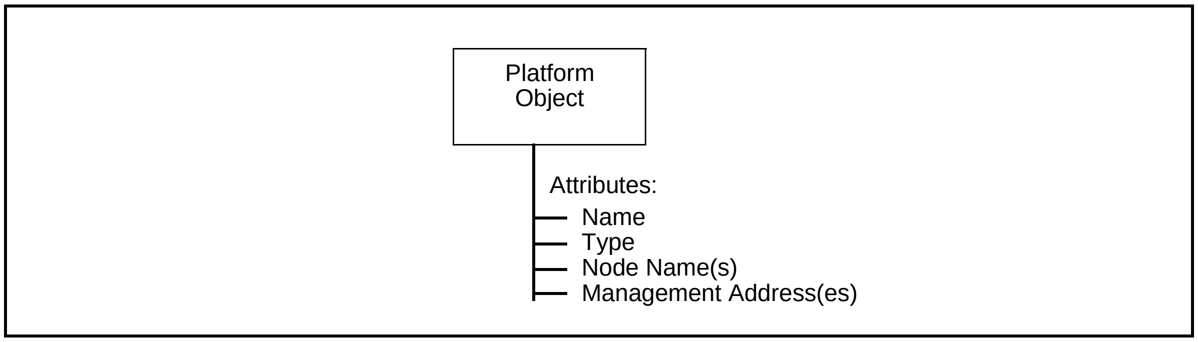


Figure 6 – Platform attributes

6.1.2.3.1 Platform Name

The format of the Platform Name attribute, as used by the Fabric Configuration Server, shall be shall be as shown in table 127. The contents of these bytes are not defined and shall not be restricted by the Fabric Configuration Server..

Table 127 – Platform Name Format

Item	Size (Bytes)
Platform Name length (m)	1
Platform Name	m
Reserved	m-255

This attribute may be registered using the protocol described in 6.1.1.2. The null value for the Platform Name attribute is a zero-length Platform Name.

6.1.2.3.2 Platform Type

The format of the Platform Type attribute, shall be as shown in table 128.

Table 128 – Platform Type – encoding

Encoded value (hex)	**Description**
00 00 00 00	Unknown
00 00 00 01	Host
00 00 00 02	Host Bus Adapter (internal controller)
00 00 00 03	External controller
00 00 00 04	Smart enclosure

Table 128 – Platform Type – encoding

Encoded value (hex)	Description
00 00 00 05	Bridge
00 00 00 06	Gateway
00 00 00 07	Hard disk
00 00 00 08	WORM
00 00 00 09	DVD
00 00 00 0A	Storage subsystem
00 00 00 0B	Magneto optical
00 00 00 0C	Media changer
00 00 00 0D	Tape
00 00 00 0E	CD
00 00 00 0F	Diskette
00 00 00 10	Other storage entity
00 00 00 11	Other entity
all others	Reserved

This attribute may be registered using the protocol described in 6.1.1.2. The null Platform Type attribute value is set to ‘Unknown’.

6.1.2.3.3 Platform Node Name

The format of the Platform Node Name attribute, as used by the Fabric Configuration Server, shall be identical to the Name_Identifier format defined in FC-FS. Zero or more Platform Node Name attributes may be associated with a Platform object. Node Names are registered to associate a Platform with the Nodes.

This attribute may be registered using the protocol described in 6.1.1.2. The null value for the platform Node Name attribute is hex’00 00 00 00 00 00 00 00’.

6.1.2.3.4 Platform Management Address

The format of the Platform Management Address attribute, as used by the Fabric Configuration Server, is identical to the Interconnect Element Management Address attribute, and shall be as defined in 6.1.2.1.7.

6.1.3 Reason code explanations

A Reject CT_IU (see 4.4.3) shall notify the requestor that the request has been unsuccessfully completed. The first error condition encountered shall be the error reported by the Reject CT_IU.

If a valid Fabric Configuration 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 Fabric Configuration Server request shall not be rejected with a Reason Code of “Command not supported”

If a Fabric Configuration Server request is rejected with a reason code of ‘Unable to perform command request’, then one of the reason code explanations, shown in table 129, are returned.

Table 129 – Reject CT_IU Reason code explanations

Encoded value (hex)	Description
00	No additional explanation
01	Invalid Name_Identifier for Interconnect Element or Port
10	Interconnect Element List not available
11	Interconnect Element Type not available
12	Domain Identifier not available
13	Management Identifier not available
14	Fabric Name not available
15	Interconnect Element Logical Name not available
16	Management Address List not available
17	Interconnect Element Information List not available
30	Port List not available
31	Port Type not available
32	Physical Port Number not available
34	Attached Port Name List not available
36	Port State not available
37	Topology Discovery not complete {?}
50	Unable To register Interconnect Element Logical Name
60	Platform Name does not exist
61	Platform Name already exists.
62	Platform Node Name does not exist
63	Platform Node Name already exists.
F0	Authorization Exception
F1	Authentication Exception
{ns?}	Access denied
{ns?}	Data base empty
Others	Reserved

The use of these codes is further defined as follows:

– If a Fabric Configuration Server request is rejected by the Fabric Configuration Server because of the identity of the requestor, then the Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation of ‘Access denied’.

– If a Fabric Configuration Server Query request is rejected by the Fabric Configuration Server because no Fabric Configuration Server entries exist, then the Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation of ‘Data base empty’.

– If a Fabric Configuration Server Query request other than GIEL and GPL is rejected by the Fabric Configuration Server because the attribute specified in the request is not found in the Fabric Configuration Server data base, then the Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation that indicates the specified attribute is not available.

– Additional uses may be defined for specific Fabric Configuration Server requests.

6.1.4 Commands

The commands defined for the Fabric Configuration Server are summarized in table 117.

6.1.4.1 Query – Get Topology Information (GTIN)

The Fabric Configuration Server shall, when it receives a GTIN request, return the Topology Information for the specified destination. The format of the GTIN Request CT_IU is shown in table 130.

Table 130 – GTIN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
RTIN request payload	see text

The RTIN request payload shall contain the request payload for the Request Topology Information ELS, not including the first (opcode) word. See FC-FS (reference [13]) for the definition and length of this payload. {note - the max size field in RTIN is redundant to the CT header - we probably need to think about this - ed }

The format of the Accept CT_IU to a GTIN request is shown in table 131.

Table 131 – Accept CT_IU to GTIN Request

Item	Size (Bytes)
CT_IU preamble	16
RTIN accept payload	see text

The RTIN accept payload shall contain the accept payload for the Request Topology Information ELS, not including the first (opcode) word. See FC-FS (reference [13]) for the definition and length of this payload. {note - the residual size field in RTIN ACC is redundant to the CT header - we probably need to think about this - ed }

6.1.4.2 Query – Get Interconnect Element List (GIEL)

The Fabric Configuration Server shall, when it receives a GIEL request, return all Interconnect Element Names in the fabric. The format of the GIEL Request CT_IU is shown in table 132.

Table 132 – GIEL Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16

The format of the Accept CT_IU to a GIEL request is shown in table 133.

Table 133 – Accept CT_IU to GIEL Request

Item	Size (Bytes)
CT_IU preamble	16
Number of Interconnect Element entries (n)	4
Interconnect Element Name #1	8
Reserved	3
Interconnect Element Type #1	1
Interconnect Element Name #2	8
Reserved	3
Interconnect Element Type #2	1
...	
Interconnect Element Name #n	8
Reserved	3
Interconnect Element Type #n	1

One or more Interconnect Element entries are returned, and the Interconnect Element entries may be returned in any order. Furthermore, the order may be different for every request even if the same Interconnect Element entries are returned and the requestor is the same.

6.1.4.3 Query – Get Interconnect Element Type (GIET)

The Fabric Configuration Server shall, when it receives a GIET request, return the Interconnect Element Type for the specified Interconnect Element Name. The format of the GIET Request CT_IU is shown in table 134.

Table 134 – GIET Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Interconnect Element Name	8

The format of the Accept CT_IU to a GIET request is shown in table 135.

Table 135 – Accept CT_IU to GIET Request

Item	Size (Bytes)
CT_IU preamble	16
Reserved	3
Interconnect Element Type	1

6.1.4.4 Query – Get Interconnect Element Domain Identifier (GDID)

The Fabric Configuration Server shall, when it receives a GDID request, return the Interconnect Element Domain Identifier for the specified Interconnect Element Name. The format of the GDID Request CT_IU is shown in table 136.

Table 136 – GDID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Interconnect Element Name	8

The format of the Accept CT_IU to a GDID request is shown in table 137.

Table 137 – Accept CT_IU to GDID Request

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1

Table 137 – Accept CT_IU to GDID Request

Interconnect Element Domain Identifier	1
Reserved	2

6.1.4.5 Query – Get Interconnect Element Management Identifier (GMID)

The Fabric Configuration Server shall, when it receives a GMID request, return the Interconnect Element Management Identifier for the specified Interconnect Element Name. The format of the GMID Request CT_IU is shown in table 138.

Table 138 – GMID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Interconnect Element Name	8

The format of the Accept CT_IU to a GMID request is shown in table 139.

Table 139 – Accept CT_IU to GMID Request

Item	Size (Bytes)
CT_IU preamble	16
Reserved	1
Interconnect Element Management Identifier	3

6.1.4.6 Query – Get Interconnect Element Fabric Name (GFN)

The Fabric Configuration Server shall, when it receives a GFN request, return the Interconnect Element Fabric Name for the specified Interconnect Element Name. The format of the GFN Request CT_IU is shown in table 140.

Table 140 – GFN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Interconnect Element Name	8

The format of the Accept CT_IU to a GFN request is shown in table 141.

Table 141 – Accept CT_IU to GFN Request

Item	Size (Bytes)
CT_IU preamble	16
Interconnect Element Fabric Name	8

6.1.4.7 Query – Get Interconnect Element Logical Name (GIELN)

The Fabric Configuration Server shall, when it receives a GIELN request, return the Interconnect Element Management Identifier for the specified Interconnect Element Name. The format of the GIELN Request CT_IU is shown in table 142.

Table 142 – GIELN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Interconnect Element Name	8

The format of the Accept CT_IU to a GIELN request is shown in table 143.

Table 143 – Accept CT_IU to GIELN Request

Item	Size (Bytes)
CT_IU preamble	16
Interconnect Element Logical Name	256

6.1.4.8 Query – Get Interconnect Element Management Address List (GMAL)

The Fabric Configuration Server shall, when it receives a GMAL request, return all Interconnect Element Management Address attributes for the specified Interconnect Element Name. The format of the GIEL Request CT_IU is shown in table 144.

Table 144 – GMAL Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Interconnect Element Name	8

The format of the Accept CT_IU to a GMAL request is shown in table 145.

Table 145 – Accept CT_IU to GMAL Request

Item	Size (Bytes)
CT_IU preamble	16
Number of Management Address entries (n)	4
Management Address #1	256
Management Address #2	256
...	
Management Address #n	256

One or more Interconnect Element Management Address entries are returned, and the entries may be returned in any order. Furthermore, the order may be different for every request even if the same entries are returned and the requestor is the same.

6.1.4.9 Query – Get Interconnect Element Information List (GIEIL)

The Fabric Configuration Server shall, when it receives a GIEIL request, return the Interconnect Element Information List for the specified Interconnect Element Name. The format of the GIEIL Request CT_IU is shown in table 146.

Table 146 – GIEIL Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Interconnect Element Name	8

The format of the Accept CT_IU to a GIEIL request is shown in table 147.

Table 147 – Accept CT_IU to GIEIL Request

Item	Size (Bytes)
CT_IU preamble	16
Interconnect Element Information List	X

{NOTE - The value of “X” is TBD pending until we settle the format of this list - ED}

6.1.4.10 Query – Get Port List (GPL)

The Fabric Configuration Server shall, when it receives a GPL request, return all Port Names and their associated Port Types for the specified Interconnect Element Name. The format of the GPL Request CT_IU is shown in table 148.

Table 148 – GPL Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Interconnect Element Name	8

The format of the Accept CT_IU to a GPL request is shown in table 149.

Table 149 – Accept CT_IU to GPL Request

Item	Size (Bytes)
CT_IU preamble	16
Number of Port entries (n)	4
Port Name #1	8
Reserved	3
Port Type #1	1
Port Name #2	8
Reserved	3
Port Type #2	1
...	
Port Name #n	8
Reserved	3
Port Type #n	1

One or more Port entries are returned, and the entries may be returned in any order. Furthermore, the order may be different for every request even if the same entries are returned and the requestor is the same.

6.1.4.11 Query – Get Port Type (GPT)

The Fabric Configuration Server shall, when it receives a GPT request, return the Port Type for the specified Port Name. The format of the GPT Request CT_IU is shown in table 150.

Table 150 – GPT Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Port Name	8

The format of the Accept CT_IU to a GPT request is shown in table 151.

Table 151 – Accept CT_IU to GPT Request

Item	Size (Bytes)
CT_IU preamble	16
Reserved	3
Port Type	1

6.1.4.12 Query – Get Physical Port Number (GPPN)

The Fabric Configuration Server shall, when it receives a GPPN request, return the Physical Port Number for the specified Port Name. The format of the GPPN Request CT_IU is shown in table 152.

Table 152 – GPPN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Port Name	8

The format of the Accept CT_IU to a GPPN request is shown in table 153.

Table 153 – Accept CT_IU to GPPN Request

Item	Size (Bytes)
CT_IU preamble	16
Physical Port Number	4

6.1.4.13 Query – Get Attached Port Name List (GAPNL)

The Fabric Configuration Server shall, when it receives a GAPNL request, return all Attached Port Name attributes for the specified Port Name. The format of the GAPNL Request CT_IU is shown in table 154.

Table 154 – GAPNL Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Port Name	8

The format of the Accept CT_IU to a GAPNL request is shown in table 155.

Table 155 – Accept CT_IU to GAPNL Request

Item	Size (Bytes)
CT_IU preamble	16
Number of Attached Port entries (n)	4
Attached Port Name #1	12
Attached Port Name #2	12
...	
Attached Port Name #n	12

One or more Attached Port entries are returned, and the entries may be returned in any order. Furthermore, the order may be different for every request even if the same entries are returned and the requestor is the same.

6.1.4.14 Query – Get Port State (GPS)

The Fabric Configuration Server shall, when it receives a GPS request, return the Port Type and Port State for the specified Port Name. The format of the GPS Request CT_IU is shown in table 156.

Table 156 – GPS Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Port Name	8

The format of the Accept CT_IU to a GPS request is shown in table 157.

Table 157 – Accept CT_IU to GPS Request

Item	Size (Bytes)
CT_IU preamble	16
Reserved	3
Port Type	1
Reserved	3
Port State	1

6.1.4.15 Query – Get Attached Topology Information (GATIN)

The Fabric Configuration Server shall, when it receives a GATIN request, return the Attached Topology Information for the specified Port Name. The format of the GATIN Request CT_IU is shown in table 158.

Table 158 – GATIN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Port Name	8

The format of the Accept CT_IU to a GATIN request is shown in table 159.

Table 159 – Accept CT_IU to GATIN Request

Item	Size (Bytes)
CT_IU preamble	16
Attached Topology Information	n

NOTE – This request may be used to discover topologies “behind” a Port, such as arbitrated loops or bridges to other interconnects. A typical approach might be to collect all of the Ports within an Interconnect Element using GPL (see 6.1.4.10), then issue this command to each Port in turn to discover the additional topologies.

6.1.4.16 Query – Get Platform Node Name List (GPLNL)

The Fabric Configuration Server shall, when it receives a GPLNL request, return all Node Name attributes for the specified Platform Name. The format of the GPLNL Request CT_IU is shown in table 160.

Table 160 – GPLNL Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Platform Name	256

The format of the Accept CT_IU to a GPLNL request is shown in table 161.

Table 161 – Accept CT_IU to GPLNL Request

Item	Size (Bytes)
CT_IU preamble	16
Number of Platform Node Name entries (n)	4
Platform Node Name #1	8
Platform Node Name #2	8
...	
Platform Node Name #n	8

One or more Platform Node Name entries are returned, and the entries may be returned in any order. Furthermore, the order may be different for every request even if the same entries are returned and the requestor is the same.

6.1.4.17 Query – Get Platform Type (GPLT)

The Fabric Configuration Server shall, when it receives a GPLT request, return the Platform Type attribute for the specified Platform Name. The format of the GPLT Request CT_IU is shown in table 162.

Table 162 – GPLT Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Platform Name	256

The format of the Accept CT_IU to a GPLT request is shown in table 163.

Table 163 – Accept CT_IU to GPLT Request

Item	Size (Bytes)
CT_IU preamble	16
Platform Type	4

6.1.4.18 Query – Get Platform Management Address List (GPLML)

The Fabric Configuration Server shall, when it receives a GPLML request, return all Interconnect Element Management Address attributes for the specified Platform Name. The format of the GPLML Request CT_IU is shown in table 164.

Table 164 – GPLML Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Platform Name	256

The format of the Accept CT_IU to a GPLML request is shown in table 165.

Table 165 – Accept CT_IU to GPLML Request

Item	Size (Bytes)
CT_IU preamble	16
Number of Management Address entries (n)	4
Management Address #1	256
Management Address #2	256
...	
Management Address #n	256

One or more Platform Management Address entries are returned, and the entries may be returned in any order. Furthermore, the order may be different for every request even if the same entries are returned and the requestor is the same.

6.1.4.19 Query – Get Platform Name - Node Name (GNPL)

The Fabric Configuration Server shall, when it receives a GNPL request, return the Platform Name attribute for the specified Platform Node Name. The format of the GNPL Request CT_IU is shown in table 166.

Table 166 – GNPL Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Platform Node Name	8

The format of the Accept CT_IU to a GNPL request is shown in table 167.

Table 167 – Accept CT_IU to GNPL Request

Item	Size (Bytes)
CT_IU preamble	16
Platform Name	256

6.1.4.20 Query – Get Node Identification Data (GNID)

The Fabric Configuration Server shall, when it receives a GNID request, return the Topology Information for the specified destination. The format of the GNID Request CT_IU is shown in table 168.

Table 168 – GNID Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Platform Node Name	8
Node Identification Data format	1
Reserved	

See the Request Node Identification Data ELS in FC-FS (reference [13]) for the definition of the Node Identification Data format.

The format of the Accept CT_IU to a GNID request is shown in table 169.

Table 169 – Accept CT_IU to GNID Request

Item	Size (Bytes)
CT_IU preamble	16
RNID accept payload	see text

The RNID accept payload shall contain the accept payload for the Request Node Identification Data ELS, not including the first (opcode) word. See FC-FS (reference [13]) for the definition and length of this payload.

6.1.4.21 Register Interconnect Element Logical Name (RIELN)

The RIELN Fabric Configuration Server request shall be used to associate a Logical Name with a given Interconnect Element.

The Fabric Configuration Server shall not attempt validation of the Logical Name attribute. This means that any Logical Name value shall be accepted.

Deregistration may be accomplished by registering a null Logical Name (see 6.1.2.1.6).

The format of the RIELN Request CT_IU is shown in table 170.

Table 170 – RIELN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Interconnect Element Name	8
Logical Name	256

The format of the RNN_ID Accept CT_IU is shown in table 171.

Table 171 – RIELN Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6.1.4.22 Register Platform (RPL)

The RPL Fabric Configuration Server request shall be used to associate a Platform Name with its attributes. This request allows the registration of a Platform and all of its attributes using a single transaction.

The Platform Name field of the Request CT_IU shall not be equal to a currently registered Platform Name. If the value of the Platform Name is equal to a currently registered Platform Name, the Fabric Configuration Server shall reject this request; the Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation of ‘Platform Name already exists’

No Platform Node Name field of the Request CT_IU shall be equal to a currently registered Platform Name. If the value of any Platform Node Name is equal to a currently registered Platform Node Name, the Fabric Configuration Server shall reject this request; the Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation of ‘Platform Node Name already exists’

In the absence of a reject condition, the Fabric Configuration Server shall create a new Platform object and register the Platform attributes with the new Platform.

The Fabric Configuration Server shall not attempt validation of the Platform Type or Platform Management Address attributes. This means that any value shall be accepted for these attributes.

Deregistration may be accomplished by a DPL request (see 6.1.4.26).

The format of the RPL Request CT_IU is shown in table 172.

Table 172 – RPL Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Platform Name	256
Platform Type	4
Number of Management Address entries (n)	4
Management Address #1	256
Management Address #2	256
...	
Management Address #n	256
Number of Platform Node Name entries (n)	4
Platform Node Name #1	8
Platform Node Name #2	8
...	
Platform Node Name #n	8

The format of the RPL Accept CT_IU is shown in table 173.

Table 173 – RPL Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6.1.4.23 Register Platform Node Name (RPLN)

The RPLN Fabric Configuration Server request shall be used to associate a Platform Node Name with a Platform.

The Platform Name field of the Request CT_IU may be equal to a currently registered Platform Name. If the value of the Platform Name is equal to a currently registered Platform Name, the Fabric Configuration Server shall register the Platform Node Name with the existing Platform. If the value of the Platform Name is not equal to a currently registered Platform Name, the Fabric Configuration Server shall create a new Platform object and register the Platform Node Name with the new Platform.

The Fabric Configuration Server shall not attempt validation of the Platform Node Name attribute. This means that any value shall be accepted for this attribute.

Deregistration may be accomplished by a DPLN request (see 6.1.4.27).

The format of the RPLN Request CT_IU is shown in table 174.

Table 174 – RPLN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Platform Name	256
Platform Node Name	8

The format of the RPLN Accept CT_IU is shown in table 175.

Table 175 – RPLN Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6.1.4.24 Register Platform Type (RPLT)

The RPLT Fabric Configuration Server request shall be used to associate a Platform Type with a Platform.

The Platform Name field of the Request CT_IU may be equal to a currently registered Platform Name. If the value of the Platform Name is equal to a currently registered Platform Name, the Fabric Configuration Server shall register the Platform Type with the existing Platform. If the value of the Platform Name is not equal to a currently registered Platform Name, the Fabric Configuration Server shall create a new Platform object and register the Platform Type with the new Platform.

The Fabric Configuration Server shall not attempt validation of the Platform Type attribute. This means that any value shall be accepted for this attribute.

Deregistration may be accomplished by registering a null Platform Type (see 6.1.2.3.2).

The format of the RPLT Request CT_IU is shown in table 176.

Table 176 – RPLT Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Platform Name	256
Platform Type	4

The format of the RPLT Accept CT_IU is shown in table 177.

Table 177 – RPLT Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6.1.4.25 Register Platform Management Address (RPLM)

The RPLM Fabric Configuration Server request shall be used to associate a Platform Management Address with a Platform.

The Platform Name field of the Request CT_IU may be equal to a currently registered Platform Name. If the value of the Platform Name is equal to a currently registered Platform Name, the Fabric Configuration Server shall register the Platform Management Address with the existing Platform. If the value of the Platform Name is not equal to a currently registered Platform Name, the Fabric Configuration Server shall create a new Platform object and register the Platform Management Address with the new Platform.

The Fabric Configuration Server shall not attempt validation of the Platform Management Address attribute. This means that any value shall be accepted for this attribute.

Deregistration may be accomplished by a DPLML request (see 6.1.4.28).

The format of the RPLM Request CT_IU is shown in table 178.

Table 178 – RPLM Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Platform Name	256
Platform Management Address	256

The format of the RPLM Accept CT_IU is shown in table 179.

Table 179 – RPLM Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6.1.4.26 Deregister Platform (DPL)

The DPL Fabric Configuration Server request shall be used to destroy a registered Platform object and all of its attributes.

The Platform Name field of the Request CT_IU shall be equal to a currently registered Platform Name. If the value of the Platform Name is not equal to a currently registered Platform Name, the Fabric Configuration Server shall reject this request; the Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation of ‘Platform Name does not exist’

If the value of the Platform Name is equal to a currently registered Platform Name, the Fabric Configuration Server shall delete the Platform object from its database.

The format of the DPL Request CT_IU is shown in table 180.

Table 180 – DPL Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Platform Name	256

The format of the DPL Accept CT_IU is shown in table 181.

Table 181 – DPL Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6.1.4.27 Deregister Platform Node Name (DPLN)

The DPL Fabric Configuration Server request shall be used to disassociate a registered Platform Node Name from a Platform object.

The Platform Node Name field of the Request CT_IU shall be equal to a currently registered Platform Node Name. If the value of the Platform Node Name is not equal to a currently registered Platform Node Name, the Fabric Configuration Server shall reject this request; the Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation of ‘Platform Node Name does not exist’.

If the value of the Platform Node Name is equal to a currently registered Platform Node Name, the Fabric Configuration Server shall delete the Platform Node Name from its database.

The format of the DPLN Request CT_IU is shown in table 182.

Table 182 – DPLN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Platform Node Name	8

The format of the DPLN Accept CT_IU is shown in table 183.

Table 183 – DPLN Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6.1.4.28 Deregister Platform Management Address List (DPLML)

The DPL Fabric Configuration Server request shall be used to disassociate all registered Platform Management Addresses from a Platform object.

The Platform Name field of the Request CT_IU shall be equal to a currently registered Platform Name. If the value of the Platform Name is not equal to a currently registered Platform Name, the Fabric Configuration Server shall reject this request; the Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation of ‘Platform Name does not exist’

If the value of the Platform Name is equal to a currently registered Platform Name, the Fabric Configuration Server shall delete all Platform Management Addresses, associated with the Platform object, from its database.

The format of the DPLML Request CT_IU is shown in table 184.

Table 184 – DPLML Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Platform Name	256

The format of the DPLML Accept CT_IU is shown in table 185.

Table 185 – DPLML Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6.2 Unzoned Name Server

The Unzoned Name Server is provided by the Management Service to give a management application access to the Name Server database without Zone constraints. An example of Zoned and Unzoned access is illustrated in figure 7.

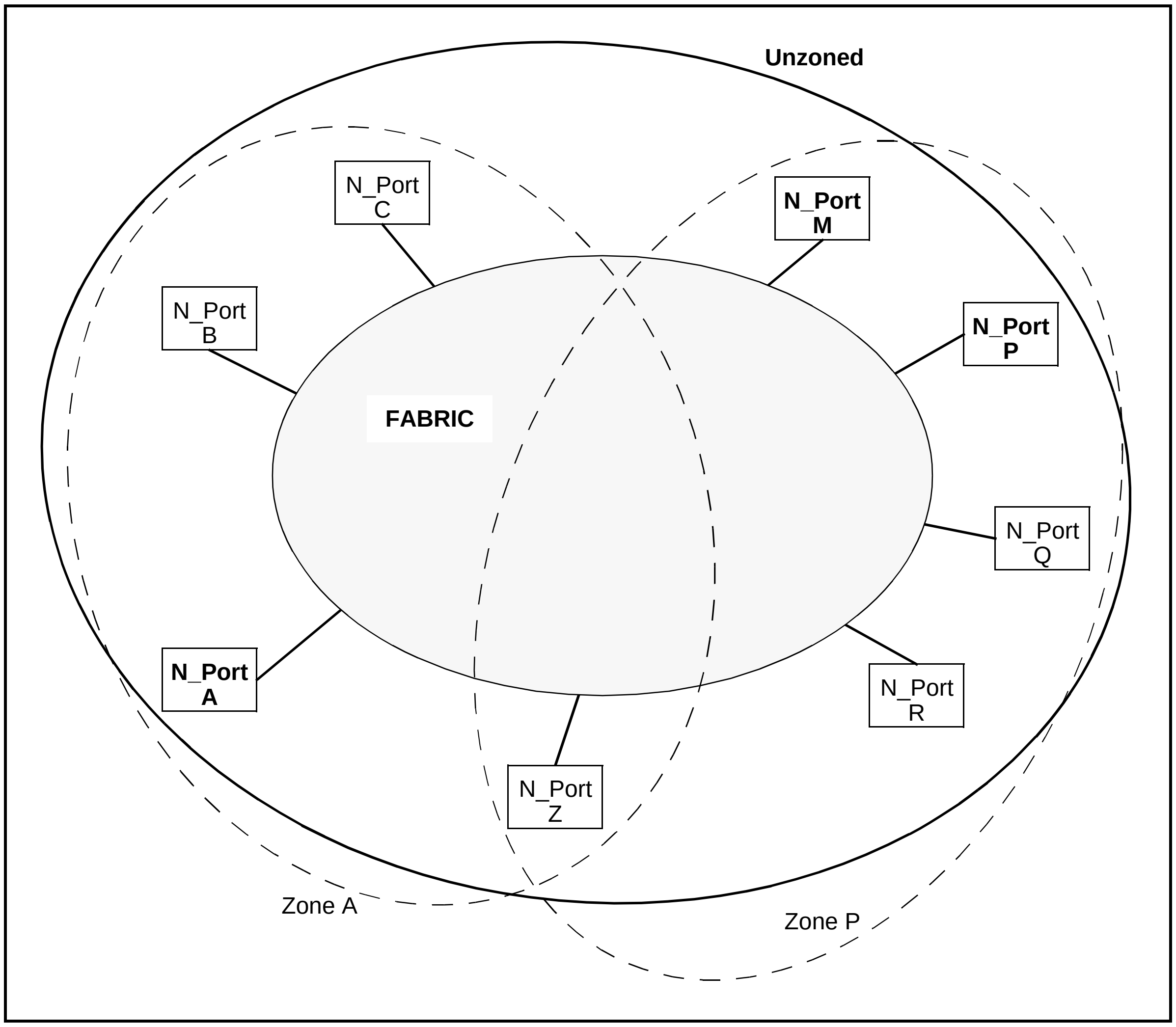


Figure 7 – Name Server Zone Constraints

In the example, N_Port A is in a Zone that allows access to, and visibility of, N_Ports B, C, and Z. N_Port P is in a Zone that allows access to, and visibility of, N_Ports Q, R, M, and Z. N_Port M is in the same Zone as P, and is allowed access to, and visibility of, N_Ports Q, R, P, and Z.

The Name Server provided by the Directory Service is required to constrain access based on Zones (see 5.1.1). So, if each N_Port issues a GID_PT (Get IDs based on Port Type) request, the answers they get are:

- Response for N_Port A contains N_Ports A, B, C, and Z;
- Response for N_Port P contains N_Ports P, Q, R, M, and Z;
- Response for N_Port M contains N_Ports P, Q, R, M, and Z.

Now, let's suppose that N_Port M is running a management application, and makes the same GID_PT request via the Unzoned Name Server provided by the Management Service. The response for N_Port M in this case contains N_Ports A, B, C, P, Q, R, M, and Z. The response is unconstrained by the Zone configuration.

6.2.1 Protocol

Unzoned Name Server registration, deregistration and queries are managed through protocols containing a set of Request CT_IUs and Response CT_IUs supported by the Unzoned Name Server.

Synchronous transactions shall be used, as defined in 4.5.1. For a Unzoned Name Server request, the Unzoned Name Server payload shall be transported from the requestor to the Unzoned Name Server using a Request CT_IU. The corresponding Unzoned Name Server response is transported from the Unzoned Name Server to the requestor, in the Exchange established by the requestor, using a Response CT_IU.

If Zones exist within the fabric, the Unzoned Name Server shall not restrict access to information in the Name Server database based on the Zone configuration.

6.2.1.1 CT_IU preamble values

The following values shall be set in the CT_IU preamble for Unzoned Name Server request and their responses; fields not specified here shall be set as defined in 4.3.1:

– Revision: hex '01';

– GS_Subtype: as indicated in table 116;

– Command Code: see table 12 for Request command codes.

6.2.1.2 Registration

The Unzoned Name Server shall not perform Registrations. If the Unzoned Name Server receives any Registration command defined in 5.1.1.2, it shall reject the command. The Reject CT_IU reason code shall be 'Unable to perform command request', with a reason code explanation of 'Access denied'.

6.2.1.3 Queries

Queries shall be performed as defined in 5.1.1.3.

6.2.1.4 Deregistration

The Unzoned Name Server shall not perform Deregistration. If the Unzoned Name Server receives any Deregistration command defined in 5.1.1.4, it shall reject the command. The Reject CT_IU reason code shall be 'Unable to perform command request', with a reason code explanation of 'Access denied'.

6.2.2 Unzoned Name Server objects – Formats

Unzoned Name Server objects are as defined in 5.1.2.

6.2.3 Reason code explanations

Unzoned Name Server reason code explanations are as defined in 5.1.3, with the following additional requirements. All reason code explantions listed in table 17 apply to the Unzoned Name Server, additional codes for the Unzoned Name Server are listed in table 186.

Table 186 – Additional Reason code explanations for Unzoned Name Server

Encoded value (hex)	Description
F0	Authorization Exception
F1	Authentication Exception

The use of these codes is further defined as follows:

– If an Unzoned Name Server request is rejected by the Name Server because the requestor is not authorized to make that request, then the Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation of ‘Authorization Exception’.

– If an Unzoned Name Server request is rejected by the Name Server because the requestor fails authentication, then the Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation of ‘Authentication Exception’.

{I DON’T HAVE ANY MATERIAL TO DEFINE AUTHENTICATION ETC. - ED}

6.2.4 Commands

The commands defined for the Unzoned Name Server are summarized in table 12, and are defined in 5.1.4.

6.3 Fabric Zone Server

Fabric Zoning provides a mechanism to expose selected views of Name Server information to Clients. This technique is similar to “virtual private networks” in that the fabric can group Fibre Channel address identifiers into Zones. The following discussion provides a brief architecutural overview of Zones, describes the parameters that define a Zone, and describes the relationships, and protocols for managing Zone information and configurations.

Administrators create Zones to increase network security, and prevent data loss or corruption, by controlling access between devices or user groups. Zoning can be specifically used to create:

– Barriers between devices that use different operating systems. It is often critical to separate servers and storage devices with different operating systems because accidental transfer of information from one to another can delete or corrupt data.

– Logical subsets of closed user groups. Administrators can authorize access rights to specific Zones for specific user groups, thereby protecting confidential data from unauthorized access.

– Groups of devices that are separate from devices in the rest of a fabric. Zoning allows certain processes to be performed on devices in a group without interrupting devices in other groups.

– Temporary access between devices for specific purposes. Administrators can remove zoning restrictions temporarily, then restore zoning restrictions to perform normal processes.

Zones can be configured through a switch vendors' managment tool, or via this Server.

A device can be a member of multiple Zones. Zone membership may be specified by:

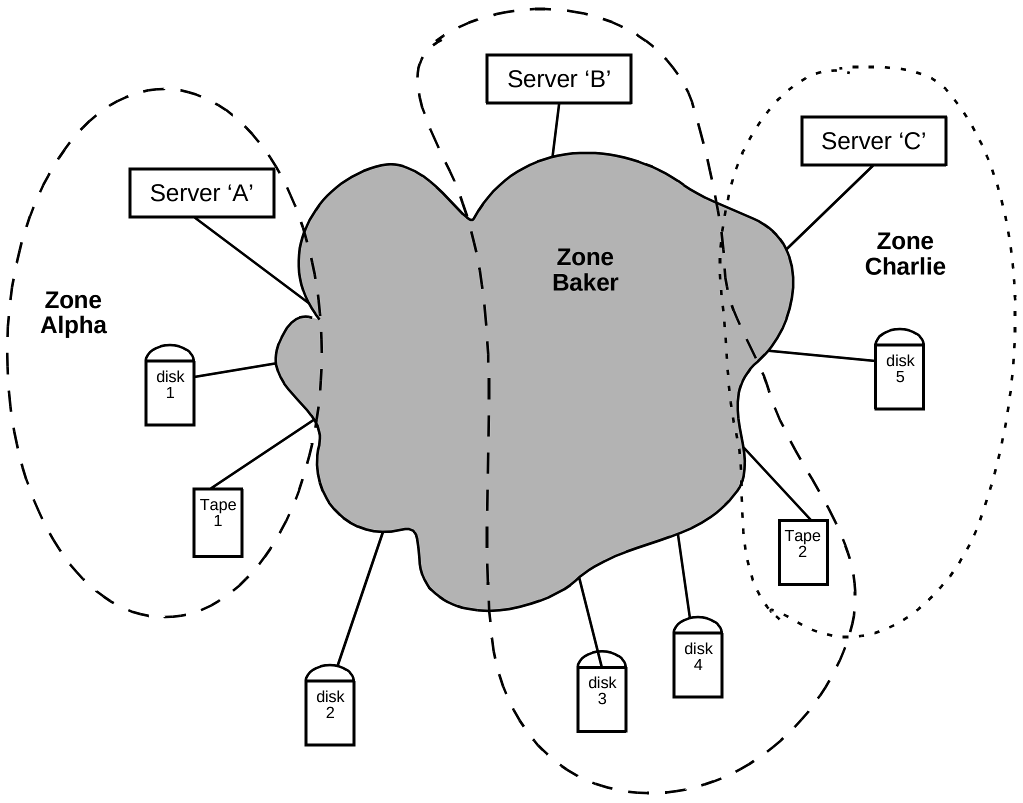
– The domain identification (Domain_ID) and physical port number of the Switch Port to which the device is attached; e.g., as "Domain 1, Port 1".

– The N_Port_Name assigned to the device connected to the switch.

– The N_Port address identifier assigned to the device during Fabric Login.

The Fabric Zone Server may be used to create a Zone by specifying the Zone Members. One or more Zones may be collected into a Zone Set. A Zone Set creates a collection of Zones which may be activated or deactivated as a single entity across all Switches in the Fabric. For example, you could have two Zone Sets, one for normal operation, and another for backup during off-hours. Note that only one Zone Set may be activated at one time.

The following is an example of an Active Zone Set used to restrict visibility amongst different operating system images.



	Zone Set			Zone Set
Zone Set Name	"OS_partition"			"Different_partition"
Zone Names	"Alpha"	"Baker"	"Charlie"	"Baker"
Zone Members	Server A	Server B	Server C	Server B
	disk 1	disk 3	disk 5	disk 3
	tape 1	tape 2	tape 2	tape 2
		disk 4		disk 4

Switches are linked through Inter-Switch Links (ISLs) to form multiswitch Fabrics. In a multiswitch Fabric, the Active Zone Set applies to the entire Fabric. Any change to the Zone Set definition, or to any Zone in the Zone Set, applies to all Switches in the Fabric.

When Fabrics attempt to join, participating Switches exchange Active Zone Set information and determine if their Active Zone Sets are compatible. If the Active Zone Sets are compatible, the Fabrics join. The resulting Active Zone Set is a single Zone Set containing Zone definitions from each Fabric. If the Active Zone Sets cannot merge, the E_Ports connecting the Fabrics become Isolated (see FC-SW, reference [6]). Zone definitions are compatible if there are no duplicate Domain_IDs, the Active

Zone Set name is the same for each Fabric, and Zones with the same names in each Fabric have identical members. {HUH??? how can different fabrics have identical members?}

6.3.1 Protocol

Fabric Zone Server registrations, activations, and queries are managed through protocols containing a set of Request CT_IUs and Response CT_IUs supported by the Fabric Zone Server.

Synchronous transactions shall be used, as defined in 4.5.1. For a Fabric Zone Server request, the payload shall be transported from the requestor to the Fabric Zone Server using a Request CT_IU. The corresponding Fabric Zone Server response is transported from the Fabric Zone Server to the requestor, in the Exchange established by the requestor, using a Response CT_IU.

6.3.1.1 CT_IU preamble values

The following values shall be set in the CT_IU preamble for Fabric Zone Server request and their responses; fields not specified here shall be set as defined in 4.3.1:

– Revision: hex ‘01’;

– GS_Subtype: as indicated in table 116;

– Command Code: see table 187 for Request command codes.

Table 187 – Fabric Zone Server – Request Command Codes

Code (hex)	Mnemonic & Description	Attribute(s) in Request CT_IU	Attribute(s) in Accept CT_IU
0100	GZC Get Zone Capabilities	none	[Zone capabilities] (see note 1)
0111	GZS Get Zone State	none	[Zone state] (see note 1)
0112	GZSN Get Zone Set List	none	List of Zone Set Name and Number of Zones
0113	GZSD Get Zone List	Zone Set Name	List of Zone Names and Number of Zone Members
0114	GZM Get Zone Member List	Zone Set Name; Zone Name	List of Zone Member Identifier Types and Zone Member Identifiers
0115	GAZS Get Active Zone Set	none	Zone Set Name, Number of Zones
0200	ADZS Register Zone Set	Zone Set Name; Number of Zones; List of Zone Names, Number of Zone Members, List of Zone Member Identifier Types and Zone Member Identifiers	none

Table 187 – Fabric Zone Server – Request Command Codes

Code (hex)	Mnemonic & Description	Attribute(s) in Request CT_IU	Attribute(s) in Accept CT_IU
0201	AZSD Register and Activate Zone Set	Zone Set Name; Number of Zones; List of Zone Names, Number of Zone Members, List of Zone Member Identifier Types and Zone Member Identifiers	none
0202	AZS Activate Zone Set	Zone Set Name	none
0203	DZS Deactivate Zone Set	none	none

0204	AZM
	Register Zone Member
		Zone Name;
		Zone Member Identifier Type;
		Zone Member Identifier
			none

0205	AZD Register Zone	Zone Set Name; Zone Name	none

0300	RZM
	Deregister Zone Member
		Zone Name;
		Zone Member Identifier Type;
		Zone Member Identifier
			none

0301	RZD Deregister Zone	Zone Set Name; Zone Name	none
	{Deregister Zone Set?}		
Other	Reserved		

Notes:
1 This is not an attribute.

6.3.1.2 Registration and Deregistration

Fabric Zone Server registrations are used to define Zones and Zone Sets within the Fabric. An entire Zone Set and its contained Zones and Zone Members may be defined in a single request; or, Zones and Zone Members may be registered and deregistered one at a time. The registration requests defined for the Fabric Zone Server are summarized in table 187.

The Fabric Zone Server may reject registrations due to Fabric Zone Server resource limitations. However, the Fabric Zone Server shall support registration of all attributes, once registration of a single attribute has been accepted for a given object.

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

{Do we need to say anything about overlapping registrations from multiple sources? - ed}

6.3.1.3 Activate and Deactivate

{we need to define these! if I activate A then later activate B, does this imply deactivate A? when I activate A and then deactivate A, does this disable zoning? how do I activate soft/hard/broadcast? - ed}

A change in the Active Zone set shall cause a Registered State Change Notification (RSCN) Extended Link Service to be originated to all Zone Members affected that are registered to receive RSCN.

6.3.1.4 Queries

The Fabric Zone Server may reject any query requests for reasons not specified in this document. The queries defined for the Fabric Zone Server are summarized in table 187.

6.3.2 Data Structures

The Fabric Zone Server object model is shown in figure 8.

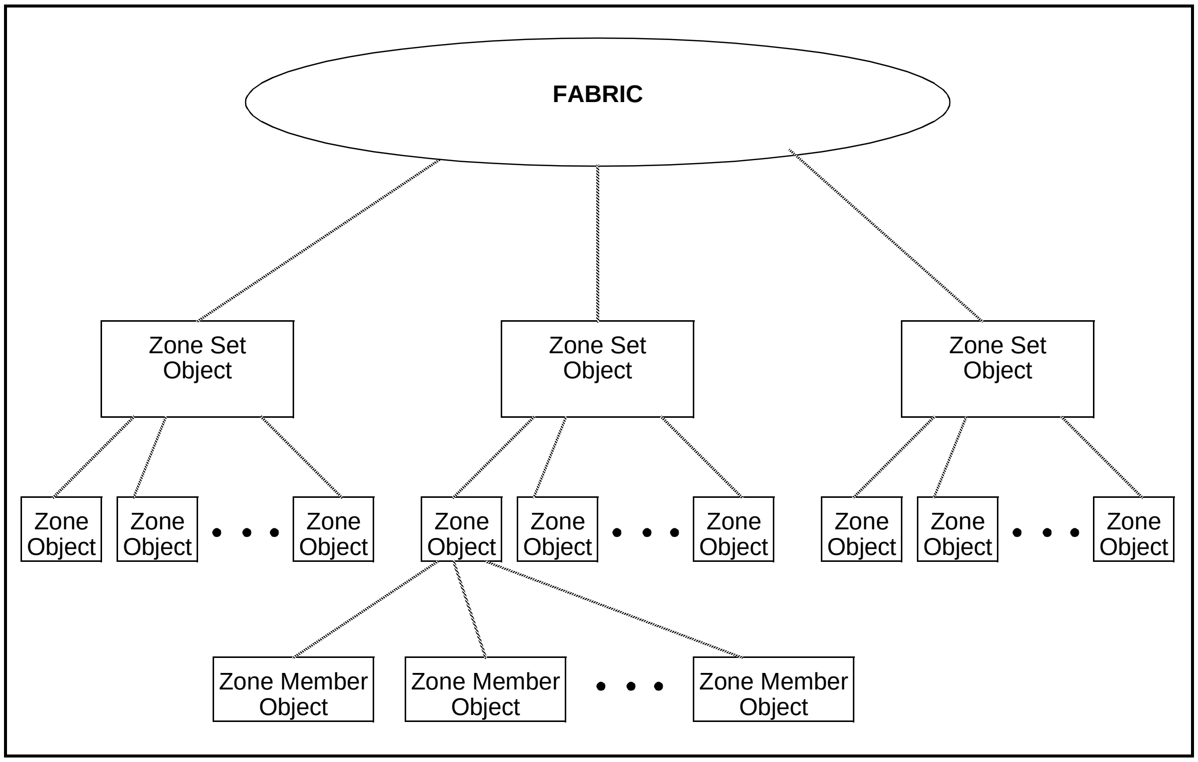


Figure 8 – Fabric Zone Server Object Model

The base object class managed by the Fabric Zone Server is the Zone Set object. One or more Zone Set objects belong to a Fabric. Zone Set objects have one or more associated Zone objects. Zone objects have one or more associated Zone Member objects. Zone Set objects, Zone objects, and Zone Member objects may have attributes associated with them, as shown in figure 9.

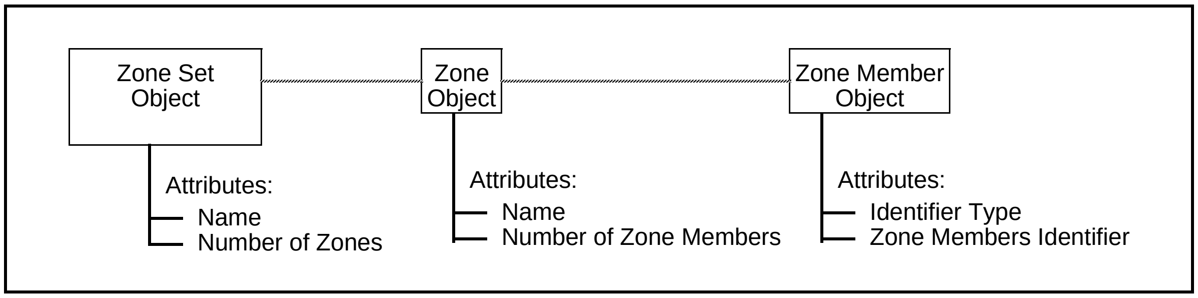


Figure 9 – Zone Set, Zone, and Zone Member attributes

6.3.2.1 Zone Set Object

Zone Set objects have the attributes described below.

6.3.2.1.1 Zone Set Name

The format of the Zone Set Name attribute, as used by the Fabric Zone Server, shall be as shown in table 188. The contents of these bytes are not defined and shall not be restricted by the Fabric Zone Server.

Table 188 – Zone Set Name Format

Item	Size (Bytes)
Zone Set Name length (m)	1
Zone Set Name	m
Reserved	63-m

This attribute may be defined using the protocol described in 6.3.1.2. The null value for the Zone Set Name attribute is a zero-length Zone Set Name.

6.3.2.1.2 Number of Zones

The format of the Number of Zones attribute, as used by the Fabric Zone Server, shall be as shown in table 189. The contents of these bytes are not defined and shall not be restricted by the Fabric Zone Server.

Table 189 – Number of Zones Format

Item	Size (Bytes)
Number of Zones	4

This attribute may be defined using the protocol described in 6.3.1.2. The null value for the Number of Zones attribute is hex ‘00 00 00 00’.

6.3.2.2 Zone Object

Zone objects have the attributes described below.

6.3.2.2.1 Zone Name

The format of the Zone Name attribute, as used by the Fabric Zone Server, shall be as shown in table 190. The contents of these bytes are not defined and shall not be restricted by the Fabric Zone Server.

Table 190 – Zone Name Format

Item	Size (Bytes)
Zone Name length (m)	1
Zone Name	m
Reserved	63-m

This attribute may be defined using the protocol described in 6.3.1.2. The null value for the Zone Name attribute is a zero-length Zone Name.

6.3.2.2.2 Number of Zone Members

The format of the Number of Zone Members attribute, as used by the Fabric Zone Server, shall be as shown in table 191. The contents of these bytes are not defined and shall not be restricted by the Fabric Zone Server.

Table 191 – Number of Zone Members Format

Item	Size (Bytes)
Number of Zone Members	4

This attribute may be defined using the protocol described in 6.3.1.2. The null value for the Number of Zone Members attribute is hex ‘00 00 00 00’.

6.3.2.3 Zone Member Object

Zone Member objects have the attributes described below.

6.3.2.3.1 Zone Member Identifier Type

The format of the Zone Member Identifier Type attribute, as used by the Fabric Zone Server, shall be as shown in table 192. This attribute establishes the format of the information contained in the Zone Member Identifier attribute.

Table 192 – Zone Member Identifier Type– encoding

Encoded value (hex)	**Description**
00	Null Zone Member Identifier
01	N_Port_Name
02	Domain_ID and Port
03	N_Port Address Identifer
all others	Reserved

This attribute may be defined using the protocol described in 6.3.1.2. The null value for the Zone Member Identifier Type attribute is shown in the table.

6.3.2.3.2 Zone Member Identifier

The format of the Zone Member Identifier attribute, as used by the Fabric Zone Server, is indicated by the Zone Member Identifier Type. For any Zone Member Identifier Type and format, the specific value of these bytes shall not be restricted by the Fabric Zone Server.

The Null Zone Member Identifier format shall be as shown in table 193.

Table 193 – Zone Member Identifier Format - Null value

Item	Size (Bytes)
hex ‘00’	8

The N_Port_Name Zone Member Identifier format shall be as shown in table 194.

Table 194 – Zone Member Identifier Format - N_Port_Name

Item	Size (Bytes)
N_Port_Name	8

The Domain_ID and Port Zone Member Identifier format shall be as shown in table 195.

Table 195 – Zone Member Identifier Format - Domain_ID and Port

Item	Size (Bytes)
Domain_ID	1
Port number	2
Reserved	5

The N_Port address identifier Zone Member Identifier format shall be as shown in table 195.

Table 196 – Zone Member Identifier Format - N_Port address identifier

Item	Size (Bytes)
Reserved	1
N_Port ID	3
Reserved	4

6.3.3 Reason code explanations

A Reject CT_IU (see 4.4.3) shall notify the requestor that the request has been unsuccessfully completed. The first error condition encountered shall be the error reported by the Reject CT_IU.

If a valid Fabric Zone 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 Fabric Zone Server request shall not be rejected with a Reason Code of “Command not supported”

If a Fabric Zone Server request is rejected with a reason code of ‘Unable to perform command request’, then one of the reason code explanations, shown in table 129, are returned.

Table 197 – Reject CT_IU Reason code explanations

Encoded value (hex)	**Description**
00	No additional explanation
01	Zones not supported
10	Zone Set Name unknown
11	No Zone Set active
12	Zone Name unknown
13	Zone State unknown
14	Incorrect payload length

Table 197 – Reject CT_IU Reason code explanations

Encoded value (hex)	Description
15	Activate Zone Set failed
16	Deactivate Zone Set failed
17	Request not supported
18	Capability not supported
19	Zone Member Identifier Type not supported
1A	Invalid Zone Set definition
Others	Reserved

The use of these codes is further defined as follows:

– If a Fabric Zone Server request is rejected by the Fabric Zone Server because of the identity of the requestor, then the Reject CT_IU reason code shall be ‘Unable to perform command request’, with a reason code explanation of ‘Access denied’. {?}

– Additional uses may be defined for specific Fabric Zone Server requests.

6.3.4 Commands

The commands defined for the Fabric Zone Server are summarized in table 187.

6.3.4.1 Query – Get Zone Capabilities (GZC)

The Fabric Zone Server shall, when it receives a GZC request, return the Zone capabilities supported by the Fabric. The format of the GZC Request CT_IU is shown in table 198.

Table 198 – GZC Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16

The format of the Accept CT_IU to a GZC request is shown in table 199.

Table 199 – Accept CT_IU to GZC Request

Item	Size (Bytes)
CT_IU preamble	16
Capability flags	1
Reserved	3
Vendor specific capabilities	4

The capability flags indicate the Zone capabilities supported by the Fabric, as shown in table 200.

Table 200 – Capability flags

Bit Position	Description
7	Soft Zones supported. When this bit is one, the Fabric supports Soft Zones.
6	Hard Zones supported. When this bit is one, the Fabric supports Hard Zones.
5	Broadcast Zones supported. When this bit is one, the Fabric supports Broad-cast Zones.
4-0	Reserved

The vendor specific capabilities field is not defined by this standard.

6.3.4.2 Query – Get Zone State (GZS)

The Fabric Zone Server shall, when it receives a GZS request, return the Zone capabilities supported by the Fabric. The format of the GZS Request CT_IU is shown in table 198.

Table 201 – GZS Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16

The format of the Accept CT_IU to a GZS request is shown in table 199.

Table 202 – Accept CT_IU to GZS Request

Item	Size (Bytes)
CT_IU preamble	16
Zone state	1
Reserved	3
Vendor specific state {?}	4

The Zone state indicates indicates whether the Fabric has an Active Zone Set, and the current enforcement of the Zone Set, as shown in table 200.

Table 203 – Capability flags

Bit Position	**Description**
7	Soft Zone Set Activated. When this bit is one, the Fabric currently has a Soft Zone Activated.
6	Hard Zone Set Activated. When this bit is one, the Fabric currently has a Hard Zone Activated.
5	Broadcast Zone Set Activated. When this bit is one, the Fabric currently has a Broadcast Zone Activated.
4-0	Reserved

The vendor specific state field is not defined by this standard. {I put this in for symmetry - is OK? - ed}

6.3.4.3 Query – Get Zone Set List (GZSN)

The Fabric Zone Server shall, when it receives a GZSN request, return the Zone Set attributes of all Zone Sets in the Fabric. The format of the GZSN Request CT_IU is shown in table 204.

Table 204 – GZSN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16

The format of the Accept CT_IU to a GZSN request is shown in table 205.

Table 205 – Accept CT_IU to GZSN Request

Item	Size (Bytes)
CT_IU preamble	16
Number of Zone Set attribute entries (n)	4
Zone Set Name #1	64
Number of Zones #1	4
Zone Set Name #2	64
Number of Zones #2	4
...	

Table 205 – Accept CT_IU to GZSN Request

Zone Set Name #n	64
Number of Zones #n	4

One or more Zone Set attribute entries are returned, and the Zone Set attribute entries may be returned in any order. Furthermore, the order may be different for every request even if the same Zone Set attribute entries are returned and the requestor is the same.

6.3.4.4 Query – Get Zone List (GZN)

The Fabric Zone Server shall, when it receives a GZN request, return the Zone attributes of all Zones in the specified Zone Set. The format of the GZN Request CT_IU is shown in table 206.

Table 206 – GZN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Zone Set Name	64

The format of the Accept CT_IU to a GZN request is shown in table 207.

Table 207 – Accept CT_IU to GZN Request

Item	Size (Bytes)
CT_IU preamble	16
Number of Zone attribute entries (n)	4
Zone Name #1	64
Number of Zone Members #1	4
Zone Name #2	64
Number of Zone Members #2	4
...	
Zone Name #n	64
Number of Zone Members #n	4

One or more Zone attribute entries are returned, and the Zone attribute entries may be returned in any order. Furthermore, the order may be different for every request even if the same Zone attribute entries are returned and the requestor is the same.

6.3.4.5 Query – Get Zone Member List (GZM)

The Fabric Zone Server shall, when it receives a GZM request, return the Zone Member attributes of all Zone Members in the specified Zone . The format of the GZM Request CT_IU is shown in table 208.

Table 208 – GZM Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Zone Set Name	64

The format of the Accept CT_IU to a GZM request is shown in table 209.

Table 209 – Accept CT_IU to GZM Request

Item	Size (Bytes)
CT_IU preamble	16
Number of Zone Member attribute entries (n)	4
Zone Member Identifier Type #1	1
Reserved	3
Zone Member Identifier #1	4
Zone Member Identifier Type #2	1
Reserved	3
Zone Member Identifier #2	4
...	
Zone Member Identifier Type #n	1
Reserved	3
Zone Member Identifier #n	4

One or more Zone Member attribute entries are returned, and the Zone Member attribute entries may be returned in any order. Furthermore, the order may be different for every request even if the same Zone Member attribute entries are returned and the requestor is the same.

6.3.4.6 Query – Get Active Zone Set (GAZS)

The Fabric Zone Server shall, when it receives a GAZS request, return the Zone Set attributes of the Active Zone Set. The format of the GAZS Request CT_IU is shown in table 204.

Table 210 – GAZS Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16

The format of the Accept CT_IU to a GAZS request is shown in table 205.

Table 211 – Accept CT_IU to GAZS Request

Item	Size (Bytes)
CT_IU preamble	16
Zone Set Name of Active Zone Set	64
Number of Zones in Active Zone Set	4

6.3.4.7 Register Zone Set (ADZS)

The AZSD Fabric Zone Server request shall be used to create a new Zone Set or add to an existing Zone Set {?}.

The Zone Set Name field of the Request CT_IU may be equal to a currently registered Zone Set Name. If the value of the Zone Set Name is equal to a currently registered Zone Set Name, the Fabric Zone Server shall add the specified Zones and Zone Members to the existing Zone Set. If the value of the Zone Set Name is not equal to a currently registered Zone Set Name, the Fabric Zone Server shall create a new Zone Set object and register the specified Zones and Zone Members to the new Zone Set. {true?}

Deregistration may be accomplished by {what?}.

The format of the ADZS Request CT_IU is shown in table 212.

Table 212 – ADZS Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Zone Set Name	64
Number of Zones	4
Number of Zone attribute entries (n)	4

Table 212 – ADZS Request CT_IU

Zone Name #1	64
Number of Zone Members #1	4
Number of Zone Member attribute entries (m)	4
Zone Member Identifier Type #1	1
Reserved	3
Zone Member Identifier #1	4
Zone Member Identifier Type #2	1
Reserved	3
Zone Member Identifier #2	4
...	
Zone Member Identifier Type #m	1
Reserved	3
Zone Member Identifier #m	4
Zone Name #2	64
Number of Zone Members #2	4
Number of Zone Member attribute entries (m)	4
Zone Member Identifier Type #1	1
Reserved	3
Zone Member Identifier #1	4
Zone Member Identifier Type #2	1
Reserved	3
Zone Member Identifier #2	4
...	
Zone Member Identifier Type #m	1
Reserved	3
Zone Member Identifier #m	4

Table 212 – ADZS Request CT_IU

...	
Zone Name #n	64
Number of Zone Members #n	4
Number of Zone Member attribute entries (m)	4
Zone Member Identifier Type #1	1
Reserved	3
Zone Member Identifier #1	4
Zone Member Identifier Type #2	1
Reserved	3
Zone Member Identifier #2	4
...	
Zone Member Identifier Type #m	1
Reserved	3
Zone Member Identifier #m	4

The format of the ADZS Accept CT_IU is shown in table 213.

Table 213 – ADZS Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6.3.4.8 Register and Activate Zone Set (AZSD)

The AZSD Fabric Zone Server request shall be used to create a new Zone Set or add to an existing Zone Set {?}, and then activate the specified Zone Set.

The Zone Set Name field of the Request CT_IU may be equal to a currently registered Zone Set Name. If the value of the Zone Set Name is equal to a currently registered Zone Set Name, the Fabric Zone Server shall add the specified Zones and Zone Members to the existing Zone Set. If the value of the Zone Set Name is not equal to a currently registered Zone Set Name, the Fabric Zone Server shall create a new Zone Set object and register the specified Zones and Zone Members to the new Zone Set. {true?}

Deregistration may be accomplished by {what?}.

The format of the AZSD Request CT_IU is shown in table 214.

Table 214 – AZSD Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Zone Set Name	64
Number of Zones	4
Number of Zone attribute entries (n)	4
Zone Name #1	64
Number of Zone Members #1	4
Number of Zone Member attribute entries (m)	4
Zone Member Identifier Type #1	1
Reserved	3
Zone Member Identifier #1	4
Zone Member Identifier Type #2	1
Reserved	3
Zone Member Identifier #2	4
...	
Zone Member Identifier Type #m	1
Reserved	3
Zone Member Identifier #m	4
Zone Name #2	64
Number of Zone Members #2	4
Number of Zone Member attribute entries (m)	4
Zone Member Identifier Type #1	1
Reserved	3
Zone Member Identifier #1	4
Zone Member Identifier Type #2	1

Table 214 – AZSD Request CT_IU

Reserved	3
Zone Member Identifier #2	4
...	
Zone Member Identifier Type #m	1
Reserved	3
Zone Member Identifier #m	4
...	
Zone Name #n	64
Number of Zone Members #n	4
Number of Zone Member attribute entries (m)	4
Zone Member Identifier Type #1	1
Reserved	3
Zone Member Identifier #1	4
Zone Member Identifier Type #2	1
Reserved	3
Zone Member Identifier #2	4
...	
Zone Member Identifier Type #m	1
Reserved	3
Zone Member Identifier #m	4

The format of the AZSD Accept CT_IU is shown in table 215.

Table 215 – AZSD Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6.3.4.9 Activate Zone Set (AZS)

The Fabric Zone Server shall, when it receives a AZS request, activate the specified Zone Set. The format of the AZS Request CT_IU is shown in table 216.

Table 216 – AZS Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Zone Set Name to Activate	64

The format of the Accept CT_IU to a AZS request is shown in table 217.

Table 217 – Accept CT_IU to AZS Request

Item	Size (Bytes)
CT_IU preamble	16

6.3.4.10 Deactivate Zone Set (DZS)

The Fabric Zone Server shall, when it receives a DZS request, deactivate the specified Zone Set. The format of the DZS Request CT_IU is shown in table 218.

Table 218 – DZS Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16

The format of the Accept CT_IU to a DZS request is shown in table 219.

Table 219 – Accept CT_IU to DZS Request

Item	Size (Bytes)
CT_IU preamble	16

6.3.4.11 Register Zone Member (AZM)

The AZM Fabric Zone Server request shall be used to add a Zone Member to an existing Zone.

The format of the AZM Request CT_IU is shown in table 220.

Table 220 – AZM Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Zone Name	64
Zone Member Identifier Type	1
Reserved	3
Zone Member Identifier	4

The format of the AZM Accept CT_IU is shown in table 221.

Table 221 – AZM Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6.3.4.12 Register Zone (AZD)

The AZD Fabric Zone Server request shall be used to add a Zone to an existing Zone Set.

The format of the AZD Request CT_IU is shown in table 222.

Table 222 – AZD Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Zone Set Name	64
Zone Name	64

The format of the AZD Accept CT_IU is shown in table 223.

Table 223 – AZD Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6.3.4.13 Deregister Zone Member (RZM)

The RZM Fabric Zone Server request shall be used to remove a Zone Member from an existing Zone.

The format of the RZM Request CT_IU is shown in table 224.

Table 224 – RZM Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Zone Name	64
Zone Member Identifier Type	1
Reserved	3
Zone Member Identifier	4

The format of the RZM Accept CT_IU is shown in table 225.

Table 225 – RZM Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

6.3.4.14 Deregister Zone (RZD)

The RZD Fabric Zone Server request shall be used to remove a Zone from an existing Zone Set.

The format of the RZD Request CT_IU is shown in table 222.

Table 226 – RZD Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Zone Set Name	64
Zone Name	64

The format of the RZD Accept CT_IU is shown in table 223.

Table 227 – RZD Accept CT_IU

Item	Size (Bytes)
CT_IU preamble	16

7 Time Service

The Time Service is provided to serve time information that is sufficient for managing expiration time. This Service is provided at the well-known address identifier, hex 'FFFFFB'.

7.1 Time Server

The functional model of the Time Server consists of primarily two entities:

- Time Service Client: the entity representing a user accessing the Time Service;
- Time Server: the entity that provides the time information through the Time Service.

There may be more than one distributed Time Server instance within the Fibre Channel network. However, from a user's perspective, the Time Service appears to come from the entity that is accessible at the Time Service well-known address identifier. If the Time Service is distributed, it shall be transparent to the Client. Figure 10 illustrates this model.

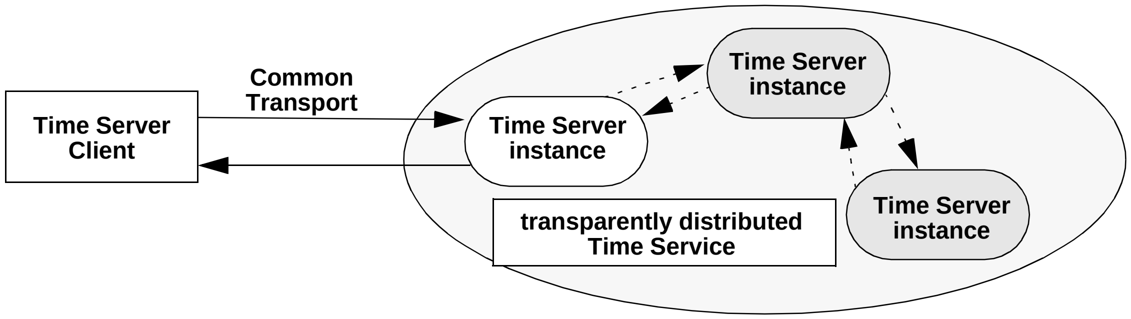


Figure 10 – Distributed model of Time Service

7.1.1 Time Server protocol

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.

Synchronous transactions shall be used. For a Time Server request, the Time Server payload shall be transported from the requestor to the Time Server using a Request CT_IU. 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 CT_IUs are associated with the Time Service protocol:

- Get_Time request (Request CT_IU);
- Get_Time response-accept (Accept CT_IU);
- Get_Time response-reject (Reject CT_IU).

7.1.1.1 CT_IU preamble values

The following values shall be set in the CT_IU preamble for Time Server requests and their responses; fields not specified here shall be set as defined in :

- Revision: hex’01’;
- GS_Type: hex’FB’ (Time Service);
- GS_Subtype: hex’01’ (Time Server);
- Command/Response Code: as defined below.

7.1.1.2 Distributed Time Server

If the Time Server is distributed, all instances of the Time Server shall be synchronized within an accuracy of ±2 seconds. The mechanism and protocol for time distribution and synchronization among multiple Time Server instances are currently not defined.

7.1.2 Data formats

No data formats have been defined for the Time Server.

7.1.3 Reason code explanations

No specific reason code explanations have been defined for the Time Server.

7.1.4 Commands

Time Server commands are described below.

7.1.4.1 Get_Time request

An Request CT_IU may be used by a Time Service client to request the time information from the Time Server. An Request CT_IU is sent from the client to the Time Server. A command code of hex ’00B1’ designates the Get_Time request. No additional information is defined for the Get_Time request; see table 228.

Table 228 – Get_Time Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16

7.1.4.2 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 Accept CT_IU, as shown in table 229.

Table 229 – Get_Time response - Accept CT_IU

Item	Size - Bytes
CT_IU preamble	16
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.

7.1.4.3 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 a Reject CT_IU to the requesting client.

The reason codes are described in 4.4.3.

{ ED. NOTE: The following comment against this clause in GS-2 was received from Edward A. Gardner, Ophidian Designs, and rejected:

“8. Time service. Again, is this used, if not, shouldn't it be declared obsolete? As specified, time servers appear to be unusable or unimplementable in multi-switch fabrics. There is no defined mechanism for initializing the time. It requires that distributed fabrics synchronize their time, yet the applicable projects (FC-SW, FC-SW-2) make no mention of such synchronization. A better time service would be the identical messages directed not at a well known address, but rather directed at any port in the FC network to determine that port's opinion of the current time.”

}

8 Alias Service

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

The Alias Service 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 Service operations is provided.

Table 230 defines the GS_Subtype codes for the Alias Service.

Table 230 – Alias Service subtype values

Values in hex	Description
01	Alias Server
other values	Reserved

8.1 Alias Server

Alias registration and de-registration are managed through protocols containing a set of request/reply IUs supported by the Alias Server.

8.1.1 Protocol

Synchronous transactions shall be used. For an Alias Server request, the Alias Server payload shall be transported from the requestor to the Alias Server using a Request CT_IU. 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.

8.1.1.1 CT_IU preamble values

The following values shall be set in the CT_IU preamble for Name Server request and their responses; fields not specified here shall be set as defined in 4.3.1:

– Revision: This field shall be set to hex'01';

– GS_Subtype: This field shall be set to hex'01';

– Command Code: see table 231 for Request command codes.

Table 231 – Alias Server Requests

Code (hex)	Mnem.	Description
0001	JNA	Join Alias Group
0002	RMA	Remove from Alias Group

Table 231 – Alias Server Requests

Code (hex)	Mnem.	Description
0003	LSN	Listen
0004	SLSN	Stop Listen
0005	RAG	Read Alias Group

8.1.1.2 Alias server functions

The following sections describe the functions performed by the Alias Server for each of the supported requests. Figure 11 illustrates the flow among the Originator N_Port, participating N_Ports, Alias Server, and Fabric Controller to create an Alias Group.

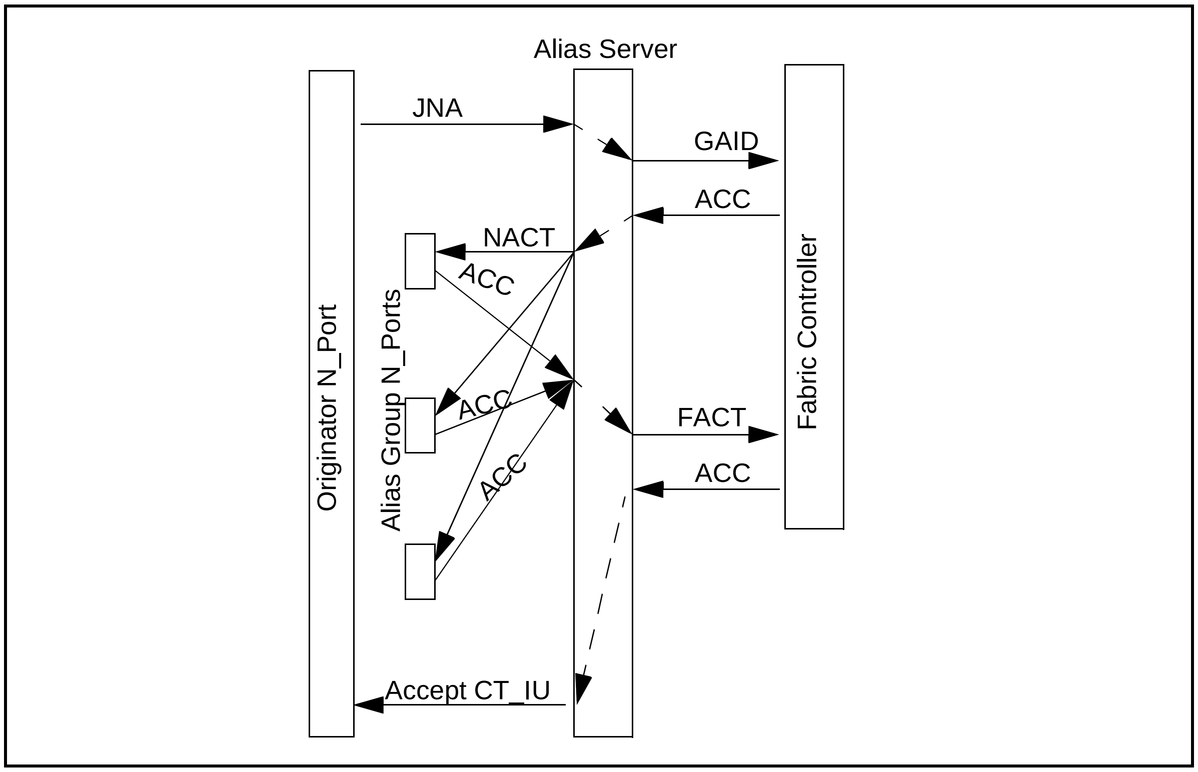


Figure 11 –Function flow

8.1.1.2.1 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 Accept CT_IU 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.

8.1.1.2.2 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 Accept CT_IU to indicate that the indicated N_Ports have been removed;

8.1.1.2.3 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 Accept CT_IU 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 Accept CT_IU 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 8.1.4.5, Membership, and 8.1.1.2.2g).

8.1.1.2.4 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 Accept CT_IU 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 Accept CT_IU 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 Accept CT_IU 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.

8.1.1.2.5 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 Accept CT_IU 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 Accept CT_IU 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.

8.1.1.3 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.

8.1.1.4 IPA Considerations

8.1.1.4.1 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.

8.1.1.4.2 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.

8.1.1.4.3 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.

8.1.2 Data Fields

Two common data fields for the Alias Server are defined here.

8.1.2.1 Alias_Token

Table 232 defines the format of the Alias_Token field .

Table 232 – 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 8.1.1.4 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 associated FC-4, as to the meaning of these values.

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

8.1.2.2 Authorization_Control

Table 233 defines the format of the Authorization_Control field .

Table 233 – 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 Authori-zation_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”.

8.1.3 Reason code explanations

A Reject CT_IU (see 4.4.3) shall notify the requestor that the request has been unsuccessfully completed. The The first error condition encountered shall be the error reported by the Reject CT_IU.

When a Reason Code of “Unable to perform command request” is generated, table 234 defines the various Reason Code Explanations.

NOTE – Reason Code Explanations hex ’30’ through hex ’38’ are identical to those defined for the Extended Link Services that 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 234 – Reject CT_IU 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

8.1.4 Commands

The Alias Server requests are summarized in table 231.

8.1.4.1 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 235.

Table 235 – JAG Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
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 Reject CT_IU 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 233.

Alias Group Token (Alias_Token): The Alias_Token defines the Alias Group being created. The request shall be rejected if the Alias_Token is invalid (Reason Code Explanation of “Invalid Alias_Token”), or the Alias Group specified in the Flags is not supported (Reason Code Explanation of “Unsupported Alias_Token”). The format of the Alias_Token is described in Table 232.

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 Accept CT_IU 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 Accept CT_IU shall be returned indicating that a valid Alias Group is being modified and that the Service Parameters are compatible.

In either case, the Accept CT_IU 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 Accept CT_IU payload is shown in table 236.

Table 236 – Accept CT_IU to JAG Request

Field	Size (Bytes)
CT_IU preamble	16
Alias_Token	12
Alias_ID	4
Alias_SP	80

Alias Group Token (Alias_Token): The Alias_Token defines the Alias Group that was just created. It is the same Alias_Token as was passed in the request. The format is described in table 232.

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.

A Reject CT_IU shall also be returned if an Alias_ID could not be assigned by the Fabric. The Reject CT_IU Reason Code Explanation indicates the appropriate Reason Code Explanation from table 234.

8.1.4.2 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 237.

Table 237 – RMA Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
Authorization_PW	12
Reserved	4
Alias_Token	12
NP_List_Length	4
NP_List(1)	4
NP_List (2 to n-1)	(n-2) x 4
NP_List(n)	4

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

Alias Group Token (Alias_Token): The Alias_ Token defines the Alias Group from which the N_Ports are to be deleted. The request shall be rejected if the Alias_Token is invalid (Reason Code Explanation of "Invalid Alias_Token"), or the Alias_Token does not exist (Reason Code Explanation of "Alias_Token does not exist"). The format of the Alias_Token is described in table 232.

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 Accept CT_IU is returned indicating that the N_Ports in the passed list have been deleted from the indicated Alias Group, as shown in table 238.

Table 238 – Accept CT_IU to RMA Request

Item	Size (Bytes)
CT_IU preamble	16

A Reject CT_IU shall also be returned if an Alias_ID could not be de-assigned by the Fabric. The Reject CT_IU Reason Code Explanation indicates the appropriate Reason Code Explanation from table 234.

8.1.4.3 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 239. .

Table 239 – LSN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
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 232.

When the Flags field indicates a Hunt Group, no further processing is performed and an Accept CT_IU 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 that 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 Accept CT_IU 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 Accept CT_IU payload is shown in table 240 .

Table 240 – Accept CT_IU to LSN Request

Item	Size (Bytes)
CT_IU preamble	16
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 232.

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

8.1.4.4 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 that 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 241.

Table 241 – SLSN Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
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 232.

When the Flags field indicates a Hunt Group, no further processing shall be performed and an Accept CT_IU 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 that 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 Accept CT_IU shall be returned indicating that the L_N_Port has been implicitly deleted from all Alias Groups of the same Alias_Class, as shown in table 242.

Table 242 – Accept CT_IU to SLSN Request

Item	Size (Bytes)
CT_IU preamble	16

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

8.1.4.5 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 241.

Table 243 – RAG Request CT_IU

Item	Size (Bytes)
CT_IU preamble	16
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 232. The request shall be rejected with a Reason Code Explanation of “Invalid Alias_Token” if the Alias_Token is invalid. An Accept CT_IU shall be returned containing a list of all the N_Port IDs in the associated Alias Group, if any.

The format of the Accept CT_IU payload is shown in table 244.

Table 244 – Accept CT_IU to RAG Request

Item	Size (Bytes)
CT_IU preamble	16
Alias_Token	12
Alias_SP	80
NP_List_Length	4

Table 244 – Accept CT_IU to RAG Request

Item	Size (Bytes)
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 232.

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 245.

Table 245 – 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.

A Reject CT_IU shall only be returned for an Invalid Alias_Token, as described above. If the Alias_Token does not define an existing Alias Group, the Accept CT_IU shall indicate an NP_List_Length of 0.

9 Key Distribution Service

The Key Distribution Service offers a mechanism for the secure distribution of secret encryption and/or authentication keys. Secure distribution is accomplished through the use of a Key Server that 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 that 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 that may use the secret keys distributed by this service.

This Service is provided through well-known address hex ’FFFFF7’.

9.1 Key Server

{9.1 needs updates to include authentication keys - ed}

The Key Server is modeled after IEEE 802.10 (see reference [9]). IEEE 802.10 is a center-based Key Exchange Technique with authentication.

The Key Server contains a copy of each Client’s unique distribution key. The 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 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 Key Server shall have a copy of the Client’s unique distribution key. The mechanism by which the distribution keys are distributed to the Clients and to the Key Server is beyond the scope of this document.

This service employs authentication techniques to assure the validity of all Key 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.

9.1.1 Protocol

Synchronous transactions shall be used, as defined in 4.5.1. Note that the protocol described below involves three entities: the Key Server, and two Clients which will be using the keys provided by the Key Server.

9.1.1.1 Terms

9.1.1.1.1 Encryption algorithm

An encryption algorithm (see figure 12) transforms private information into a form that is unreadable during public transport by anyone without a decryption key.

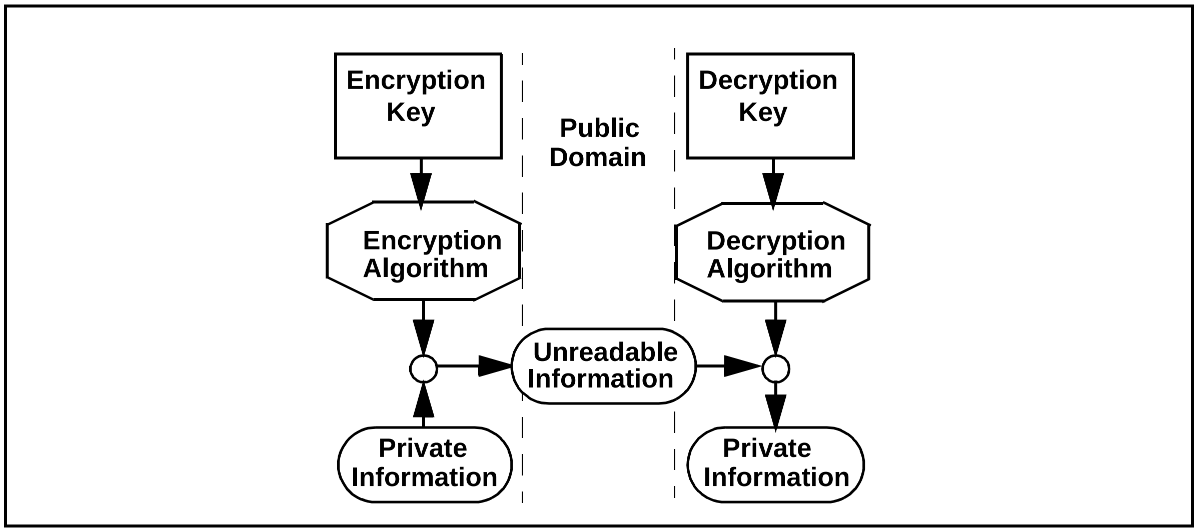


Figure 12 – Encryption Algorithm

9.1.1.1.2 Key encryption

Key encryption is used by the 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 Key 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.

9.1.1.1.3 Data encryption

Data encryption is used by a 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.

9.1.1.2 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*.

Table 246 shows the table format for data structures used within this clause.

Table 246 – Example Data Structure Format

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.

9.1.1.3 Communications model

The key distribution protocol involves a three phase process (see figure 13) consisting of a key request phase followed by a key exchange phase and a key authentication phase.

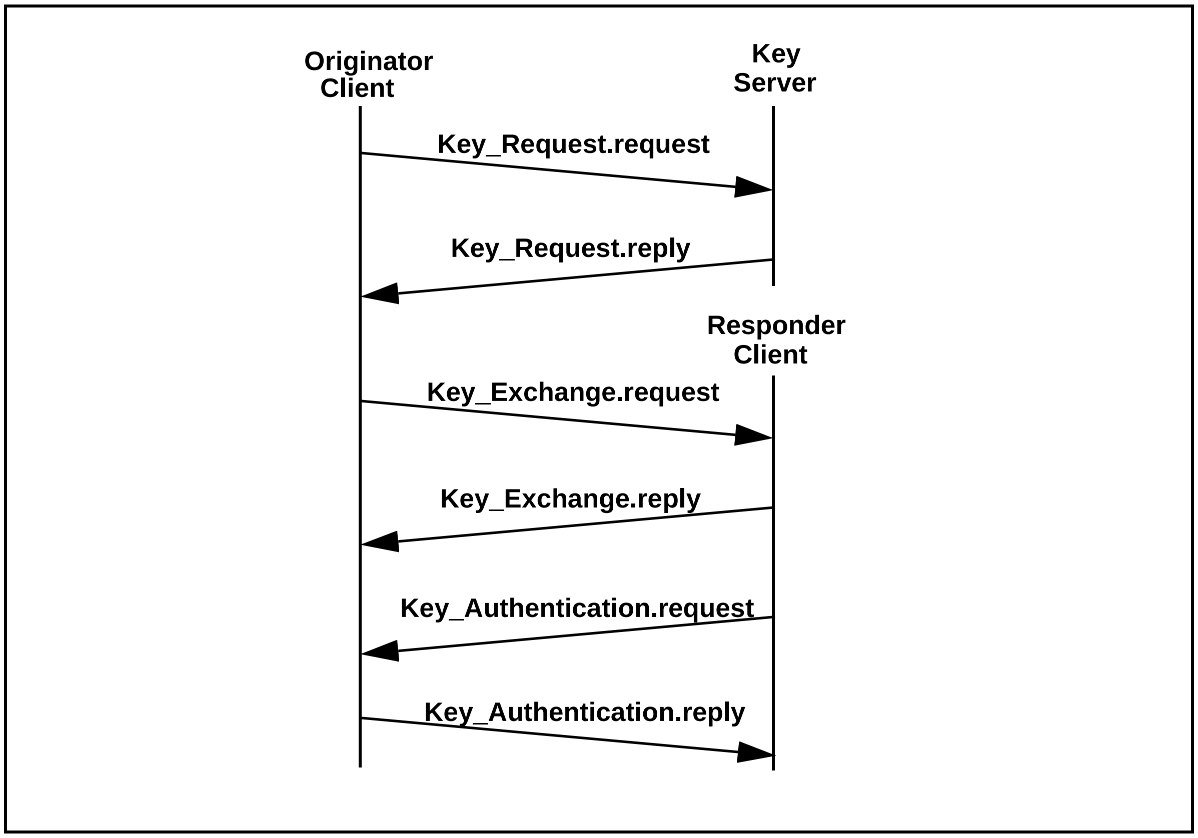


Figure 13 – Key Model

The key request phase request payload shall be transported from the requestor (Originator Client) to the Key Server using a Request CT_IU. The corresponding key request phase reply is transported from the Key Server to the Originator Client, in the Exchange established by the Originator Client, using a Response CT_IU.

The key exchange phase request payload shall be transported from the Originator Client to the Responder Client using a Request CT_IU. The corresponding key exchange phase reply payload is transported from the Responder Client to the Originator Client, in the Exchange established by the Originator Client, using a Response CT_IU.

The key authentication phase request payload shall be transported from the Responder Client to the Originator Client using a Request CT_IU. The corresponding key authentication phase reply payload is transported from the Originator Client to the Responder Client, in the Exchange established by the Responder Client, using a Response CT_IU.

NOTE – The second and third phases of this protocol are not directed at the Key Server, but rather are Client-to-Client Exchanges.

9.1.1.3.1 Key request phase

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 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 Key Server.

When the 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 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.

9.1.1.3.2 Key exchange phase

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.

9.1.1.3.2.1 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.

9.1.1.3.2.2 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.

9.1.1.3.3 Key authentication phase

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.

9.1.1.4 CT_IU preamble values

The following values shall be set in the CT_IU preamble for Key Server request and their responses; fields not specified here shall be set as defined in 4.3.1:

– Revision field: hex ‘01’;

– GS_Subtype: hex ‘00’;

– Command Code: see table 247 for request codes.

Table 247 – Key Server - Request Command Codes

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

9.1.1.5 Encryption Algorithm

The specification of encryption algorithms for key encryption and data encryption is beyond the scope of this document (see Annex B). The 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 248 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 248 – 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

9.1.2 Data Bases

The Key Server shall contain a distribution data base for each Client as defined in table 249.

Table 249 – Server distribution_key data base

Item	Size (bytes)
encryption_id	4
for each 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 Client:
  - the value of the Client’s address identifier;
  - the value of the Client’s distribution_key.

9.1.2.1 Distribution Key

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

Each Client shall contain a distribution data base as defined in table 250.

Table 250 – 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.

9.1.2.2 Secret Key

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

As defined in table 251, each Client shall contain a secret data base for each participating Client with which a secret_key was exchanged.

Table 251 – 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.

9.1.3 Reason Code Explanations

No specific reason code explanations have been defined for the Key Server.

9.1.4 Commands

Key Server commands are described below.

9.1.4.1 Request key (SKE_KEYREQ)

The key distribution request command shall be used by a Client to request a secret key from the Key Server.

The key request is performed as follows:

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

The format of the payload is shown in table 252.

Table 252 – SKE_KEYREQ Request CT_IU

Item	Size (bytes)
CT_IU preamble	16
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 key request includes the originator_id in the S_ID field described above.

A Reject CT_IU signifies rejection of the SKE_KEYREQ request.

An Accept CT_IU signifies that the Key Server has transmitted the secret_key information. The format of the accept payload is shown in table 253.

Table 253 – Accept CT_IU to SKE_KEYREQ Request

Item			Size (bytes)
CT_IU preamble			16
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).

9.1.4.2 Key exchange (SKE_KEYEXC)

The key exchange request 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 254.

Table 254 – SKE_KEYEXC Request CT_IU

Item	Size (bytes)
CT_IU preamble	16
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.

A Reject CT_IU signifies rejection of the SKE_KEYEXC request.

An Accept CT_IU signifies that the Responder Client has received the key exchange information from the Originator Client, as shown in table 255.

Table 255 – Accept CT_IU to SKE_KEYREQ Request

Item	Size (bytes)
CT_IU preamble	16

9.1.4.3 Key authentication (SKE_KEYAUT)

The key authentication request 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 256.

Table 256 – SKE_KEYAUT Request CT_IU

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).

A Reject CT_IU signifies rejection of the SKE_KEYAUT command.

An Accept CT_IU signifies that the Originator Client has received the authentication information from the Responder Client. The format of the accept payload is shown in table 257.

Table 257 – Accept CT_IU to SKE_KEYAUT Request

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).

Annex A: Service Interface Provided by FC-CT (Informative)

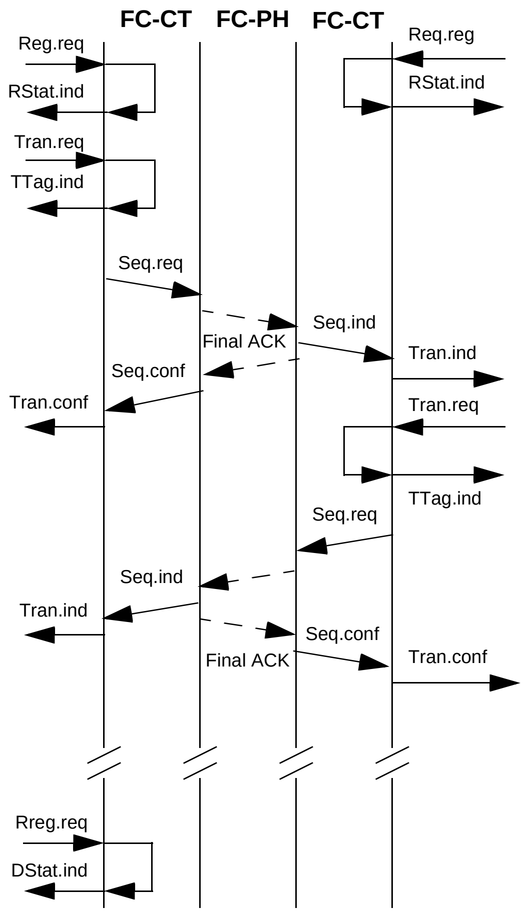
This annex specifies the services provided by FC-CT and the services required by FC-CT. The services provided by FC-CT are categorized as:

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. 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 Request-Tag 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 transac-tion 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, that 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: FC-4 Feature bits (Informative)

This clause documents proposed definitions for the FC-4 Feature object bits that can be retrieved using the GFF_ID and GID_FF Name Server Queries (see 5.1).

C.1 FCP-2

The NCITS T10 project FCP-2 has defined these bits as follows:

Table 258 – FCP-2 definition of FC-4 Feature bits

Bit	Interpretation
3	Reserved
2	Reserved
1	1 = FCP Initiator function supported 0 = not supported
0	1 = FCP Target function supported 0 = not supported

C.2 Others

{ TBD }

Annex D: Discovery (Informative)

This clause is {TBD Steve Wilson, Brocade}

.

Annex E: Future Work (Informative)

This clause documents proposed new work items for a future version of Generic Services. These work items were valid at publication time, but may change as work proceeds. Be sure to contact the T11 Chair for the latest status on these work items.

The work items are:

- Lock Services
- Additional CT sercurity features
- Additional Key Servers
- Additional Time Services