ISO/IEC 14165-432

Edition 1.0 2022-03

INTERNATIONAL STANDARD

colour inside

Information technology – Fibre channel – Part 432: Security Protocols – 2 (FC-SP-2)

ISO/IEC 14165-432:2022-03(en)

IEC Secretariat
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

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

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

IEC publications search - webstore.iec.ch/advsearchform
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee, …). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

IEC Products & Services Portal - products.iec.ch
Discover our powerful search engine and read freely all the publications previews. With a subscription you will always have access to up to date content tailored to your needs.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 300 terminological entries in English and French, with equivalent terms in 19 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

ISO/IEC 14165-432

Edition 1.0 2022-03

INTERNATIONAL STANDARD

Information technology – Fibre channel – Part 432: Security Protocols – 2 (FC-SP-2)

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ICS 35.200

ISBN 978-2-8322-1084-0

Contents Page

Figure **Page**

Table **Page**

INFORMATION TECHNOLOGY – FIBRE CHANNEL –

Part 432: Security Protocols - 2 (FC-SP-2)

FOREWORD

1. ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

2. The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC and ISO National bodies.

3. IEC and ISO documents have the form of recommendations for international use and are accepted by IEC and ISO National bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC and ISO documents is accurate, IEC and ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

4. In order to promote international uniformity, IEC and ISO National bodies undertake to apply IEC and ISO documents transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC and ISO document and the corresponding national or regional publication shall be clearly indicated in the latter.

5. IEC and ISO do not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC and ISO marks of conformity. IEC and ISO are not responsible for any services carried out by independent certification bodies.

6. All users should ensure that they have the latest edition of this document.

7. No liability shall attach to IEC and ISO or their directors, employees, servants or agents including individual experts and members of its technical committees and IEC and ISO National bodies for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this ISO/IEC document or any other IEC and ISO documents.

8. Attention is drawn to the Normative references cited in this document. Use of the referenced publications is indispensable for the correct application of this document.

9. Attention is drawn to the possibility that some of the elements of this ISO/IEC document may be the subject of patent rights. IEC and ISO shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 14165-432 has been prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology. It is an International Standard.

The text of this International Standard is based on the following documents:

Draft	Report on Voting
JTC1-SC25/2999/CDV	JTC1-SC25/3029/RVC

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1, available at www.iec.ch/members_experts/refdocs and www.iso.org/directives

A list of all parts in the ISO/IEC 14165 series, published under the general title *Information technology – Fibre Channel*, can be found on the IEC and ISO websites.

IMPORTANT – The "colour inside" logo on the cover page of this document indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

This standard describes the protocols used to implement security in a Fibre Channel Fabric. This standard includes the definition of protocols to authenticate Fibre Channel entities, protocols to set up session keys, protocols to negotiate the parameters required to ensure frame-by-frame integrity and confidentiality, and protocols to establish and distribute policies across a Fibre Channel Fabric.

INFORMATION TECHNOLOGY – FIBRE CHANNEL –

Part 432: Security Protocols - 2 (FC-SP-2)

1 Scope

This part of 14165 is one of the Fibre Channel family of standards. This standard describes the protocols used to implement security in a Fibre Channel Fabric. This standard includes the definition of protocols to authenticate Fibre Channel entities, protocols to set up session keys, protocols to negotiate the parameters required to ensure frame-by-frame integrity and confidentiality, and protocols to establish and distribute policies across a Fibre Channel Fabric.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ANSI INCITS 305-1998 (R2008), *SCSI Enclosures Services (SES).* Available at https://webstore.ansi.org/Standards/INCITS/INCITS3051998R2018

ANSI INCITS 305-1998/AM1-2000 (R2008), *SCSI Enclosures Services (SES) - Amendment 1.* Available at https://webstore.ansi.org/Standards/INCITS/INCITS3051998AM12000R2018

ANSI INCITS 461-2010, *Fibre Channel - Switch Fabric - 5 (FC-SW-5).* Available at https://webstore.ansi.org/Standards/INCITS/INCITS4612010

ANSI INCITS 463-2010, *Fibre Channel - Generic Services - 6 (FC-GS-6).* Available at https://webstore.ansi.org/Standards/INCITS/INCITS4632010

ANSI INCITS 470-2011, *Fibre Channel - Framing and Signaling - 3 (FC-FS-3).* Available at https://webstore.ansi.org/Standards/INCITS/INCITS4702011

ANSI INCITS 477-2011, *Fibre Channel - Link Services - 2 (FC-LS-2).* Available at https://webstore.ansi.org/Standards/INCITS/INCITS4772011

ANSI INCITS 487-2018, *Fibre Channel - Link Services - 3 (FC-LS-3).* Available at https://webstore.ansi.org/Standards/INCITS/INCITS4872018

INCITS TR-49-2012, *Fibre Channel - Device Attach - 2 (FC-DA-2).* Available at https://webstore.ansi.org/Standards/INCITS/INCITSTR492012R2017

RFC 1321, *The MD5 Message-Digest Algorithm*, April 1992. Available at http://www.ietf.org/rfc/rfc1321.txt

RFC 1994, *PPP Challenge Handshake Authentication Protocol (CHAP)*, August 1996. Available at http://www.ietf.org/rfc/rfc1994.txt

RFC 2104, *HMAC: Keyed-Hashing for Message Authentication*, February 1997. Available at http://www.ietf.org/rfc/rfc2104.txt

RFC 2246, *The TLS Protocol Version 1.0*, January 1999. Available at http://www.ietf.org/rfc/rfc2246.txt

RFC 2401, *Security Architecture for the Internet Protocol*, November 1998. Available at http://www.ietf.org/rfc/rfc2401.txt

RFC 2403, *The Use of HMAC-MD5-96 within ESP and AH,* November 1998. Available at http://www.ietf.org/rfc/rfc2403.txt

RFC 2404, *The Use of HMAC-SHA-1-96 within ESP and AH,* November 1998. Available at http://www.ietf.org/rfc/rfc2404.txt

RFC 2410, *The NULL Encryption Algorithm and Its Use With IPsec,* November 1998. Available at http://www.ietf.org/rfc/rfc2410.txt

RFC 2434, *Guidelines for Writing an IANA Considerations Section in RFCs,* October 1998. Available at http://www.ietf.org/rfc/rfc2434.txt

RFC 2437, PKCS #1: *RSA Cryptography Specifications Version 2,* October 1998. Available at http://www.ietf.org/rfc/rfc2437.txt

RFC 2451, *The ESP CBC-Mode Cipher Algorithms,* November 1998. Available at http://www.ietf.org/rfc/rfc2451.txt

RFC 2560, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP,* June 1999. Available at http://www.ietf.org/rfc/rfc2560.txt

RFC 2616, *Hypertext Transfer Protocol -- HTTP/1.1*, June 1999. Available at http://www.ietf.org/rfc/rfc2616.txt

RFC 2631, *Diffie-Hellman Key Agreement Method,* June 1999. Available at http://www.ietf.org/rfc/rfc2631.txt

RFC 2865, *Remote Authentication Dial In User Service (RADIUS),* June 2000. Available at http://www.ietf.org/rfc/rfc2865.txt

RFC 2945, *The SRP Authentication and Key Exchange System*, September 2000. Available at http://www.ietf.org/rfc/rfc2945.txt

RFC 3279, *Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,* April 2002. Available at http://www.ietf.org/rfc/rfc3279.txt

RFC 3526, *More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE),* May 2003. Available at http://www.ietf.org/rfc/rfc3526.txt

RFC 3602, *The AES-CBC Cipher Algorithm and Its Use with IPsec*, September 2003. Available at http://www.ietf.org/rfc/rfc3602.txt

RFC 3686, *Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP),* January 2004. Available at http://www.ietf.org/rfc/rfc3686.txt

RFC 3723, *Securing Block Storage Protocols over IP,* April 2004. Available at http://www.ietf.org/rfc/rfc3723.txt

RFC 3748, *Extensible Authentication Protocol (EAP),* June 2004. Available at http://www.ietf.org/rfc/rfc3748.txt

RFC 3852, *Cryptographic Message Syntax (CMS),* July 2004. Available at http://www.ietf.org/rfc/rfc3852.txt

RFC 3986, *Uniform Resource Identifier (URI): Generic Syntax*, January 2005. Available at http://www.ietf.org/rfc/rfc3986.txt

RFC 4072, *Diameter Extensible Authentication Protocol (EAP) Application,* August 2005. Available at http://www.ietf.org/rfc/rfc4072.txt

RFC 4086, *Randomness Requirements for Security,* June 2005. Available at http://www.ietf.org/rfc/rfc4086.txt

RFC 4106, *The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP),* June 2005. Available at http://www.ietf.org/rfc/rfc4106.txt

RFC 4303, *IP Encapsulating Security Payload (ESP),* December 2005. Available at http://www.ietf.org/rfc/rfc4303.txt

RFC 4434, *The AES-XCBC-PRF-128 Algorithm for the Internet Key Exchange Protocol (IKE),* February 2006. Available at http://www.ietf.org/rfc/rfc4434.txt

RFC 4543, *The Use of Galois Message Authentication Code (GMAC) in IPsec ESP and AH,* May 2006. Available at http://www.ietf.org/rfc/rfc4543.txt

RFC 4346, *The Transport Layer Security (TLS) Protocol Version 1.1*, April 2006. Available at http://www.ietf.org/rfc/rfc4346.txt

RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2*, August 2008. Available at http://www.ietf.org/rfc/rfc5246.txt

RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,* May 2008. Available at http://www.ietf.org/rfc/rfc5280.txt

RFC 5433, *Extensible Authentication Protocol - Generalized Pre-Shared Key (EAP-GPSK) Method,* February 2009. Available at http://www.ietf.org/rfc/rfc5433.txt

RFC 5996, *Internet Key Exchange Protocol Version 2 (IKEv2),* September 2010. Available at http://www.ietf.org/rfc/rfc5996.txt

RFC 6614, *TLS encryption for RADIUS,* May 2012. Available at http://www.ietf.org/rfc/rfc6614.txt

RFC 6818, *Updates to the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile,* January 2013. Available at http://www.ietf.org/rfc/rfc6818.txt

SRP-6, *Improvements and Refinements to the Secure Remote Password Protocol,* Submission to the IEEE P1363 Working Group, Oct 2002. Available at http://srp.stanford.edu

X.509v3, ITU-T Recommendation X.509 (1997 E), *Information Technology - Open Systems Interconnection - The Directory: Authentication Framework,* June 1997. Available at https://www.itu.int/ITU-T/recommendations

X.501, ITU-T Recommendation X.501, *Information Technology - Open Systems Interconnection - The Directory: Models*, 1993. Available at https://www.itu.int/ITU-T/recommendations

FIPS PUB 140-2, *Security Requirements for Cryptographic Modules,* May 2001. Available at https://csrc.nist.gov/publications/detail/fips/140/2/final

FIPS PUB 180-4, Secure Hash Standard (SHS), March 2012. Available at https://csrc.nist.gov/publications/detail/fips/180/4/final

FIPS PUB 197, *Advanced Encryption Standard (AES),* November 2001. Available at https://csrc.nist.gov/publications/detail/fips/197/final

FIPS PUB 198, *The Keyed-Hash Message Authentication Code (HMAC),* March 2002. Available at https://csrc.nist.gov/csrc/media/publications/fips/198/archive/2002-03-06/documents/fips-198a.pdf

NIST SP 800-57 Part 3, *Recommendation for Key Management – Part 3: Application-Specific Key Management Guidance*, December 2009. Available at https://csrc.nist.gov/publications/detail/sp/800-57-part-3/archive/2009-12-28

NIST SP 800-131A, *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths,* January 2011. Available at https://csrc.nist.gov/publications/detail/sp/800-131a/archive/2011-01-13

KMIP Spec, *Key Management Interoperability Protocol Specification Version 1.0*, June 2010. Available at http://docs.oasis-open.org/kmip/ug/v1.0/kmip-ug-1.0.pdf

KMIP Profiles, *Key Management Interoperability Protocol Profiles Version 1.0*, June 2010. Available at http://docs.oasis-open.org/kmip/profiles/v1.0/cs01/kmip-profiles-1.0-cs-01.pdf

3 Terms, definitions, symbols, abbreviated terms, and conventions

3.1 Terms and definitions

For the purposes of this document the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org

- ISO Online browsing platform: available at http://www.iso.org/obp

3.1.1
Access Control
security service that prevents unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner

3.1.2
address identifier
address value used to identify source (S_ID) or destination (D_ID) of a frame

Note 1 to entry: See FC-FS-3.

3.1.3
Anti-replay
security service that prevents processing of duplicate frames

3.1.4
Ascending order
sorting order in which each entry is positioned in accord with the value of its key(s) to precede all entries with keys of greater value

3.1.5
Authentication
process that verifies an identity

3.1.6
Authentication Initiator
entity initiating an Authentication Transaction

3.1.7
Authentication Protocol
protocol to perform Authentication

3.1.8
Authentication Responder
responding entity in an Authentication Transaction

3.1.9
Authentication Transaction
instance of an Authentication Protocol

3.1.10
Authorization
process that verifies that an entity is permitted to perform an action

3.1.11
Autonomous Switch
Switch that retains its own per Switch Policy Objects, all Fabric-wide Policy Objects, and all Node to Node (Zoning) information (see 4.8.1)

3.1.12
B_Port
Fabric inter-element port used to connect Bridge devices with E_Ports on a Switch

Note 1 to entry: The B_Port provides a subset of the E_port functionality.

Note 2 to entry: See FC-SW-5.

3.1.13
Bridge
device that encapsulates/de-encapsulates Fibre Channel frames within another protocol

Note 1 to entry: See FC-FS-2.

3.1.14
Certificate
data structure that cryptographically binds an identity to a public key

3.1.15
Certificate Revocation List
list of revoked Certificates issued by a specific CA

3.1.16
Certification Authority
organization or a function that creates, issues and manages Certificates

3.1.17
Child_SA
Security Association used to protect FC-2 frames or CT_IUs

Note 1 to entry: Its parent is an IKE_SA.

3.1.18
Client Switch
Switch that retains its per Switch Policy Objects, all Fabric-wide Policy Objects and the subset of the Node to Node (Zoning) information relevant for its operations (see 4.8.1)

3.1.19
Compliance Element
subset of the features defined in this standard

Note 1 to entry: See Annex A.

3.1.20
Confidentiality
security service that protects data from unauthorized disclosure

3.1.21
Cryptographic Integrity
security service that ensures integrity by using cryptographic techniques

3.1.22
Data Origin Authentication
security service that verifies the identity of the claimed source of data

3.1.23
E_Port
Fabric “Expansion” Port that attaches to another Interconnect_Port to create an Inter-Switch Link

Note 1 to entry: See FC-SW-5.

3.1.24
Encryption
security mechanism used to transform data from an intelligible form (plaintext) into an unintelligible form (ciphertext), in order to provide confidentiality

Note 1 to entry: The inverse transformation process is called decryption.

3.1.25
entity
something that may communicate using a Fibre Channel topology and has an identity that distinguishes it for the purpose of applying the security features specified in this standard

3.1.26
Ephemeral key
nonce used as an intermediate key within a cryptographic protocol

3.1.27
ESP_Header
optional Header defined in FC-FS-3

3.1.28
Exchange
unit of protocol activity that transfers information between a specific Originator Nx_Port and specific Responder Nx_Port using one or more related non-concurrent Sequences that may flow in the same or opposite directions

Note 1 to entry: See FC-FS-3.

3.1.29
exchange (noun)
pair of related messages one of which is a response to the other one (used in clause 6)

3.1.30
Fabric
entity that interconnects Nx_Ports attached to it and is capable of routing frames by using the D_ID information in a FC-2 frame header

Note 1 to entry: See FC-FS-3.

3.1.31
F_Port
Link Control Facility within the Fabric that attaches to an N_Port through a link

Note 1 to entry: An F_Port is addressable by the N_Port attached to it, with a common well-known address identifier (FFFFFEh).

Note 2 to entry: See FC-FS-3.

3.1.32
F_Port_Name
Name_Identifier associated with an F_Port

Note 1 to entry: See FC-FS-3.

3.1.33
FC-SP Compliance
set of Compliance Elements that are required to be implemented in order to claim compliance with this standard

Note 1 to entry: See 4.2.

3.1.34
FC-SP Zoning
variant of zoning defined in this standard (see 7.6)

3.1.35
Fx_Port
Switch port capable of operating as an F_Port or FL_Port

Note 1 to entry: See FC-FS-3.

3.1.36
IKE_SA
Security Association used to protect the messages used by the FC SA management protocol, in order to negotiate Child_SAs

3.1.37
Integrity
service that enables detection of modifications to data

3.1.38
Internet Key Exchange
security protocol used for the management of Security Associations in IP networks

Note 1 to entry: The FC SA Management protocol is based on version 2 of IKE, called IKEv2.

3.1.39
Key
value that controls the operation of a cryptographic algorithm

3.1.40
Local Fx_Port
Fx_Port to which an Nx_Port is directly attached by a link or an Arbitrated Loop

Note 1 to entry: See FC-FS-3.

3.1.41
Log (noun)
one or more collections of information concerning events or conditions that have occurred and may be of interest to security administrator(s)

Note 1 to entry: The mechanism used to maintain a log or logs is vendor specific.

Note 2 to entry: A log may contain sensitive security information and should be appropriately protected.

3.1.42
Log (verb)
to insert information about a specific event or condition into a log

3.1.43
Name_Identifier
64-bit identifier, with a 60-bit value preceded by a 4-bit Network_Address_Authority Identifier, used to identify entities in Fibre Channel

Note 1 to entry: See FC-FS-3.

3.1.44
Node
collection of one or more Nx_Ports controlled by a level above FC-2

Note 1 to entry: See FC-FS-3.

3.1.45
Node_Name
Name_Identifier associated with a Node

Note 1 to entry: See FC-FS-3.

3.1.46
Nonce
unpredictable random value used only for a single instance or invocation of a cryptographic algorithm or protocol

3.1.47
N_Port
hardware entity that includes a Link Control Facility but not Arbitrated Loop functions associated with Arbitrated Loop topology, and has the ability to act as an Originator, a Responder, or both

Note 1 to entry: Well-known addresses are considered to be N_Ports.

Note 2 to entry: See FC-FS-3.

3.1.48
N_Port_Name
Name_Identifier associated with an Nx_Port

Note 1 to entry: See FC-FS-3.

3.1.49
Nx_Port
port capable of operating as an N_Port or Public NL_Port, but not as a Private NL_Port

Note 1 to entry: See FC-FS-3.

3.1.50
Online Certificate Status Protocol
protocol for online verification of Certificate validity

Note 1 to entry: See RFC 2560.

3.1.51
Password
user generated value known to a limited group of entities

Note 1 to entry: Password based Security Protocols use an entity's knowledge of a password to establish security properties and privileges for that entity.

3.1.52
Perfect Forward Secrecy
security service that ensures that the compromise of a single key does not enable access to data protected by other keys

3.1.53
Printable ASCII characters
ASCII characters in the range 20h through 7Eh

3.1.54
Private Key
in asymmetric cryptography the element of a public-private key pair that shall be kept secret

3.1.55
Public Key
in asymmetric cryptography the element of a public-private key pair that may be made public

3.1.56
RADIUS Server
entity providing the security services defined in RFC 2865

3.1.57
Root Certificate
Certificate for a key that a Certification Authority uses to sign the Certificates that it issues

3.1.58
Secret
value known to a limited group of entities and generated with sufficient randomness to be computationally intractable to guess

Note 1 to entry: Secret based Security Protocols use an entity's knowledge of a secret to establish security properties and privileges for that entity.

3.1.59
SA_Initiator
entity initiating an SA Management Transaction

3.1.60
SA Management Protocol
protocol to perform management of Security Associations

3.1.61
SA Management Transaction
instance of a SA Management Protocol

3.1.62
SA Proposal
Proposal
set of security parameters proposed to a peer in the process of negotiating an SA

3.1.63
SA_Responder
responding entity in an SA Management Transaction

3.1.64
Salt
random value associated with a password to increase the difficulty of cryptographic attacks

3.1.65
Security Association
uni-directional logical connection created by the SA Management protocol to provide security processing

Note 1 to entry: All traffic belonging to an SA has the same security processing applied to it.

3.1.66
Security Association Database
database maintaining Security Associations

3.1.67
Security Parameters Index
value used by a receiver to identify the SA to which an incoming frame, CT_IU or FC SA message belongs

3.1.68
security relationship
relationship based on shared state among two or more entities that allows the communication among them to be protected by one or more security services

3.1.69
Server Switch
Switch that retains all Policy Objects and all Node to Node (Zoning) information (see 4.8.1)

3.1.70
Switch
member of the Fabric collective

Note 1 to entry: See FC-SW-5.

3.1.71
Switch_Name
Name_Identifier that identifies a Switch or a Bridge device for identification purposes

Note 1 to entry: See FC-SW-5.

3.1.72
T10 Vendor ID
character string that uniquely identifies a vendor (see 3.5).

3.1.73
well-known address
set of address identifiers defined to access Fabric and other functions (e.g., a name server)

Note 1 to entry: See FC-FS-3.

3.1.74
word
when used to indicate a size, 32 contiguous bits

3.2 Symbols and abbreviated terms

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

=	is equal to
#	number
||	concatenation symbol (e.g., A || B represents the concatenation of A and B)
ACA	Acquire Change Authorization SW_ILS (see FC-SW-5)
AES	Advanced Encryption Standard (see FIPS PUB 197)
CA	Certification Authority
CHAP	Challenge Handshake Authentication Protocol (see RFC 1994)
CRL	Certificate Revocation List
CT	Common Transport (see FC-GS-6)
CT_IU	Common Transport Information Unit (see FC-GS-6)
DH	Diffie-Hellman
DH-CHAP	Diffie-Hellman augmented CHAP
D_ID	Destination address identifier (see FC-FS-3)
ELS	Extended Link Service (see FC-LS-2)
HMAC	Keyed-Hashing for Message Authentication (see RFC 2104)
IANA	Internet Assigned Numbers Authority (see RFC 2434)
ICV	Integrity Check Value
IKEv2	Internet Key Exchange (IKEv2) Protocol (see RFC 5996)
IP	Internet Protocol (see RFC 791 and RFC 2460)

ISO	International Organization for Standardization
ITU	International Telecommunication Union
LS_ACC	Link Service Accept (see FC-LS-2)
LS_RJT	Link Service Reject (see FC-LS-2)
MAC	Message Authentication Code
MD5	Message Digest 5 (see RFC 1321)
mod	The modulus function
OCSP	Online Certificate Status Protocol
OID	Object Identifier (see RFC 5280)
prf	Pseudo Random Function
RADIUS	Remote Authentication Dial In User Service (see RFC 2865)
RCA	Remove Change Authorization SW_ILS (see FC-SW-5)
SA	Security Association
SADB	Security Association Database
SHA-1	Secure Hash Algorithm (see ANSI X9.30.2-1997)
SFC	Stage Fabric Configuration SW_ILS (see FC-SW-5)
SPI	Security Parameter Index
SSB	Server Session Begin CT Request (see FC-GS-6)
SSE	Server Session End CT Request (see FC-GS-6)
SW_ILS	Switch Internal Link Service (see FC-SW-5)
SW_ACC	Switch Fabric Link Service Accept (see FC-SW-5)
SW_RJT	Switch Fabric Link Service Reject (see FC-SW-5)
S_ID	Source address identifier (see FC-FS-3)
SRP	Secure Remote Password (see RFC 2945)
T_ID	Transaction Identifier
TCP	Transmission Control Protocol (see RFC 793)
UDP	User Datagram Protocol (see RFC 768)
UFC	Update Fabric Configuration SW_ILS (see FC-SW-5)
UTC	Universal Time Code

3.3 Editorial conventions

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

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

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 is 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.

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.

When the value of a bit or field is not relevant, x or xx appears in place of a specific value.

Unless stated otherwise, numbers that are not immediately followed by lower-case b or h are decimal values, numbers immediately followed by lower-case b (xxb) are binary values, and numbers or upper case letters immediately followed by lower-case h (xxh) are hexadecimal values.

A numeric range is indicated by listing the two extremes separated by ".." (e.g., "1 .. 6" indicates the range from 1 to 6, including 1 and 6).

Optional parameters are enclosed in square brackets (e.g., [X] indicates that X is an optional parameter), except in the clause 7 policy enforcement definitions, where square brackets enclose restricted identifiers (e.g., [N(α)] indicates a restriction on access for the Switch with Node_Name α. See table 141 for additional examples).

The notation SK { ... } is used in clause 6 to indicate encrypted and integrity protected IKE Payloads (see 6.1.2).

3.4 Keywords

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

3.4.2 ignored: When speaking of a bit, byte, word, field, or code value, this keyword indicates that the object is unused. The contents or value of an ignored bit, byte, word, field or code value shall not be examined by the receiving device and may be set to any value by the transmitting device. When speaking of a protocol step or event, this keyword indicates that the recipient of the protocol step or event shall take no action.

3.4.3 invalid: A keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field or code value shall be reported as an error.

3.4.4 Invocable: A keyword referring to a feature or parameter that is required to be implemented by an entity to which a request may be sent, but it is not required to be used by a requesting entity.

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

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

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

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

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

3.4.10 prohibited: A keyword referring to a feature that shall not be used between entities compliant with this standard.

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

3.4.12 restricted: A keyword referring to bits, bytes, words, and fields that are set aside for use in other Fibre Channel standards. A restricted bit, byte, word, or field shall be treated as a reserved bit, byte, word or field for the purposes of the requirements defined in this standard.

3.4.13 shall: A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

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

3.4.15 x or **xx:** The value of the bit or field is not relevant.

3.5 T10 Vendor ID

A T10 Vendor ID shall be a string of one to eight characters that is recorded in an informal list of Vendor IDs maintained by INCITS Technical Committee T10 (see http://www.t10.org).

A field described as containing a T10 Vendor ID shall contain the first character of the T10 Vendor ID in the first byte of the field, and successive characters of the T10 Vendor ID in successive bytes of the field. Any bytes of the field not filled by characters of the T10 Vendor ID shall be filled with ASCII space characters (20h).

3.6 Sorting

3.6.1 Sorting alphabetic keys

An alphabetic key is an ordered series of bytes containing printable ASCII characters. Alphabetic keys are sorted by comparison of the values in their corresponding bytes (i.e., bytes at the same offset within the key):

a) one alphabetic key is equal to another alphabetic key if the two keys are the same length and each byte of the one is equal to the corresponding byte of the other; and

b) one alphabetic key is greater than another alphabetic key if:

   A) the first byte of the one alphabetic key in which the two keys differ is greater than the corresponding byte in the other alphabetic key; or

   B) there are no corresponding bytes of the two alphabetic keys that differ, but the one alphabetic key is longer than the other alphabetic key.

NOTE 1 – Unlike dictionary sorting, alphabetic key sorting compares the actual ASCII values of alphabetic characters of different case (i.e., a lower case character is not equal to its corresponding upper case character).

3.6.2 Sorting numeric keys

A numeric key is an unsigned integer of arbitrary size. Numeric keys are sorted by numeric comparison of their values.

3.7 Terminate communication

This standard often uses language of the form “If the Authentication fails the entity shall terminate the communication.” This means that any further communication between the two involved entities is forbidden. More specifically, when an Authentication Transaction occurs in specific states of a state machine, the meaning is as follows:

a) When an Authentication Transaction is performed between E_Ports in state P17a (see 8.9.1) or in state P22 (see 8.9.3), terminate communication means going in state P16 (Invalid Attachment).

b) When an Authentication Transaction is performed between an E_Port and a B_Port in state P19 (see 8.9.2), terminate communication means going in state P16 (Invalid Attachment).

c) When an Authentication Transaction is performed between VE_Ports in state P24(k)a (see 8.9.3), terminate communication means going in state P26(k) (Logical Isolation).

d) When an Authentication Transaction is performed between two Domain Controller addresses (see FC-SW-5), terminate communication means that no further communication occurs between the involved Domain Controller addresses.

e) When an Authentication Transaction is performed between an N_Port or a VN_Port and another N_Port or VN_Port, terminate communication means that no further communication occurs between the two entities.

f) When an Authentication Transaction is performed between an N_Port and an F_Port, terminate communication means that no further communication occurs between the involved FC_Ports (the situation is equivalent to a Fabric Logout).

g) When an Authentication Transaction is performed between a VN_Port and a VF_Port, terminate communication means that no further communication occurs between the involved Virtual FC_Ports (the situation is equivalent to a Fabric Logout in that particular Virtual Fabric).

3.8 State machine notation

State machines in this standard use the style shown in figure 1.

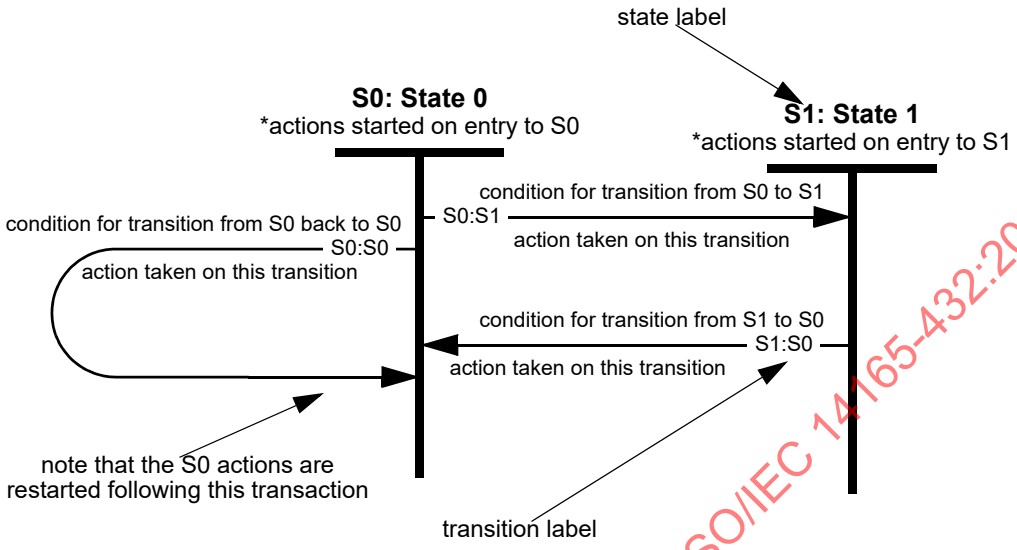

Figure 1 – State machine example

These state machines make three assumptions:

a) Time elapses only within discrete states.

b) State transitions are logically instantaneous, so the only actions taken during a transition are setting flags and variables and sending signals. These actions complete before the next state is entered.

c) Every time a state is entered, the actions of that state are started. This means that a transition that points back to the same state repeats the actions from the beginning. All the actions started upon entry complete before any tests are made to exit the state.

3.9 Using numbers in hash functions and concatenation functions

When a numeric value is used as a bit string argument to a function (e.g., inputs to hash functions and concatenation functions), unless otherwise specified, the numeric value is represented as a bit string as follows:

a) if the numeric value used as a bit string argument to a function is determined by reference to a field with size determined by this standard, its bit string representation is the entire field that contained it; and

b) if the numeric value used as a bit string argument to a function is not determined by reference to a field with size determined by this standard and the numeric value is the output of a function or the result of an arithmetic computation, the bit string representation of the numeric value is the bit string that:

A) has the same value when interpreted as a binary number; and

B) has as many bits as the shortest bit string that is capable of representing all possible outputs of the function or computation.

4 Structure and Concepts 4.1

Overview

The variety of environments in which Fibre Channel fabrics are deployed makes it difficult to rely on physical security. Storage subsystems may be accessed by different users over Fabrics that may span several sites. Security services are extremely important to prevent misconfigurations or access to data by non-authorized entities.

This standard defines mechanisms that may be used to protect against several classes of threats. These mechanisms include protocols to authenticate Fibre Channel entities, protocols to set up session keys, protocols to negotiate parameters to ensure frame-by-frame integrity and confidentiality, and protocols to define and distribute policies across a Fibre Channel Fabric.

The appropriate amount of security to deploy and the appropriate interfaces to protect are highly business and technology environment dependent. It is advisable that consumers of products based on this standard analyze their security needs, their business processes that require security technology, their flexibility requirements, and the range of available options to determine appropriate methods for mitigating identified risks for their particular environment. Risk mitigation is the core focus of any security technology deployment.

4.2 FC-SP-2 Compliance

To claim FC-SP-2 compliance, an implementation shall support the AUTH-A Compliance Element (see A.2.1). An FC-SP-2 compliant implementation may support additional Compliance Elements. Different combinations of Compliance Elements may be appropriate for different environments.

4.3 Fabric Security Architecture

The security architecture defined by this standard encompasses the following components:

a) Authentication Infrastructure (see 4.4);

b) Authentication (see 4.5);

c) Security Associations (see 4.6);

d) Cryptographic Integrity and Confidentiality (see 4.7); and

e) Authorization (see 4.8).

4.4 Authentication Infrastructure

The Fabric security architecture is defined for several authentication infrastructures. Secret-based, Certificate-based, password-based, and Pre-Shared Key based authentication infrastructures are accommodated. Specific authentication protocols that directly leverage these four authentication infrastructures are defined.

With a secret-based infrastructure, entities within the fabric environment that establish a security relationship share a common secret or centralize the secret administration in a RADIUS server (see Annex D). Entities may mutually authenticate with other entities by using the DH-CHAP protocol (see 5.4). Security Associations may be set up using the session key computed at the end of the DH-CHAP transaction. Frame integrity or confidentiality may be provided by using the ESP_Header (see FC-FS-3).

With a Certificate-based infrastructure, entities within the fabric environment are certified by a trusted Certificate Authority. The resulting Certificates bind each entity to a public-private key pair that may be used to mutually authenticate with other certified entities via the FCAP protocol (see 5.5). Security Associations may be set up by using these entity Certificates and associated keys or by using the session key computed at the end of the FCAP transaction. Frame integrity or confidentiality may be provided by using the ESP_Header (see FC-FS-3).

With a password-based infrastructure, entities within the fabric environment that establish a security relationship have knowledge of the password-based credential material of other entities. Entities may use this credential material to mutually authenticate with other entities using the FCPAP protocol (see 5.6). Security Associations may be set up using the session key computed at the end of the FCPAP transaction. Frame integrity or confidentiality may be provided by using the ESP_Header (see FC-FS-3).

With a Pre-Shared Key based infrastructure, entities within the fabric environment that establish a security relationship have knowledge of the Pre-Shared Key based credential material of other entities. An example shared key infrastructure is shown in Annex B. Entities may use this credential material to mutually authenticate with other entities using the FCEAP protocol (see 5.7). Security Associations may be set up using the session key computed at the end of the FCEAP transaction. Frame integrity or confidentiality may be provided by using the ESP_Header (see FC-FS-3).

4.5 Authentication

Authentication Protocols are defined to allow entities to ensure the identity of the entities with which they are communicating. Two entities may negotiate whether authentication is required and which Authentication Protocol may be used. Authentication is defined for Switch to Switch, Node to Switch, and Node to Node. The defined Authentication Protocols are able to perform mutual authentication with optional shared key establishment. The shared key computed at the end of an Authentication Transaction may be used to establish Security Associations, as shown in figure 2.

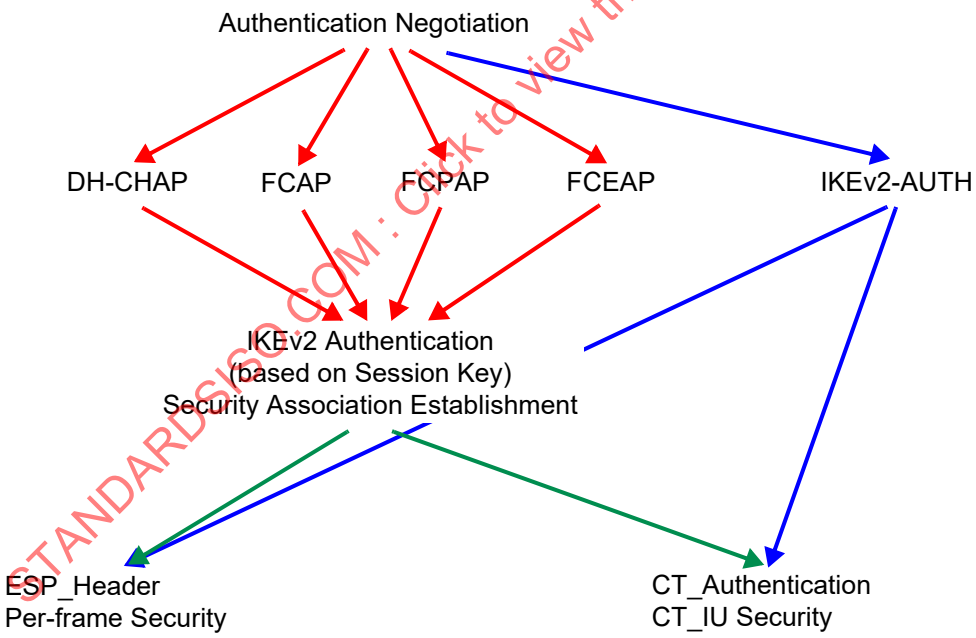

Figure 2 – Relationship between Authentication Protocols and Security Associations

The following Authentication Protocols are defined:

a) Diffie-Hellman Challenge Handshake Authentication Protocol (DH-CHAP, see 5.4);

b) Fibre Channel Certificate Authentication Protocol (FCAP, see 5.5);

c) Fibre Channel Password Authentication Protocol (FCPAP, see 5.6);

d) Fibre Channel Extensible Authentication Protocol (FCEAP, see 5.7); and

e) The Security Association Management Protocol (IKEv2-AUTH, see 6.7.3).

NOTE 2 – Using the Security Association Management Protocol for both Authentication and Security Association management (see 6.7.3) usually has better security properties than concatenating the Security Association Management Protocol to another Authentication protocol. However concatenating the Security Association Management Protocol to another Authentication protocol (see 6.7.2) leverages the authentication infrastructure associated with that Authentication protocol and this may be easier to manage.

NOTE 3 – Some Authentication Protocols (e.g., DH-CHAP with a NULL DH algorithm) do not generate the session key needed to concatenate the Security Association Management Protocol to an Authentication protocol (see 5.4.6).

4.6 Security Associations

A subset of the IKEv2 protocol suitable for Fibre Channel (i.e., the Security Association Management protocol, see clause 6) is defined in order to establish Security Associations between entities. Traffic Selectors are defined to specify which type of traffic has to be protected by the Security Association, and what the characteristics of the protection are. Two mechanisms are available to protect specific classes of traffic: the ESP_Header is used to protect FC-2 frames, and CT_Authentication is used to protect Common Transport Information Units.

An entity protecting specific classes of traffic maintains an internal Security Association Database (SADB) that contains the currently active Security Associations and Traffic Selectors.

Each active Security Association is logically associated with an entry in the SADB. A Security Association entry includes the SA's SPI, a Sequence Number counter, and the parameters for the selected transforms (e.g., encryption algorithm, integrity algorithm, mode of operation of the algorithms, keys).

Each active Traffic Selector is logically associated with an entry in the SADB. Two types of Traffic Selector entries may be present:

a) Traffic Selector entries identifying FC-2 frames or CT_IUs to be bypassed or discarded; and

b) Traffic Selector entries identifying FC-2 frames or CT_IUs to be protected or verified. These entries point to the corresponding SA entry defining the parameters and the security processing to be performed.

4.7 Cryptographic Integrity and Confidentiality

4.7.1 Overview

Frame by frame cryptographic integrity and confidentiality, replay protection, and traffic origin authentication is achieved by using the ESP_Header optional header (see FC-FS-3). CT_Authentication (see FC-GS-6) may be leveraged to provide cryptographic integrity and confidentiality, replay protection, and traffic origin authentication to Common Transport Information Units.

ESP_Header processing and CT_Authentication processing are independent, with two logically separated SADBs that apply to the respective levels (see 4.7.2 and 4.7.3).

NOTE 4 – It is then possible, although not recommended, to set up a set of Security Associations so that CT_Authentication protected CT_IUs are also protected with ESP_Header.

4.7.2 ESP_Header Processing

The ESP_Header is processed according to the model defined in RFC 2401. The ESP_Header processing is performed over selected frames according to a set of Traffic Selectors maintained in the SADB. Traffic Selectors are negotiated when Security Associations are established. Traffic Selectors may be Incoming Traffic Selectors, used to process incoming frames, or Outgoing Traffic Selectors, used to process outgoing frames. Each Traffic Selector has an action associated with it, that has to be applied to a frame that matches it. The action may be one of the following:

a) Bypass: the frame is passed unchanged;

b) Drop: the frame is discarded; or

c) Process: the frame is processed according to the SA pointed by the Traffic Selector.

Figure 3 shows an informative logical model of an entity capable of protecting selected subsets of Fibre Channel frames using the ESP_Header.

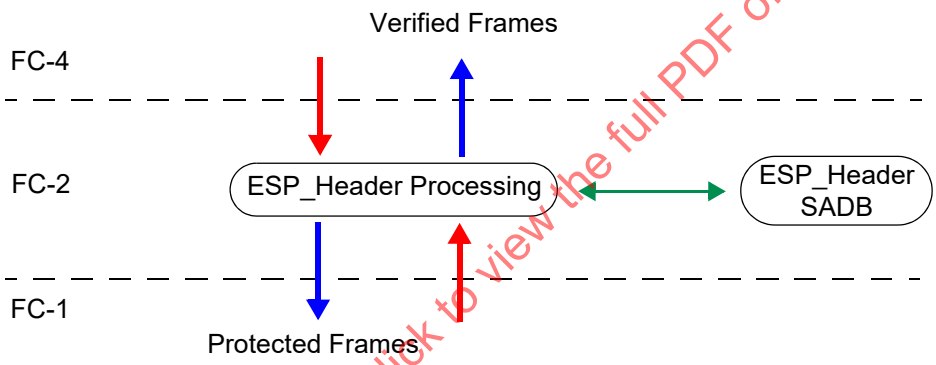

Figure 3 – Logical Model for Integrity and Confidentiality Protection with ESP_Header

When an outgoing frame is received by the ESP_Header Processing module, the SADB is checked to verify if the frame matches an Outgoing Traffic Selector. If there is no match, the frame is sent unchanged. If there is a match, the action associated with the matched Traffic Selector is applied to the frame. If the action is Process, the protecting security transforms defined by the SA pointed by the matched Traffic Selector are applied to the frame.

When an incoming frame not protected by the ESP_Header is received by the ESP_Header processing module, the SADB is checked to verify if the frame matches any of the existing Incoming Traffic Selectors. If there is a match and the action is Process, the frame is discarded. If there is a match and the action is Bypass or Drop, that action is applied to the frame. If there is no match, the frame is discarded.

This means that a frame not protected by the ESP_Header and not matching any Incoming Traffic Selector is discarded. It is possible to reverse this behavior by inserting in the SADB a last Incoming Traffic Selector associated with a Bypass action and that matches all frames. This guarantees a no-drop default behavior, however the security implications of this configuration should be carefully considered by a security administrator.

When an incoming frame protected by the ESP_Header is received by the ESP_Header processing module, the SPI contained in the frame is used to locate the protecting SA in the SADB. The security transforms associated with the matched SA are applied to the frame and:

a) if all security transforms succeed and the frame matches at least one of the Traffic Selectors associated with the located SA, the processed frame is passed to the upper levels; or

b) if at least one security transform fails or the frame does not match the Traffic Selectors associated with the located SA, the incoming frame is discarded.

The ESP_Header SADB maintains for each established ESP_Header SA the following logical parameters:

a) the 4-byte SPI, mapped on the ESP_Header Security Parameter Index (SPI) field;

b) a 32-bit sequence number counter, mapped on the ESP_Header Sequence Number field;

NOTE 5 – 64-bit extended sequence numbers are not supported by this standard.

c) the security parameters for the negotiated Transforms (e.g., encryption algorithm, integrity algorithm, mode of operation of the algorithms, keys).

d) a sequence counter overflow flag, indicating whether overflow of the sequence number counter generates an auditable event or prevents processing of additional frames on the SA;

e) a 32-bit anti-replay window counter and bit-map, used to determine whether an incoming frame carrying an ESP_Header is a replay; and

f) the lifetime of the Security Association (i.e., a time interval after which the SA has to be replaced with a new SA or terminated).

4.7.3 CT_Authentication Processing

The CT_Authentication processing is performed over selected CT_IUs according to a set of Traffic Selectors maintained in the SADB (Security Associations Database). Traffic Selectors are negotiated when Security Associations are established. Traffic Selectors may be Incoming Traffic Selectors, used to process incoming CT_IUs, or Outgoing Traffic Selectors, used to process outgoing CT_IUs. Each Traffic Selector has an action associated with it, that has to be applied to a CT_IU that matches it. The action may be one of the following:

a) Bypass: the CT_IU is passed unchanged;

b) Drop: the CT_IU is discarded; or

c) Process: the CT_IU is processed according to the SA pointed by the Traffic Selector.

Figure 4 shows an informative logical model of an entity capable of protecting selected subsets of Common Transport Information Units with CT_Authentication.

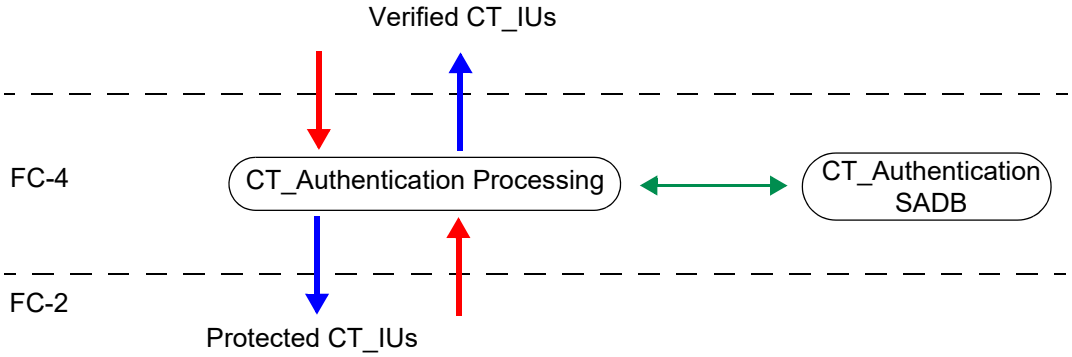

Figure 4 – Logical Model for Integrity and Confidentiality Protection with CT_Authentication

When an outgoing CT_IU is received by the CT_Authentication Processing module, the SADB is checked to verify if the CT_IU matches an Outgoing Traffic Selector. If there is no match, the CT_IU is sent unchanged. If there is a match, the action associated with the matched Traffic Selector is applied to the CT_IU. If the action is Process, the protecting security transforms defined by the SA pointed by the matched Traffic Selector are applied to the CT_IU.

When an incoming CT_IU not protected by CT_Authentication is received by the CT_Authentication processing module, the SADB is checked to verify if the CT_IU matches any of the existing Incoming Traffic Selectors. If there is a match and the action is Process, the CT_IU is discarded. If there is a match and the action is Bypass or Drop, that action is applied to the CT_IU. If there is no match, the CT_IU is discarded.

NOTE 6 – This means that a CT_IU not protected by CT_Authentication and not matching any Incoming Traffic Selector is discarded. It is possible to reverse this behavior by inserting in the SADB a last Incoming Traffic Selector associated with a Bypass action and that matches all CT_IUs. This guarantees a no-drop default behavior, however the security implications of this configuration should be carefully considered by a security administrator.

When an incoming CT_IU protected by CT_Authentication is received by the CT_Authentication processing module, the SAID contained in the CT_IU is used to locate the protecting SA in the SADB. The security transforms associated with the matched SA are applied to the CT_IU and:

a) if all security transforms succeed and the CT_IU matches at least one of the Traffic Selectors associated with the located SA, the processed CT_IU is passed to the upper levels; or

b) if at least one security transform fails or the CT_IU does not match the Traffic Selectors associated with the located SA, the incoming CT_IU is discarded.

The CT_Authentication SADB maintains for each established CT_Authentication SA the following logical parameters:

a) the 4-byte SPI, mapped on the CT_IU Authentication SAID field;

b) a 32-bit sequence number counter, mapped on the CT_IU Time Stamp field;

c) the security parameters for the negotiated Transforms (e.g., encryption algorithm, integrity algorithm, mode of operation of the algorithms, keys).

d) a sequence counter overflow flag, indicating whether overflow of the sequence number counter generates an auditable event or prevent processing of additional CT_IUs on the SA;

e) a 32-bit anti-replay window counter and bit-map, used to determine whether an incoming CT_IU protected with CT_Authentication is a replay; and

f) the lifetime of the Security Association (i.e., a time interval after which the SA has to be replaced with a new SA or terminated).

4.8 Authorization (Access Control)

4.8.1 Policy Definition

Fabric policies provide basic authorization controls in the form of access control lists (ACLs). Two basic types of policies are defined:

a) policies that contain Fabric-wide data, distributed to every Switch of the Fabric; and

b) policies that contain per Switch data, sent to an individual Switch.

Fabric policies may be used to control which Switches are allowed in a Fabric and which Nodes are allowed to connect to a Fabric. Policies may be further used to specify topology restrictions within the Fabric environment (e.g., which Switches may connect to which other Switches or which Nodes may connect to which Switches).

Fabric policies also provide the mechanism for controlling management access to the Fabric and the ability to control authentication choices and to specify security attributes for Fabric entities (e.g., Nodes and Switches). Management access to the Fabric may be controlled for Common Transport or IP access.

A policy configuration is composed by a set of Policy Objects (see 7.1). Each Policy Object is summarized in a hash value. The enforcement of a policy configuration is performed with the definition of a Policy Summary Object (i.e., a Fabric-wide Object that consists of the names of all the Policy Objects along with their associated hashes). The Policy Summary Object allows an easy comparison of policy configurations.

Zoning policies (i.e., FC-SP Zoning) are defined to encode Node to Node restrictions in a form consistent with the Policy model (see 7.6).

Fabric policies and Zoning policies allow an asymmetric distribution of policy information in the Fabric with the definition of three types of Switches (see 7.1.4):

a) Server Switches: Switches that retain all Policy Objects and all Node to Node (Zoning) information;

b) Autonomous Switches: Switches that retain their own per Switch Policy Objects, all Fabric-wide Policy Objects, and all Node to Node (Zoning) information; and

c) Client Switches: Switches that retain their per Switch Policy Objects, all Fabric-wide Policy Objects and the subset of the Node to Node (Zoning) information relevant for their operations, which is pulled from a Server Switch when needed. In addition, they maintain the Zone Set Database Hash and the Active Zone Set Hash for the Fabric (see 7.6).

4.8.2 Policy Enforcement

Policy enforcement occurs whenever a connection is attempted, a management application attempts to access the Fabric, or a new policy configuration is activated (see 7.2). The appropriate Policy Objects are

checked to determine whether the requested connection or access is to be allowed or denied. The Policy enforcement is performed locally by the entities involved in the connection or access attempt.

4.8.3 Policy Distribution

Distribution mechanisms are defined to ensure that policy information is distributed to the appropriate Switches in the Fabric (see 7.3).

4.8.4 Policy Check

When two Switches join they ensure that their enforced policy configurations are the same. They do this by exchanging their Policy Summary Objects. If the compared Policy Summary Objects are identical, then the join is allowed, otherwise the join is denied (see 7.4).

4.9 Name Format

This standard uses the name format shown in table 2 to identify entities (e.g., for Authentication or Fabric Policies).

Table 2 – Name Format

Item	Size (Bytes)
Name Tag	2
Name Length	2
Name Value	variable

Name Tag: Identifies the format of the name. Tag values are defined in 5.3.3 and 7.1.2.

Name Length: Indicates the total length in bytes of the Name Value.

Name Value: Contains the name value, according to the specified Name Tag.

5 Authentication Protocols 5.1

Overview

Secure relationships are possible only between known entities. Authentication is the process by which an entity is able to verify the identity of another entity, thus providing the foundation for secure relationships.

Different Authentication Protocols may be used to validate an entity on the basis of different parameters, (e.g., digital Certificates, secrets, or passwords). Selection of allowed and preferred security protocols and parameters is a decision made by a security administrator.

Each entity is identified by a name. The purpose of an Authentication Protocol is to verify that a claimed name is associated with the claiming entity. The Authentication Protocols may be used to authenticate Nx_Ports, B_Ports, or Switches. When an Nx_Port authenticates with a Switch, the authentication is established to the whole Fabric because the Switch identity serves as a proxy for the Fabric of which the Switch is a member.

An Authentication Transaction occurs between an Authentication Initiator and an Authentication Responder. An Authentication Transaction (see figure 5) is identified by a unique Transaction Identifier. The Authentication Initiator starts the Authentication Transaction by sending the AUTH_Negotiate message (see 5.3.2) to the Authentication Responder. In the AUTH_Negotiate message, the Authentication Initiator shall specify the Transaction Identifier, and shall send its own name, together with the list of proposed Authentication Protocols and associated parameters that may be used in the transaction. The Authentication Responder shall choose from the proposed Authentication Protocols and associated parameters the ones to be used to perform the Authentication, and reply with an appropriate AUTH message (see 5.2.1).

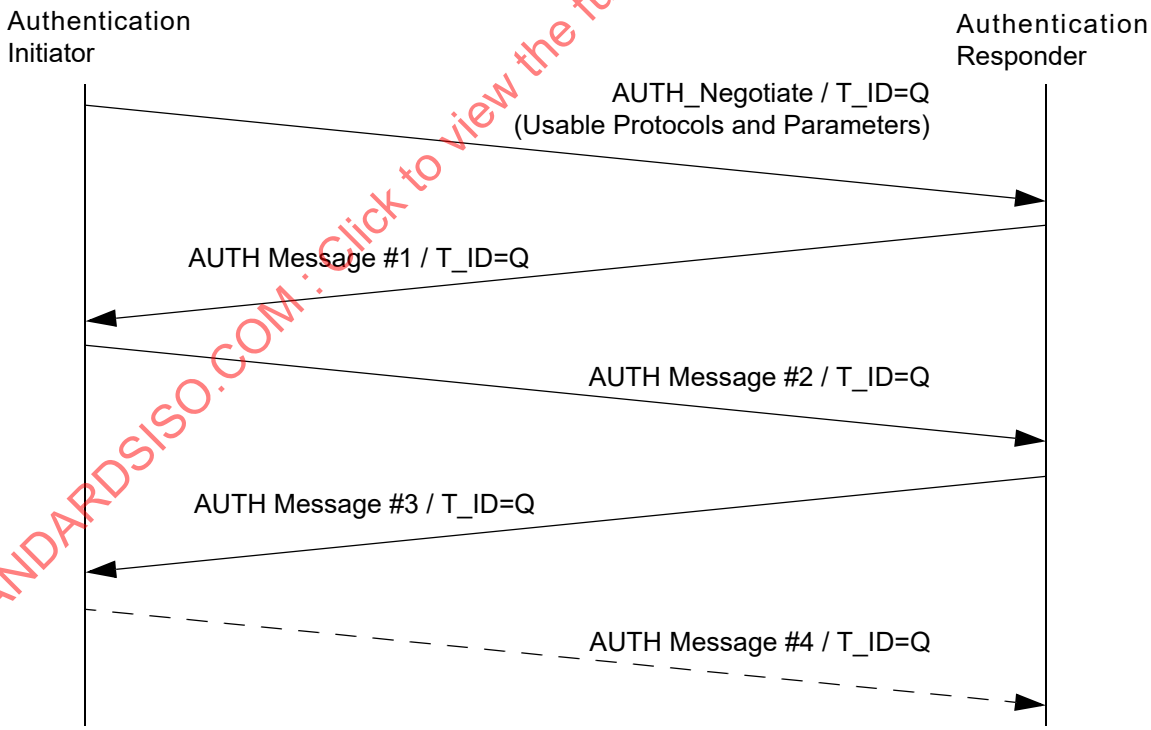

Figure 5 – A Generic Authentication Transaction

The Authentication Protocols allow any Fibre Channel entity to act as an Authentication Initiator or as an Authentication Responder. A Fibre Channel entity may initiate an Authentication Transaction whenever needed. No more than one transaction of an Authentication Protocol shall be in progress between the same two entities at a time. If two communicating entities start an Authentication Transaction at the same time, a case by case rule (see 5.8.1 and 5.10.1) determines which transaction shall be aborted and which one shall be continued.

If a Fibre Channel entity is not acting as an Authentication Initiator or Authentication Responder and receives an AUTH_Negotiate message, it shall reply to that message as specified by the Authentication Protocol of its choosing, becoming the Authentication Responder.

If a Fibre Channel entity is acting as an Authentication Initiator and receives an AUTH_Negotiate message from the designated Authentication Responder, one of the two Authentication Transactions shall be aborted (see 5.8.1 and 5.10.1). The Fibre Channel entity that remains the Authentication Initiator shall reply to the received AUTH_Negotiate message with an appropriate AUTH_Reject message. The Fibre Channel entity that becomes the Authentication Responder shall reply to the received AUTH_Negotiate message and abort its own transaction upon receipt of the AUTH_Reject message.

Each Authentication Protocol specifies the conditions required for an Authentication Transaction to complete successfully. If an Authentication Transaction does not complete successfully, an Authentication failure shall occur and appropriate actions shall be taken (e.g., terminate the communication).

An AUTH message may receive a response with an error indication of:

a) AUTH_Reject message (see 5.3.7);

b) SW_RJT (see 5.8.3 and 5.9.3); or

c) LS_RJT (see 5.10.3).

Two error indications shall not be generated in response to one AUTH message. If a lower level error is detected (e.g., the AUTH_Negotiate ELS or SW_ILS is not supported) an appropriate LS_RJT or SW_RJT shall be generated. If the received message is correct for the lower level, but there are Authentication problems, then an appropriate AUTH_Reject message shall be generated.

5.2 Authentication Messages Structure

5.2.1 Overview

The Authentication Protocols may be used to authenticate Nx_Ports, B_Ports, or Switches. The Authentication messages are transmitted as SW_ILSs when the involved entities are Switches or B_Ports, and as ELSs when the involved entities include at least one Nx_Port. All Authentication messages share the same message structure, and are referred to as AUTH messages.

5.2.2 SW_ILS Authentication Messages

When an AUTH message is transmitted between Switches (e.g., for E_Port to E_Port or Domain_Controller to Domain_Controller Authentication), it is transmitted as an SW_ILS and is referred to as an AUTH_ILS message (see table 3). The AUTH_ILS message is propagated by B_Ports.

Table 3 – AUTH_ILS Message Format

Item	Size (Bytes)
AUTH_ILS Code = 40h	1
AUTH_ILS Flags	1
AUTH Message Code	1
Protocol Version	1
Message Length	4
Transaction Identifier (T_ID)	4
Message Payload	variable

AUTH_ILS Flags: The AUTH_ILS flags are shown in table 4.

Table 4 – AUTH_ILS Flags

Bit	Description
7	Shall be set to 0
6	Concatenation Flag
5 .. 1	Reserved
0	Shall be set to 0

Concatenation Flag: Used to concatenate multiple AUTH transactions, a capability referred to as AUTH Concatenation. As an example of usage, AUTH Concatenation enables an SA Management Transaction to be concatenated to an Authentication Transaction (see 6.7.2). When the Concatenation Flag is set to one in AUTH_Negotiate, the Authentication Initiator requires AUTH Concatenation. If the Authentication Responder does not support AUTH Concatenation, an AUTH_Reject with Reason Code 'Logical Error' and Reason Code Explanation 'AUTH Concatenation not Supported' shall be returned (see 5.3.7). The Authentication Initiator may restart the Authentication Transaction without setting the Concatenation Flag, if appropriate. If the Authentication Responder supports AUTH Concatenation, the Concatenation Flag shall be set to one in all subsequent messages belonging to that transaction, otherwise an AUTH_Reject with Reason Code 'Authentication Failure' and Reason Code Explanation 'Incorrect Authentication Protocol Message' shall be returned (see 5.3.7).

See 5.2.4 for the definition of the other AUTH_ILS fields.

When an AUTH message is transmitted between an E_Port and a B_Port, it is transmitted as an SW_ILS and is referred to as a B_AUTH_ILS message (see table 5). The B_AUTH_ILS message is terminated by B_Ports.

Table 5 – B_AUTH_ILS Message Format

Item	Size (Bytes)
B_AUTH_ILS Code = 41h	1
B_AUTH_ILS Flags	1
AUTH Message Code	1
Protocol Version	1
Message Length	4
Transaction Identifier (T_ID)	4
Message Payload	variable

B_AUTH_ILS Flags: The B_AUTH_ILS flags are equal to the AUTH_ILS flags, shown in table 4.

See 5.2.4 for the definition of the other B_AUTH_ILS fields.

5.2.3 ELS Authentication Messages

When an AUTH message is transmitted between an Nx_Port and an Fx_Port or between Nx_Ports, it is transmitted as an ELS, and is referred to as an AUTH_ELS message (see table 6).

Table 6 – AUTH_ELS Message Format

Item	Size (Bytes)
AUTH_ELS Code = 90h	1
AUTH_ELS Flags	1
AUTH Message Code	1
Protocol Version	1
Message Length	4
Transaction Identifier (T_ID)	4
Message Payload	variable

Flags: The AUTH_ELS flags are shown in table 7.

Table 7 – AUTH_ELS Flags

Bit	Description
7	1 = More Fragments Follow 0 = No More Fragments (see 5.10.4)
6	Concatenation Flag
5 .. 1	Reserved
0	Sequence Number (see 5.10.4)

Concatenation Flag: Used to concatenate multiple AUTH transactions, a capability referred to as AUTH Concatenation. As an example of usage, AUTH Concatenation enables an SA Management Transaction to be concatenated to an Authentication Transaction (see 6.7.2). When the Concatenation Flag is set to one in AUTH_Negotiate, the Authentication Initiator requires AUTH Concatenation. If the Authentication Responder does not support AUTH Concatenation, an AUTH_Reject with Reason Code 'Logical Error' and Reason Code Explanation 'AUTH Concatenation not Supported' shall be returned (see 5.3.7). The Authentication Initiator may restart the Authentication Transaction without setting the Concatenation Flag, if appropriate. If the Authentication Responder supports AUTH Concatenation, the Concatenation Flag shall be set to one in all subsequent messages belonging to that transaction, otherwise an AUTH_Reject with Reason Code 'Authentication Failure' and Reason Code Explanation 'Incorrect Authentication Protocol Message' shall be returned (see 5.3.7).

See 5.2.4 for the definition of the other AUTH_ELS fields.

5.2.4 Fields Common to All AUTH Messages

AUTH Message Code: specifies the AUTH message that is to be transmitted from the source to the destination. The AUTH Message Codes are listed in table 8.

Table 8 – AUTH Message Codes

Value	Description
01h .. 09h	Reserved for legacy implementations
0Ah	AUTH_Reject (see 5.3.7)
0Bh	AUTH_Negotiate (see 5.3.2)
0Ch	AUTH_Done (see 5.3.8)
10h	DHCHAP_Challenge (see 5.4.3)
11h	DHCHAP_Reply (see 5.4.4)
12h	DHCHAP_Success (see 5.4.5)
13h	FCAP_Request (see 5.5.3)
14h	FCAP_Acknowledge (see 5.5.4)
15h	FCAP_Confirm (see 5.5.5)
16h	FCPAP_Init (see 5.6.3)
17h	FCPAP_Accept (see 5.6.4)
18h	FCPAP_Complete (see 5.6.5)
22h	IKE_SA_Init (see 6.3)
23h	IKE_Auth (see 6.4)
24h	IKE_Create_Child_SA (see 6.5)
25h	IKE_Informational (see 6.6)
26h	FCEAP_Request (see 6.3)
27h	FCEAP_Response (see 6.4)
28h	FCEAP_Success (see 6.5)
29h	FCEAP_Failure (see 6.6)
F0h .. FEh	Vendor Specific (see 5.2.5)
all others	Reserved

Protocol Version: Specifies the version of the AUTH protocol. This value shall be set to 01h. If an entity receives an AUTH message with a Protocol Version value that is higher than its highest supported value, the entity shall reply with an AUTH_Reject with Reason Code 'Logical Error' and Reason Code Explanation 'Unsupported Protocol Version'. The Protocol Version value of the AUTH_Reject shall be set to the entity's highest supported Protocol Version value.

Message Length: Specifies the total length in bytes of the Message Payload of the AUTH message. It is needed for the fragmentation support (see 5.10.4).

Transaction Identifier: Uniquely identifies an Authentication Transaction between two entities. The Transaction Identifier shall be set by the Authentication Initiator, and each subsequent Authentication message between the same two entities shall contain the same value, until the Authentication Transaction is completed. Its value shall be unique for each Authentication Transaction between two entities.

NOTE 7 – The usage of the Transaction Identifier is very similar to the usage of an OX_ID when an Exchange Originator is enforcing uniqueness via the OX_ID mechanism (see FC-FS-2), but it is not related in any way to the OX_ID present in the Fibre Channel frames carrying the Authentication messages.

When DH-CHAP is used as the Authentication Protocol (see 5.4), the least significant byte of the Transaction Identifier field shall be changed for each transaction of the Protocol between two entities, allowing its use as the DH-CHAP identifier in the DH-CHAP hash computation. This allows compatibility with a back-end authentication infrastructure based on RADIUS (see RFC 2865).

NOTE 8 – Incrementing the Transaction Identifier by one for each new Authentication Transaction satisfies the uniqueness requirement stated in this subclause.

Message Payload: contains the payload related to the specific AUTH message specified in the AUTH Message Code field.

5.2.5 Vendor Specific Messages

The message payload of Vendor Specific messages shall have the format shown in table 9.

Table 9 – Vendor Specific Message Payload Format

Item	Size (Bytes)
Vendor_ID	8
Vendor Specific Information	variable

Vendor_ID: This field shall contain the vendor's T10 Vendor ID.

Support for Vendor Specific messages is optional. A system shall be able to operate correctly when all Vendor Specific messages are rejected.

5.3 Authentication Messages Common to Authentication Protocols

5.3.1 Overview

The AUTH_Negotiate and the AUTH_Reject messages are common to all Authentication Protocols.

An Authentication Transaction is initiated with an AUTH_Negotiate message and terminated by:

a) a successful completion of the Authentication Transaction;

b) an AUTH_Reject message (see 5.3.7);

c) an SW_RJT (see 5.8.3 and 5.9.3); or

d) an LS_RJT (see 5.10.3).

5.3.2 AUTH_Negotiate Message

An Authentication Initiator transmits an AUTH_Negotiate message to negotiate the Authentication Protocol and the associated parameters to be used for the remainder of this Authentication Transaction. The Authentication Protocols and parameters that may be used are listed in order of preference within the AUTH_Negotiate message payload. The first Authentication Protocol listed is the most preferred and the last one is the least preferred. The message payload of the AUTH_Negotiate message is shown in table 10.

Table 10 – AUTH_Negotiate Message Payload

Item	Size (Bytes)
Authentication Initiator Name	variable
Number of Usable Authentication Protocols	4
Authentication Protocol Parameters #1 Length	4
Authentication Protocol Identifier #1 (most preferred)	4
Authentication Protocol Parameters #1	variable
Authentication Protocol Parameters #2 Length	4
Authentication Protocol Identifier #2	4
Authentication Protocol Parameters #2	variable
...	...
Authentication Protocol Parameters #q Length	4
Authentication Protocol Identifier #q (least preferred)	4
Authentication Protocol Parameters #q	variable

Authentication Initiator Name: Shall be set to the Authentication Initiator Name (see 5.3.3).

Number of Usable Authentication Protocols: Specifies the number of Authentication Protocols the Authentication Initiator proposes for use in this Authentication Transaction.

Authentication Protocol Parameters Length: Specifies the length in bytes of the corresponding Authentication Protocol Identifier and Authentication Protocol Parameters fields.

Authentication Protocol Identifier: Identifies the Authentication Protocol. The Authentication Protocol Identifiers are listed in table 11.

Table 11 – Authentication Protocol Identifiers

Value	Authentication Protocol
0000 0001h	DH-CHAP (see 5.4.2)
0000 0002h	FCAP (see 5.5.2)
0000 0003h	FCPAP (see 5.6.2)
0000 0004h	IKEv2 (see 6.7.2)
0000 0005h	IKEv2-AUTH (see 6.7.3)
0000 0006h	FCEAP (see 5.7.2)
0000 00F0h .. 0000 00FEh	Vendor Specific Protocols
all others	Reserved

Authentication Protocol Parameters: Lists the usable parameters for the Authentication Protocol. Each Authentication Protocol defines the format of this field.

The Authentication Protocol Parameters for Vendor Specific protocols shall have the structure shown in table 12.

Table 12 – AUTH_Negotiate Vendor Specific Protocol Parameters

Item	Size (Bytes)
Vendor_ID	8
Vendor Specific Information	variable

Vendor_ID: This field shall contain the vendor's T10 Vendor ID.

Support for Vendor Specific protocols is optional. A system shall be able to operate correctly when all vendor specific protocols are rejected.

5.3.3 Names used in Authentication

The name used in the Authentication Protocol payloads identifies an entity for Authentication purposes, and shall have the format specified in table 2 with the Name Tag, Name Length, and Name Value content shown in table 13.

Table 13 – Names used in Authentication

Name Tag	Name Length (Bytes)	Name Value content
0001h	8	Name_Identifier[a]
[a] The IEEE Registered Extended Name_Identifier format (i.e., NAA=6h) shall not be used.		

5.3.4 Hash Functions

The list of hash function identifiers used in the Authentication Protocol payloads is shown in table 14.

Table 14 – Hash Functions Identifiers

Identifier	Hash Function	Hash Length (bytes)
0000 0005h[a]	MD5 hash function	16
0000 0006h[a]	SHA-1 hash function	20
0000 0007h	SHA-256 hash function[b]	32
0000 0008h	SHA-384 hash function[b]	48
0000 0009h	SHA-512 hash function[b]	64
all others	Reserved	

[a] These values are the same as those specified by IANA in the "Authentication Algorithms" section of the Point-to-Point Protocol Field Assignments registry (see http://www.iana.org/assignments/ppp-numbers). However, IANA identifiers are 8-bit values, whereas these identifiers are 32-bit values.

[b] See FIPS PUB 180-4.

5.3.5 Diffie-Hellman Groups

The list of Diffie-Hellman groups identifiers used in the Authentication Protocol payloads is shown in table 15. These groups identifiers are used by the Authentication Protocols defined in clause 5, but not by the SA Management protocol defined in clause 6.

Table 15 – Diffie-Hellman Group Identifiers (part 1 of 2)

Identifier	DH Group	Generator (g)	Modulus (p or n) (Hex)
0000 0000h	NULL	N/A	N/A
0000 0001h	1 024[a]	2	EEAF0AB9ADB38DD69C33F80AFA8FC5E8 6072618775FF3C0B9EA2314C9C256576 D674DF7496EA81D3383B4813D692C6E0 E0D5D8E250B98BE48E495C1D6089DAD1 5DC7D7B46154D6B6CE8EF4AD69B15D49 82559B297BCF1885C529F566660E57EC 68EDBC3C05726CC02FD4CBF4976EAA9A FD5138FE8376435B9FC61D2FC0EB06E3
0000 0002h	1 280[a]	2	D77946826E811914B39401D56A0A7843 A8E7575D738C672A090AB1187D690DC4 3872FC06A7B6A43F3B95BEAEC7DF04B9 D242EBDC481111283216CE816E004B78 6C5FCE856780D41837D95AD787A50BBE 90BD3A9C98AC0F5FC0DE744B1CDE1891 690894BC1F65E00DE15B4B2AA6D87100 C9ECC2527E45EB849DEB14BB2049B163 EA04187FD27C1BD9C7958CD40CE7067A 9C024F9B7C5A0B4F5003686161F0605B

[a] See RFC 3723

Table 15 – Diffie-Hellman Group Identifiers (part 2 of 2)

Identifier	DH Group	Generator (g)	Modulus (p or n) (Hex)
0000 0003h	1 536[a]	2	9DEF3CAFB939277AB1F12A8617A47BBB DBA51DF499AC4C80BEEEA9614B19CC4D 5F4F5F556E27CBDE51C6A94BE4607A29 1558903BA0D0F84380B655BB9A22E8DC DF028A7CEC67F0D08134B1C8B9798914 9B609E0BE3BAB63D47548381DBC5B1FC 764E3F4B53DD9DA1158BFD3E2B9C8CF5 6EDF019539349627DB2FD53D24B7C486 65772E437D6C7F8CE442734AF7CCB7AE 837C264AE3A9BEB87F8A2FE9B8B5292E 5A021FFF5E91479E8CE7A28C2442C6F3 15180F93499A234DCF76E3FED135F9BB
0000 0004h	2 048[a]	2	AC6BDB41324A9A9BF166DE5E1389582F AF72B6651987EE07FC3192943DB56050 A37329CBB4A099ED8193E0757767A13D D52312AB4B03310DCD7F48A9DA04FD50 E8083969EDB767B0CF6095179A163AB3 661A05FBD5FAAAE82918A9962F0B93B8 55F97993EC975EEAA80D740ADBF4FF74 7359D041D5C33EA71D281E446B14773B CA97B43A23FB801676BD207A436C6481 F1D2B9078717461A5B9D32E688F87748 544523B524B0D57D5EA77A2775D2ECFA 032CFBDBF52FB3786160279004E57AE6 AF874E7303CE53299CCC041C7BC308D8 2A5698F3A8D0C38271AE35F8E9DBFBB6 94B5C803D89F7AE435DE236D525F5475 9B65E372FCD68EF20FA7111F9E4AFF73
0001 0006h	3 072	5	see RFC 3526
0001 0007h	4 096	5	see RFC 3526
0001 0008h	6 144	5	see RFC 3526
0001 0009h	8 192	19	see RFC 3526
all others	Reserved		

[a] See RFC 3723

5.3.6 Accepting an AUTH_Negotiate Message

To accept an AUTH_Negotiate message, an Authentication Responder shall send the next message specified for the selected Authentication Protocol. The Authentication Responder shall select an Authentication Protocol in accord with applicable policy of the Authentication Responder. This policy may require that the preference expressed by the Authentication Initiator in the AUTH_Negotiate message be honored.

5.3.7 AUTH_Reject Message

The AUTH_Reject message reports error conditions that occur at the Authentication level (e.g., figure 6 shows what happens if the Authentication Responder receives a message that contains a list of

Authentication Protocols and the Responder is unable to use any of them. This may happen when the Authentication Responder is not yet configured with an appropriate secret).

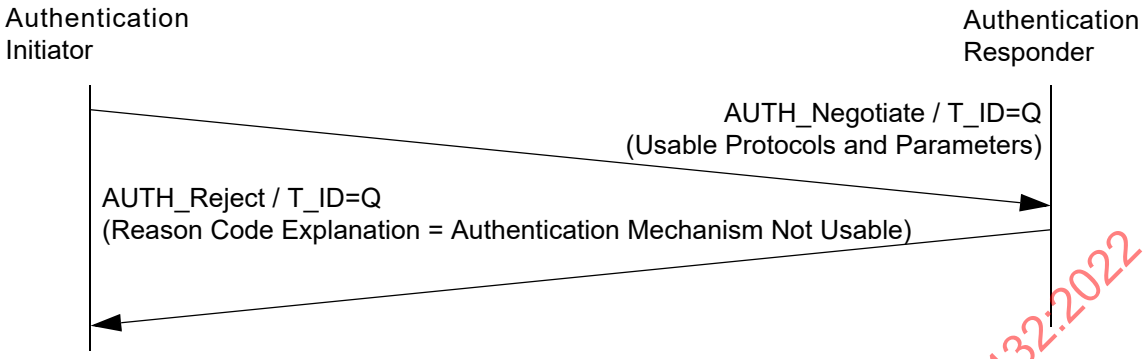

Figure 6 – Example of AUTH_Reject

The message payload of the AUTH_Reject message is shown in table 16.

Table 16 – AUTH_Reject Message Payload

Item	Size (Bytes)
Reason Code	1
Reason Code Explanation	1
Reserved	2

The AUTH_Reject Reason Codes are listed in table 17.

Table 17 – AUTH_Reject Reason Codes

Value	Description
01h	Authentication Failure
02h	Logical Error
all others	Reserved

An ‘Authentication Failure’ Reason Code indicates that the AUTH_Reject sending entity failed the Authentication Transaction and shall terminate the communication.

A ‘Logical Error’ Reason Code indicates that the Authentication Transaction is not completed, and may be restarted or continued. The entity sending an AUTH_Reject with a ‘Logical Error’ Reason Code terminates the Authentication Transaction.

The AUTH_Reject Reason Code Explanations are listed in table 18.

Table 18 – AUTH_Reject Reason Code Explanations

Value	Description
01h	Authentication Mechanism Not Usable
02h	DH Group Not Usable
03h	Hash Function Not Usable
04h[a]	Authentication Transaction Already Started
05h	Authentication Failed
06h[b]	Incorrect Payload
07h[b]	Incorrect Authentication Protocol Message
08h	Restart Authentication Protocol
09h	AUTH Concatenation not Supported
0Ah	Unsupported Protocol Version
all others	Reserved

[a] This Reason Code Explanation shall be used only for the cases described in 5.8.1 and 5.10.1.

[b] Message type, payload format, and payload structure shall be validated before performing any other security computation.

The use of AUTH_Reject Reason Codes and Reason Code Explanations under some error conditions are listed in table 19.

Table 19 – Error Conditions (part 1 of 2)

Error Condition	After Message	Reason Code	Reason Code Explanation	Action performed by the entity receiving AUTH_Reject
None of the proposed Authentication mechanisms are usable	AUTH_Negotiate	02h	01h	Based on policy: a) continue if Authentication is optional; b) retry with different parameters if specific parameters are not required; or c) terminate communication if specific parameters are required
None of the proposed DH groups are usable	AUTH_Negotiate, DHCHAP_Challenge, FCAP_Request, FCPAP_Init	02h	02h	
None of the proposed hash functions are usable	AUTH_Negotiate, DHCHAP_Challenge, FCAP_Request, FCPAP_Init	02h	03h	
An Authentication Transaction is already in progress	AUTH_Negotiate	02h	04h	Continue by replying to the Authentication Transaction initiated by the other entity

Table 19 – Error Conditions (part 2 of 2)

Error Condition	After Message	Reason Code	Reason Code Explanation	Action performed by the entity receiving AUTH_Reject
Authentication Failed	DHCHAP_Reply, DHCHAP_Success, FCAP_Request, FCAP_Acknowledge, FCAP_Confirm, FCPAP_Accept, FCPAP_Complete	01h	05h	Terminate communication
Erroneous payload received	Any	01h	06h	Terminate communication
The received Authentication message is not consistent with the Authentication Protocol in progress[a]	Any except AUTH_Negotiate	01h	07h	Terminate communication
The Authentication Transaction needs to be restarted	Any	02h	08h	Restart the Authentication Transaction by sending a new AUTH_Negotiate
The Concatenation Flag is set to one in a received AUTH message, but AUTH Concatenation is not supported by the receiving entity	AUTH_Negotiate	02h	09h	see 5.2.2, 5.2.3, 6.2.4 and 6.7.2
The Protocol Version of the received AUTH message is not supported	AUTH_Negotiate	02h	0Ah	Retry using a lower Protocol Version (e.g., the Protocol Version received in the AUTH_Reject)

[a] This error condition shall be detected also when a received AUTH message has the Concatenation Flag set to one outside the conditions defined in 5.2.2, 5.2.3, 6.2.4 and 6.7.2.

5.3.8 AUTH_Done Message

The AUTH_Done message completes the Authentication Transaction for some Authentication Protocols, and provides an indication that Authentication has been successful. The AUTH_Done message has no Message Payload.

5.4 DH-CHAP Protocol

5.4.1 Protocol Operations

DH-CHAP is a secret based Authentication and key management protocol that uses the CHAP algorithm (see RFC 1994) augmented with an optional Diffie-Hellman algorithm (see RFC 2631, clause 2.2.1). DH-CHAP provides bidirectional or unidirectional Authentication between an Authentication Initiator and an Authentication Responder.

When the Diffie-Hellman part of the protocol is not used, DH-CHAP reduces its operations to those of the CHAP protocol, and it is referred to as DH-CHAP with a NULL DH algorithm.

An implementation that supports Authentication shall support DH-CHAP with a NULL DH algorithm.

In order to authenticate with the DH-CHAP protocol, each entity, identified by a unique Name, shall be provided with a secret. Two entities may impersonate one another if they have the same secret, therefore when the assigned secrets are not different for each entity there is a security vulnerability.

To Authenticate another entity, an entity is required to either:

a) Know the secret associated with the entity to be Authenticated; or

b) Rely on a third party that knows the secret (e.g., a RADIUS server) to verify the Authentication.

An example of a DH-CHAP protocol transaction is shown in figure 7, with the notation shown in table 20. The final DHCHAP_Success message that is shown as a dashed line is used only for bidirectional Authentication.

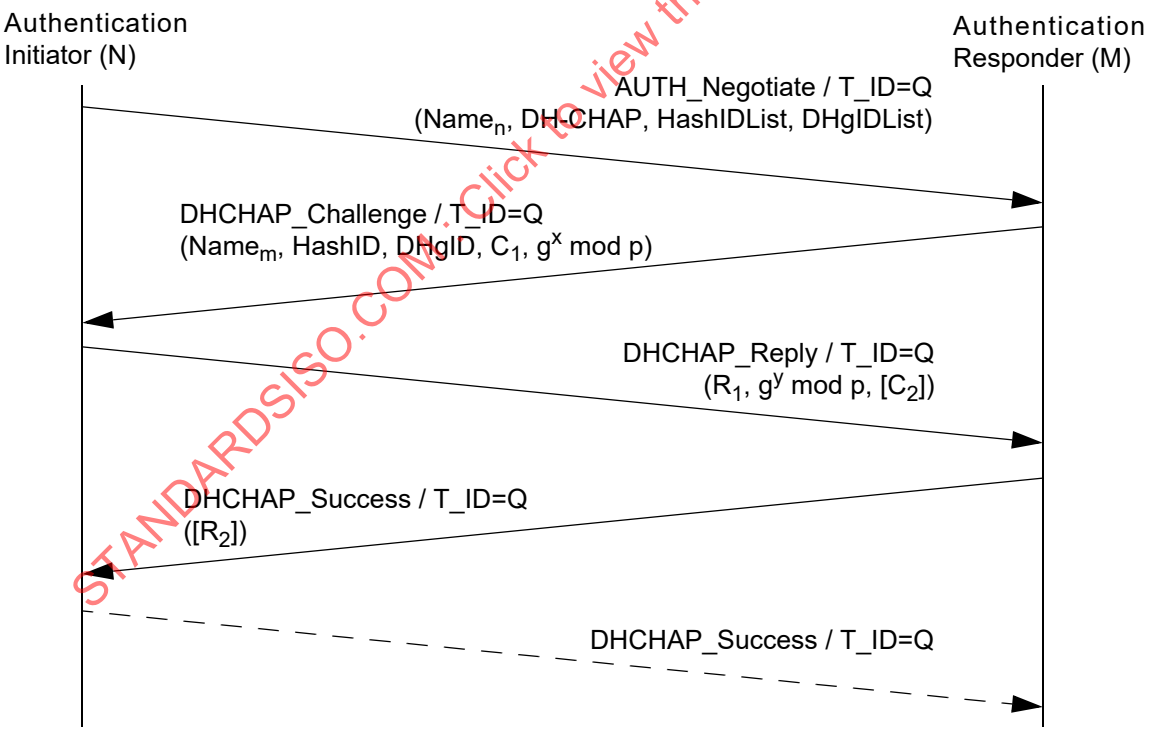

Figure 7 – A DH-CHAP Protocol Transaction Example

Table 20 – Mathematical Notation for DH-CHAP

Symbols	Description
p, g	The modulus (p) and generator (g) of the chosen DH Group (see table 15). All computations are performed modulo p
K_n, K_m	Administratively configured secrets
x, y	Nonces, used as random numbers
C_1, C_2	Challenge Values
R_1, R_2	Reply Values
C_{a1}, C_{a2}	Augmented Challenge Values
T_i	The least significant byte of the Transaction Identifier
H( )	One-way hash function
K_S	Computed session key

The DH-CHAP protocol proceeds as follows:

1) Operations shall start by negotiating the hash functions and the Diffie-Hellman group identifier to be used in the Authentication process with the AUTH_Negotiate message (see 5.3.2). In the AUTH_Negotiate message, the Authentication Initiator shall send its own name, and the list of the Authentication Protocols and the associated parameters to be used for the remainder of this Authentication Transaction. For the DH-CHAP protocol the parameters are the list of hash functions (e.g., SHA-1, MD5) and the list of Diffie-Hellman Group Identifiers that may be used (see 5.4.2);

NOTE 9 – Although all implementations compliant with this standard are required to support DH-CHAP with a NULL DH algorithm, in certain environments the administrator may require different Authentication Protocols to be used. The administrator may disable the use of DH-CHAP or of DH-CHAP without DH algorithm, and in such a case DH-CHAP or the NULL DH group identifier, respectively, may not be included in the AUTH_Negotiate message.

2) The Authentication Responder shall reply with a DHCHAP_Challenge message (see 5.4.3) carrying the name of the Authentication Responder, the hash function and the DH group identifier selected among the ones proposed by the Authentication Initiator, a Challenge Value C_1, and the DH parameter. If the responder selects a NULL DH group identifier, the DH portion of the DH-CHAP protocol shall not be used, and the Authentication Protocol is equivalent to a CHAP transaction;

3) The Authentication Initiator shall send a DHCHAP_Reply message (see 5.4.4) with the response R_1 to the challenge C_1, and its own DH parameter. The DH Value Length shall be set to zero if the Authentication Responder has sent a NULL group identifier in the DHCHAP_Challenge message. To request bidirectional authentication, the DHCHAP_Reply message shall have the Challenge Value Length set to a non-zero value and the Challenge Value shall contain a Challenge Value C_2 that differs from the Challenge Value C_1 received in the DHCHAP_Challenge message;

4) If the Authentication succeeds, the Authentication Responder shall reply with a DHCHAP_Success message (see 5.4.5), to indicate that the Authentication Initiator has been authenticated. If the Authentication Initiator requested authentication of the Authentication Responder, the DHCHAP_Success message shall have the Response Value Length field set to a non-zero value and the Response Value field shall contain a response value R_2 for the Challenge Value C_2 received in the DHCHAP_Reply message. If the Authentication fails, the Authentication Responder shall reply with an appropriate AUTH_Reject message (see 5.4.4), and shall terminate the communication; and

5) The Authentication Transaction may end here, unless bidirectional Authentication has been requested. In this case, as shown by the dashed arrow in figure 7, if the Authentication succeeds, the Authentication Initiator shall send a DHCHAP_Success message with no Response field to confirm that the Responder has been authenticated. If the Authentication fails, the Authentication Initiator shall send an appropriate AUTH_Reject message (see 5.4.5), and shall terminate the communication.

NOTE 10 – The DH-CHAP protocol does not use the AUTH_Done message.

5.4.2 AUTH_Negotiate DH-CHAP Parameters

5.4.2.1 Overview

The Authentication Protocol Parameters in the AUTH_Negotiate message for DH-CHAP are formatted as shown in table 21.

Table 21 – AUTH_Negotiate DH-CHAP Protocol Parameters

Item	Size (Bytes)
Parameter #1 = HashList	variable
Parameter #2 = DHgIDList	variable
...	...
Parameter #k	variable

Parameter: Each parameter shall be formatted as shown in table 22.

Table 22 – AUTH_Negotiate DH-CHAP Parameter Format

Item	Size (Bytes)
Parameter Tag	2
Parameter Word Count	2
Parameter Value	variable

Parameter Tag: Identifies the format of the Parameter Value. Parameter Tags are shown in table 23.

Table 23 – AUTH_Negotiate DH-CHAP Parameter Tags

Parameter Tag	Parameter Value Format
0001h	HashList (see 5.4.2.2)
0002h	DHgIDList (see 5.4.2.3)
all others	Reserved

Parameter Word Count: Indicates the number of words composing the Parameter Value.

Parameter Value: Contains the parameter value.

5.4.2.2 HashList Parameter

The HashList parameter shall be included as the first parameter in the AUTH_Negotiate DH-CHAP Protocol Parameters (see table 21).

Hashlist Parameter Word Count: Shall be set to the number of hash functions proposed by the Authentication Initiator. Each hash function identifier is encoded into one word.

Hashlist Parameter Value: Each word of this field contains an identifier of a proposed hash function, in order of preference. The first word contains the most preferred, the last word contains the least preferred hash function. The list of defined hash function identifiers is shown in table 14.

Support for the MD5 hash function is mandatory for DH-CHAP.

NOTE 11 – Although all implementations compliant with this standard are required to support the MD5 hash function, in certain environments the administrator may require different hash functions to be used.

5.4.2.3 DHgIDList Parameter

The DHgIDList parameter shall be included as the second parameter in the AUTH_Negotiate DH-CHAP Protocol Parameters (see table 21).

DHgIDList Parameter Word Count: Shall be set to the number of proposed DH groups. Each DH group identifier is encoded into one word.

DHgIDList Parameter Value: Each word of this field contains a DH group identifier (see table 15) proposed by the Authentication Initiator, in order of preference.

Support for the NULL DH-CHAP algorithm (i.e., DH group identifier = 0000 0000h) is mandatory.

Support for the DH group 1 536 (i.e., DH group identifier = 0000 0003h) is mandatory for DH-CHAP implementations supporting a non-NULL DH-CHAP algorithm.

NOTE 12 – Although all implementations compliant with this standard are required to support the NULL DH-CHAP algorithm or the DH group 1536, in certain environments the administrator may require different functions to be used.

5.4.3 DHCHAP_Challenge Message

The DHCHAP_Challenge message is sent from the Authentication Responder to the Authentication Initiator. The payload of the DHCHAP_Challenge message is shown in table 24.

Table 24 – DHCHAP_Challenge Message Payload

Item	Size (Bytes)
Authentication Responder Name	variable
Hash Identifier	4
DH Group Identifier	4
Challenge Value Length	4
Challenge Value	variable
DH Value Length	4
DH Value	variable

Authentication Responder Name: Shall be set to the Authentication Responder Name (see 5.3.3).

Hash Identifier: Shall be set to the identifier of the selected hash function among those proposed in the AUTH_Negotiate message. The Authentication Responder shall select a hash function in accord with applicable policy of the Authentication Responder. This policy may require that the preference expressed by the Authentication Initiator in the AUTH_Negotiate message be honored.

DH Group Identifier: Shall be set to the DH Group Identifier (see table 15) selected for this Authentication Transaction. The Authentication Responder shall select a DH Group Identifier in accord with applicable policy of the Authentication Responder. This policy may require that the preference expressed by the Authentication Initiator in the AUTH_Negotiate message be honored. If this field is set to zero, the DH portion of the DH-CHAP protocol shall not be performed.

Challenge Value Length: Shall be set to the length in bytes of the Challenge Value. This length shall have the value specified in table 14 for the selected hash function. If the Challenge Value Length does not match the value specified in table 14, the Authentication Initiator shall reply with an AUTH_Reject having Reason Code 'Authentication Failure' and Reason Code Explanation 'Incorrect Payload'.

Challenge Value: Shall be set to a random challenge value C_1 (see C.1). Each challenge value should be unique and unpredictable, since repetition of a challenge value in conjunction with the same secret may reveal information about the secret or the correct response to this challenge. The algorithm for generating the challenge value is outside the scope of this standard. Randomness of the challenge value is crucial to the security of the protocol (see C.1).

DH Value Length: Diffie-Hellman parameter length. This length shall be a multiple of 4. If the DH group identifier is set to zero (i.e., NULL DH algorithm), this field shall be set to zero. Otherwise, it shall be set to the length in bytes of the DH Value.

DH Value: Diffie-Hellman parameter. If the DH Value Length is set to zero, this field is not present. The DH Value shall be set to the value of g^x mod p, where x is a random number selected by the Authentication Responder, and p and g shall have the values indicated in table 15, based on the selected DH group identifier. A value of zero is illegal for the DH Value. If the DH Value is set to zero, the Authentication Initiator shall reply with an AUTH_Reject having Reason Code 'Authentication Failure' and Reason Code Explanation 'Incorrect Payload'.

5.4.4 DHCHAP_Reply Message

The DHCHAP_Reply message is sent from the Authentication Initiator to the Authentication Responder. The Authentication Initiator may request authentication of the Authentication Responder to enable bidirectional authentication, including a DH-CHAP challenge C_2 in this message. The challenge C_2 shall be different from the challenge C_1 received in the DHCHAP_Challenge message.

When a DHCHAP_Reply is received, the Authentication Responder shall verify the response R_1 using the negotiated hash function. If the response R_1 is not verified, the Authentication Responder shall reply with an AUTH_Reject message with Reason Code 'Authentication Failure' and Reason Code Explanation 'Authentication Failed', and shall terminate the communication.

The Authentication Responder shall check the challenge C_2, if any, to verify it is different from the challenge C_1 the Authentication Responder previously sent. If C_2 is equal to C_1, the Authentication Responder shall reply with an AUTH_Reject message with Reason Code 'Authentication Failure' and Reason Code Explanation 'Incorrect Payload', and shall terminate the communication.

The payload of the DHCHAP_Reply message is shown in table 25.

Table 25 – DHCHAP_Reply Message Payload

Item	Size (Bytes)
Response Value Length	4
Response Value	variable
DH Value Length	4
DH Value	variable
Challenge Value Length	4
Challenge Value	variable

Response Value Length: Shall be set to the length in bytes of the Response Value. This length shall have the value specified in table 14 for the selected hash function. If the Response Value Length does not match the value specified in table 14, the Authentication Responder shall reply with an AUTH_Reject having Reason Code 'Authentication Failure' and Reason Code Explanation 'Incorrect Payload'.

Response Value: DH-CHAP response R_1. The value of R_1 is computed using the hash function H( ) selected by the HashID parameter in the DHCHAP_Challenge message, and the augmented challenge C_{a1}. If the NULL DH group has been selected, the augmented challenge C_{a1} is equal to the challenge C_1 received from the Authentication Responder (i.e., $C_{a1} = C_1$). If a non-NULL DH group has been selected, the augmented challenge is computed applying the hash function H( ) to the concatenation of the challenge C_1, and the ephemeral DH key resulting from the combination of the random value y selected by the Authentication Initiator with the DH parameter (i.e., g^x mod p) received from the Authentication Responder (i.e., $C_{a1} = H(C_1 \,||\, (g^x \bmod p)^y \bmod p) = H(C_1 \,||\, g^{xy} \bmod p)$). The value of R_1 shall be computed applying the hash function H( ) to the concatenation of the least significant byte of the Transaction Identifier T_i, the secret K_n associated with the Authentication Initiator, and the augmented challenge C_{a1} (i.e., $R_1 = H(T_i \,||\, K_n \,||\, C_{a1})$).

DH Value Length: Diffie-Hellman parameter length. This length shall be a multiple of 4. If the DH group identifier is set to zero (i.e., NULL DH algorithm), this field shall be set to zero. Otherwise, it shall be set to the length in bytes of the DH Value.

DH Value: Diffie-Hellman parameter. If the DH Value Length is set to zero, this field is not present. The DH Value shall be set to the value of g^y mod p, where y is a random number selected by the Authentication Initiator, and p and g shall have the values indicated in table 15, based on the selected DH group identifier. A value of zero is illegal for the DH Value. If the DH Value is set to zero, the Authentication Responder shall reply with an AUTH_Reject having Reason Code 'Authentication Failure' and Reason Code Explanation 'Incorrect Payload'.

Challenge Value Length: If the Authentication Initiator does not require bidirectional Authentication, this field shall be set to zero. Otherwise, this field shall be set to the length in bytes of the Challenge Value. In this case the length shall have the value specified in table 14 for the selected hash function. If the Challenge Value Length does not match the value specified in table 14, the Authentication Responder shall reply with an AUTH_Reject having Reason Code 'Authentication Failure' and Reason Code Explanation 'Incorrect Payload'.

Challenge Value: Shall be set to a random challenge value C_2 (see C.1). Each challenge value should be unique and unpredictable, since repetition of a challenge value in conjunction with the same secret may

reveal information about the secret or the correct response to this challenge. The algorithm for generating the challenge value is outside the scope of this standard. Randomness of the challenge value is crucial to the security of the protocol (see C.1). If the corresponding length field is set to zero, this field is not present.

5.4.5 DHCHAP_Success Message

The DHCHAP_Success message is sent from the Authentication Responder to the Authentication Initiator. If bidirectional Authentication is requested, the DHCHAP_Success message shall also be sent from the Authentication Initiator to the Authentication Responder.

When sent from the Authentication Responder to the Authentication Initiator, with bidirectional authentication requested, the DHCHAP_Success message contains the response R_2 to the challenge C_2 received in the DHCHAP_Reply message. In this case, when a DHCHAP_Success message is received, the Authentication Initiator shall verify the response R_2 using the selected hash function. If the response R_2 is not verified, the Authentication Initiator shall reply with an AUTH_Reject message with a Reason Code 'Authentication Failure' and Reason Code Explanation 'Authentication Failed', and shall terminate the communication.

When sent from the Authentication Responder to the Authentication Initiator without bidirectional authentication, or when sent from the Authentication Initiator to the Authentication Responder, the DHCHAP_Success message indicates that the authentication has been completed successfully.

The payload of the DHCHAP_Success message is shown in table 26.

Table 26 – DHCHAP_Success Message Payload

Item	Size (Bytes)
Response Value Length	4
Response Value	variable

Response Value Length: If the Authentication Initiator did not request Authentication of the Responder (i.e., bidirectional Authentication), or when the DHCHAP message is sent from the Authentication Initiator to the Authentication Responder, this field shall be set to zero to indicate that no response is conveyed. If the Authentication Initiator did request Authentication of the Responder, this field shall be set to the length in bytes of the Response Value. In this case the length shall have the value specified in table 14 for the selected hash function. If the Response Value Length does not match the value specified in table 14, the Authentication Initiator shall reply with an AUTH_Reject having Reason Code 'Authentication Failure' and Reason Code Explanation 'Incorrect Payload'.

Response Value: DH-CHAP response R_2. The value of R_2 is computed using the hash function H( ) selected by the HashID parameter of the DHCHAP_Challenge message, and the augmented challenge C_{a2}. If the NULL DH group has been selected, the augmented challenge C_{a2} is equal to the challenge C_2 received from the Authentication Initiator (i.e., $C_{a2} = C_2$). If a non-NULL DH group has been selected, the augmented challenge is computed applying the hash function H( ) to the concatenation of the challenge C_2, and the ephemeral DH key resulting from the combination of the random value x selected by the Authentication Responder with the DH parameter (i.e., g^y mod p) received from the Authentication Initiator (i.e., $C_{a2} = H(C_2 \,||\, (g^y \bmod p)^x \bmod p) = H(C_2 \,||\, g^{xy} \bmod p)$). The value of R_2 shall be computed applying the hash function H( ) to the concatenation of the least significant byte of the Transaction Identifier T_i, the secret K_m associated with the Authentication Responder, and the augmented challenge C_{a2} (i.e., $R_2 = H(T_i \,||\, K_m \,||\, C_{a2})$).

5.4.6 Key Generation for the Security Association Management Protocol

The DH-CHAP protocol, when used with a non-NULL DH algorithm, enables the Security Association Management Protocol by generating a session key K_S that shall be used by the SA Management Transaction (see 6.7). When the DH group used in the DH-CHAP transaction is NULL, the results from the DH-CHAP transaction shall not be used to generate a session key K_S for the SA Management Transaction.

The session key K_S shall be computed as the complete hash, with no padding, of the shared key (i.e., g^{xy} mod p) generated during the DH-CHAP transaction (i.e., $K_S = H(g^{xy} \text{ mod } p)$). The hash function H( ) is selected by the Authentication Responder in the HashID field of the DHCHAP_Challenge message. The size of the session key K_S is determined by the selected hash function, as shown in table 14.

5.4.7 Reuse of Diffie-Hellman Exponential

DH-CHAP implementations may reuse a DH exponential (see 6.8.10).

The basic risk in allowing reuse of a DH exponential (g^x or g^y) is replay of a prior authentication sequence based on the attacker reusing the other exponential.

For DH-CHAP, replay is prevented by the requirement that any challenge be randomly generated from scratch (see 5.4.8). An implementation that fails to do this bears the consequences of its failure, as when a challenge is reused, a responder is not able to prevent replay of an old authentication whether or not DH is used, and the security compromise that results from the replay attack is of the system that failed to generate the challenge properly.

5.4.8 DH-CHAP Security Considerations

DH-CHAP secrets shall be randomly generated (see C.1). Support for the following is mandatory:

a) 128 bit random secrets;

b) generation of such secrets; and

c) acceptance of such secrets from an external source.

These requirements protect DH-CHAP from off-line dictionary attacks based on either passive observation of the communication or active attempts to obtain information sufficient to mount an off-line dictionary attack.

Secrets shall be an integral number of bytes and at least 96 bits in size and based on at least 96 bits of independent randomness (see C.1). Configuration of secrets of less than 96 bits shall not be permitted (e.g., the configuration interface may reject the attempt, or force the secret to be at least 96 bits in size). Any secret shall be used for authentication of only one Fibre Channel entity.

For DH-CHAP with a null DH group, upon receipt of a response (i.e., R_1 or R_2), the receiving entity shall compute the response to the same challenge using the entity's own secret. If the computed response is equal to the received response, the received response shall cause an authentication failure. This ensures that the same secret is not used for authentication in both directions. The reason for this authentication failure (i.e., the same secret is used for authentication in both directions) should be logged. This computing and comparing of responses should also be performed for DH-CHAP with a non-NULL DH group.

For example, an attack prevented by computing and comparing responses is:

1) An attacker wants to impersonate entity A to entity B, and knows that a single secret is used for both directions of entity A-B authentication.

2) The attacker convinces entity B to open two connections to the attacker, and the attacker identifies itself as entity A on both connections.

3) The attacker issues a DH-CHAP challenge on connection 1, waits for entity B to respond, and then reflects entity B's challenge as the initial challenge to entity B on connection 2.

4) If entity B does not check for the reflection across connections, entity B's response on connection 2 enables the attacker to impersonate entity A on connection 1, even though the attacker does not know the entity A-B DH-CHAP secret.

Use of a non-NULL DH group also prevents this attack because an attacker is not able to force two different DH algorithms to produce the same results.

An entity shall not reuse the challenge sent by another entity for the opposite direction of a bidirectional authentication. All entities shall check for this condition and cause an authentication failure if it occurs.

When a Fibre Channel entity authenticates itself to counterparts in multiple administrative domains, a different secret should be used for each administrative domain to avoid propagating security compromises across domains.

Within a single administrative domain, a single DH-CHAP secret may be used for authentication of an entity to multiple other entities when an external server (e.g., RADIUS) is used to verify DH-CHAP responses.

If an external response verification server (e.g., RADIUS) is not used, employing a single DH-CHAP secret for authentication of an entity to multiple other entities results in all such entities knowing the original entity's secret. Any such entity is able to impersonate any other entity whose secret it knows. If an attacker compromises any of the involved entities and obtains its known secrets, the attacker is able to impersonate all of the involved entities. Separate DH-CHAP secrets should be used by an entity for authentication to each other entity to mitigate such risks when they are of concern.

When the used DH group is not NULL, DH-CHAP is vulnerable to a denial of service attack if the attacker initiates concurrent authentication from a sufficient number of different S_IDs, because the attacker may cause the responder to compute $g^x \bmod p$ without the attacker engaging in any exponentiation.

This vulnerability is not present in the cases of E_Port to E_Port authentication, E_Port to B_Port authentication and Nx_Port to Fx_Port authentication because S_ID and D_ID have fixed values. For Nx_Port to Nx_Port authentication the fact that a Port Login is required before performing authentication (see 5.10) mitigates the issue, because the attacker has to be able to respond from any S_ID used to mount the attack. Implementations that may exhibit non-responsive behavior under overload should limit the number of simultaneous authentication computations by issuing an LS_RJT with Reason Code 'Logical Busy' (see 5.10.3).

5.5 FCAP Protocol

5.5.1 Protocol Operations

FCAP is an authentication and key management protocol between an Authentication Initiator and an Authentication Responder. FCAP is based on digital Certificates. When the FCAP protocol successfully completes, the Authentication Initiator and Authentication Responder are mutually authenticated and may share a session key. The FCAP design is based on IKE and IKEv2 authentication based on Certificates and signatures (see RFC 5996).

In order to authenticate with the FCAP protocol, each entity, identified by a unique Name, shall be provided with a digital Certificate associated with that Name, the private/public key pair that corresponds to the Certificate, and with the Certificate of the signing Certification Authority. To authenticate another entity, an entity is required to be provided with the Certificate of the associated Certification Authority.

Entities shall support the ability to configure at least four Root Certificates and shall be able to validate a received Certificate against the corresponding Root Certificate. See RFC 5280 for how to perform Certificate validation. Entities shall not accept Certificates that may not be validated against a configured Root Certificate. A Fabric should have at least one Root Certificate that is configured on all participating entities. Support for Certificate chains and verification of Certificate chains containing more than one Certificate is optional.

The authenticity of Root Certificates is critical to the security of a Certificate based authentication infrastructure, therefore the configuration and distribution of them should be carefully controlled.

Entities shall support base-64 encoded X.509 Certificates and may support other Certificate formats. Certificates used by FCAP shall be in base-64 encoded X.509 format.

NOTE 13 – The FC SA Management protocol allows other Certificate formats.

Entities shall be able to access a Certificate revocation list (CRL) for each configured Root Certificate, if one is available from the CA. Certificates on the CRL shall be considered invalid. The mechanisms to provide CRLs or CRL access to an entity are outside the scope of this standard. Entities may support online Certificate validation mechanisms such as OCSP. Entities shall support the ability to reject a Certificate for which neither an online Certificate validation mechanism nor a CRL are available.

Table 27 – Mathematical Notation for FCAP

Symbols	Description
p, g	The modulus (p) and generator (g) of the chosen DH Group (see table 15). All computations are performed modulo p
x, y	Random numbers
C_a, C_b	Digital Certificates
R_a, R_b	Random nonces (see C.1)
$S_a( )$, $S_b( )$	Digital signature functions
H( )	One-way hash function
K_S	Computed session key

An example of a FCAP protocol transaction is shown in figure 8, with the notation shown in table 27.

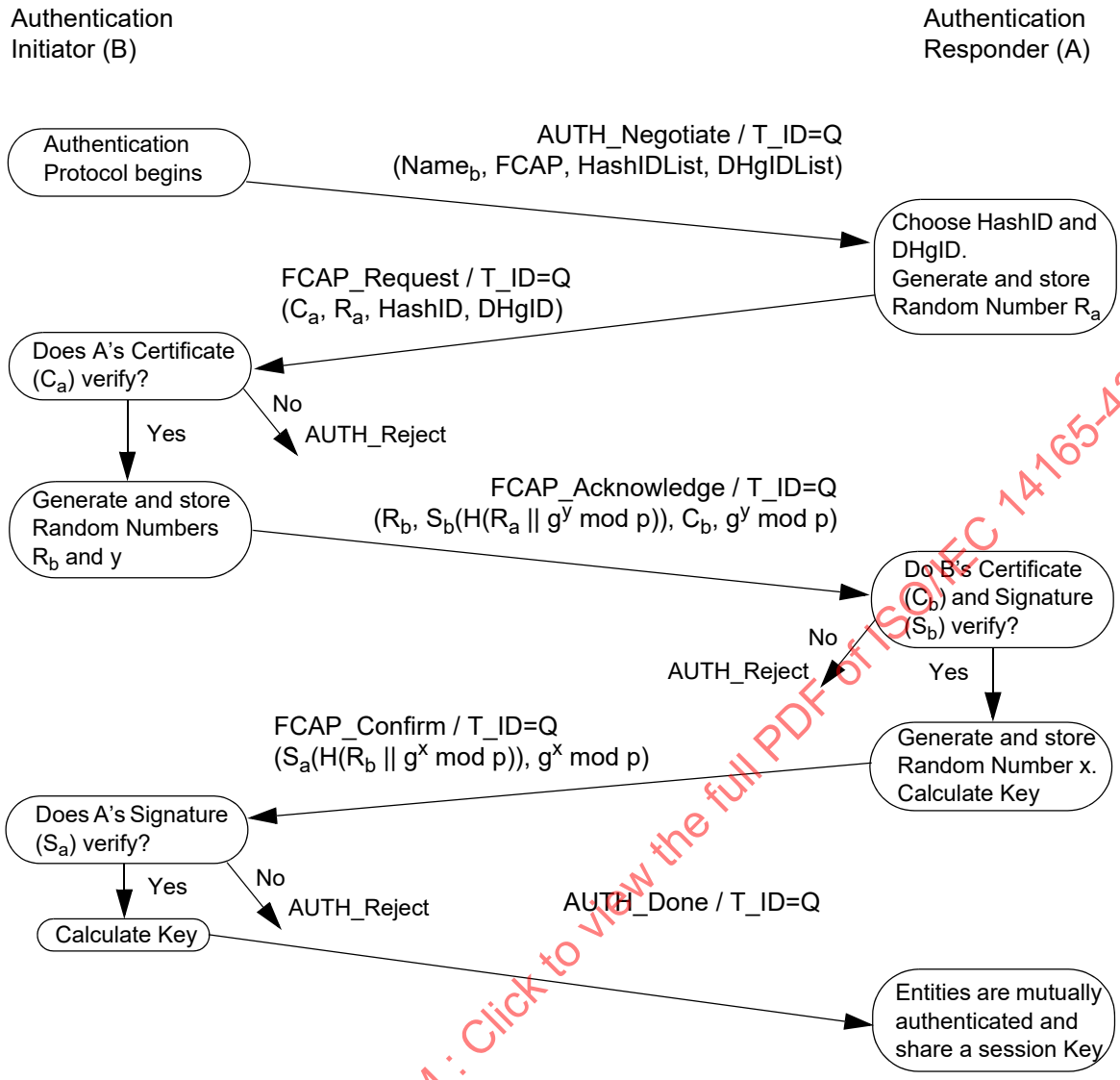

Figure 8 – A FCAP Protocol Transaction Example

The FCAP protocol proceeds as follows:

1) Operations shall start by negotiating the hash functions and the Diffie-Hellman group identifier to be used in the Authentication process with the AUTH_Negotiate message (see 5.3.2). In the AUTH_Negotiate message, the Authentication Initiator shall send its own name, and the list of the Authentication Protocols and the associated parameters to be used for the remainder of this Authentication Transaction. For the FCAP protocol the parameters are the list of hash functions (e.g., SHA-1) and the list of Diffie-Hellman Group Identifiers that may be used (see 5.4.2);

2) On receiving the AUTH_Negotiate message, the Authentication Responder shall choose the hash function and the DH group identifier to be used in the Authentication Transaction among those proposed by the Authentication Initiator, and shall generate a new random nonce R_a. Then the Authentication Responder shall reply to the AUTH_Negotiate message with a FCAP_Request

message (see 5.5.3), carrying the nonce R_a, and the chosen HashID and DHgID, along with the Certificate C_a of the Authentication Responder;

3) On receiving the FCAP_Request message, the Authentication Initiator shall verify the Certificate C_a of the Authentication Responder with the Certificate of the corresponding Certification Authority. If the verification fails, the Authentication Initiator shall reply with an AUTH_Reject message with Reason Code 'Authentication Failure' and Reason Code Explanation 'Authentication Failed', and shall terminate the communication. If the verification succeeds, the Authentication Initiator shall generate a new random nonce R_b, and a random number y, then reply to the Authentication Responder with a FCAP_Acknowledge message (see 5.5.4) carrying the nonce R_b, its Certificate C_b, the Diffie-Hellman parameter g^y mod p, and its signature S_b. The signature of the Authentication Initiator S_b shall be generated by computing, with the negotiated hash function H( ), the hash of the received nonce R_a concatenated with the Diffie-Hellman parameter g^y mod p (i.e., R_a || g^y mod p), then by encrypting the hash with the RSA private key of the Authentication Initiator;

NOTE 14 – The Diffie-Hellman parameter g^y mod p is included in the signature to protect the signature's integrity.

4) On receiving the FCAP_Acknowledge message, the Authentication Responder shall verify both the Certificate C_b and the Signature S_b of the Authentication Initiator. The Certificate C_b shall be verified with the Certificate of the corresponding Certification Authority. The identity of the Authentication Initiator shall be taken from the Certificate C_b. The Signature S_b shall be verified after the Certificate C_b is verified. To verify the Signature S_b, the Authentication Responder shall decrypt the Signature S_b with the RSA public key of the Authentication Initiator, obtained from the verified Certificate C_b, then perform a hash of its nonce R_a and the received Diffie-Hellman parameter g^y mod p using the negotiated hash function. If the hash value calculated by the Authentication Responder is equal to the hash resulting from the decryption of the received signature S_b, the signature is verified successful. If the signature verification is successful, the Authentication Responder shall generate a random number x, then calculate the session key K_S (see 5.5.6). If either the Certificate or signature verifications fail, then the Authentication Responder shall reply with an AUTH_Reject message with Reason Code 'Authentication Failure' and Reason Code Explanation 'Authentication Failed', and shall terminate the communication. If both the Certificate and signature verifications complete successfully, the Authentication Responder shall generate its Signature S_a by computing, with the negotiated hash function H( ), the hash of the received nonce R_b concatenated with the Diffie-Hellman parameter g^x mod p (i.e., R_b || g^x mod p), then by encrypting the hash with its RSA private key. Then the Authentication Responder shall send its Signature S_a and its Diffie-Hellman parameter g^x mod p to the Authentication Initiator in a FCAP_Confirm message (see 5.5.5); and

5) On receiving the FCAP_Confirm message, the Authentication Initiator shall verify the Signature S_a of the Authentication Responder. To verify the Signature S_a, the Authentication Initiator shall decrypt the Signature S_a with the RSA public key of the Authentication Responder, then perform a hash of its nonce R_b and the received Diffie-Hellman parameter g^x mod p using the negotiated hash function. If the hash value calculated by the Authentication Initiator is equal to the hash resulting from the decryption of the received signature S_a, the signature is verified successful. If the signature verification is successful, the Authentication Initiator shall calculate the session key K_S (see 5.5.6). If the signature verification fails, then the Authentication Initiator shall reply with an AUTH_Reject message with Reason Code 'Authentication Failure' and Reason Code Explanation 'Authentication Failed', and shall terminate the communication. If signature verification is successful, then the Authentication Initiator shall send an AUTH_Done message (see 5.3.8).

5.5.2 AUTH_Negotiate FCAP Parameters

5.5.2.1 Overview

The Authentication Protocol Parameters in the AUTH_Negotiate message for FCAP are formatted as shown in table 28.

Table 28 – AUTH_Negotiate FCAP Protocol Parameters

Item	Size (Bytes)
Parameter #1 = HashList	variable
Parameter #2 = DHgIDList	variable
...	...
Parameter #k	variable

Parameter: Each parameter shall be formatted as shown in table 29.

Table 29 – AUTH_Negotiate FCAP Parameter Format

Item	Size (Bytes)
Parameter Tag	2
Parameter Word Count	2
Parameter Value	variable

Parameter Tag: Identifies the format of the Parameter Value. Parameter Tags are shown in table 30.

Table 30 – AUTH_Negotiate FCAP Parameter Tags

Parameter Tag	Parameter Value Format
0001h	HashList (see 5.5.2.2)
0002h	DHgIDList (see 5.5.2.3)
all others	Reserved

Parameter Word Count: Indicates the number of words composing the Parameter Value.

Parameter Value: Contains the parameter value.

5.5.2.2 HashList Parameter

The HashList parameter shall be included as the first parameter in the AUTH_Negotiate FCAP Protocol Parameters (see table 28).

Hashlist Parameter Word Count: Shall be set to the number of hash functions proposed by the Authentication Initiator. Each hash function identifier is encoded into one word.

Hashlist Parameter Value: Each word of this field contains an identifier of a proposed hash function, in order of preference. The first word contains the most preferred, the last word contains the least preferred hash function. The list of defined hash function identifiers is shown in table 14.

Support for the SHA-1 hash function is mandatory for FCAP. The MD5 hash function shall not be used with FCAP.

5.5.2.3 DHgIDList Parameter

The DHgIDList parameter shall be included as the second parameter in the AUTH_Negotiate FCAP Protocol Parameters (see table 28).

DHgIDList Parameter Word Count: Shall be set to the number of proposed DH groups. Each DH group identifier is encoded into one word.

DHgIDList Parameter Value: Each word of this field contains a DH group identifier (see table 15) proposed by the Authentication Initiator, in order of preference.

Support for the NULL FCAP algorithm (i.e., DH group identifier = 0000 0000h) is mandatory. When selected, the FCAP protocol does not provide a session key K_S between the Authentication Initiator and the Authentication Responder at the end of the Authentication Transaction (see 5.5.6).

Support for the DH group 1 536 (i.e., DH group identifier = 0000 0003h) is mandatory for FCAP implementations supporting a non-NULL FCAP algorithm.

5.5.3 FCAP_Request Message

5.5.3.1 Message Format

The FCAP_Request message is sent from the Authentication Responder to the Authentication Initiator. The format of the FCAP_Request message payload is shown in table 31.

Table 31 – FCAP_Request Message Payload

Item	Size (Bytes)
Authentication Responder Certificate	variable
Authentication Responder Nonce	variable
Hash Identifier	4
DH Group Identifier	4

Authentication Responder Certificate: Shall contain the Certificate C_a of the Authentication Responder. The format of the Certificate C_a is defined in 5.5.3.2.

Authentication Responder Nonce: This field shall contain the nonce R_a created by the Authentication Responder. The format of the nonce R_a is defined in 5.5.3.3.

Hash Identifier: Shall be set to the identifier of the selected hash function among those proposed in the AUTH_Negotiate message. The Authentication Responder shall select a hash function in accord with applicable policy of the Authentication Responder. This policy may require that the preference expressed by the Authentication Initiator in the AUTH_Negotiate message be honored.

DH Group Identifier: Shall be set to the DH Group Identifier (see table 15) selected for this Authentication Transaction. The Authentication Responder shall select a DH Group Identifier in accord with applicable policy of the Authentication Responder. This policy may require that the preference expressed by the Authentication Initiator in the AUTH_Negotiate message be honored. If this field is set to zero, the DH portion of the FCAP protocol shall not be performed.

5.5.3.2 FCAP Certificate Format

The FCAP Certificate format is shown in table 32.

Table 32 – FCAP Certificate Format

Item	Size (bytes)
Certificate Identifier	2
Certificate Length	2
Certificate Value	variable

Certificate Identifier: Identifies the format of the Certificate. Certificate identifiers are shown in table 33.

Certificate Length: Indicates the total length in bytes of the Certificate Value. Length values are shown in table 33.

Table 33 – Certificate Formats

Certificate Identifier	Certificate Type	Certificate Length (bytes)
0001h	FCAP X.509	variable
all others	Reserved	

Certificate Value: Contains the Certificate value.

NOTE 15 – Certificates used by FCAP are in base-64 encoded X.509 format.

FCAP X.509 Certificate Value:
RFC 5280 and RFC 6818 define the Certificate syntax for Certificates consistent with X.509v3. Certificates for FCAP shall use the RFC 5280 and RFC 6818 Certificate syntax as described in table 34.

Table 34 – FCAP usage of X.509v3 Certificate fields (part 1 of 2)

Certificate Field	Usage	Description
signatureAlgorithm	Mandatory	Support for RSA-SHA1 is mandatory (see RFC 3279)[f]. Support for RSA2048-SHA256, RSA2048-SHA384, and RSA2048-SHA512 is optional (see FIPS PUB 180-4). A Certificate may be rejected if another algorithm is used.
signatureValue	Mandatory	As per RFC 5280 and 6818.
version	Mandatory	Version 3 is required for Certificates with extensions. Legacy use of version 1 is prohibited.
serialNumber	Mandatory	As per RFC 5280 and 6818.
signature	Mandatory	Support for RSA-SHA1 is mandatory (see RFC 3279)[f]. Support for RSA2048-SHA256, RSA2048-SHA384, and RSA2048-SHA512 is optional (see FIPS PUB 180-4). A Certificate may be rejected if another algorithm is used.
issuer	Mandatory	As per RFC 5280 and 6818.

Table 34 – FCAP usage of X.509v3 Certificate fields (part 2 of 2)

Certificate Field	Usage	Description
validity	Mandatory	As per RFC 5280 and 6818.
subject	Mandatory	The subject field, if not empty[c], contains a Distinguished Name (DN) in ITU X.501 Name format. This format is a sequence of attributes, each of them consisting of a (type, value) pair. The type is expressed as an OID. One of the attributes is the CommonName attribute, whose type is OID 2.5.4.3. The CommonName attribute shall be used to represent a Fibre Channel Name_Identifier. The value of the CommonName attribute shall be a Fibre Channel Name_Identifier represented as 23 hexadecimal UTF-8 characters in colon separated format (e.g., 10:00:00:60:69:90:0F:A7).[d]
subjectPublicKeyInfo	Mandatory	As per RFC 5280 and 6818.
issuerUniqueID	Optional	May be ignored.[a]
subjectUniqueID	Optional	May be ignored.[a]
Subject Alternative Name extension	Invocable[b]	The Subject Alternative Name extension field contains a sequence of GeneralName elements, that support multiple formats. Exactly one of the GeneralName elements shall represent a Fibre Channel Name_Identifier, and the otherName format shall be used for this purpose. The otherName format consists of a (type-id, value) pair. The type-id field of the otherName shall contain the OID 1.2.840.114402.1.1.1[e] to indicate that the otherName represents a Fibre Channel Name_Identifier. The value field of the otherName shall contain a Fibre Channel Name_Identifier represented as 23 hexadecimal UTF-8 characters in colon separated format (e.g., 10:00:00:60:69:90:0F:A7).[d]
Key Usage extension	Invocable	As per RFC 5280 and 6818.
Other critical extensions	Prohibited	The Certificate shall be rejected.
Other extensions	Optional	May be ignored.[a]

[a] This field should not be generated in a Certificate. If it is received in a Certificate, its value shall be parsed and ignored, except for applicable signature calculations.

[b] This field is mandatory for version 3 Certificates.

[c] This field may be empty for version 3 Certificates and shall not be empty for version 1 Certificates.

[d] The following terms are ASN.1 syntax elements (see RFC 5280 and 6818): Name, GeneralName, otherName, type, type-id, and value.

[e] This OID represents: { iso(1) member-body(2) us(840) INCITS(114402) T11(1) element-identification(1) Name-Identifier-format-1(1) }.

[f] For compliance with NIST SP 800-131A, use of RSA-SHA1 is prohibited and use of RSA2048-SHA256, RSA2048-SHA384, and RSA2048-SHA512 is allowed.

5.5.3.3 FCAP Nonce Format

The FCAP nonce format is shown in table 35.

Table 35 – FCAP Nonce Format

Item	Size (Bytes)
Nonce Identifier	2
Nonce Length	2
Nonce Value	variable

Nonce Identifier: Identifies the format of the nonce. Nonce Identifiers are shown in table 36.

Nonce Length: Indicates the total length in bytes of the Nonce Value. Length values are shown in table 36.

Table 36 – Nonce Formats

Nonce Identifier	Nonce Type	Nonce Length (bytes)
0001h	Binary String	256
all others	Reserved	

Nonce Value: Contains a random value of the type shown in table 36.

5.5.4 FCAP_Acknowledge Message

5.5.4.1 Message Format

The FCAP_Acknowledge message is sent from the Authentication Initiator to the Authentication Responder. The format of the FCAP_Acknowledge message payload is shown in table 37.

Table 37 – FCAP_Acknowledge Message Payload

Item	Size (Bytes)
Authentication Initiator Nonce	variable
Authentication Initiator Signature	variable
Authentication Initiator Certificate	variable
DH Value Length	4
DH Value	variable

Authentication Initiator Nonce: Shall contain the nonce R_b generated by the Authentication Initiator. The format of the nonce R_b is defined in 5.5.3.3.

Authentication Initiator Signature: Shall contain the signature S_b generated by the Authentication Initiator. The format of the signature S_b is defined in 5.5.4.2.

Authentication Initiator Certificate: This field shall contain the Certificate C_b of the Authentication Initiator. The format of the Certificate C_b is defined in 5.5.3.2.

DH Value Length: Diffie-Hellman parameter length. This length shall be a multiple of 4. If the DH group Identifier is set to zero (i.e., NULL DH algorithm), this field shall be set to zero. Otherwise, it shall be set to the length in bytes of the DH Value.

DH Value: Diffie-Hellman parameter. If the DH Value Length is set to zero, this field is not present. The DH Value shall be set to the value of g^y mod p, where y is a random number selected by the Authentication Initiator, and p and g shall have the values indicated in table 15, based on the selected DH group Identifier. A value of zero is illegal for the DH Value. If the DH Value is set to zero, the Authentication Responder shall reply with an AUTH_Reject having Reason Code 'Authentication Failure' and Reason Code Explanation 'Incorrect Payload'.

5.5.4.2 FCAP Signature Format

The FCAP signature format is shown in table 38.

Table 38 – FCAP Signature Format

Item	Size (bytes)
Signature Identifier	2
Signature Length	2
Signature Value	variable

Signature Identifier: Identifies the format of the signature. Signature identifiers are defined in table 39.

Signature Length: Indicates the total length in bytes of the Signature Value. Length values are shown in table 39.

Table 39 – Signature Formats

Signature Identifier	Signature Type	Signature Length (bytes)
0001h	RSA-SHA1	128
0002h	RSA2048-SHA256	256
0003h	RSA2048-SHA384	256
0004h	RSA2048-SHA512	256
all others	Reserved	

Signature Value: This field contains the signature value.

RSA-SHA1 Signature Value: The RSA-SHA1 signature is generated by computing the concatenation of the nonce with the Diffie-Hellman parameter g^y mod p, then applying the SHA-1 hash function to the concatenated quantity, then by encrypting the hash with the RSA private key of the sending entity (see RFC 3279).

RSA2048-SHA256 Signature Value: The RSA2048-SHA256 signature is generated by computing the concatenation of the nonce with the Diffie-Hellman parameter g^y mod p, then applying the SHA-256 hash function to the concatenated quantity, then by encrypting the hash with the RSA private key of the sending entity (see FIPS PUB 180-4).

RSA2048-SHA384 Signature Value: The RSA2048-SHA384 signature is generated by computing the concatenation of the nonce with the Diffie-Hellman parameter g^y mod p, then applying the SHA-384 hash

function to the concatenated quantity, then by encrypting the hash with the RSA private key of the sending entity (see FIPS PUB 180-4).

RSA2048-SHA512 Signature Value: The RSA2048-SHA512 signature is generated by computing the concatenation of the nonce with the Diffie-Hellman parameter g^y mod p, then applying the SHA-512 hash function to the concatenated quantity, then by encrypting the hash with the RSA private key of the sending entity (see FIPS PUB 180-4).

Support for the RSA-SHA1 signature format is mandatory for FCAP. Support for the RSA2048-SHA256, RSA2048-SHA384, and RSA2048-SHA512 signature formats is optional.

NOTE 16 – For compliance with NIST SP 800-131A, use of RSA-SHA1 is prohibited and use of RSA2048-SHA256, RSA2048-SHA384, and RSA2048-SHA512 is allowed.

5.5.5 FCAP_Confirm Message

The FCAP_Confirm message is sent from the Authentication Responder to the Authentication Initiator. The format of the FCAP_Confirm message payload is shown in table 40.

Table 40 – FCAP_Confirm Message Payload

Item	Size (Bytes)
Authentication Responder Signature	variable
DH Value Length	4
DH Value	variable

Authentication Responder Signature: Shall contain the signature S_a generated by the Authentication Responder. The format of the signature is defined in 5.5.4.2.

DH Value Length: Diffie-Hellman parameter length. This length shall be a multiple of 4. If the DH group Identifier is set to zero (i.e., NULL DH algorithm), this field shall be set to zero. Otherwise, it shall be set to the length in bytes of the DH Value.

DH Value: Diffie-Hellman parameter. If the DH Value Length is set to zero, this field is not present. The DH Value shall be set to the value of g^x mod p, where x is a random number selected by the Authentication Responder, and p and g shall have the values indicated in table 15, based on the selected DH group Identifier. A value of zero is illegal for the DH Value. If the DH Value is set to zero, the Authentication Initiator shall reply with an AUTH_Reject having Reason Code 'Authentication Failure' and Reason Code Explanation 'Incorrect Payload'.

5.5.6 Key Generation for the Security Association Management Protocol

The FCAP protocol enables the Security Association Management Protocol by generating a session key K_S that shall be used by the SA Management Transaction (see 6.7).

The session key K_S shall be computed as the complete hash, with no padding, of the shared key (i.e., g^{xy} mod p) generated during the FCAP transaction (i.e., $K_S = H(g^{xy} \text{ mod } p)$). The hash function H( ) is selected by the Authentication Responder in the HashID field of the FCAP_Request message. The size of the session key K_S is determined by the selected hash function, as shown in table 14.

5.5.7 Reuse of Diffie-Hellman Exponential

FCAP implementations may reuse a DH exponential (see 6.8.10) only when the generated session key K_S (see 5.5.6) is not used.

The basic risk in allowing reuse of a DH exponential (g^x or g^y) is replay of a prior authentication sequence based on the attacker reusing the other exponential.

For FCAP, replay of authentication is prevented by the nonces, but the nonces don't contribute to generation of the session key K_S, so DH exponential reuse by both parties results in the same session key. This needs to be avoided, therefore reuse of DH exponential is allowed only when the generated FCAP key is not used. If the generated FCAP session key is used for any purpose (e.g., as session key for the SA Management protocol, see 6.7.2), the DH exponentials shall not be reused.

5.6 FCPAP Protocol

5.6.1 Protocol Operations

FCPAP is a password based Authentication and key management protocol that uses the SRP algorithm (see RFC 2945 and SRP-6). FCPAP provides bidirectional Authentication between an Authentication Initiator and an Authentication Responder. When the FCPAP protocol successfully completes, Authentication Initiator and Responder are authenticated and share a session key.

The parameters for Authentication in the SRP algorithm are a password, a salt, and a verifier. In order to authenticate with the FCPAP protocol, each entity, identified by a unique Name, shall be provided with a password. To Authenticate another entity, an entity is required to be provided with a random salt, a verifier computed from the salt and the password, and the hash function used to perform this computation. Using the mathematical notation indicated in table 41, for each entity Z provided with a password P_z, the verifier v_z is generated by picking a random salt s_z, computing the intermediate value $x_z = H(s_z \,||\, Name_z \,||\, P_z)$, where H is a one-way hash function, and computing the verifier $v_z = g^{x_z} \bmod n$. The random salt s_z shall be a 16 bytes quantity, while the password P_z shall be at least 8 bytes long. The hash function and the DH group used during the Authentication Transaction shall be those chosen to compute the verifier.

NOTE 17 – In a sense, the used hash function, DH group and modulus are a Fabric property, because they are all administratively configured and then checked, rather than negotiated, during an Authentication Transaction. If two entities are configured with different parameters, an AUTH_Reject with reason code 'Logical Error' and Reason Code Explanation 'Hash Function Not Usable' or 'DH Group Not Usable' is sent in response to an AUTH_Negotiate message. If the AUTH_Negotiate message carries a list of hash functions or DH groups, this means that the sender has a verifier for each of the offered possibilities. This allows a graceful change of the hash function or the DH group across a Fabric by adding first the new verifiers, and by removing then the old ones.

Table 41 – Mathematical Notation for FCPAP

Symbols	Description
n, g	The modulus (n) and generator (g) of the chosen DH Group (see table 15). All computations are performed modulo n
P_z, P_y	Administratively configured passwords
s_z, s_y	Administratively configured random salts
v_z, v_y	Administratively configured pre-computed verifiers
x_z, x_y	Private keys computed from password and salt
a_z, b_z, a_y, b_y	Nonces, used as ephemeral private keys, generated randomly and not publicly revealed
A_z, B_z, A_y, B_y	Corresponding public keys
u	Scrambling parameter, publicly revealed
M_1, M_2	Hash values
S, S_z, S_y	Exponential values
H( )	One-way hash function
K_S	Computed session key

FCPAP operates with shared verifiers to verify a password assigned to entity Z (i.e., each entity has the same pair of salt and verifier to authenticate entity Z). A double SRP transaction to verify the passwords of both entities involved is required to provide a complete bidirectional Authentication.

An example of a FCPAP protocol transaction is shown in figure 9 with the notation shown in table 41. All computations are performed modulo n.

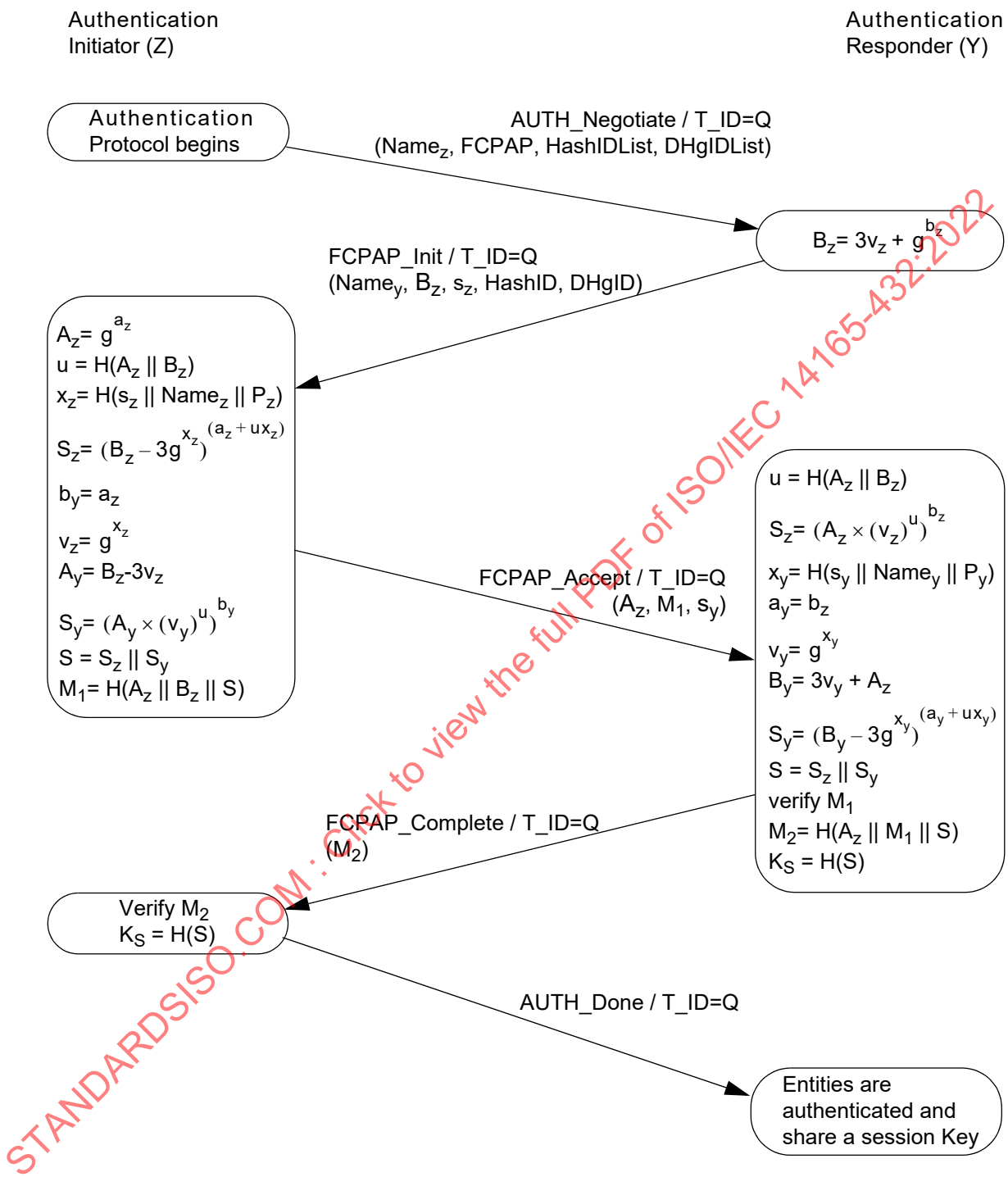

Figure 9 – A FCPAP Protocol Transaction Example

The FCPAP protocol proceeds as follows:

1) Operations shall start by negotiating the hash functions and the Diffie-Hellman group identifier to be used in the Authentication process with the AUTH_Negotiate message (see 5.3.2). In the AUTH_Negotiate message, the Authentication Initiator shall send its own name, and the list of the Authentication Protocols and the associated parameters to be used for the remainder of this Authentication Transaction. For the FCPAP protocol the parameters are the list of hash functions (e.g., SHA-1) and the list of Diffie-Hellman Group Identifiers that may be used (see 5.4.2);

2) On receiving the AUTH_Negotiate message, the Authentication Responder shall pick the salt s_z and the verifier v_z corresponding to the Authenticator Initiator, a random ephemeral private key b_z and compute the public key $B_z = 3v_z + g^{b_z}$. The Authentication Responder shall then reply with a FCPAP_Init message (see 5.6.3), carrying its name, the one-way hash function identifier HashID that has been used to generate the verifier, the DH group identifier selected among the ones proposed by the Authentication Initiator, the public key B_z, and the salt s_z;

3) On receiving the FCPAP_Init message, the Authentication Initiator shall pick a random ephemeral private key a_z, and compute the corresponding public key $A_z = g^{a_z}$. The generated public key A_z and the received public key B_z shall be combined to compute the scrambling parameter $u = H(A_z \| B_z)$. The Authentication Initiator shall then compute the private key $x_z = H(s_z \| Name_z \| P_z)$ from the received salt and its own password, and compute the exponential value $S_z = (B_z - 3g^{x_z})^{(a_z + ux_z)}$. The Authentication Initiator shall then use the Authentication Responder's salt s_y and verifier v_y. It shall pick the ephemeral private key $b_y = a_z$, compute the verifier $v_z = g^{x_z}$, and compute the ephemeral public key $A_y = B_z - 3v_z$. It shall then compute the exponential value $S_y = (A_y \times (v_y)^u)^{b_y}$ that is concatenated with the exponential value S_z to compute the common exponential value $S = S_z \| S_y$. The Authentication Initiator shall then compute the hash $M_1 = H(A_z \| B_z \| S)$ as evidence that it has the correct session key, and compute the session key $K_S = H(S)$. The Authentication Initiator shall then reply with a FCPAP_Accept message (see 5.6.4), carrying the public key A_z, the hash M_1, and the salt s_y corresponding to the Authentication Responder;

4) On receiving the FCPAP_Accept message, the Authentication Responder shall compute the scrambling parameter $u = H(A_z \| B_z)$, and the common exponential value $S_z = (A_z \times (v_z)^u)^{b_z}$. The Authentication Responder shall then compute the private key $x_y = H(s_y \| Name_y \| P_y)$ computed from the received salt, and its own password. It shall pick the ephemeral private key $a_y = b_z$, compute the verifier $v_y = g^{x_y}$, and compute the ephemeral public key $B_y = 3v_y + A_z$. Then it shall compute the exponential value $S_y = (B_y - 3g^{x_y})^{(a_y + ux_y)}$ that is concatenated to S_z to compute the common exponential value $S = S_z \| S_y$. The Authentication Responder shall then compute the hash $M_1' = H(A_z \| B_z \| S)$, and shall verify the hash matches the value of the received hash M_1. If the computed hash M_1' does not match the value of the received hash M_1, the Authentication fails and the Authentication Responder shall reply with an AUTH_Reject message with Reason Code 'Authentication Failure' and Reason Code Explanation 'Authentication Failed', and shall terminate the communication. If the authentication succeeds, the Authentication Responder shall compute the session key $K_S = H(S)$, and the hash $M_2 = H(A_z \| M_1 \| S)$ as evidence that it has the correct session key, and shall send a FCPAP_Complete message (see 5.6.5), carrying the hash M_2; and

5) On receiving the FCPAP_Complete message, the Authentication Initiator shall compute the hash M_2' = H(A_z || M_1 || S) and verify that it matches the value of the received hash M_2. If the computed hash M_2' does not match the value of the received hash M_2, the authentication fails then the Authentication Initiator shall reply with an AUTH_Reject message with Reason Code 'Authentication Failure' and Reason Code Explanation 'Authentication Failed', and shall terminate the communication. If the verification is successful, then the Authentication Initiator shall send an AUTH_Done message (see 5.3.8).

5.6.2 AUTH_Negotiate FCPAP Parameters

5.6.2.1 Overview

The Authentication Protocol Parameters in the AUTH_Negotiate message for FCPAP are formatted as shown in table 42.

Table 42 – AUTH_Negotiate FCPAP Protocol Parameters

Item	Size (Bytes)
Parameter #1 = HashList	variable
Parameter #2 = DHgIDList	variable
...	...
Parameter #k	variable

Parameter: Each parameter shall be formatted as shown in table 43.

Table 43 – AUTH_Negotiate FCPAP Parameter Format

Item	Size (Bytes)
Parameter Tag	2
Parameter Word Count	2
Parameter Value	variable

Parameter Tag: Identifies the format of the Parameter Value. Parameter Tags are shown in table 44.

Table 44 – AUTH_Negotiate FCPAP Parameter Tags

Parameter Tag	Parameter Value Format
0001h	HashList (see 5.6.2.2)
0002h	DHgIDList (see 5.6.2.3)
all others	Reserved

Parameter Word Count: Indicates the number of words composing the Parameter Value.

Parameter Value: Contains the parameter value.

5.6.2.2 HashList Parameter

The HashList parameter shall be included as the first parameter in the AUTH_Negotiate FCPAP Protocol Parameters (see table 42).

Hashlist Parameter Word Count: Shall be set to the number of hash functions proposed by the Authentication Initiator. Each hash function identifier is encoded into one word.

Hashlist Parameter Value: Each word of this field contains an identifier of a proposed hash function, in order of preference. The first word contains the most preferred, the last word contains the least preferred hash function. The list of defined hash function identifiers is shown in table 14.

Support for the SHA-1 hash function is mandatory for FCPAP. The MD5 hash function shall not be used with FCPAP.

5.6.2.3 DHgIDList Parameter

The DHgIDList parameter shall be included as the second parameter in the AUTH_Negotiate FCPAP Protocol Parameters (see table 42).

DHgIDList Parameter Word Count: Shall be set to the number of proposed DH groups. Each DH group identifier is encoded into one word.

DHgIDList Parameter Value: Each word of this field contains a DH group identifier (see table 15) proposed by the Authentication Initiator, in order of preference.

Support for the DH group 1 536 (i.e., DH group identifier = 0000 0003h) is mandatory.

The NULL DH group (i.e., DH group identifier = 0000 0000h) shall not be used.

5.6.3 FCPAP_Init Message

The FCPAP_Init message is sent from the Authentication Responder to the Authentication Initiator. The format of the FCPAP_Init message payload is shown in table 45.

Table 45 – FCPAP_Init Message Payload

Item	Size (Bytes)
Name	variable
Authentication Data Length	4
Authentication Data Value	variable
SRP Salt Length	4
SRP Salt Value	variable
Hash Identifier	4
DH Group Identifier	4

Name: Shall be set to the Authentication Responder Name. See 5.3.3 for name formats.

Authentication Data Length: Shall be set to the length in bytes of the Authentication Data Value.

Authentication Data Value: Shall be set to the value of the ephemeral public key B_x computed by the Authentication Responder.

SRP Salt Length: Shall be set to the length in bytes of the SRP Salt Value.

SRP Salt Value: Shall be set to the value of the salt s_z selected by the Authentication Responder.

Hash Identifier: Shall be set to the selected hash function among those proposed in the corresponding AUTH_Negotiate message. The Authentication Responder shall select a hash function in accord with applicable policy of the Authentication Responder. This policy may require that the preference expressed by the Authentication Initiator in the AUTH_Negotiate message be honored.

DH Group Identifier: Shall be set to the identifier of the DH group selected for this Authentication Transaction. The list of defined DH group identifiers is specified in table 15. The Authentication Responder shall select a DH Group Identifier in accord with applicable policy of the Authentication Responder. This policy may require that the preference expressed by the Authentication Initiator in the AUTH_Negotiate message be honored.

5.6.4 FCPAP_Accept Message

The FCPAP_Accept message is sent from the Authentication Initiator to the Authentication Responder. The format of the FCPAP_Accept message payload is shown in table 46.

Table 46 – FCPAP_Accept Message Payload

Item	Size (Bytes)
Authentication Data Length	4
Authentication Data Value	variable
Hash Length	4
Hash Value	variable
SRP Salt Length	4
SRP Salt Value	variable

Authentication Data Length: Shall be set to the length in bytes of the Authentication Data Value.

Authentication Data Value: Shall be set to the value of the ephemeral public key A_z computed by the Authentication Initiator.

Hash Length: Shall be set to the length in bytes of the Hash value.

Hash Value: Shall be set to the value of the hash M_1 computed by the Authentication Initiator.

SRP Salt Length: Shall be set to the length in bytes of the SRP Salt Value.

SRP Salt Value: Shall be set to the value of the salt s_y selected by the Authentication Initiator.

5.6.5 FCPAP_Complete Message

The FCPAP_Complete message is sent from the Authentication Responder to the Authentication Initiator. The format of the FCPAP_Complete message payload is shown in table 47.

Table 47 – FCPAP_Complete Message Payload

Item	Size (Bytes)
Hash Length	4
Hash Value	variable

Hash Length: Shall be set to the length in bytes of the Hash Value.

Hash Value: Shall be set to the value of the hash M_2 computed by the Authentication Responder.

5.6.6 Key Generation for the Security Association Management Protocol

The FCPAP protocol enables the Security Association Management Protocol by generating a session key K_S that shall be used by the SA Management Transaction (see 6.7).

The session key K_S shall be computed as the complete hash, with no padding, of the common exponential value S computed during the FCPAP transaction (i.e., $K_S = H(S)$). The hash function H( ) is selected by the Authentication Responder in the HashID field of the FCPAP_Init message. The size of the session key K_S is determined by the selected hash function, as shown in table 14.

5.6.7 Reuse of Diffie-Hellman Exponential

FCPAP implementations shall not reuse a DH exponential (see 6.8.10) because there is no other source of randomness in the protocol (e.g., no additional nonces). Any exponential reuse risks replay of an old authentication.

5.7 FCEAP Protocol

5.7.1 Protocol Operations

The Extensible Authentication Protocol (EAP) defined in RFC 3748 is an authentication protocol that supports multiple authentication methods. FCEAP defines how to encapsulate EAP messages over the AUTH messages defined in 5.2. This is done by mapping the four EAP packet formats defined by RFC 3748 (i.e., Request, Response, Success, Failure) into the AUTH Messages (see table 8).

An example of a FCEAP protocol transaction is shown in figure 10.

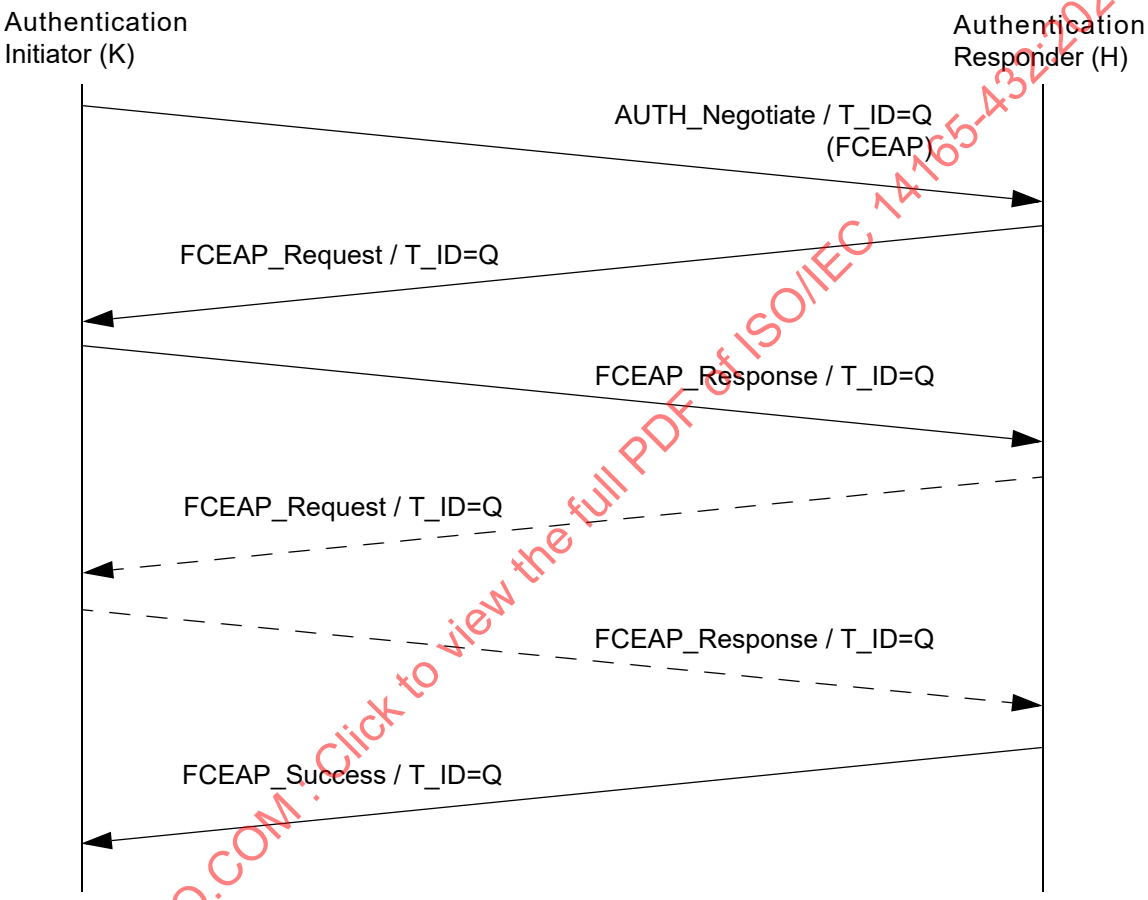

Figure 10 – A FCEAP Protocol Transaction Example

5.7.2 AUTH_Negotiate FCEAP Parameters

The FCEAP protocol does not use Authentication Protocol Parameters as part of the AUTH_Negotiate message payload. See table 11 for the FCEAP protocol identifier.

5.7.3 FCEAP_Request Message

The FCEAP_Request message is sent from the Authentication Responder to the Authentication Initiator. The payload of the FCEAP_Request message is shown in table 48.

Table 48 – FCEAP_Request Message Payload

Item	Size (Bytes)
Authentication Responder Name	variable
EAP_Code	1
EAP_Identifier	1
EAP_Length	2
EAP_Data	variable

Authentication Responder Name: Shall be set to the Authentication Responder Name (see 5.3.3).

EAP_Code: The EAP_Code Field shall be set to one to identify the EAP Request packet at the EAP Protocol level.

EAP_Identifier: The EAP_Identifier field is one byte long and aids in matching FCEAP_Responses with FCEAP_Requests.

EAP_Length: The EAP_Length field indicates the length, in bytes, of the EAP packet including the EAP_Code, EAP_Identifier, EAP_Length, and EAP_Data fields. A message with the EAP_Length field set to a value larger than the number of received bytes shall be discarded.

EAP_Data: The EAP_Data field is zero or more bytes. The format of the EAP_Data field is determined by the EAP_Code field as specified in RFC 3748.

5.7.4 FCEAP_Response Message

The FCEAP_Response message is sent from the Authentication Initiator to the Authentication Responder. The payload of the FCEAP_Response message is shown in table 49.

Table 49 – FCEAP_Response Message Payload

Item	Size (Bytes)
Authentication Initiator Name	variable
EAP_Code	1
EAP_Identifier	1
EAP_Length	2
EAP_Data	variable

Authentication Initiator Name: Shall be set to the Authentication Initiator Name (see 5.3.3).

EAP_Code: The EAP_Code Field shall be set to two to identify the EAP Response packet at the EAP Protocol level.

EAP_Identifier: The EAP_Identifier field is one byte long and aids in matching FCEAP_Responses with FCEAP_Requests.

EAP_Length: The EAP_Length field indicates the length, in bytes, of the EAP packet including the EAP_Code, EAP_Identifier, EAP_Length, and EAP_Data fields. A message with the EAP_Length field set to a value larger than the number of received bytes shall be discarded.

EAP_Data: The EAP_Data field is zero or more bytes. The format of the EAP_Data field is determined by the EAP_Code field as specified in RFC 3748.

5.7.5 FCEAP_Success Message

The FCEAP_Success message is sent from the Authentication Responder to the Authentication Initiator. The payload of the FCEAP_Success message is shown in table 50.

Table 50 – FCEAP_Success Message Payload

Item	Size (Bytes)
Authentication Responder Name	variable
EAP_Code	1
EAP_Identifier	1
EAP_Length	2

Authentication Responder Name: Shall be set to the Authentication Responder Name (see 5.3.3).

EAP_Code: The EAP_Code Field shall be set to three to identify the EAP Success packet at the EAP Protocol level.

EAP_Identifier: The EAP_Identifier field is one byte long and aids in matching FCEAP_Responses with FCEAP_Requests. The EAP_Identifier field shall match the EAP_Identifier field of the FCEAP_Response message to which the EAP_Success message is responding.

EAP_Length: Shall be set to four.

5.7.6 FCEAP_Failure Message

The FCEAP_Failure message is sent from the Authentication Responder to the Authentication Initiator. The payload of the FCEAP_Failure message is shown in table 51.

Table 51 – FCEAP_Failure Message Payload

Item	Size (Bytes)
Authentication Responder Name	variable
EAP_Code	1
EAP_Identifier	1
EAP_Length	2

Authentication Responder Name: Shall be set to the Authentication Responder Name (see 5.3.3).

EAP_Code: The EAP_Code Field shall be set to four to identify the EAP Failure packet at the EAP Protocol level.

EAP_Identifier: The EAP_Identifier field is one byte long and aids in matching FCEAP_Responses with FCEAP_Requests. The EAP_Identifier field shall match the EAP_Identifier field of the FCEAP_Response message to which the EAP_Success message is responding.

EAP_Length: Shall be set to four.

5.7.7 AUTH_Reject Use

A FCEAP Protocol transaction shall be terminated as described in this subclause.

If an error is detected at the AUTH Protocol level (e.g., if an Authentication Initiator receives a FCEAP_Request with an AUTH payload that is not properly formatted), then the FCEAP Protocol shall be terminated with an AUTH_Reject message (see 5.3.7) with Reason Code set to 01h (Authentication Failure) and Reason Code Explanation set to 06h (Incorrect Payload).

If an error is detected at the EAP Protocol level, then the FCEAP Protocol transaction shall proceed to conclusion as described in RFC 3748.

An example of an FCEAP Protocol transaction that ends with a failure at the EAP Protocol level is shown in figure 11.

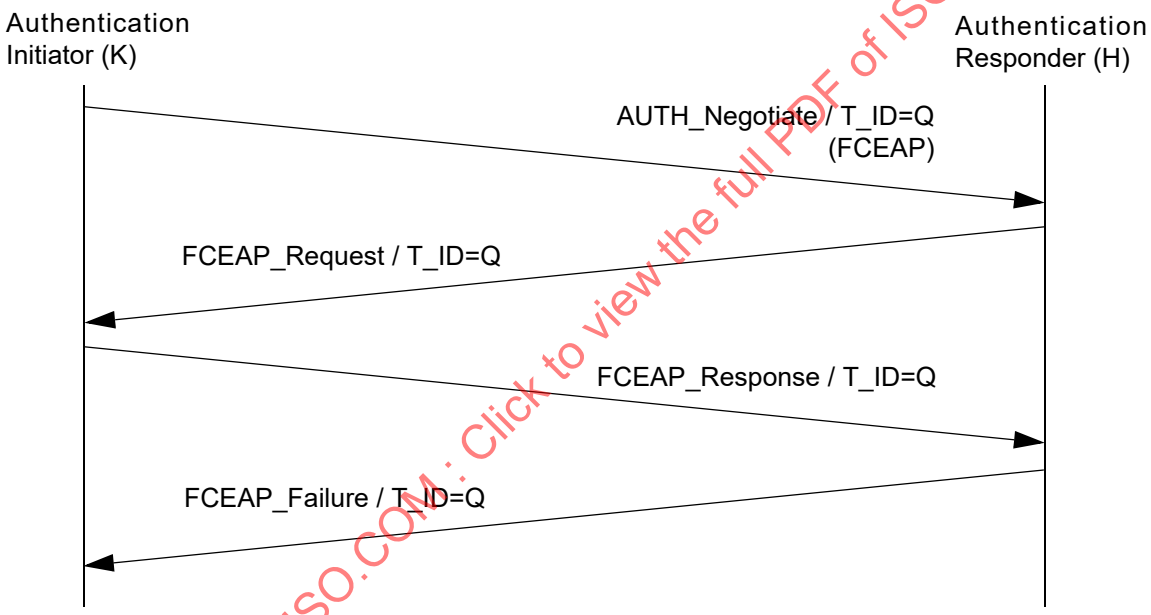

Figure 11 – A Failing FCEAP Protocol Transaction Example

5.7.8 AUTH_ELS and AUTH_ILS Size Requirements

FCEAP requires an AUTH_ELS and AUTH_ILS to be able to transport a payload of 1 040 bytes or greater. The reason is that EAP, as specified in RFC 3748, requires such a transport.

5.7.9 Supported EAP Methods

FCEAP supports the EAP methods shown in table 52.

Table 52 – Supported EAP Methods

EAP Type[a]	EAP Name	Description
51	EAP-GPSK	EAP Generalized Pre-Shared Key (EAP-GPSK) Method (see RFC 5433)
all others		Reserved to IANA
[a] These values are a subset of those specified in http://www.iana.org/assignments/eap-numbers		

5.7.10 Key Generation for the Security Association Management Protocol

With use of some EAP methods (e.g., EAP-GPSK), the FCEAP protocol enables the Security Association Management Protocol by generating a session key K_S that shall be used by the SA Management Transaction (see 6.7).

If

a) FCEAP negotiates to use an EAP method that exports a Master Session Key (see RFC 3748);

b) the Authentication Initiator and the Authentication Responder both have access to the EAP Master Session Key; and

c) FCEAP negotiates to concatenate an SA Management Transaction to the Authentication Protocol,

then the session key K_S shall be bytes 32 .. 63 of the Master Session Key exported by the EAP method. When exported by an EAP method, the Master Session Key has a length greater than or equal to 64 bytes.

Where K_S is treated as an integer, the most significant byte shall be byte 0 of K_S and the least significant byte shall be byte 32 of K_S.

If

a) FCEAP negotiates to use an EAP method that does not export a Master Session Key (see RFC 3748); or

b) the Authentication Initiator and the Authentication Responder do not both have access to the EAP Master Session Key,

then FCEAP shall not negotiate to concatenate an SA Management Transaction to the Authentication Protocol.

NOTE 18 – Access to the EAP Master Session Key may be an issue for FCEAP peers that act as clients to an authentication server (e.g., RADIUS), because in these cases, EAP exports its Master Session Key to the authentication server rather than the FCEAP peer. There is no current standard for a RADIUS authentication server

to securely pass an EAP Master Session Key to its client. RFC 4072 specifies such a protocol for DIAMETER authentication servers.

NOTE 19 – Access to the EAP Master Session Key is less likely to be an issue for the Authentication Initiator because it is an EAP client, and there is no current standard by which an EAP client can act as a client to an authentication server.

5.8 AUTH_ILS Specification

5.8.1 Overview

The AUTH_ILS SW_ILS shall be used to convey Authentication messages between Switches, via either the Fabric Controller Address Identifier (i.e., FFFFFDh to FFFFFDh) or the Domain Controller Address Identifier (i.e., FFFCxxh to FFFCxxh).

Any Switch may act as Authentication Initiator or as Authentication Responder. A Switch may initiate an Authentication Transaction whenever needed. No more than one Authentication Transaction shall be in progress between a pair of E_Ports, using the Fabric Controller Address Identifier, or a pair of Domain_Controller Address Identifiers, at any time.

NOTE 20 – The usage of the AUTH_ILS SW_ILS between Domain Controller Address Identifiers is not specified by this standard.

If two Switches start an Authentication Transaction at the same time, one of the two Authentication Transactions shall be aborted as described in this subclause.

If a Switch is acting as an Authentication Initiator and receives an AUTH_Negotiate message from the designated Authentication Responder, one of the two Authentication Transactions shall be aborted. The Switch that sent the AUTH_Negotiate message with the numerically higher Name shall remain the Authentication Initiator, while the Switch that sent the AUTH_Negotiate message with the numerically lower Name shall become the Authentication Responder. The Switch that remains the Authentication Initiator shall reply to the received AUTH_Negotiate message with an AUTH_Reject message with Reason Code ‘Logical Error’ and Reason Code Explanation ‘Authentication Transaction Already Started’. The Switch that becomes the Authentication Responder shall reply to the received AUTH_Negotiate message and abort its own transaction upon receipt of the AUTH_Reject message.

If a Switch is not acting as an Authentication Initiator or Authentication Responder and it receives an AUTH_Negotiate message, then it shall reply to that message as specified by the Authentication Protocol of its choosing, becoming the Authentication Responder.

In order to not tie the timeouts of Authentication Protocols with the timeouts already defined for SW_ILSs, each AUTH message is carried in a separate bidirectional Exchange (see FC-FS-3). Each AUTH_ILS shall be replied to with a null SW_ACC or with a SW_RJT. Failures at the Authentication level shall be indicated by the AUTH_Reject message, not by the SW_RJT SW_ILS.

As an example, figure 12 shows the flow of AUTH_ILSs for the E_Port to E_Port Authentication case.

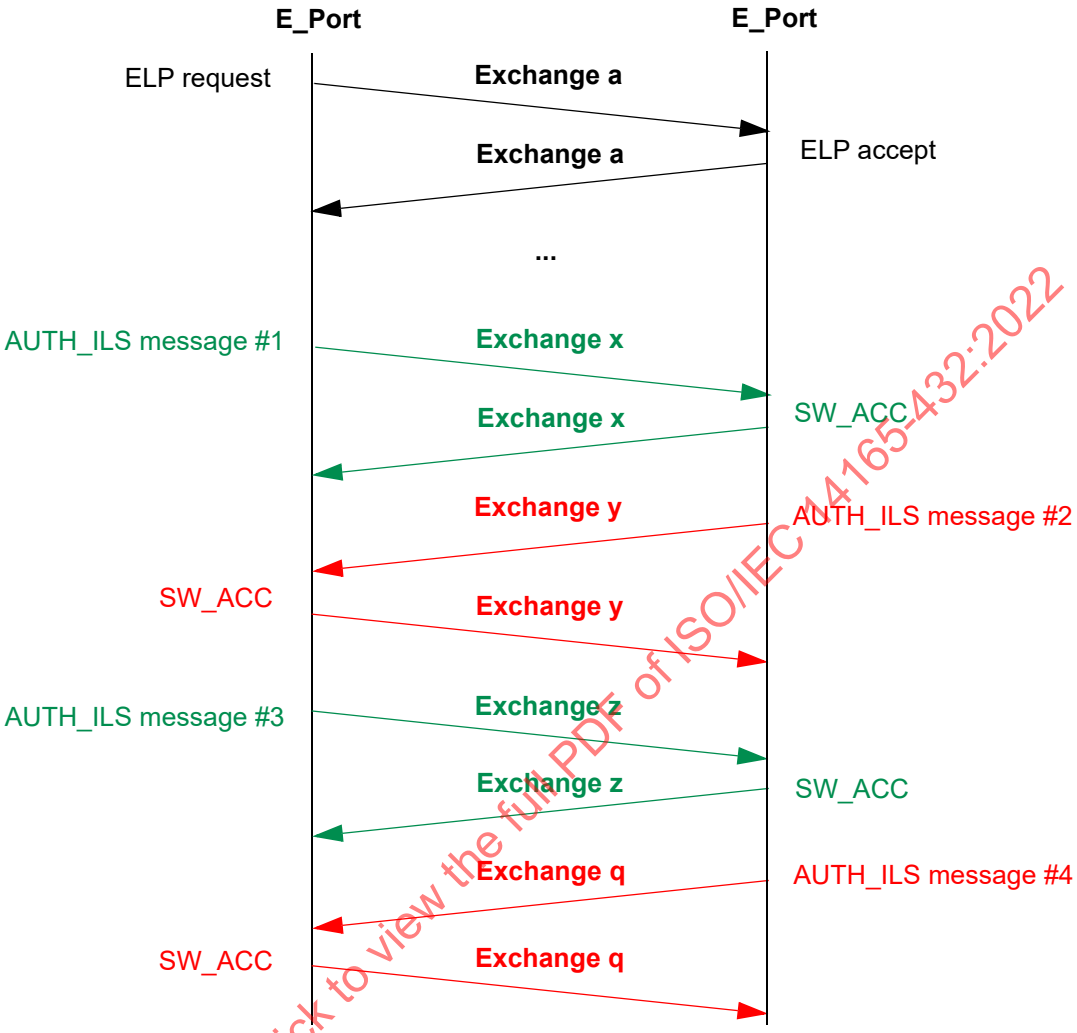

Figure 12 – FC-2 AUTH_ILS Mapping Example for the E_Port to E_Port Case

5.8.2 AUTH_ILS Request Sequence

Protocol: AUTH_ILS SW_ILS Request Sequence

Addressing: For use between two adjacent E_Ports, the S_ID field shall be set to FFFFFDh, indicating the Fabric Controller of the originating Switch, and the D_ID field shall be set to FFFFFDh, indicating the Fabric Controller of the destination Switch. For use between two Domain_Controllers, the S_ID field shall be set to FFFCxxh, indicating the Domain_Controller of the originating Switch, and the D_ID field shall be set to FFFCyyh, indicating the Domain_Controller of the destination Switch.

Payload: The format of the AUTH_ILS Request Sequence Payload is shown in table 3.

5.8.3 AUTH_ILS Reply Sequence

SW_RJT: SW_RJT shall be sent as a reply to signify the rejection of the AUTH_ILS Request Sequence for reasons shown in table 53. SW_RJT shall not be used to indicate a failure of Authentication detected by the Authentication Protocol during processing. Such failure shall be indicated by SW_ACC followed by an AUTH_Reject message.

Table 53 – AUTH_ILS SW_RJT Reasons

Reason	Reason Code	Reason Code Explanation
AUTH_ILS not supported	0Bh	2Ch
Logical Busy	05h	00h

The receiver of an SW_RJT signaling a Logical Busy error condition should restart the Authentication Protocol after a random delay. The receiver should compute the random delay by using the following binary exponential backoff algorithm. Before restarting the Authentication Protocol the receiver should delay a random amount of time between E_D_TOV and R_A_TOV. For each additional SW_RJT signaling a Logical Busy error condition that occurs after an attempt at restarting, the receiver should double the upper limit of the range from which the delay is chosen, until the upper limit reaches AUTH_TOV (i.e., for the second restart the random delay is between E_D_TOV and 2 times R_A_TOV; for the third restart the random delay is between E_D_TOV and 4 times R_A_TOV). If the Logical Busy error condition is ignored, an AUTH_TOV timeout shall occur (see 5.12).

SW_ACC: SW_ACC shall be sent as a reply to signify the acceptance of the AUTH_ILS Request Sequence for processing. The format of the AUTH_ILS SW_ACC Payload is shown in table 54.

Table 54 – AUTH_ILS SW_ACC Payload

Item	Size (Bytes)
0200 0000h	4

5.9 B_AUTH_ILS Specification

5.9.1 Overview

The B_AUTH_ILS SW_ILS shall be used to convey Authentication messages between an E_Port and a B_Port. The E_Port shall always act as the Authentication Initiator and the B_Port shall always act as the Authentication Responder. The B_Port shall never initiate an Authentication Transaction.

The B_AUTH_ILS allows B_Ports to be authenticated when B_Ports are used to interconnect remote E_Ports, as shown in figure 13.

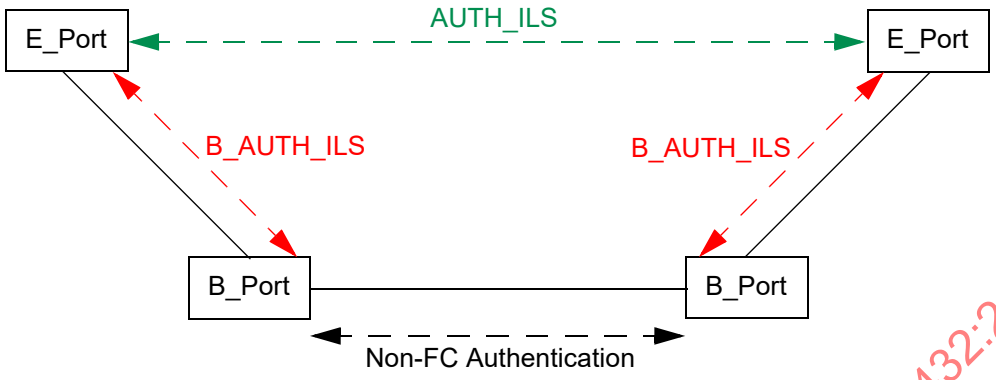

Figure 13 – Usage of B_AUTH_ILS

The two remote E_Ports may authenticate each other by using the AUTH_ILS, propagated by B_Ports. The two B_Ports may authenticate each other using methods outside the scope of this standard. The B_AUTH_ILS, terminated by B_Ports, shall be used to authenticate the link between E_Port and B_Port. If performed, a B_AUTH_ILS transaction shall precede an AUTH_ILS transaction over the same link (see FC-SW-5).

Other than the different SW_ILS code, the only difference between B_AUTH_ILS and AUTH_ILS is that one is terminated by B_Ports and the other is propagated by B_Ports.

Each B_AUTH_ILS Authentication Protocol message is carried in a separate bidirectional Exchange (see FC-FS-3). Each B_AUTH_ILS shall be replied to with a null SW_ACC or with a SW_RJT. Failures at the Authentication level shall be indicated by the AUTH_Reject message, not by the SW_RJT SW_ILS.

As an example, figure 14 shows the flow of B_AUTH_ILSs for E_Port to B_Port Authentication.

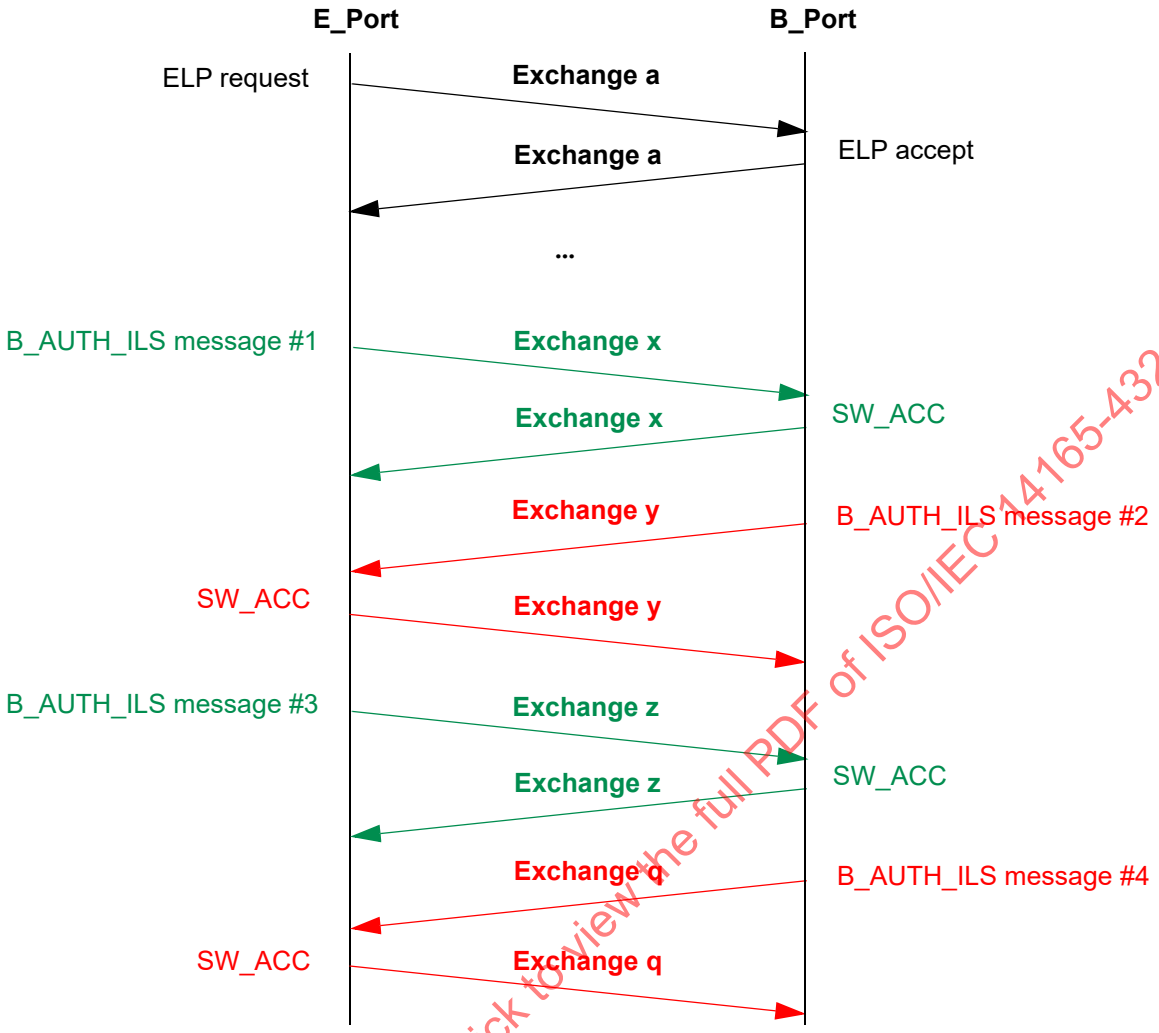

Figure 14 – FC-2 B_AUTH_ILS Mapping Example

An E_Port that detects it is connected to a B_Port may use B_AUTH_ILSs to authenticate the B_Port, and then AUTH_ILSs to authenticate with the remote E_Port.

5.9.2 B_AUTH_ILS Request Sequence

Protocol: B_AUTH_ILS SW_ILS Request Sequence

Addressing: For use between an E_Port and a B_Port, the S_ID field shall be set to FFFFFDh, and the D_ID field shall be set to FFFFFDh.

Payload: The format of the B_AUTH_ILS Request Sequence Payload is shown in table 5.

5.9.3 B_AUTH_ILS Reply Sequence

SW_RJT: SW_RJT shall be sent as a reply to signify the rejection of the AUTH_ILS Request Sequence for reasons shown in table 55. SW_RJT shall not be used to indicate a failure of Authentication detected by the Authentication Protocol during processing. Such failure shall be indicated by SW_ACC followed by an AUTH_Reject message.

Table 55 – B_AUTH_ILS SW_RJT Reasons

Reason	Reason Code	Reason Code Explanation
B_AUTH_ILS not supported	0Bh	2Ch
Logical Busy	05h	00h

The receiver of an SW_RJT signaling a Logical Busy error condition should restart the Authentication Protocol after a random delay. The receiver should compute the random delay by using the following binary exponential backoff algorithm. Before restarting the Authentication Protocol the receiver should delay a random amount of time between E_D_TOV and R_A_TOV. For each additional SW_RJT signaling a Logical Busy error condition that occurs after an attempt at restarting, the receiver should double the upper limit of the range from which the delay is chosen, until the upper limit reaches AUTH_TOV (i.e., for the second restart the random delay is between E_D_TOV and 2 times R_A_TOV; for the third restart the random delay is between E_D_TOV and 4 times R_A_TOV). If the Logical Busy error condition is ignored, an AUTH_TOV timeout shall occur (see 5.12).

SW_ACC: SW_ACC shall be sent as a reply to signify the acceptance of the B_AUTH_ILS Request Sequence for processing. The format of the B_AUTH_ILS SW_ACC Payload is shown in table 56.

Table 56 – B_AUTH_ILS SW_ACC Payload

Item	Size (Bytes)
0200 0000h	4

5.10 AUTH_ELS Specification

5.10.1 Overview

The AUTH_ELS ELS is sent by:

a) An Nx_Port to another Nx_Port;

b) An Nx_Port to an Fx_Port; or

c) An Fx_Port to an Nx_Port.

An AUTH_ELS transfers an Authentication message or a fragment of an Authentication message. AUTH_ELS requires Login between the two associated Nx_Ports (e.g., for authentication with a Fabric, one Nx_Port is an F_Port Controller) prior to its use.

Any Nx_Port or F_Port Controller may act as Authentication Initiator or as Authentication Responder. An Nx_Port or a F_Port Controller may initiate an Authentication Transaction whenever appropriate (see clause 8). No more than one Authentication Transaction shall be in progress between a pair of Nx_Ports, or an Nx_Port and a F_Port Controller, at any time.

If two Nx_Ports start an Authentication Transaction at the same time, one of the two Authentication Transactions shall be aborted, as described in this subclause.

If an Nx_Port or F_Port Controller is acting as an Authentication Initiator and receives an AUTH_Negotiate message from the designated Authentication Responder, one of the two Authentication Transactions shall be aborted. In the case of Nx_Port to Nx_Port Authentication, the Nx_Port that sent the AUTH_Negotiate message with the numerically higher Name shall remain the Authentication Initiator, while the Nx_Port that sent the AUTH_Negotiate message with the numerically lower Name shall become the Authentication Responder. In the case of Nx_Port to F_Port Controller Authentication, the Nx_Port shall remain the Authentication Initiator, while the F_Port Controller shall become the Authentication Responder. The Nx_Port that remains the Authentication Initiator shall reply to the received AUTH_Negotiate message with an AUTH_Reject message with Reason Code 'Logical Error' and Reason Code Explanation 'Authentication Transaction Already Started'. The Nx_Port that becomes the Authentication Responder shall reply to the received AUTH_Negotiate message and abort its own transaction upon receipt of the AUTH_Reject message.

If an Nx_Port or F_Port Controller is not acting as an Authentication Initiator or Authentication Responder and it receives an AUTH_Negotiate message, then it shall reply to that message as specified by the Authentication Protocol of its choosing, becoming the Authentication Responder.

If performed, an Authentication Transaction between an Nx_Port and an F_Port Controller should be completed before any Nx_Port to Nx_Port Authentication Transaction involving the same Nx_Port.

In order to not tie the timeouts of Authentication Protocols with the timeouts already defined for ELSs, each Authentication Protocol message or fragment (see 5.10.4) is carried in a separate bidirectional Exchange (see FC-FS-3). Each AUTH_ELS shall be replied to with a null LS_ACC or with a LS_RJT. Failures at the Authentication level shall be indicated by the AUTH_Reject message, not by the LS_RJT ELS.

As an example, figure 15 shows the flow of ELSs for the Nx_Port to Nx_Port Authentication case.

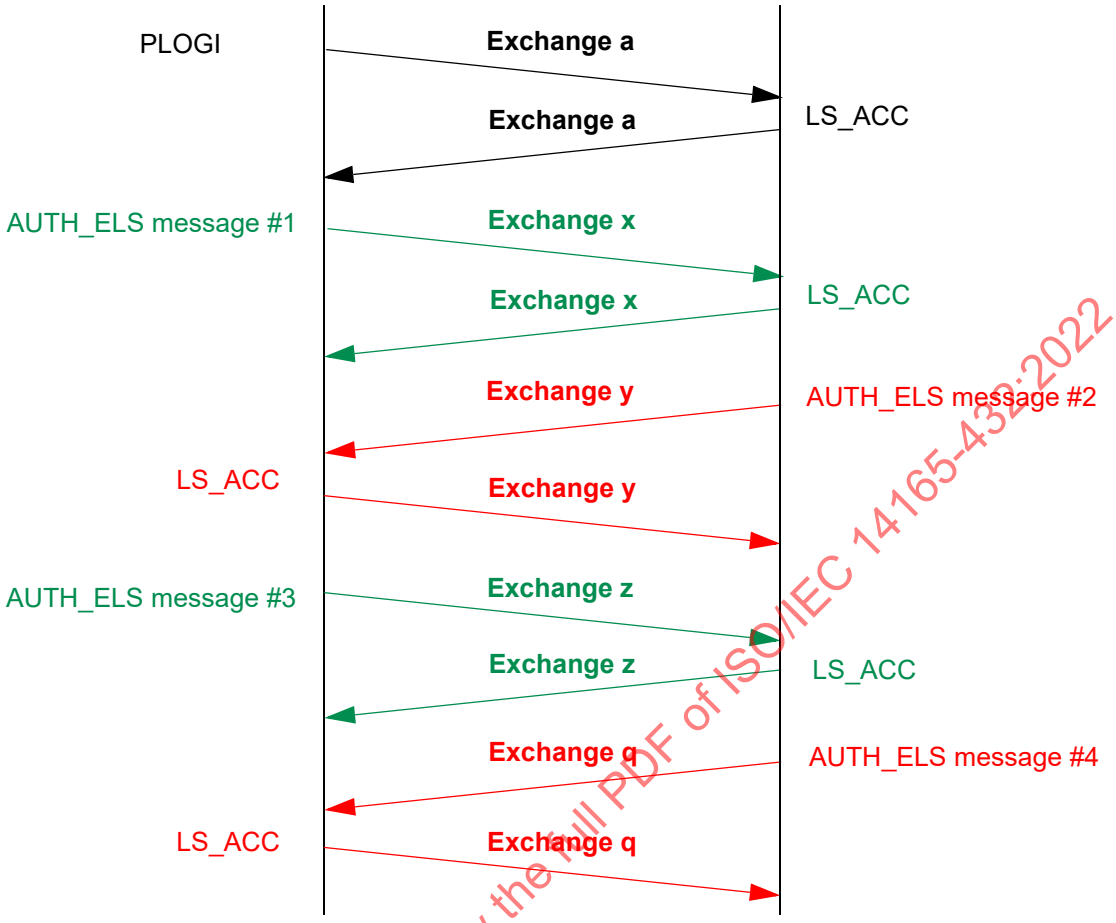

Figure 15 – FC-2 AUTH_ELS Mapping Example for the Nx_Port to Nx_Port Case

5.10.2 AUTH_ELS Request Sequence

Protocol: AUTH_ELS ELS Request Sequence

Addressing: The S_ID field shall be the address identifier of the Nx_Port sending the AUTH_ELS message, or FFFFF0h (i.e. the N_Port Controller address), or FFFFFEh (i.e., the F_Port Controller address) if the AUTH_ELS message is being sent from an Fx_Port to an Nx_Port, or the Well Known Address of a Generic Service. The D_ID field shall designate the Nx_Port to which Authentication is being performed, and shall be one of:

a) The address identifier of another Nx_Port to designate that Nx_Port as the Nx_Port to which Authentication is being performed;

b) The N_Port Controller address (i.e., FFFFF0h) to designate a VFT capable PN_Port (see FC-FS-3 and FC-LS-2);

c) The Well Known Address of a Generic Service to designate that Generic Service as the Nx_Port to which Authentication is being performed; or

d) FFFFFEh to designate the local F_Port Controller as the Nx_Port to which Authentication is being performed. Authentication of an Nx_Port with its local F_Port Controller shall be equivalent to Authentication with the Fabric.

Payload: The format of the AUTH_ELS Request Sequence Payload is shown in table 6.

5.10.3 AUTH_ELS Reply Sequence

LS_RJT: LS_RJT shall be sent as a reply to signify the rejection of the AUTH_ELS Request Sequence for reasons shown in table 57. LS_RJT shall not be used to indicate a failure of Authentication detected by the Authentication Protocol during processing. Such failure shall be indicated by LS_ACC followed by an AUTH_Reject message.

Table 57 – AUTH_ELS LS_RJT Reasons

Reason	Reason Code	Reason Code Explanation
AUTH_ELS not supported	0Bh	2Ch
AUTH_ELS received by an Nx_Port from an Nx_Port with which it is not logged in	09h	1Eh
Logical Busy	05h	00h

The receiver of an LS_RJT signaling a Logical Busy error condition should restart the Authentication Protocol after a random delay. The receiver should compute the random delay by using the following binary exponential backoff algorithm. Before restarting the Authentication Protocol the receiver should delay a random amount of time between E_D_TOV and R_A_TOV. For each additional LS_RJT signaling a Logical Busy error condition that occurs after an attempt at restarting, the receiver should double the upper limit of the range from which the delay is chosen, until the upper limit reaches AUTH_TOV (i.e., for the second restart the random delay is between E_D_TOV and 2 times R_A_TOV; for the third restart the random delay is between E_D_TOV and 4 times R_A_TOV). If the Logical Busy error condition is ignored, an AUTH_TOV timeout shall occur (see 5.12).

LS_ACC: LS_ACC shall be sent as a reply to signify the acceptance of the AUTH_ELS Request Sequence for processing. The format of the AUTH_ELS LS_ACC Payload is shown in table 58.

Table 58 – AUTH_ELS LS_ACC Payload

Item	Size (Bytes)
0200 0000h	4

5.10.4 AUTH_ELS Fragmentation

The size of an Authentication message may exceed the size of an ELS payload that a specific Nx_Port is able to handle. An Nx_Port may report that is has some ELS size limitations by setting to one the Query Buffer Condition (QBC) bit in the Common Service Parameters field of the FLOGI or PLOGI message. The Report Port Buffer Conditions (RPBC) ELS (see FC-LS-2) allows to quantify the ELS size limitations. All fragments of an AUTH_ELS message, except the last one, shall be of the maximum supported size reported by the RPBC ELS.

In order to support AUTH_ELS, an Nx_Port with buffer limitations is required to support the RPBC ELS and the QBC bit in the FLOGI and PLOGI ELSs (see FC-LS-2). If an Nx_Port with buffer limitations does not

support the RPBC ELS, or does not support the QBC bit in the FLOGI and PLOGI ELSs, that Nx_Port shall reject any AUTH_ELS message with a Reason Code 'ELS not supported'.

The AUTH_ELS provides a means to indicate either that:

a) The AUTH_ELS contains the final or the only fragment of an Authentication message; or

b) A sequentially subsequent fragment of the same Authentication message shall be sent in the next AUTH_ELS from the same S_ID to the same D_ID.

NOTE 21 – Fragmentation as defined in this subclause does not apply to AUTH_ILS and B_AUTH_ILS.

The common part of the AUTH_ELS message (i.e., the first 12 bytes of the message shown in table 6) shall be included in each fragment. A sender with ELS size limitations, or that has to send a message to a receiver with ELS size limitations, shall fragment each Authentication message as shown in figure 16.

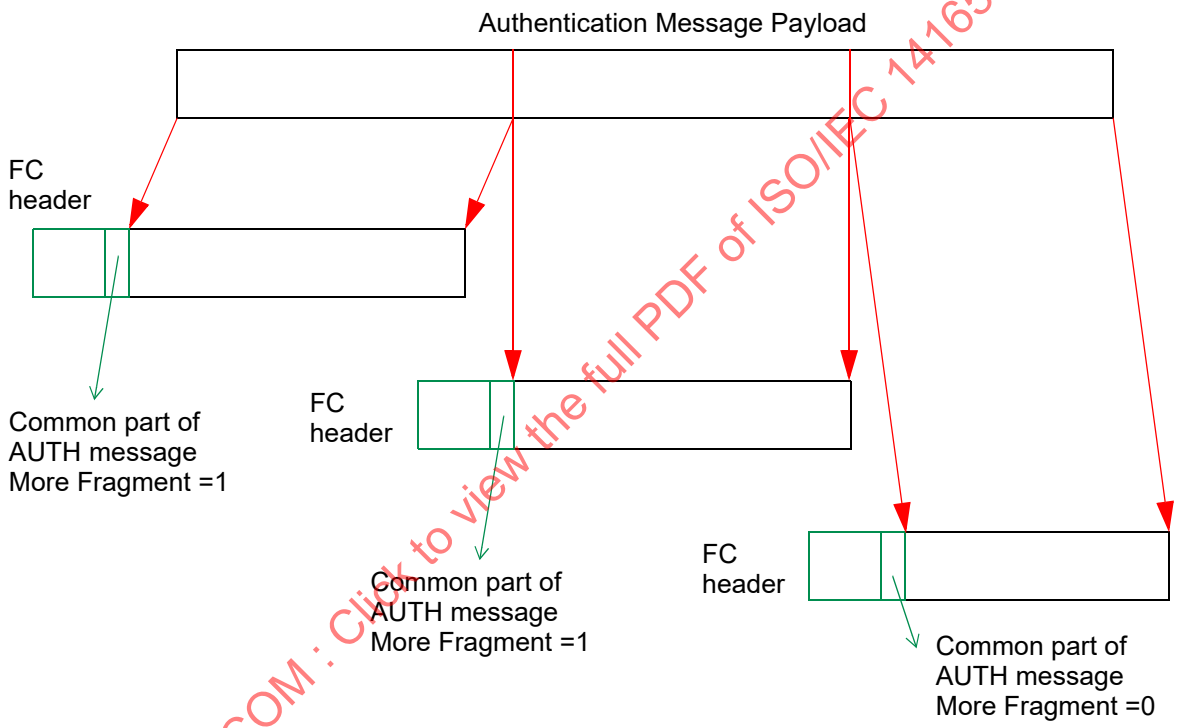

Figure 16 – AUTH_ELS Fragmentation Process

A receiver shall use the value of the More Fragments bit (see table 7) to determine if an Authentication message is complete or if more fragments follow. The Message Length field of the common part of the AUTH_ELS allows the receiver to establish an appropriately sized buffer to reassemble the entire Authentication message. Each fragment shall be accepted by replying with an LS_ACC to the sender. The sender shall not transmit the next fragment of an Authentication message until the LS_ACC for the sent fragment has been received. When the last fragment is received (i.e., More Fragment bit = 0b) the receiver is able to process the complete Authentication message.

If the sender does not receive the LS_ACC associated with a specific fragment within 2 times R_A_TOV, it shall retransmit the fragment. The Sequence Number bit (see table 7) allows the receiver to recognize a retransmitted fragment. The Sequence Number bit shall be initialized to zero in the first fragment of an Authentication message to be fragmented, and shall be alternated in each subsequent fragment of the

same Authentication message. Given that any fragment needs to be accepted with an LS_ACC before the following fragment may be sent, only one fragment may be in transit at any given time. The receiver shall detect a retransmitted fragment when it has the same Sequence Number as the one previously received, if it belongs to the same Authentication message.

Figure 17 shows an example of how the Sequence Number is used to perform error recovery. The second fragment of an Authentication message is delivered to the receiver, but the responding LS_ACC is lost. When the 2 times R_A_TOV timeout expires, the Exchange x2 is closed and the fragment is retransmitted. Any LS_ACC arriving in error after the 2 times R_A_TOV timeout are discarded. The receiver is able to recognize the fragment as a duplicated fragment, because the Sequence Number does not change in respect to the previously received fragment of the same Authentication message.

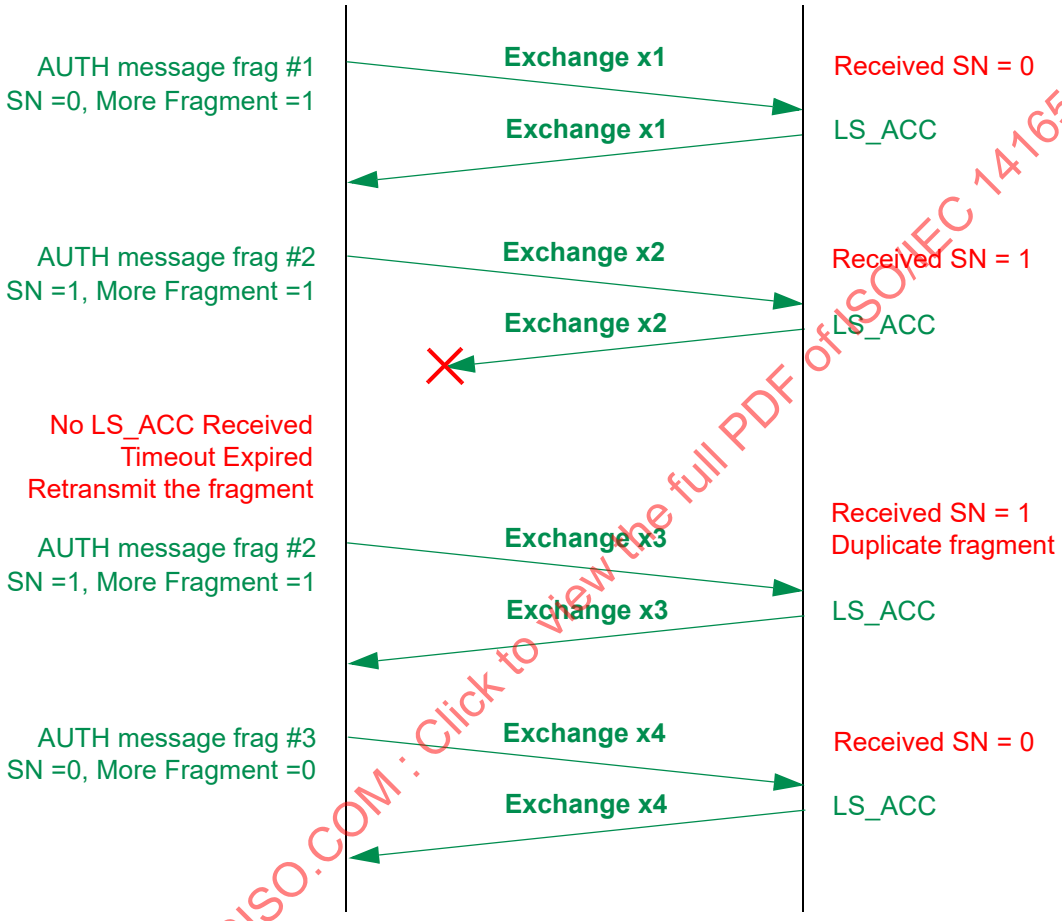

Figure 17 – Use of the Sequence Number Bit Example

An example of an Authentication Transaction using the fragmentation capability of AUTH_ELS and the RPBC ELS (see FC-LS-2) to discover ELS size limitations is shown in figure 18.

In this example the PLOGI process discovers particular buffer conditions to report, with the Query Buffer Condition bit set to one in the Common Service Parameters field. The RPBC ELS is then used, so both parties understand their limitations in handling ELS buffers. Then the Authentication is performed, by fragmenting each Authentication message in two or more, if required, AUTH_ELS fragments.

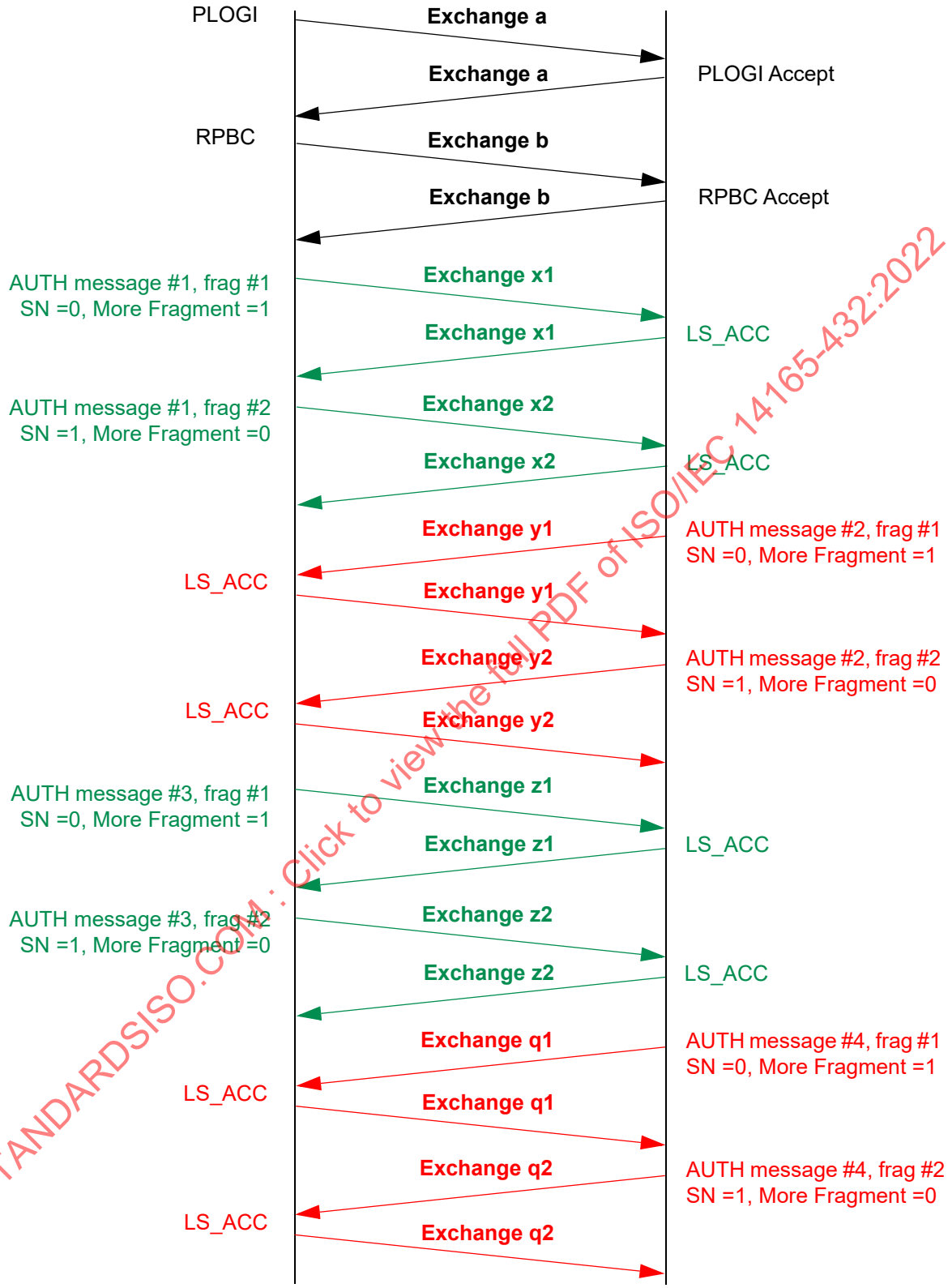

Figure 18 – FC-2 Authentication Mapping with AUTH_ELS Fragmentation Example

Implementations supporting only DH-CHAP with a NULL DH algorithm do not need to use AUTH_ELSs larger than 128 bytes. Implementations supporting other Authentication protocols are required to support at least 2048 bytes single frame ELSs. The FC SA Management protocol may require multi frame ELS Sequences or the use of AUTH fragmentation.

5.10.5 Authentication and Login

The AUTH_ELS ELS requires Login (e.g., it shall be accepted by an Nx_Port only when an N_Port Login is in effect between the sending Nx_Port and the destination Nx_Port).

The Login process may trigger Authentication between an Nx_Port and its local Fx_Port, as well as between two Nx_Ports. The Security Bit of the Common Service parameters (bit 21, word 1 of the FLOGI/PLOGI/LS_ACC ELSs, see FC-LS-2) is used for that purpose. Table 59 shows the applicability of the Security Bit.

Table 59 – Security Bit Applicability

Service Parameter	Word	Bits	PLOGI and PLOGI LS_ACC Parameter applicability		FLOGI Parameter applicability		FLOGI LS_ACC Parameter applicability	
			Class		Class		Class	
			2	3	2	3	2	3
Common Features	1	31 .. 16						
Security Bit	1	21	y	y	y	y	y	y
Key: y - indicates yes, applicable (i.e. has meaning); n - indicates no, not applicable (i.e. has no meaning)								

When set to one in the FLOGI or PLOGI request, the Security Bit indicates that the sending Nx_Port is able to perform Authentication. When set to one in the FLOGI LS_ACC reply, the Security Bit indicates that the Fabric requires the Nx_Port to Authenticate with the Fabric before granting access. When set to one in the PLOGI LS_ACC reply, the Security Bit indicates that the responding Nx_Port requires the sending Nx_Port to Authenticate before granting access.

Table 60 shows the usage of the Security Bit in the FLOGI process.

Table 60 – Security Bit usage with FLOGI

Requesting Nx_Port	Fabric Policy	Behavior
Security Bit = 0b	Authentication not Required	LS_ACC with Security Bit = 0b
Security Bit = 1b	Authentication not Required	LS_ACC with Security Bit = 0b
Security Bit = 0b	Authentication Required	LS_RJT (see table 62)
Security Bit = 1b	Authentication Required	LS_ACC with Security Bit = 1b

Table 61 shows the usage of the Security Bit in the PLOGI process.

Table 61 – Security Bit usage with PLOGI

Requesting Nx_Port	Responding Nx_Port Policy	Behavior
Security Bit = 0b	Authentication not Required	LS_ACC with Security Bit = 0b
Security Bit = 1b	Authentication not Required	LS_ACC with Security Bit = 0b
Security Bit = 0b	Authentication Required	LS_RJT (see table 62)
Security Bit = 1b	Authentication Required	LS_ACC with Security Bit = 1b

When the Security Bit is set to one in the FLOGI or PLOGI LS_ACC, the Requesting Nx_Port shall initiate an Authentication Transaction. The Fx_Port or the Responding Nx_Port shall not accept any other kind of traffic before Authentication is successfully completed. The only exception to this rule is when the Query Data Buffer Conditions bit is set to one in the Login process, when the Requesting Nx_Port shall issue an RPBC ELS before starting the Authentication Transaction. This allows the two FC_Ports to discover their possible ELS size limitations, and to overcome these limitations by using the fragmentation method specified in 5.10.4.

If the Fabric or the Responding Nx_Port require Authentication, but the Requesting Nx_Port is not capable to perform Authentication, the Login shall be rejected with the Reason Code and Reason Code Explanation shown in table 62.

Table 62 – Login LS_RJT Reasons

Reason	Reason Code	Reason Code Explanation
Authentication Required	03h	48h

5.11 Re-Authentication

The protocols described in clause 5 may be used for re-authentication. Either of the two authenticated entities may re-authenticate the other one whenever needed by starting a new Authentication Transaction. Re-authentication begins by sending an AUTH_Negotiate message with a new Transaction Identifier to the other entity, and may also generate a new session key between the two associated entities (see 5.8.1 and 5.10.1). The Authentication Protocol shall proceed as described in 5.8, 5.9 or 5.10. Re-authentication should be implemented in a way that does not cause traffic disruption unless re-authentication fails.

Re-authentication may use different parameters or a different Authentication Protocol. However, in most cases the same protocol and parameters used for the first Authentication are used for re-authentication.

When unidirectional re-authentication is desired (see 5.4), an entity may request the other to become the initiator of re-authentication by sending an AUTH_Reject message with any value for the Transaction Identifier, and with Reason Code ‘Logical Error’ and Reason Code Explanation ‘Restart Authentication Protocol’. An alternative way to re-authenticate is to send an AUTH_Negotiate message.

NOTE 22 – Sending an AUTH_Negotiate to restart a unidirectional Authentication requires some unnecessary computations. However, receiving an AUTH_Reject message to restart a unidirectional Authentication may be logged as an error.

A receiver shall be able to process both AUTH_Reject and AUTH_Negotiate messages to restart a unidirectional re-authentication, a sender may select one of the two methods.

5.12 Timeouts

The originator of an AUTH_ILS or B_AUTH_ILS Exchange shall detect an Exchange error following Sequence Initiative transfer if the SW_ACC or SW_RJT Sequence is not received within a timeout interval of 2 times R_A_TOV. When this error is detected, the recovery action should be to retransmit in a different Exchange the AUTH_ILS or B_AUTH_ILS up to two times. If the error persists for three consecutive times, the Authentication Transaction shall be aborted and the entity shall act as if the Authentication Transaction has failed. The receiver should be able to process duplicated messages up to three times. An alternative recovery action may be to restart the Authentication Transaction.

The originator of an Exchange for an AUTH_ELS or for a fragment of an AUTH_ELS shall detect an Exchange error following Sequence Initiative transfer if the LS_ACC or LS_RJT Sequence is not received within a timeout interval of 2 times R_A_TOV. When this error is detected, the recovery action should be to retransmit in a different Exchange the AUTH_ELS or the fragment of AUTH_ELS up to two times. If the error persists for three consecutive times, the Authentication Transaction shall be aborted and the entity shall act as if the Authentication Transaction has failed. The receiver should be able to process duplicated messages up to three times. An alternative recovery action may be to restart the Authentication Transaction.

The sender of an AUTH message shall detect an error following the reception of the related Accept (i.e., LS_ACC or SW_ACC) if the reply AUTH message is not received within AUTH_TOV. When this error is detected, the entity may:

a) Act as if the Authentication Transaction has failed and terminate the communication; or

b) Restart a new Authentication Transaction, by sending an AUTH_Reject message with Reason Code ‘Logical Error’ and Reason Code Explanation ‘Restart Authentication Protocol’, possibly followed by a new AUTH_Negotiate.

The default value for AUTH_TOV is 45 seconds.

6 Security Association Management Protocol

6.1 Overview

6.1.1 General

A Fibre Channel Security Association (SA) Management transaction occurs between an SA_Initiator and an SA_Responder. The SA Management protocol may use the session key produced by a transaction of an FC Authentication protocol (see clause 5) as the initial key to establish a Security Association. The SA Management protocol may be initiated by either the Authentication Initiator or the Authentication Responder of the FC Authentication protocol. SAs are unidirectional and always exist as an SA pair of the same type but in opposite directions. There are two types of SAs:

a) An IKE_SA (one per direction) for secure communication of SA management functions; and

b) Child_SAs, created using the IKE_SA, for secure communication of FC traffic.

The SA Management protocol is a subset of the IKEv2 protocol (see RFC 5996) suitable for Fibre Channel. Notes are used to explain instances where the SA Management protocol differs from the IKEv2 specification. The term exchange is used in this clause to signify a pair of related messages one of which is a response to the other one.

SA messages are encapsulated in AUTH messages. An SA Management Transaction is then identified by a Transaction Identifier (T_ID). The SA Management protocol begins with a set of four messages that establish the IKE_SA. To establish one or more Child_SAs additional messages are required (see 6.1.4). An example of the SA Management protocol with the transactions above the dashed line showing the establishment of the IKE_SA and the transactions below the dashed line showing the establishment of a Child_SA is shown in figure 19 using the notation shown in table 63.

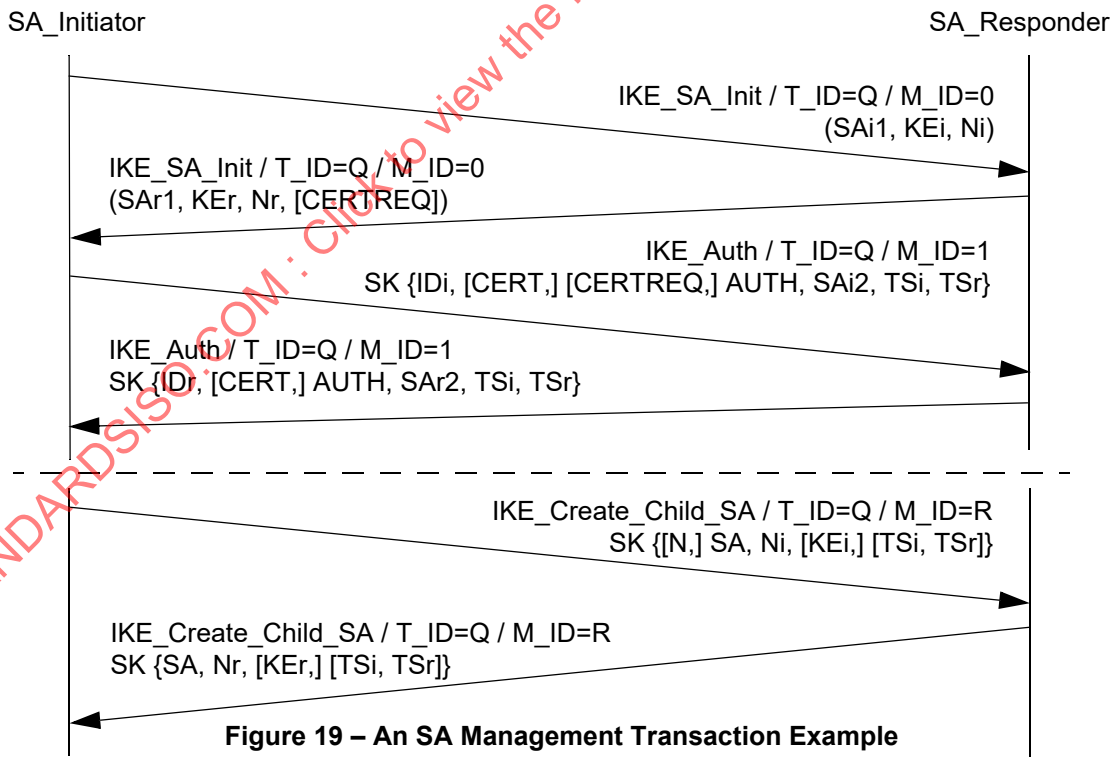

Figure 19 – An SA Management Transaction Example

Each message of the SA Management protocol encapsulates an IKEv2 message, composed of an IKE_Header Payload followed by a sequence of different IKE Payloads.

The IKE_Header Payload (see 6.2.2) includes the Security Parameters Indexes (SPIs), the IKE protocol version number, a set of flags, and a Message_ID (M_ID) used to match IKEv2 message requests to responses.

Each SA Management message (see 6.3, 6.4, and 6.5) is composed of a sequence of one or more of the IKE Payloads shown in table 63.

Table 63 – IKE Payloads Summary

IKE Payload	Notation	Description	Reference
Security_Association	SAi	Contains Security Association parameters proposed by SA_Initiator	6.3.2
	SAr	Contains Security Association parameters selected by SA_Responder	
Key_Exchange	KEi	Used by the SA_Initiator to perform a cryptographically secure Key exchange	6.3.3
	KEr	Used by the SA_Responder to perform a cryptographically secure Key exchange	
Nonce	Ni	Used by the SA_Initiator for anti-replay protection	6.3.4
	Nr	Used by the SA_Responder for anti-replay protection	
Identification	IDi	Used to specify the identity of the SA_Initiator	6.4.3
	IDr	Used to specify the identity of the SA_Responder	
Traffic Selector	TSi	Used by the SA_Initiator to specify the intended scope of the Security Association negotiated during the SA transaction	6.4.5
	TSr	Used by the SA_Responder to specify the intended scope of the Security Association negotiated during the SA transaction	
Authentication	AUTH	Contains cryptographic material used for the authentication and integrity protection of the SA Management message	6.4.4
Certificate	CERT, CERTREQ	Used to include Certificate material	6.4.6 6.4.7
Encrypted	E	Contains other IKE Payloads in encrypted form	6.4.2
Notify	N	Used to transmit informational data	6.6.2
Delete	D	Used to delete Security Associations	6.6.3
Vendor_ID	V	Used to identify and recognize Vendor Specific implementations	6.6.4

The Notify, Delete, and Vendor_ID payloads are not shown in figure 19. See 6.6.2, 6.6.3, and 6.6.4, respectively, for how they are used.

An SA Management transaction begins by exchanging a pair of IKE_SA_Init messages, that negotiate cryptographic algorithms, exchange nonces, and perform a Diffie-Hellman computation. The result of the IKE_SA_Init exchange is the establishment of the IKE_SA (see 6.1.2), used to protect the subsequent messages of the SA Management protocol.

The SA Management transaction continues by exchanging a pair of IKE_Auth messages, that authenticate the IKE_SA_Init messages, exchange identities and Certificates, if used, and establish the first Child_SA. Parts of these messages are encrypted and integrity protected with keys established through the IKE_SA_Init exchange, therefore the identities are hidden from eavesdroppers and all fields in all the messages are authenticated. The result of the IKE_Auth exchange is the establishment of a Child_SA (see 6.1.4), used to protect Fibre Channel frames. Additional Child_SAs may be established by exchanging IKE_Create_Child_SA messages (see 6.1.4).

6.1.2 IKE_SA_Init Overview

As shown in figure 19, the SA_Initiator starts an SA Management Transaction by sending the IKE_SA_Init message to the SA_Responder. The SAi1 Payload indicates the cryptographic algorithms the SA_Initiator supports for the IKE_SA (see 6.3.2.2). The KEi Payload indicates the SA_Initiator's Diffie-Hellman value. The Ni Payload indicates the SA_Initiator's nonce.

The SA_Responder selects a cryptographic suite from those proposed by the SA_Initiator and indicates that selection in the SAr1 Payload, completes the Diffie-Hellman exchange with the KEr Payload, and sends its nonce in the Nr Payload. The optional CERTREQ payload (see 6.4.7) may be used to specify the SA_Responder's trust anchors list (see X.509v3).

At this point in the negotiation each party may generate SKEYSEED (i.e., the shared key that is resulting from the Key exchange, see 6.8.11), from which all keys are derived for that IKE_SA. All the messages that follow are encrypted and integrity protected, except the headers. The keys used for the encryption (SK_e) and integrity protection (SK_a) are derived from SKEYSEED. A separate SK_e and SK_a is computed for each direction. In addition to the keys SK_e and SK_a computed from the DH value for protection of the IKE_SA, another quantity SK_d is computed and used for computation of further keying material for Child_SAs. The notation SK { ... } indicates that these IKE Payloads are encrypted and integrity protected using that direction's SK_e and SK_a.

All messages following the IKE_SA_Init exchange are cryptographically protected using the cryptographic algorithms and keys negotiated in the IKE_SA_Init exchange. The subsequent messages use the syntax of the Encrypted Payload described in 6.4.2.

6.1.3 IKE_Auth Overview

As shown in figure 19, the SA_Initiator asserts its identity with the IDi Payload, proves knowledge of the secret corresponding to IDi and integrity protects the contents of the first message using the Authentication Payload (see 6.8.13). The SA_Initiator may send its Certificate(s) in CERT Payload(s) and a list of its trust anchors in CERTREQ Payload(s). If any CERT Payloads are included, the first Certificate provided shall contain the public key used to verify the Authentication Payload. The SA_Initiator begins negotiation of a Child_SA using the SAi2 Payload. The remaining IKE Payloads, starting with SAi, are specified in the description of the IKE_Create_Child_SA exchange (see 6.5).

NOTE 23 – Unlike IKEv2, the SA_Initiator optional Identification Payload for the SA_Responder (IDr) is not used in this standard. In IKEv2, the SA_Initiator may include an IDr in the IKE_Auth message. This allows the SA_Initiator to specify which of the SA_Responder’s identities it wants to communicate with. This would be useful when an SA_Responder has multiple identities, but in Fibre Channel each entity has a single identity.

The SA_Responder asserts its identity with the IDr Payload, optionally sends one or more Certificates, with the Certificate containing the public key used to verify the Authentication Payload listed first, authenticates its identity and protects the integrity of the second message with the Authentication Payload, and completes negotiation of a Child_SA with an IKE_Create_Child_SA exchange (see 6.1.4).

The recipients of an IKE_Auth message shall verify that all signatures and MACs are computed correctly and that the names in the Identification Payloads correspond to the keys used to generate the Authentication Payload.

6.1.4 IKE_Create_Child_SA Overview

As shown in figure 19, an IKE_Create_Child_SA exchange consists of a single request/response pair. The result of an IKE_Create_Child_SA exchange is a Child_SA pair. an IKE_Create_Child_SA exchange may be initiated by either end of the IKE_SA after the IKE_SA_Init and IKE_Auth exchanges are completed. The term SA_Initiator refers to the endpoint initiating this exchange.

A Child_SA pair is created by exchanging IKE_Create_Child_SA messages. The IKE_Create_Child_SA request may contain a Key_Exchange Payload for an additional Diffie-Hellman exchange to enable stronger guarantees of forward secrecy for the Child_SA. The keying material for the Child_SA is a function of SK_d, established during the establishment of the IKE_SA, and of the nonces exchanged during the IKE_Create_Child_SA exchange. The keying material for the Child_SA is also a function of the Diffie-Hellman value if Key_Exchange Payloads are included in the IKE_Create_Child_SA exchange. In the Child_SA created as part of the initial exchange, a second Key_Exchange Payload and nonce shall not be sent. The nonces from the initial exchange are used in computing the keys for the Child_SA.

The SA_Initiator sends one or more SA Proposals in the SA Payload, a nonce in the Ni Payload, an optional Diffie-Hellman value in the KEi Payload, and the proposed Traffic Selectors in the TSi and TSr Payloads. If an IKE_Create_Child_SA exchange is rekeying an existing SA other than the IKE_SA, the leading Notify Payload of type REKEY_SA shall identify the SA being rekeyed. If an IKE_Create_Child_SA exchange is not rekeying an existing SA, the Notify Payload shall be omitted. If the SA proposes multiple Diffie-Hellman groups, KEi shall be an element of the group the SA_Initiator expects the SA_Responder to accept. If the SA_Initiator guesses wrong, the IKE_Create_Child_SA exchange fails and the SA_Initiator has to retry with a different KEi.

The IKE_Create_Child_SA message, including the header, is integrity protected and the part of the message following the header is encrypted using the cryptographic algorithms negotiated for the IKE_SA.

The SA_Responder replies, using the same Message_ID to respond, with the accepted Proposal in an SA Payload, and a Diffie-Hellman group in the KEr Payload, if KEi was included in the request and the selected cryptographic suite includes that Diffie-Hellman group. If the SA_Responder chooses a cryptographic suite with a different Diffie-Hellman group, it shall reject the received request (see 6.6.2). The SA_Initiator should repeat the request, but now with a KEi Payload from the Diffie-Hellman group the SA_Responder selected.

The Traffic Selectors for traffic to be sent on that SA are specified in the TS Payloads, that may be a subset of what the SA_Initiator of the Child_SA proposed. Traffic Selectors are omitted if an IKE_Create_Child_SA request is being used to change the key of the IKE_SA.

6.2 SA Management Messages

6.2.1 General Structure

The Fibre Channel SA Management protocol shall be used to negotiate SAs between Nx_Ports, between Nx_Ports and Fx_Ports, and between Switches.

To allow the capability of using the SA Management protocol as an Authentication Protocol, the SA messages are encoded as AUTH messages, having the general structure defined in 5.2. SA messages are encoded as AUTH_ILSs when the involved entities are Switches, and as AUTH_ELSs when the involved entities include at least one Nx_Port.

All the IKE Payload definitions in this clause are relative to the Message Payload field of the AUTH_ILS or AUTH_ELS format. The AUTH Message Codes used by the SA Management protocol are listed in table 8.

6.2.2 IKE_Header Payload

All SA messages begin with the same general IKE_Header Payload, as shown in table 64

Table 64 – IKE_Header Payload Format

Item	Size (Bytes)
IKE_SA SA_Initiator's SPI	8
IKE_SA SA_Responder's SPI	8
Next IKE Payload	1
IKE Protocol Version	1
Reserved	1
IKE Flags	1
IKE Message_ID	4

IKE_SA SA_Initiator's SPI: A value selected by the SA_Initiator to identify a unique IKE Security Association. This value shall not be zero.

IKE_SA SA_Responder's SPI: A value selected by the SA_Responder to identify a unique IKE Security Association. This value shall be zero in the first message of an IKE initial exchange and shall not be zero in any other message.

Next IKE Payload: Indicates the type of IKE Payload (see table 67) that immediately follows the IKE_Header Payload. The format and value of each IKE Payload is defined in this clause.

IKE Protocol Version: Contains an 8-bit unsigned binary integer that specifies the version of the SA protocol. The four most significant bits contain the Major Version, while the four least significant bits contain the Minor Version. Since this standard is implementing IKEv2, this field shall be set to 20h (i.e., Major Version 2h, Minor Version 0h). Messages containing a Major Version other than 2h shall be rejected or ignored. The Minor Version number shall be ignored.

IKE Flags: indicates specific options that are used for the message. The presence of options is indicated by the appropriate bit in the flags field being set to one. Table 65 describes the defined flags.

Table 65 – IKE Flags

Bit	Notation	Description
7 .. 6		Reserved
5	R	Response: Indicates that this message is a response to a message containing the same Message_ID. This bit shall be set to zero in all request messages and shall be set to one in all responses. An IKE endpoint shall not generate a response to a message that is marked as being a response.
4	V	Version: Indicates that the transmitter is capable of supporting a higher major version number of the protocol than the one indicated in the major version number field. Implementations shall set to zero this bit when sending and shall ignore it in incoming messages.
3	I	Initiator: Shall be set to one in messages sent by the original SA_Initiator of the IKE_SA and shall be set to zero in messages sent by the original SA_Responder. It is used by the recipient to determine which eight bytes of the SPI was generated by the recipient.
2 .. 0		Reserved

IKE Message_ID: used to control retransmission of lost messages and matching of requests and responses. This field is used to prevent message replay attacks (see 6.8.1 and 6.8.2).

6.2.3 Chaining Header

Each SA Management message is composed of one or more different IKE Payload types. Each IKE Payload type begins with a Chaining Header (see table 66) that indicates the next IKE Payload type, specifies if the current IKE Payload is critical, and specifies the length of the current IKE Payload. IKE Payloads are processed in the order in which they appear in an SA Management message according to the Next IKE Payload field in the IKE_Header Payload. IKE Payloads following the first one are processed according to the Next IKE Payload field in the previous IKE Payload, until a Next IKE Payload field of zero indicates that no IKE Payloads follow. If an Encrypted Payload is found, it is decrypted and its content parsed as additional IKE Payloads. An Encrypted Payload shall be the last IKE Payload in a message and an Encrypted Payload shall not contain another Encrypted Payload.

Table 66 – Chaining Header Format

Item	Size (Bytes)
Next IKE Payload	1
Chaining Flags	1
IKE Payload Length	2

Next IKE Payload: Identifier for the IKE Payload type of the next IKE Payload in the message. This field provides a chaining capability whereby additional IKE Payloads may be added to a message by appending them to the end of the message and setting the Next IKE Payload field of the preceding IKE Payload to indicate the new IKE Payload's type. If the current IKE Payload is the last in the message, then this field shall be set to zero, except for the Encrypted Payload. The Encrypted Payload, which shall always be the last IKE Payload of a message, is an exception because it contains data structures in the format of additional IKE Payloads. In the header of an Encrypted Payload the Next IKE Payload field is set to the IKE Payload type of the first contained IKE Payload instead of zero (see 6.4.2).

Table 67 describes the IKE Payload types and lists the messages where they are used.

Table 67 – IKE Payload Type Values

Value[a]	IKE Payload Type	Notation	Messages
33	Security_Association	SA	IKE_SA_Init, IKE_Auth, IKE_Create_Child_SA
34	Key_Exchange	KE	IKE_SA_Init, IKE_Create_Child_SA (opt.)
35	Identification - SA_Initiator	IDi	IKE_Auth
36	Identification - SA_Responder	IDr	IKE_Auth
37	Certificate	CERT	IKE_Auth (opt.)
38	Certificate Request	CERTREQ	IKE_Init (response, opt.), IKE_Auth (request, opt.)
39	Authentication	AUTH	IKE_Auth
40	Nonce (SA_Initiator, SA_Responder)	Ni, Nr	IKE_SA_Init, IKE_Create_Child_SA
41	Notify	N	All (optional)
42	Delete	D	All (optional)
43	Vendor_ID	V	All (optional)
44	Traffic Selector - SA_Initiator	TSi	IKE_Auth, IKE_Create_Child_SA (opt.)
45	Traffic Selector - SA_Responder	TSr	IKE_Auth, IKE_Create_Child_SA (opt.)
46	Encrypted	E	IKE_Auth, IKE_Create_Child_SA, IKE_Informational
128 .. 255	Vendor Specific		All (optional)
all others	Reserved to IANA		

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

Chaining Flags: Table 68 shows the defined flags.

Table 68 – Chaining Flags

Bit	Description
7	Critical Bit. For the purpose of defining the Critical bit, the term Payload type refers to the Payload type of the current Payload (i.e., the value in the Next Payload field of the previous Payload). The sender shall set the Critical bit to zero to specify the recipient shall skip this Payload if the recipient does not understand the Payload type. The sender shall set the Critical bit to one to specify the recipient shall reject the entire message if the recipient does not understand the Payload type. If the recipient does not understand a Payload type and the Critical bit is set to one, the recipient shall reject the entire message. If the recipient does not understand a Payload type and the Critical bit is set to zero, the recipient shall skip the Payload and continue processing the message. Recipients shall ignore the Critical bit if the Payload type is understood. Implementations are required to understand the Payload types defined in this standard. Senders shall set the Critical bit to zero for Payload types defined in this standard. The recipient shall reject the entire message if any skipped Payload contains an invalid Next Payload or Payload Length field.
6 .. 0	Reserved

IKE Payload Length: Length in bytes of the current IKE Payload, including the Chaining Header.

6.2.4 AUTH_Reject Message Use

An SA Management Transaction may be concatenated to an Authentication Transaction (see 6.7.2), or may replace an Authentication Transaction (see 6.7.3).

When an SA Management Transaction is concatenated to an Authentication Transaction (see 6.7.2), AUTH Concatenation is required (see 5.2.2 and 5.2.3). If an implementation does not support AUTH Concatenation, an AUTH_Reject with Reason Code ‘Logical Error’ and Reason Code Explanation ‘AUTH Concatenation not Supported’ shall be returned to a received AUTH_Negotiate message having the Concatenation Flag set to one, while an AUTH_Reject with Reason Code ‘Authentication Failure’ and Reason Code Explanation ‘Incorrect Authentication Protocol Message’ shall be returned to any other received AUTH message having the Concatenation Flag set to one (see 5.3.7).

An SA Management Transaction used for both authentication and SA management is indicated by the Authentication Initiator including the IKEv2-AUTH protocol identifier in the list of usable Authentication Protocols contained in the AUTH_Negotiate message (see 6.7.3). If the Authentication responder does not support the SA Management protocol and receives only the IKEv2-AUTH protocol identifiers in the list of usable Authentication Protocols contained in the AUTH_Negotiate message, it shall reply with an AUTH_Reject message with Reason Code ‘Logical Error’ and Reason Code Explanation ‘Authentication Mechanism Not Usable’ (see 5.3.7).

6.3 IKE_SA_Init Message

6.3.1 Overview

The IKE_SA_Init message is used to establish the IKE_SA. This message contains a Security_Association (SA) Payload that is used to negotiate the SA Transforms and parameters, a Key_Exchange (KE) Payload used to generate new key material, and a Nonce (Ni or Nr) Payload. The same message format is used by

the SA_Initiator and the SA_Responder, but the SA_Initiator proposes a set of Transforms, while the SA_Responder selects a Transform among the proposed ones. An optional CERTREQ Payload may be included by the SA_Responder. The IKE_SA_Init exchange is shown in figure 20.

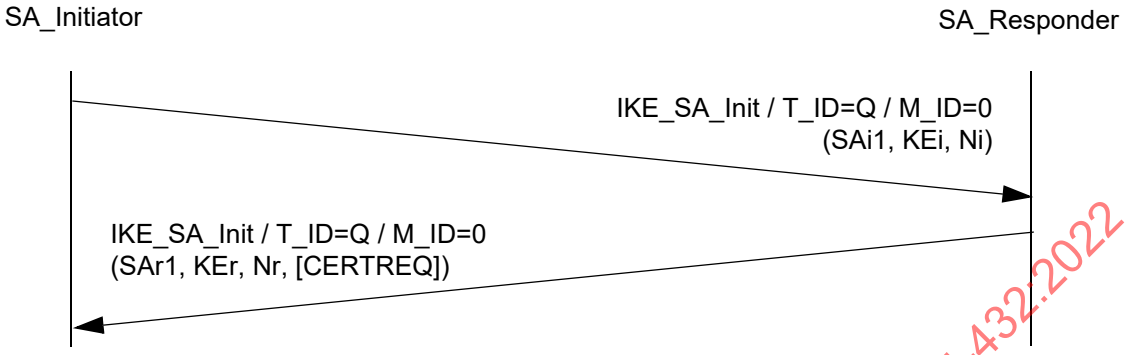

Figure 20 – An IKE_SA_Init exchange.

The Message Payload of the IKE_SA_Init message is shown in table 69.

Table 69 – IKE_SA_Init Message Payload

Item	Size (Bytes)	Reference
IKE_Header Payload	24	6.2.2
Security_Association Payload	variable	6.3.2
Key_Exchange Payload	variable	6.3.3
Nonce Payload	variable	6.3.4
Optional Certificate Request Payload (possibly included only by the SA_Responder)	variable	6.4.7
Optional Vendor_ID Payload	variable	6.6.4

6.3.2 Security_Association Payload

6.3.2.1 Negotiation of Security Association Parameters

During Security Association negotiation SA_Initiators present Proposals to SA_Responders. SA_Responders shall select a single complete set of parameters from the Proposals or reject all Proposals if none are acceptable. If there are multiple Proposals, the SA_Responder shall select a single Proposal Number and return all of the Proposal substructures with that Proposal Number. If there are multiple Transforms with the same type the SA_Responder shall select a single one. Any Attributes of a selected Transform shall be returned unmodified. The SA_Initiator of an exchange shall check that the accepted Proposal is consistent with one of its Proposals, and if not that response shall be rejected.

When negotiating Diffie-Hellman groups, SA Proposals include proposed Attributes and a Diffie-Hellman public number in a Key_Exchange Payload in the message. If in the initial exchange the SA_Initiator offers to use one of several Diffie-Hellman groups, it should pick the one the SA_Responder is most likely to accept and include a Key_Exchange Payload corresponding to that group. If the guess turns out to be wrong, the SA_Responder indicates the correct group in the response and the SA_Initiator should pick an element of that group for its Key_Exchange Payload when retrying the first message. It should, however, continue to propose its full supported set of groups in order to prevent a man in the middle downgrade attack.

Certain negotiable Attributes may have ranges or multiple acceptable values. These include the key length of a variable key length symmetric cipher. To improve interoperability and to support upgrading end-points independently, implementations should accept values that they deem to supply greater security (e.g., if a peer is configured to accept a variable length cipher with a key length of X bits and is offered that cipher with a larger key length, the implementation should accept the offer if it supports use of the longer key). Support of this capability allows an implementation to express a concept of "at least" a certain level of security (i.e., a key length of at least X bits for cipher Y).

6.3.2.2 Payload Structure

The Security_Association Payload is used to negotiate Attributes of a Security Association. An SA Payload may contain multiple Proposals. If an SA Payload contains more than one Proposal, they shall be ordered from most preferred to least preferred. Each Proposal shall contain a single protocol, where a protocol is IKE, ESP_Header, or CT_Authentication, each protocol may contain multiple Transforms, and each Transform may contain multiple Attributes. When parsing an SA Payload, the receiving entity shall check that the total Payload Length is consistent with the SA Payload's internal lengths and counts. Proposals, Transforms, and Attributes each have their own variable length encodings. They are nested such that the Payload Length of an SA Payload includes the combined contents of the SA, Proposal, Transform, and Attribute information. The length of a Proposal includes the lengths of all Transforms and Attributes it contains. The length of a Transform includes the lengths of all Attributes it contains.

The hierarchy of the syntax of the SA Payload allows for multiple possible combinations of algorithms to be encoded in a single SA. Sometimes there is a choice of multiple algorithms, while other times there is a combination of algorithms (e.g., an SA_Initiator may want to propose using (CT_Authentication with AUTH_HMAC_MD5_96) or (ESP_Header with AUTH_HMAC_MD5_96 and ENCR_3DES)).

The Proposal structure contains within it a Proposal Number and a Security Protocol_ID. The first Proposal shall have a Proposal number set to one. The other Proposals shall have a Proposal number that is one greater of the previous Proposal. A Proposal for CT_Authentication or ESP_Header has two Proposal structures, one for CT_Authentication as Proposal #1 and one for ESP_Header as Proposal #2.

NOTE 24 – Unlike IKEv2, multiple security protocols within a single Proposal (e.g., CT_Authentication and ESP_Header) are not used in this standard.

In order to propose both normal ciphers and combined-mode ciphers, that include both integrity and encryption in a single encryption algorithm, an implementation shall include two proposals in the SA Payload: one listing all the combined-mode ciphers, and the other one listing all the normal ciphers with the integrity algorithms.

Each Proposal/Protocol structure is followed by one or more Transform structures. The number of different Transforms is generally determined by the protocol. CT_Authentication may have two Transforms, an integrity check algorithm and an encryption algorithm. ESP_Header may have two Transforms, an integrity check algorithm and an encryption algorithm. IKE generally has four Transforms, a Diffie-Hellman group, an integrity check algorithm, a pseudo-random function, and an encryption algorithm. If an algorithm that combines encryption and integrity protection is proposed, it shall be proposed as an encryption algorithm, and the AUTH_NONE integrity protection algorithm shall be proposed.

NOTE 25 – Unlike IKEv2, this standard specifies that the AUTH_NONE integrity protection algorithm is always proposed when a combined mode encryption algorithm is proposed.

For each protocol, the set of permissible Transforms are assigned Transform_ID numbers that appear in the header of each Transform.

If there are multiple Transforms with the same Transform Type, the Proposal is an 'OR' of those Transforms. If there are multiple Transforms with different Transform Types, the Proposal is an 'AND' of the different groups. As an example, to propose ESP_Header with (AES_CBC or AES_CTR) and (HMAC_MD5_96 or HMAC_SHA1), the SA_Initiator has to construct one Proposal containing two Transform Type 1 candidates (one for AES_CBC and one for AES_CTR) and two Transform Type 2 candidates (one for HMAC_MD5_96 and one for HMAC_SHA1). The result is four combinations of algorithms, as shown by example 1 in table 71. This standard does not specify any method for a SA_Initiator to encode multiple Transforms within a single Proposal (e.g., (AES_CBC and HMAC_MD5_96) or (AES_CTR and HMAC_SHA1)). Instead, the SA_Initiator has to construct two different Proposals, each with two Transforms, as shown by example 2 in table 70.

Table 70 – Examples of Proposals

Proposal #	Proposal Definition	Example 1	Example 2
Proposal #1	Security Protocol_ID	ESP_Header	ESP_Header
	Transform #1	Transform Type: 1 Transform_ID: AES_CBC	Transform Type: 1 Transform_ID: AES_CBC
	Transform #2	Transform Type: 1 Transform_ID: AES_CTR	Transform Type: 3 Transform_ID: HMAC_MD5_96
	Transform #3	Transform Type: 3 Transform_ID: HMAC_MD5_96	
	Transform #4	Transform Type: 3 Transform_ID: HMAC_SHA1	
Proposal #2	Security Protocol_ID		ESP_Header
	Transform #1		Transform Type: 1 Transform_ID: AES_CTR
	Transform #2		Transform Type: 3 Transform_ID: HMAC_SHA1
Example 1 allows the recipient to select either AES_CBC and HMAC_MD5_96, or AES_CBC and HMAC_SHA1, or AES_CTR and HMAC_MD5_96, or AES_CTR and HMAC_SHA1. Example 2 allows the recipient to select either AES_CBC and HMAC_MD5_96, or AES_CTR and HMAC_SHA1.			

A given Transform may have one or more Attributes. Attributes are necessary when the Transform is used in more than one way, (e.g., when an encryption algorithm has a variable key size). The Transform specifies the algorithm and the Attribute specifies the key size. Most Transforms do not have Attributes. A Transform shall not have multiple Attributes of the same type. To propose alternate values for an Attribute (e.g., multiple key sizes for the AES encryption algorithm), an implementation shall include multiple Transforms with the same Transform Type, each with a single Attribute.

The Security_Association Payload has the format shown in table 71.

Table 71 – Security_Association Payload Format

Item	Size (Bytes)
Chaining Header	4
Last/More Proposal	1
Reserved	1
Proposal Length	2
Proposal Number	1
Security Protocol_ID	1
SPI Size	1
Number of Transforms	1
SPI	variable
Transforms Definition	variable
Last/More Proposal	1
...	
Last/More Proposal	1
...	

Chaining Header: See 6.2.3 .

Last/More Proposal: Shall be set to zero to indicate this is the last Proposal Substructure in the SA Payload. Shall be set to two to indicate there are more Proposal Substructures in the SA Payload. All other values are reserved.

Proposal Length: Length in bytes of this Proposal, including all Transforms and Attributes that follow.

Proposal Number: When a Proposal is made, the Proposal Number of the first Proposal in an SA shall be set to one, and subsequent Proposals shall be numbered one more than the previous Proposal (i.e., indicating an 'OR' of the two Proposals). When a Proposal is accepted, all of the Proposal Numbers in the SA shall be the same and shall match the number on the Proposal sent that was accepted.

NOTE 26 – Unlike IKEv2, multiple protocols on a single Proposal are not used in this standard.

Security Protocol_ID: Specifies the Security Protocol identifier for the current negotiation. The defined values are shown in table 72.

Table 72 – Security Protocol Identifiers

Security Protocol_ID[a]	Protocol
1	IKE
4[b]	ESP_Header[b]
5	CT_Authentication
201 .. 255	Vendor Specific
all others	Reserved to IANA

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

[b] This identifier is used to indicate ESP_Header in both end-to-end and link-by-link variant. End-to-end ESP_Header (see FC-FS-3) shall be used to protect N_Port to N_Port traffic. Link-by-link ESP_Header (see FC-FS-3) shall be used to protect E_Port to E_Port and N_Port to F_Port traffic, as specified in 8.9.

SPI Size: For an initial IKE_SA negotiation, this field shall be set to zero, because the SPI is obtained from the IKE_Header Payload. During subsequent negotiations, it is equal to the size in bytes of the SPI of the corresponding protocol (i.e., eight for IKE, four for ESP_Header and CT_Authentication).

Number of Transforms: Specifies the number of Transforms in this Proposal.

SPI: The sending entity's SPI. When the SPI Size field is zero, this field shall not be present in the Security_Association Payload. SPI values in the range 0 .. 255 are reserved (see RFC 4303).

Transforms Definition: This field has the format shown in table 73.

Table 73 – Transforms Definition

Item	Size (Bytes)
Last/More Transform	1
Reserved	1
Transform Length	2
Transform Type	1
Reserved	1
Transform_ID	2
Optional Transform Attributes Definition	variable
Last/More Transform	1
...	
Last/More Transform	1
...	

Last/More Transform: Shall be set to zero to indicate this is the last Transform Substructure in the SA Payload. Shall be set to three to indicate there are more Transform Substructures in the SA Payload. All other values are reserved.

Transform Length: The length in bytes of the Transform Substructure, including Attributes.

Transform Type: The type of Transform being specified in this Transform. Different protocols support different Transform types. For some protocols, some of the Transforms may be optional. If a Transform is optional and the SA_Initiator proposes that the Transform be omitted, no Transform of the given type is included in the Proposal. For both ESP_Header and CT_Authentication encryption is optional to propose and integrity is mandatory to propose.

The SA_Initiator instructs the SA_Responder not to use encryption by omitting the encryption Transform in the proposal. The SA_Initiator allows the SA_Responder to select whether encryption is used by including the ENCR_NULL encryption Transform with additional non-NULL encryption Transforms.

The AUTH_NONE authentication Transform shall be proposed only in conjunction with combined mode encryption Transforms, and shall not be proposed with ENCR_NULL or an encryption only Transform.

NOTE 27 – Unlike IKEv2, this standard specifies that an integrity Transform be always proposed and allows an encryption Transform not to be proposed.

Transform_ID: The specific instance of the Transform type being proposed (see 6.3.2.3).

Transform Attributes Definition: An optional field that contains one or more Attributes that modify or complete the specification of the Transform (see 6.3.2.5).

See Annex E for some examples of Proposals negotiation.

6.3.2.3 Transform Types

The defined Transform Types are shown in table 74.

Table 74 – Transform Type Values

Value[a]	Transform Type	Used In
1	Encryption Algorithm	IKE, optional in ESP_Header, optional in CT_Authentication
2	Pseudo-random Function	IKE
3	Integrity Algorithm	IKE, ESP_Header, CT_Authentication
4	Diffie-Hellman Group	IKE
241 .. 255	Vendor Specific	
all others[b]	Reserved to IANA	

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

[b] The "IKEv2 Parameters" registry defines the value 05h for negotiation of 64-bit extended sequence numbers. 64-bit extended sequence numbers are not supported by this standard. The SA_Initiatior and the SA_Responder use 32-bit sequence numbers for both ESP_Header and CT_Authentication.

For Transform Type 1 (Encryption Algorithms), Transform_IDs are defined in table 75.

Table 75 – Encryption Algorithms Transform_IDs (Transform Type 1)

Transform_ID[a]	Encryption Algorithm	Reference
3	ENCR_3DES	RFC 2451
11	ENCR_NULL	RFC 2410
12	ENCR_AES_CBC	RFC 3602
13	ENCR_AES_CTR	RFC 3686
20[b]	ENCR_AES_GCM[c] (with a 16 bytes ICV)	RFC 4106[d]
21[e]	ENCR_NULL_AUTH_AES_GMAC[c]	RFC 4543[d]
1 024 .. 65 535	Vendor Specific	
all others	Reserved to IANA	

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

[b] ENCR_AES_GCM with a 8 or 12 bytes ICV shall not be used.

[c] ENCR_AES_GCM and ENCR_NULL_AUTH_AES_GMAC may be used with a 128 bit key, a 192 bit key or a 256 bit key. If ENCR_AES_GCM or ENCR_NULL_AUTH_AES_GMAC is implemented, support for the 128 bit key is mandatory, support for the 192 bit and 256 bit key is optional. The key size is specified by using the Key Length Transform Attribute (see 6.3.2.5).

[d] This standard requires a variation in the content of the Additional Authentication Data (AAD) field from that specified in the RFC. The AAD field specified by the RFC shall be prefixed by the modified Fibre Channel Frame_Header (see FC-FS-3) to construct the AAD field required by this standard.

[e] ENCR_NULL_AUTH_AES_GMAC is used only for authentication, but is documented as an encryption algorithm so that it can use an initialization value.

For Transform Type 2 (Pseudo-random Functions), Transform_IDs are defined in table 76.

Table 76 – Pseudo-random Functions Transform_IDs (Transform Type 2)

Transform_ID[a, b]	Pseudo-random Function	Reference
1	PRF_HMAC_MD5	RFC 2104
2	PRF_HMAC_SHA1	RFC 2104
4	PRF_AES128_CBC	RFC 4434
1 024 .. 65 535	Vendor Specific	
all others	Reserved to IANA	

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

[b] All groups in this table are modp groups; elliptic curve groups shall not be used.

For Transform Type 3 (Integrity Algorithms), Transform_IDs are defined in table 77.

Table 77 – Integrity Algorithms Transform_IDs (Transform Type 3)

Transform_ID[a]	Integrity Algorithm	Reference
0	AUTH_NONE	
1	AUTH_HMAC_MD5_96[b]	RFC 2403
2	AUTH_HMAC_SHA1_96[c]	RFC 2404
6	AUTH_HMAC_MD5_128[d]	RFC 1321, RFC 2104
7	AUTH_HMAC_SHA1_160[e]	RFC 2104, SHA-1
1 024 .. 65 535	Vendor Specific	
all others	Reserved to IANA	

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

[b] This integrity Algorithm shall not be used with CT_Authentication. AUTH_HMAC_MD5_128 shall be used instead.

[c] This integrity Algorithm shall not be used with CT_Authentication. AUTH_HMAC_SHA1_160 shall be used instead.

[d] This integrity Algorithm shall not be used with ESP_Header. AUTH_HMAC_MD5_96 shall be used instead.

[e] This integrity Algorithm shall not be used with ESP_Header. AUTH_HMAC_SHA1_96 shall be used instead.

For Transform Type 4 (Diffie Hellman Groups), Transform_IDs are defined in table 78.

Table 78 – Diffie-Hellman Group Transform_IDs (Transform Type 4)

Transform_ID[a]	DH Group	Generator	Modulus (Hex)
0	NONE		
2	1 024 bit	2	FFFFFFFFFFFFFFFFC90FDAA22168C234 C4C6628B80DC1CD129024E088A67CC74 020BBEA63B139B22514A08798E3404DD EF9519B3CD3A431B302B0A6DF25F1437 4FE1356D6D51C245E485B576625E7EC6 F44C42E9A637ED6B0BFF5CB6F406B7ED EE386BFB5A899FA5AE9F24117C4B1FE6 49286651ECE65381FFFFFFFFFFFFFFFF
5	1 536 bit	2	see RFC 3526
14	2 048 bit	2	see RFC 3526
15	3 072 bit	2	see RFC 3526
16	4 096 bit	2	see RFC 3526
17	6 144 bit	2	see RFC 3526
18	8 192 bit	2	see RFC 3526
1 024 .. 65 535	Vendor Specific		
all others	Reserved to IANA		

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

The number and type of Transforms included in an SA Payload are dependent on the protocol in the SA itself. An SA Payload proposing the establishment of an SA has the following mandatory and optional Transform types. A compliant implementation shall understand all mandatory and optional types for each protocol it supports, although that implementation need not accept Proposals with unacceptable suites. A Proposal may omit the optional types if the only value for them it accepts is NONE. The mandatory Transform types are shown in table 79.

Table 79 – Mandatory Transform Types

Protocol	Mandatory Types	Optional Types
IKE	Encryption Algorithms, Pseudo-random Functions, Integrity Algorithms, Diffie-Hellman Groups	
ESP_Header	Integrity Algorithms[a]	Encryption Algorithms
CT_Authentication	Integrity Algorithms[a]	Encryption Algorithms

[a] A combined mode algorithm satisfies the requirement for integrity even if it is negotiated as an encryption only algorithm. No additional integrity algorithm is proposed in this case.

6.3.2.4 Mandatory Transform_IDs

The mandatory and recommended Transform_IDs for the SA Management protocol, the ESP_Header protocol and the CT_Authentication protocol are shown in table 80.

Table 80 – Mandatory and Recommended Transform_IDs (part 1 of 2)

	Encryption Algorithms (see table 75)	Pseudo-random Functions (see table 76)	Integrity Algorithms (see table 77)	DH Groups (see table 78)
Mandatory[a] Transforms for the SA Management protocol	ENCR_AES_CBC (Key length 128-bit)	PRF_HMAC_SHA1	AUTH_HMAC_SHA1_96	14[e] (2 048 bit)
Mandatory[a] Transforms for the ESP_Header protocol	ENCR_NULL, ENCR_AES_GCM (Key length 128-bit, 16 bytes ICV)	-	ENCR_NULL_-_AUTH_AES_GMAC[c] (Key length 128-bit)	-

[a] These Transforms are mandatory to implement.

[b] These Transforms are recommended to be implemented as recommended algorithms to protect against the possibility that major flaws are found in the mandatory algorithms.

[c] ENCR_NULL_AUTH_AES_GMAC is an integrity algorithm, although it is defined as a combined mode encryption algorithm in the IKEv2 registry (see table 75). This standard re-uses this definition for consistency with IKEv2.

[d] ENCR_AES_CBC is required for CT_Authentication because it is required by IKEv2, and the implementation of the algorithm may be common between the two protocols.

[e] Implementations should include a management facility that allows specification, by a user or system administrator, of Diffie-Hellman parameters (i.e., the generator, modulus, and exponent lengths and values) for new DH groups. Implementations should provide a management interface via which these parameters and the associated Transform_IDs may be entered, by a user or system administrator, to enable negotiating such groups.

Table 80 – Mandatory and Recommended Transform_IDs (part 2 of 2)

	Encryption Algorithms (see table 75)	Pseudo-random Functions (see table 76)	Integrity Algorithms (see table 77)	DH Groups (see table 78)
Mandatory[a] Transforms for the CT_Authentication protocol	ENCR_NULL, ENCR_AES_CBC[d] (key length 128-bit)	-	AUTH_HMAC_SHA1_160	-
Recommended[b] Transforms for the SA Management protocol	ENCR_3DES	-	AUTH_HMAC_MD5_96	-
Recommended[b] Transforms for the ESP_Header protocol	ENCR_3DES	-	AUTH_HMAC_MD5_96	-
Recommended[b] Transforms for the CT_Authentication protocol	ENCR_3DES	-	AUTH_HMAC_MD5_128	-

[a] These Transforms are mandatory to implement.

[b] These Transforms are recommended to be implemented as recommended algorithms to protect against the possibility that major flaws are found in the mandatory algorithms.

[c] ENCR_NULL_AUTH_AES_GMAC is an integrity algorithm, although it is defined as a combined mode encryption algorithm in the IKEv2 registry (see table 75). This standard re-uses this definition for consistency with IKEv2.

[d] ENCR_AES_CBC is required for CT_Authentication because it is required by IKEv2, and the implementation of the algorithm may be common between the two protocols.

[e] Implementations should include a management facility that allows specification, by a user or system administrator, of Diffie-Hellman parameters (i.e., the generator, modulus, and exponent lengths and values) for new DH groups. Implementations should provide a management interface via which these parameters and the associated Transform_IDs may be entered, by a user or system administrator, to enable negotiating such groups.

Although the security of negotiated Child_SAs does not depend on the strength of the encryption and integrity protection negotiated in the IKE_SA, peers shall not negotiate ENCR_NULL as the IKE encryption algorithm and shall not negotiate AUTH_NONE as the IKE integrity protection algorithm unless a combined mode encryption algorithm is used. All implementations shall include a management facility that enables a user or system administrator to specify the suites that are acceptable for use with IKE. Upon receipt of a payload with a set of Transform_IDs, the implementation shall compare the transmitted Transform_IDs against those locally configured via the management controls, to verify that the proposed suite is acceptable based on local policy. The implementation shall reject SA Proposals that are not authorized by these IKE suite controls.

6.3.2.5 Transform Attributes

Each Transform in a Security Association payload may include Attributes that modify or complete the specification of the Transform (e.g., if an encryption algorithm has a variable length key, the key length to be used may be specified as an Attribute). Attributes are type/value pairs that may have a value of a fixed

two byte length or of variable length. For the latter, the Attribute is encoded as type/length/value, as shown in table 81.

Table 81 – Transform Attributes Definition

Item	Size (Bytes)
Attribute Type	2
Attribute Length/Value	2
Optional Attribute Value	variable
Attribute Type	2
...	

Attribute Type: A unique identifier for each type of Attribute. The most significant bit of this field is the Attribute Format bit (AF). It indicates whether the Attribute is of Type/Length/Value (TLV) format or of a shortened Type/Value (TV) format. If the AF bit is set to zero, then the Attribute is of the Type/Length/Value (TLV) form. If the AF bit is set to one, then the Attribute is of the Type/Value form.

Attribute Length/Value: When the AF bit is set to zero it is the length in bytes of the following Attribute Value field. When the AF bit is set to one it is the 2 bytes value of the Attribute, and there is no Optional Attribute Value field.

Attribute Value: The value of the Attribute associated with the Attribute Type. If the AF bit is set to zero, this field has a variable length defined by the Attribute Length/Value field. If the AF bit is set to one this field is not present.

Only one Attribute Type (i.e., Key Length) is defined, as shown in table 82, and it is of fixed length. The variable length encoding specification is included only for future extensions. The only algorithms defined in this standard that accept Attributes are the AES based encryption, including the GMAC algorithm, that require a single Attribute specifying the key length. Attributes described as basic shall not be encoded using the variable length encoding. Variable length Attributes shall not be encoded as basic even if their value may fit into two bytes.

Table 82 – Attribute Type Values

Value[a,b]	Attribute Type	Attribute Format
14	Key Length (in bits)	Type/Value (AF=1)
16 384 .. 32 767	Vendor Specific	
all others	Reserved to IANA	

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

[b] These are 15 bits values that do not include the most significant bit of the field (the AF bit).

Key Length: When using an encryption algorithm that has a variable length key, this Attribute specifies the key length in bits, using network byte order. This Attribute shall not be used when the specified encryption algorithm uses a fixed length key.

6.3.3 Key_Exchange Payload

The Key_Exchange Payload has the format shown in table 83.

Table 83 – Key_Exchange Payload Format

Item	Size (Bytes)
Chaining Header	4
DH Group number	2
Reserved	2
Key_Exchange Data	variable

Chaining Header: See 6.2.3 .

DH Group number: Identifies the Diffie-Hellman group in which the Key_Exchange Data was computed (see table 78). If the selected Proposal uses a different Diffie-Hellman group, the message shall be rejected with a Notify Payload of type INVALID_KE_PAYLOAD.

Key_Exchange Data: A Key_Exchange Payload is constructed by copying one's Diffie-Hellman public value into the Key_Exchange Data portion of the Payload. The length of the Diffie-Hellman public value shall be equal to the length of the prime modulus over which the exponentiation was performed, prepending zero bits to the value if necessary.

6.3.4 Nonce Payload

The Nonce Payload has the format shown in table 84.

Table 84 – Nonce Payload Format

Item	Size (Bytes)
Chaining Header	4
Nonce Data	variable

Chaining Header: See 6.2.3 .

Nonce Data: The Nonce Payload, denoted Ni and Nr for the SA_Initiator's and SA_Responder's nonce respectively, contains random data used to protect against replay attacks and ensure that each peer is actively involved in the SA Management Transaction. The Nonce Data field contains the random data generated by the transmitting entity. The size of a Nonce shall be between 16 and 256 bytes inclusive. Nonce values shall not be reused.

6.4 IKE_Auth Message

6.4.1 Overview

The IKE_Auth Message is used after the IKE_SA_Init exchange has been completed. The IKE_Auth messages authenticate the IKE_SA_Init messages, exchange identities, and establish the first Child_SA. Parts of these messages are encrypted and integrity protected with keys established through the IKE_SA_Init exchange, so the identities are hidden and all fields in all messages are authenticated.

After the IKE_SA_Init exchange has been completed each party may generate SKEYSEED (see 6.8.12), from which all keys are derived for that IKE_SA. All but the headers of all the messages that follow are encrypted and integrity protected. The keys used for the encryption and integrity protection are derived from SKEYSEED and are known as SK_e (encryption) and SK_a (authentication (i.e., integrity protection)). A separate SK_e and SK_a is computed for each direction. In addition to the keys SK_e and SK_a derived from the DH value for protection of the IKE_SA, another quantity SK_d is derived and used for derivation of further keying material for Child_SAs. The notation SK {...} in figure 21 indicates that the IKE Payloads are encrypted and integrity protected using that direction's SK_e and SK_a.

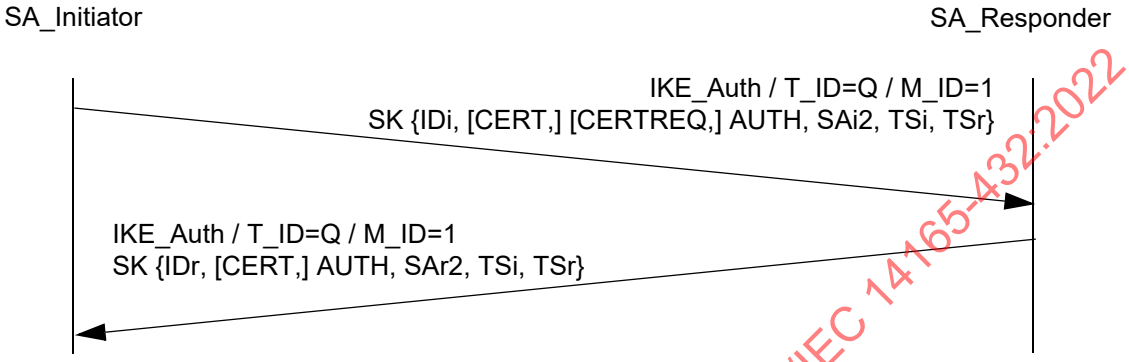

Figure 21 – An IKE_Auth exchange.

The SA_Initiator asserts its identity with the IDi Payload, proves knowledge of the secret corresponding to IDi and integrity protects the contents of the IKE_Auth messages using the Authentication Payload. The SA_Initiator begins negotiation of a Child_SA using the SAi2 Payload. The remaining IKE Payloads, starting with SAi2, are specified in the description of the IKE_Create_Child_SA message (see 6.5).

The SA_Responder specifies its identity in the IDr Payload, authenticates its identity with the Authentication Payload, and completes negotiation of a Child_SA with the additional IKE Payloads specified in the IKE_Create_Child_SA message (see 6.5).

The recipients of an IKE_Auth message shall verify that all signatures and MACs are computed correctly and that the names in the Identification Payloads correspond to the keys used to generate the AUTH Payload. The Message Payload of the IKE_Auth message is shown in table 85.

Table 85 – IKE_Auth Message Payload

Item	Size (Bytes)
IKE_Header Payload	24
Encrypted Payload	variable

IKE_Header Payload: See 6.2.2 .

Encrypted Payload: Contains other IKE Payloads in encrypted form. See 6.4.2 for additional information on the Encrypted Payload. For the IKE_Auth message the IKE Payloads contained in the Encrypted Payload are shown in table 86.

Table 86 – IKE Payloads Contained in the IKE_Auth Message

Item	Size (Bytes)	Reference
Identification Payload	16	6.4.3
Optional Certificate Payload	variable	6.4.6
Optional Certificate Request Payload	variable	6.4.7
Authentication Payload	variable	6.4.4
Security_Association Payload	variable	6.3.2
Traffic Selector Payload - SA_Initiator	variable	6.4.5
Traffic Selector Payload - SA_Responder	variable	6.4.5
Optional Vendor_ID Payload	variable	6.6.4

6.4.2 Encrypted Payload

The Encrypted Payload contains other IKE Payloads in encrypted form. The Encrypted Payload, if present, shall be the last IKE Payload in the message. The Encrypted Payload may be the only IKE Payload in the message. The algorithms for encryption and integrity protection are negotiated during the IKE_SA setup, and the keys are computed as specified in 6.8.11 and 6.8.12.

The encryption and integrity protection algorithms are modelled after the ESP algorithms described in RFC 2104, RFC 4303, and RFC 2451. This standard completely specifies the cryptographic processing of IKE data, but the RFCs describe the design rationale. This standard assumes a block cipher with a fixed block size and an integrity check algorithm that computes a fixed length checksum over a variable size message.

The Encrypted Payload format is shown in table 87.

Table 87 – Encrypted Payload Format

Item	Size (Bytes)
Chaining Header	4
Initialization Vector	variable
Encrypted IKE Payloads	variable
Padding	0 .. 255
Pad Length	1
Integrity Checksum Data	variable

Chaining Header: See 6.2.3 .

Initialization Vector: A randomly selected value whose length is equal to the block length of the underlying encryption algorithm. Recipients shall accept any value for this field. Senders should either pick this value pseudo-randomly and independently for each message or use the final ciphertext block of the previous message sent. Senders shall not use the same value for each message, use a sequence of values with low hamming distance (e.g., a sequence number), or use ciphertext from a received message.

Encrypted IKE Payloads: Other IKE Payloads encrypted with the negotiated cipher. The IKE Payloads contained in the IKE_Auth message are shown in table 86. The IKE Payloads contained in the IKE_Create_Child_SA message are shown in table 99. The IKE Payloads contained in the IKE_Informational message are shown in table 101.

Padding: May contain any value selected by the sender, and shall have a length that makes the combination of the IKE Payloads, the Padding, and the Pad Length field to be a multiple of the encryption block size. This field is encrypted with the negotiated cipher.

Pad Length: The length in bytes of the Padding field. The sender should set the Pad Length to the minimum value that makes the combination of the IKE Payloads, the Padding, and the Pad Length a multiple of the block size, but the recipient shall accept any length that results in proper alignment. This field is encrypted with the negotiated cipher.

Integrity Checksum Data: The cryptographic checksum of the entire message starting with the IKE_Header Payload through the Pad Length. The checksum shall be computed over the encrypted message.

6.4.3 Identification Payload

The Identification Payload allows peers to specify an identity to one another. The Identification Payload format is shown in table 88.

Table 88 – Identification Payload Format

Item	Size (Bytes)
Chaining Header	4
ID_Type	1
Reserved	3
Identification Data	variable

Chaining Header: See 6.2.3 .

ID_Type: Specifies the type of Identification being used. This field shall be set to the value 12 that represents the ID_Type Name_Identifier. The defined values are shown in table 89.

Table 89 – Type Identifiers

ID_Type[a]	Description	Size (Bytes)
12	Name_Identifier[b]	8
201 .. 255	Vendor Specific	
all others	Reserved to IANA	

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

[b] The IEEE Registered Extended Name_Identifier format (i.e., NAA=6h) shall not be used.

Identification Data: Shall be set to the SA_Initiator or SA_Responder Name.

6.4.4 Authentication Payload

The Authentication Payload contains data used for authentication purposes. The Authentication Payload format is shown in table 90.

Table 90 – Authentication Payload Format

Item	Size (Bytes)
Chaining Header	4
Auth_Method	1
Reserved	3
Authentication_Data	variable

Chaining Header: See 6.2.3 .

Auth_Method: Specifies the method of authentication used. The defined values are shown in table 91.

Table 91 – Authentication Methods

Value	Description
1	RSA Digital Signature. Computed as specified in 6.8.13 using an RSA private key over a PKCS#1 padded hash (see RFC 2437).
2	Shared Key Message Integrity Code. Computed as specified in 6.8.13 using the shared key associated with the Identity in the Identification Payload and the negotiated PRF functions.
201 .. 255	Vendor Specific
all others	Reserved

Authentication_Data: See 6.8.13 .

6.4.5 Traffic Selector Payload

The Traffic Selector Payload allows peers to identify packet flows for processing by Fibre Channel security services. Traffic Selector Payloads are used in pairs (TSi, TSr) in order to fully specify a traffic flow to protect. For both SA_Initiator and SA_Responder (TSi and TSr) the Traffic Selector Payload format is shown in table 92.

Table 92 – Traffic Selector Payload Format

Item	Size (Bytes)
Chaining Header	4
Number of TS Definitions	1
Reserved	3
Traffic Selector Definition #1	see table 93
Traffic Selector Definition #2	see table 93
...	
Traffic Selector Definition #k	see table 93

Chaining Header: See 6.2.3 .

Number of TS Definitions: Number of Traffic Selector Definitions being provided.

Traffic Selectors are defined as a list of address identifier ranges and Protocol ranges. A range of address identifiers is a set of two 3-bytes values. The first value is the beginning address identifier (inclusive), and the second value is the ending address identifier (inclusive). All address identifiers falling between the two specified values are considered to be within the list. For a given traffic flow the TSi defines the S_ID address range, and the TSr defines the D_ID address range.

The Protocol ranges are specified as a range of R_CTLs values and Types (see FC-FS-3). The Type attribute has different semantic depending on the protocol to which the Selector is applied. For the ESP_Header protocol that operates at FC-2 layer, the Type range is a range of 00h || TYPE (see FC-FS-3). For the CT_Authentication protocol that operates at CT layer, the Type range is a range of CT GS_Type || GS_Subtype (see FC-GS-6).

NOTE 28 – There is no ambiguity in this definition because in a selector that applies to Common Transport traffic, the TYPE has the implicit value 20h.

A range of R_CTLs or Types is represented by two values. The first value is the beginning R_CTL or Type (inclusive). The second value is the ending R_CTL or Type (inclusive). For a given traffic flow the protocol definitions are the same in TSi and TSr.

Traffic Selector Definition: as shown in table 93.

Table 93 – Traffic Selector Definition

Item	Size (Bytes)
TS Type	1
Reserved	1
Selector Length	2
Starting R_CTL	1
Starting Address	3
Ending R_CTL	1
Ending Address	3
Starting Type	2
Ending Type	2

TS Type: The type of Traffic Selector. This field shall be set to the value 9, that represents a range of address identifiers. TS Types are defined in table 94.

Table 94 – TS Type Identifiers

TS Type[a]	Description
9	FC_Address_Range
241 .. 255	Vendor Specific
all others	Reserved to IANA

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

Selector Length: The length in bytes of the Traffic Selector Substructure. The Selector Length field shall be set to 16.

Starting R_CTL: In both TSi and TSr, the smallest R_CTL value included in the Traffic Selector.

Starting Address: in TSi, the smallest S_ID included in the Traffic Selector. In TSr, the smallest D_ID included in the Traffic Selector.

Ending R_CTL: in both TSi and TSr, the largest R_CTL value included in the Traffic Selector.

NOTE 29 – A value of 00h for the Starting R_CTL, and a value of FFh for the Ending R_CTL means that the R_CTL is not relevant to this Traffic Selector, and the SA includes all the R_CTLs.

Ending Address: in TSi, the largest S_ID included in the Traffic Selector. In TSr, the largest D_ID included in the Traffic Selector.

Starting Type: in both TSi and TSr, the smallest Type included in the Traffic Selector. If the Traffic Selector applies to the ESP_Header protocol, this field is an 00h || TYPE (see FC-FS-3). If the Traffic Selector applies to the CT_Authentication protocol, this field is a GS_Type || GS_Subtype (see FC-GS-6).

Ending Type: in both TSi and TSr, the largest Type included in the Traffic Selector. If the Traffic Selector applies to the ESP_Header protocol, this field is an 00h || TYPE (see FC-FS-3). If the Traffic Selector applies to the CT_Authentication protocol, this field is a GS_Type || GS_Subtype (see FC-GS-6).

NOTE 30 – A value of 0000h for the Starting Type, and a value of FFFFh for the Ending Type means that the Type is not relevant to this Traffic Selector, and the SA includes all the Types.

6.4.6 Certificate Payload

The optional Certificate Payload provides a means to transport digital Certificates or other authentication related information. Certificate payloads should be included in an exchange if Certificates are available to the sender. The Hash and URL formats of the Certificate payloads should be used in case the peer has indicated an ability to retrieve this information from elsewhere using an HTTP_CERT_LOOKUP_SUPPORTED Notify payload. When any Hash and URL format (see RFC 3986) is supported, the HTTP method (see RFC 2616) for Hash and URL lookup shall be supported and other URL methods for lookup should not be supported.

NOTE 31 – The term "Certificate Payload" is somewhat misleading, because not all authentication mechanisms use Certificates and data other than Certificates may be passed in this payload.

The Certificate Payload format is shown in table 95.

Table 95 – Certificate Payload Format

Item	Size (Bytes)
Chaining Header	4
Certificate Encoding	1
Reserved	3
Certificate Data	variable

Chaining Header: See 6.2.3 .

Certificate Encoding: Indicates the Certificate type or Certificate-related information contained in the Certificate Data field. The defined Certificate encodings are shown in table 96. These types shall be supported by each implementation supporting Certificates. Base-64 encoded Certificates used by FCAP shall be converted in DER format (value 4, see table 96) for use with the SA Management protocol.

Table 96 – Certificate Encodings

Value[a]	Certificate Encoding	Certificate Syntax
1	PKCS #7 wrapped X.509 Certificate	See RFC 3852.
4	X.509 Certificate - Signature	Contains a DER encoded X.509 Certificate whose public key is used to validate the sender's AUTH Payload.
7	Certificate Revocation List (CRL)	Contains a DER encoded X.509 Certificate revocation list.
11	Raw RSA Key	Contains a PKCS #1 encoded RSA key.
12	Hash and URL of X.509 Certificate	These encodings replace long data structures with a 20 byte SHA-1 hash of the replaced value followed by a variable length URL that resolves to the DER encoded data structure itself. This improves efficiency when the endpoints have Certificate data cached.
13	Hash and URL of X.509 Bundle (see RFC 5996, clause 3.6)	
201 .. 255	Vendor Specific	
all others	Reserved to IANA	

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

Certificate Data: Contains the actual encoding of Certificate data. The Certificate format is defined in 5.5.3.2.

Implementations shall be capable of being configured to send and accept up to four X.509 Certificates in support of authentication.

Implementations should be capable of being configured to send and accept Raw RSA keys and the two Hash and URL formats. If multiple Certificates are sent, the first Certificate shall contain the public key used to sign the Authentication Payload. The other Certificates may be sent in any order.

6.4.7 Certificate Request Payload

The optional Certificate Request Payload provides a means to request preferred Certificates and may appear in the IKE_SA_Init response and/or the IKE_Auth request. Certificate Request Payloads may be included in an exchange when the sender needs to get the Certificate of the receiver. If multiple CAs are

trusted and the Certificate encoding does not allow a list, then multiple Certificate Request payloads should be transmitted. The Certificate Request Payload format is shown in table 97.

Table 97 – Certificate Request Payload Format

Item	Size (Bytes)
Chaining Header	4
Certificate Encoding	1
Reserved	3
Certification Authority	variable

Chaining Header: See 6.2.3 .

Certificate Encoding: Indicates the Certificate type or Certificate-related information contained in the Certification Authority field. The defined Certificate encodings are shown in table 96.

Certification Authority: Contains an encoding of an acceptable certification authority for the type of Certificate requested.

The Certification Authority field contains an indicator of trusted authorities for this Certificate type. The Certification Authority value is a concatenated list of SHA-1 hashes of the public keys of trusted CAs. Each is encoded as the SHA-1 hash of the Subject Public Key Info element (see section 4.1.2.7 of RFC 5280) from each Trust Anchor Certificate. The twenty-byte hashes are concatenated and included with no other formatting.

NOTE 32 – The term "Certificate Request" is somewhat misleading, in that values other than Certificates are defined in a "Certificate" payload and requests for those values may be present in a Certificate Request Payload. The syntax of the Certificate Request payload in such cases is outside the scope of this standard.

The Certificate Request Payload is processed by inspecting the Certificate Encoding field to determine whether the processor has any Certificates of this type. If so, the Certification Authority field is inspected to determine if the processor has any Certificates that may be validated up to one of the specified certification authorities. This may be a chain of Certificates. If an end-entity Certificate exists that satisfies the criteria specified in the Certificate Request Payload, a Certificate or Certificate chain should be sent back to the Certificate requestor if the recipient of the CERTREQ Payload:

a) is configured to use Certificate authentication;

b) is allowed to send a CERT Payload;

c) has matching CA trust policy governing the current negotiation; and

d) has at least one time-wise and usage appropriate end-entity Certificate chaining to a CA provided in the CERTREQ Payload.

Certificate revocation checking shall be considered during the chaining process used to select a Certificate. Even if two peers are configured to use two different CAs, cross-certification relationships should be supported by appropriate selection logic. This requirement does not prevent an alternate Certificate from being selected by the sender that enables the recipient to successfully validate and trust the Certificate through trust conveyed by cross-certification, CRLs or other out-of-band configured means. The processing of a Certificate Request Payload CA name is a suggestion for a Certificate to select, not a

mandated one. If no Certificates exist then the Certificate Request Payload is ignored. This is not an error condition of the protocol.

6.5 IKE_Create_Child_SA Message

The IKE_Create_Child_SA exchange consists of a single request/response pair. It may be initiated by either end of the IKE_SA after the initial exchanges are completed. All messages following the initial exchanges are protected using the syntax described in 6.4.2 with the cryptographic algorithms and keys negotiated with the IKE_SA_Init exchange.

Given that either endpoint may initiate an IKE_Create_Child_SA exchange, in this subclause the term SA_Initiator refers to the endpoint initiating the IKE_Create_Child_SA exchange.

A Child_SA is created by sending an IKE_Create_Child_SA request. The IKE_Create_Child_SA request may contain a Key_Exchange Payload for an additional Diffie-Hellman algorithm to enable stronger guarantees of forward secrecy for the Child_SA. The keying material for the Child_SA is a function of SK_d computed during the establishment of the IKE_SA, the nonces exchanged during the IKE_Create_Child_SA exchange, and the Diffie-Hellman value if Key_Exchange Payloads are included in the IKE_Create_Child_SA message. The process is represented in figure 22.

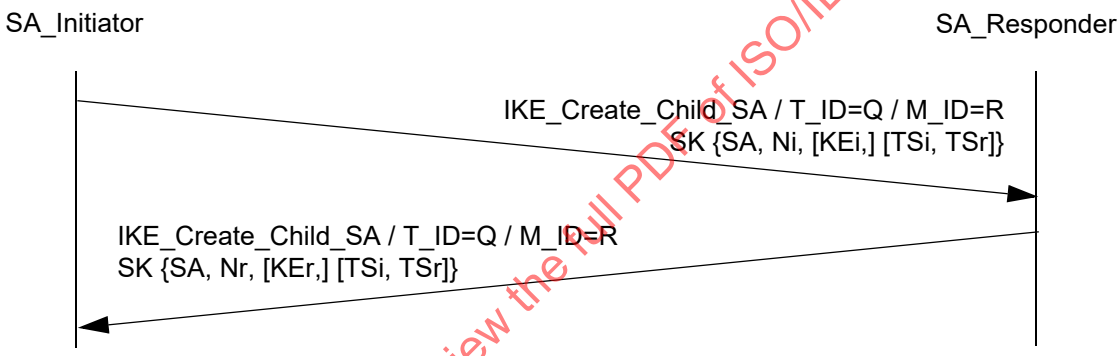

Figure 22 – An IKE_Create_Child_SA exchange.

In the Child_SA created as part of the initial exchange, a second Key_Exchange Payload and nonce shall not be sent. The nonces from the initial exchange are used in computing the keys for the Child_SA.

The SA_Initiator sends one or more SA Proposals in the SA Payload, a nonce in the Ni Payload, an optional Diffie-Hellman value in the KEi Payload, and the proposed Traffic Selectors in the TSi and TSr Payloads. If the SA Proposals include different Diffie-Hellman groups, KEi shall be an element of the group the SA_Initiator expects the SA_Responder to accept. If the SA_Initiator guesses wrong, the IKE_Create_Child_SA exchange fails and the SA_Initiator has to retry with a different KEi. The part of message following the header is encrypted and all the message, including the header, is integrity protected using the cryptographic algorithms negotiated for the IKE_SA.

The SA_Responder replies, using the same Message_ID to respond, with the accepted Proposal in an SA Payload, and a Diffie-Hellman value in the KEr Payload if KEi was included in the request and the selected cryptographic suite includes that group. If the SA_Responder chooses a cryptographic suite with a different group, it shall reject the request. The SA_Initiator should repeat the request with a KEi Payload from the group selected by the SA_Responder.

The Traffic Selectors for traffic to be sent on that SA are specified in the TS Payloads, that may be a subset of what the SA_Initiator of the Child_SA proposed. Traffic Selectors are omitted if this IKE_Create_Child_SA request is being used to change the key of the IKE_SA.

The IKE Payloads that compose an IKE_Create_Child_SA message are shown in table 99. The Message Payload of the IKE_Create_Child_SA message is shown in table 98.

Table 98 – IKE_Create_Child_SA Message Payload

Item	Size (Bytes)
IKE_Header Payload	24
Encrypted Payload	variable

IKE_Header Payload: See 6.2.2 .

Encrypted Payload: Contains other IKE Payloads in encrypted form. See 6.4.2 for additional information on the Encrypted Payload. For the IKE_Create_Child_SA message the IKE Payloads contained in the Encrypted Payload are shown in table 99.

Table 99 – IKE Payloads Contained in the IKE_Create_Child_SA Message

Item	Size (Bytes)	Reference
Security_Association Payload	variable	6.3.2
Nonce Payload	variable	6.3.4
Optional Key_Exchange Payload	variable	6.3.3
Optional SA_Initiator Traffic Selector Payload	variable	6.4.5
Optional SA_Responder Traffic Selector Payload	variable	6.4.5
Optional Vendor_ID Payload	variable	6.6.4

6.6 IKE_Informational Message

6.6.1 Overview

At various points during the operation of an IKE_SA, peers may send control messages to each other regarding errors or notifications of events using an IKE_Informational exchange. IKE_Informational exchanges shall occur only after the initial exchanges and shall be cryptographically protected with the negotiated keys.

Control messages that pertain to an IKE_SA shall be sent under that IKE_SA. Control messages that pertain to Child_SAs shall be sent under the IKE_SA that generated them, or its successor if the IKE_SA was replaced for the purpose of rekeying.

Messages in an IKE_Informational exchange contain zero or more Notification, Delete, and Vendor_ID Payloads. The Recipient of an IKE_Informational exchange request shall send a response, otherwise the Sender shall assume the message was lost and shall retransmit it. The response may be a message with no IKE Payloads. The request message in an IKE_Informational exchange may contain no IKE Payloads.

This is the normal way an endpoint determines if the other endpoint is functional. The process is represented in figure 23.

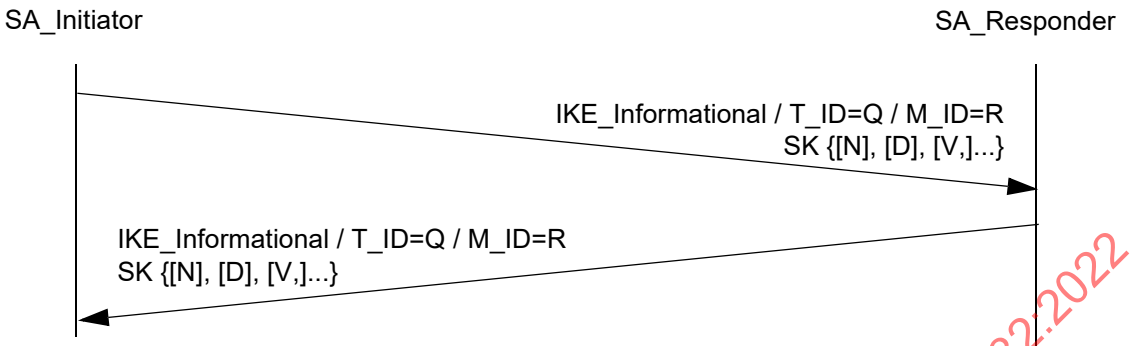

Figure 23 – An IKE_Informational exchange.

NOTE 33 – Unlike IKEv2, the Configuration Payload is not used in this standard.

SAs always occur in pairs, with one SA in each direction. When an SA is closed, both members of the pair shall be closed. Each endpoint shall close its incoming SAs and allow the other endpoint to close the other SA in each pair. To delete an SA, an IKE_Informational exchange with one or more Delete Payloads is sent listing the SPIs, in the format defined for the headers of inbound packets, of the SAs to be deleted. The recipient shall close the designated SAs.

The reply in the IKE_Informational exchange shall contain Delete Payloads for the paired SAs going in the other direction, unless both ends of a set of SAs independently decide to close them. In this case each entity may send a Delete Payload and the two requests may cross in the fabric. If an entity receives a delete request for SAs for which it has already issued a delete request, it shall delete the outgoing SAs while processing the request and the incoming SAs while processing the response. In this case, the responses shall not include Delete Payloads for the deleted SAs, thus avoiding a duplicate deletion that may delete the wrong SA.

An entity should regard half closed SA pairs (i.e., with one SA open and the other one closed) as anomalous and audit their existence if they persist. This standard does not specify timeout periods, so it is up to individual endpoints to decide how long to wait. An entity may refuse to accept incoming data on half closed SA pairs but shall not unilaterally close them and reuse the SPIs. An entity may close the IKE_SA implicitly closing all SAs negotiated under it. An entity may then rebuild the SAs it needs under a new IKE_SA.

The processing of an IKE_Informational exchange is determined by its component IKE Payloads. The Message Payload of the IKE_Informational message is shown in table 100.

Table 100 – IKE_Informational Message Payload

Item	Size (Bytes)
IKE_Header Payload	24
Encrypted Payload	variable

IKE_Header Payload: See 6.2.2 .

Encrypted Payload: Contains other IKE Payloads in encrypted form. See 6.4.2 for additional information on the Encrypted Payload. For the IKE_Informational message the IKE Payloads contained in the Encrypted Payload are shown in table 101.

Table 101 – IKE Payloads Contained in the IKE_Informational Message

Item	Size (Bytes)	Reference
Optional Notify Payload	variable	6.6.2
Optional Delete Payload	variable	6.6.3
Optional Vendor_ID Payload	variable	6.6.4

6.6.2 Notify Payload

The Notify Payload is used to transmit informational data (e.g., error conditions and state transitions) to an IKE peer. A Notify Payload may appear in a response message usually specifying why a request was rejected, in an IKE_Informational message to report an error not in an IKE request, or in any other message to indicate sender capabilities or to modify the meaning of the request. The Notify Payload format is shown in table 102.

Table 102 – Notify Payload Format

Item	Size (Bytes)
Chaining Header	4
Security Protocol_ID	1
SPI Size	1
Notify Message Type	2
SPI	variable
Notification Data	variable

Chaining Header: See 6.2.3 .

Security Protocol_ID: If this notification is related to an existing SA, this field indicates the type of that SA. The defined Security Protocol_IDs are shown in table 72. For notifications that do not relate to an existing SA, this field shall be sent as zero and shall be ignored on receipt.

SPI Size: Length in bytes of the SPI as defined by the Security Protocol_ID, or zero if no SPI is applicable. For a notification concerning the IKE_SA, the SPI Size shall be zero.

SPI: Security Parameter Index indentifying the SA being notified.

Notification Data: IKE_Informational or error data transmitted in addition to the Notify Message Type. Values for this field are dependent upon the value of the Notify Message Type field.

Notify Message Type: Specifies the type of notification message. Notification information may be error messages specifying why it was not possible to establish an SA. It may also be status data that a process managing an SA database wishes to communicate with a peer process. Table 103 and table 104 list the Notification messages and their corresponding values.

Types in the range 0 .. 16 383 are intended for reporting errors. An implementation receiving a Notify Payload with one of these types that it does not recognize in a response shall assume that the corresponding request has failed entirely. Unrecognized error types in a request and status types in a

request or response shall be ignored except that they should be logged. Table 103 lists the Notification messages types used for errors.

Table 103 – Notify Message Types - Errors (part 1 of 2)

Type[a]	Mnemonic	Description
1	UNSUPPORTED_ CRITICAL_PAYLOAD	Sent if the IKE Payload has the critical bit set to one and the IKE Payload type is not recognized. Notification Data contains the one byte IKE Payload type.
4	INVALID_IKE_SPI	Indicates an IKE message was received with an unrecognized destination SPI.
5	INVALID_MAJOR_VERSION	Indicates the recipient is not able to handle the version of IKE specified in the header. The closest version number that the recipient may support is in the reply header.
7	INVALID_SYNTAX	Indicates the received IKE message was invalid because some type, length, or value was out of range or because the request was rejected for policy reasons. To avoid a denial of service attack using forged messages, this status may only be returned for and in an encrypted packet if the MESSAGE_ID and cryptographic checksum were valid. To avoid leaking information to someone probing an entity, this status shall be sent in response to any error not covered by one of the other status types. To aid debugging, more detailed error information should be written to a console or log.
9	INVALID_MESSAGE_ID	Sent when an IKE MESSAGE_ID outside the supported window is received. This Notify shall not be sent in a response and the invalid request shall not be acknowledged. Instead, inform the other side by initiating an IKE_Informational exchange with Notification data containing the four byte invalid MESSAGE_ID. Sending this notification is optional and notifications of this type shall be rate limited.
11	INVALID_SPI	May be sent in an IKE_Informational exchange when an entity receives an ESP_Header or CT_Authentication packet with an invalid SPI. The Notification Data contains the SPI of the invalid packet. If this IKE_Informational Message is sent outside the context of an IKE_SA, it should only be used by the recipient as a hint that something may be wrong because it may be forged.

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

[b] In the range 0 .. 42.

Table 103 – Notify Message Types - Errors (part 2 of 2)

Type[a]	Mnemonic	Description
14	NO_PROPOSAL_CHOSEN	None of the proposed crypto suites was acceptable.
17	INVALID_KE_PAYLOAD	The DH Group number field in the Key_Exchange Payload is not the group number selected by the SA_Responder for this exchange. There are two bytes of data associated with this notification: the accepted DH Group number.
24	AUTHENTICATION_FAILED	Sent in the response to an IKE_Auth message when for some reason the authentication failed. There is no associated data.
34	SINGLE_PAIR_REQUIRED	This error indicates that an IKE_Create_Child_SA request is unacceptable because its sender only accepts Traffic Selectors specifying a single pair of addresses. The requestor should respond by requesting an SA for only the specific traffic the SA_Responder is trying to forward.
35	NO_ADDITIONAL_SAS	This error indicates that an IKE_Create_Child_SA request is unacceptable because the SA_Responder is unable to accept any more Child_SAs on this IKE_SA. Some minimal implementations may only accept a single Child_SA setup in the context of an initial IKE exchange and reject any subsequent attempts to add more.
38	TS_UNACCEPTABLE	Indicates that none of the address/protocol combinations in the supplied Traffic Selectors is acceptable.
39	INVALID_SELECTORS	May be sent in an IKE_Informational exchange when an entity receives an ESP_Header or CT_Authentication packet whose selectors do not match those of the SA on which it was delivered and that caused the packet to be dropped. The Notification Data SPI field is set to match the SPI of the SA.
43	TEMPORARY_FAILURE	See 6.8.3.
44	CHILD_SA_NOT_FOUND	See 6.8.3.
45 .. 8 191		Reserved to IANA - Errors
8 192 .. 16 383		Vendor Specific - Errors
all others[b]		Reserved to IANA

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

[b] In the range 0 .. 42.

Notify Payloads with status types may be added to any message and shall be ignored if not recognized. They are intended to indicate capabilities, and as part of SA negotiation are used to negotiate non-cryptographic parameters. Table 104 lists the Notification messages types used for status.

Table 104 – Notify Message Types - Status

Type[a]	Mnemonic	Description
16 384	INITIAL_CONTACT	This notification specifies that this IKE_SA is the only IKE_SA currently active between the authenticated identities. It may be sent when an IKE_SA is established after a failure, and the recipient may use this information to delete any other IKE_SAs it has to the same authenticated identity without waiting for a timeout. This notification shall not be sent by an entity that may be replicated.
16 386	ADDITIONAL_TS_POSSIBLE	This notification specifies that the sending endpoint narrowed the proposed Traffic Selectors but that other Traffic Selectors may also have been acceptable, though only in a separate SA. There is no data associated with this Notify type. It may only be sent as an additional IKE Payload in a message including accepted Traffic Selectors.
16 392	HTTP_CERT_LOOKUP_ SUPPORTED	This notification may be included in any message that may include a CERTREQ payload and indicates that the sender is capable of looking up Certificates based on an HTTP-based URL.
16 393	REKEY_SA	This notification shall be included in a Create_Child_SA exchange if the purpose of the exchange is to replace an existing ESP_Header or CT_Authentication SA. The SPI field identifies the SA being rekeyed. The notification carries no data. REKEY_SA is needed to specify that this is a rekeying and not a new SA establishment (see 6.8.7).
16 396 .. 40 959		Reserved to IANA - Status
40 960 .. 65 535		Vendor Specific - Status
all others[b]		Reserved to IANA

[a] These values are a subset of those specified by IANA in the "IKEv2 Parameters" registry (see http://www.iana.org/assignments/ikev2-parameters).

[b] In the range 16 385 .. 16 395.

6.6.3 Delete Payload

The Delete Payload contains protocol specific SPIs that the sender has removed from its Security Association database and thus are no longer valid. A delete payload may appear only in Informational messages. It is possible to send multiple SPIs in a Delete Payload, however, each SPI shall be for the same protocol. Mixing of Protocol_IDs shall not be performed in a Delete Payload. Multiple Delete

Payloads may be included in a single IKE_Informational message where each Delete Payload lists SPIs for a different protocol.

Table 105 shows the format of the Delete Payload.

Table 105 – Delete Payload Format

Item	Size (Bytes)
Chaining Header	4
Security Protocol_ID	1
SPI Size	1
Number of SPIs	2
SPI #1	variable
SPI #2	variable
...	
SPI #k	variable

Chaining Header: See 6.2.3 .

Security Protocol_ID: The defined Security Protocol_IDs are shown in table 72.

SPI Size: Length in bytes of the SPI as defined by the Security Protocol_ID. It shall be set to zero for IKE (SPI is in IKE_Header) or set to four for ESP_Header and CT_AUthentication.

Number of SPI: The number of SPIs contained in the Delete Payload.

SPI: Identifies the specific Security Association to delete. The size of this field is determined by the SPI Size field.

Deletion of the IKE_SA is indicated by a Security Protocol_ID of one with no SPIs. Deletion of a Child_SA (e.g., ESP_Header or CT_Authentication), is indicated by the Security Protocol_ID of that Protocol and the SPI set to the SPI value the sending endpoint expects in inbound ESP_Header frames or CT_Authenticated CT_IUs.

6.6.4 Vendor_ID Payload

The Vendor_ID Payload contains a vendor defined constant. The constant is used by vendors to identify and recognize remote instances of their implementations. This mechanism allows a vendor to experiment with new features while maintaining backwards compatibility.

A Vendor_ID Payload may announce that the sender is capable of accepting certain extensions to the protocol, or it may simply identify the implementation as an aid in debugging. A Vendor_ID Payload shall not change the interpretation of any information defined in this standard (i.e., it shall be non-critical). Multiple Vendor_ID Payloads may be sent. An implementation is not required to send Vendor_ID Payloads.

A Vendor_ID Payload may be sent as part of any message. Reception of a known Vendor_ID Payload allows an implementation to make use of Vendor Specific numbers described in this standard (e.g., private

Payloads, private exchanges, private notifications). Unknown Vendor_IDs shall be ignored. Table 106 shows the format of the Vendor_ID Payload.

Table 106 – Vendor_ID Payload Format

Item	Size (Bytes)
Chaining Header	4
Vendor_ID	8
Vendor Specific Information	4

Chaining Header: See 6.2.3 .

Vendor_ID: This field shall contain the vendor's T10 Vendor ID.

6.7 Interaction with the Authentication Protocols

6.7.1 Overview

An SA Management Transaction may occur after the Authentication Initiator and Authentication Responder have successfully completed an Authentication Transaction, and a session key has been established as a result of the Authentication Transaction. This session key is used by the SA Management protocol to construct the Authentication Payload in the IKE_Auth message used during creation of the IKE_SA. The Authentication Initiator declares the capability to concatenate an SA Management Transaction to an Authentication Transaction by including the IKEv2 protocol identifier in the list of usable Authentication Protocols contained in the AUTH_Negotiate message (see 6.7.2). Either Authentication Initiator or Authentication Responder may require the concatenation by setting to one the Concatenation Flag (see 5.2.2 and 5.2.3) in the AUTH messages of the Authentication Transaction (see 6.7.2).

There are situations where an SA Management Transaction may be used to perform both functions of authentication and SA management, replacing the Authentication Protocols defined in clause 5. An SA Management Transaction that performs both functions of authentication and SA management is referred to with the name IKEv2-AUTH protocol. The credentials used in an IKEv2-AUTH transaction are either strong shared secrets or Certificates. The Authentication Initiator declares the capability to use an SA Management Transaction for both authentication and SA management by including the IKEv2-AUTH protocol identifier in the list of usable Authentication Protocols contained in the AUTH_Negotiate message (see 6.7.3).

NOTE 34 – Using the Security Association Management protocol for both Authentication and Security Association management (see 6.7.3) usually has better security properties than concatenating the Security Association Management protocol to another Authentication protocol. However concatenating the Security Association Management protocol to another Authentication protocol (see 6.7.2) leverages the authentication infrastructure associated with that Authentication protocol and this may be easier to manage.

6.7.2 Concatenation of Authentication and SA Management Transactions

When an SA Management Transaction is concatenated to an Authentication Transaction, the session key that has been established as a result of the Authentication Transaction itself shall be used by the SA Management protocol to setup the IKE_SA. In this case, the IKE_Auth message does not carry the optional Certificate and Certificate Request Payloads. The SA Management Transaction shall proceed as specified in clause 6.

The capability to concatenate an SA Management Transaction to an Authentication Transaction shall be indicated by the Authentication Initiator including the IKEv2 protocol identifier in the list of usable Authentication Protocols carries in the AUTH_Negotiate message (see 5.3.2). The IKEv2 protocol has no Authentication Protocol Parameters in the AUTH_Negotiate message.

Once the Authentication Initiator declared the capability to concatenate an SA Management Transaction to an Authentication Transaction by including the IKEv2 protocol identifier in the list of usable Authentication Protocols contained in the AUTH_Negotiate message, either the Authentication Initiator or the Authentication Responder may require to concatenate an SA Management Transaction to the Authentication Transaction by setting the Concatenation Flag to one in the AUTH Flags field of the first AUTH message sent.

The Authentication Initiator requires to concatenate an SA Management Transaction to the Authentication Transaction by setting the Concatenation Flag to one in the AUTH_Negotiate message. If the Authentication Responder does not support AUTH Concatenation, an AUTH_Reject with Reason Code 'Logical Error' and Reason Code Explanation 'AUTH Concatenation not Supported' shall be returned (see 5.3.7). If the Authentication Responder supports AUTH Concatenation, the Concatenation Flag shall be set to one in all subsequent messages belonging to the Authentication Transaction, otherwise an AUTH_Reject with Reason Code 'Authentication Failure' and Reason Code Explanation 'Incorrect Authentication Protocol Message' shall be returned (see 5.3.7).

When the Authentication Initiator does not require to concatenate an SA Management Transaction to the Authentication Transaction, the Authentication Responder may require it by setting the Concatenation Flag to one in the AUTH message sent in reply to the AUTH_Negotiate. Having the Authentication Initiator declared support to concatenate an SA Management Transaction to the Authentication Transaction by including the IKEv2 protocol identifier in the list of usable Authentication Protocols contained in the AUTH_Negotiate message, the Concatenation Flag shall be set to one in all subsequent messages belonging to the Authentication Transaction, otherwise an AUTH_Reject with Reason Code 'Authentication Failure' and Reason Code Explanation 'Incorrect Authentication Protocol Message' shall be returned (see 5.3.7).

When the SA Management Transaction begins after the Authentication Transaction, the Concatenation flag shall be set to zero. The IKE_SA_Init message should be received in AUTH_TOV after the Authentication Transaction completes successfully. The SA Management Transaction shall use the same Transaction Identifier used by the Authentication Transaction.

The SA Management Transaction may be initiated either by the Authentication Initiator or by the Authentication Responder. If both parties send an IKE_SA_Init at the same time, then the entity that receives the request with the lowest of the four nonces contained in the two IKE_SA_Init requests shall send an AUTH_Reject for that request having Reason Code 'Logical Error' and Reason Code Explanation 'Authentication Transaction Already Started', and continue with the other transaction.

Figure 24 shows a typical Authentication Transaction concatenated to an SA Management Transaction. In this example the Authentication Initiator declares the capability to concatenate an SA Management Transaction to an Authentication Transaction by including the IKEv2 protocol identifier in the list of usable Authentication Protocols contained in the AUTH_Negotiate message, but does not require AUTH Concatenation. The Authentication Responder requires to concatenate an SA Management Transaction to

the Authentication Transaction by setting the Concatenation Flag to one when replying to the AUTH_Negotiate message.

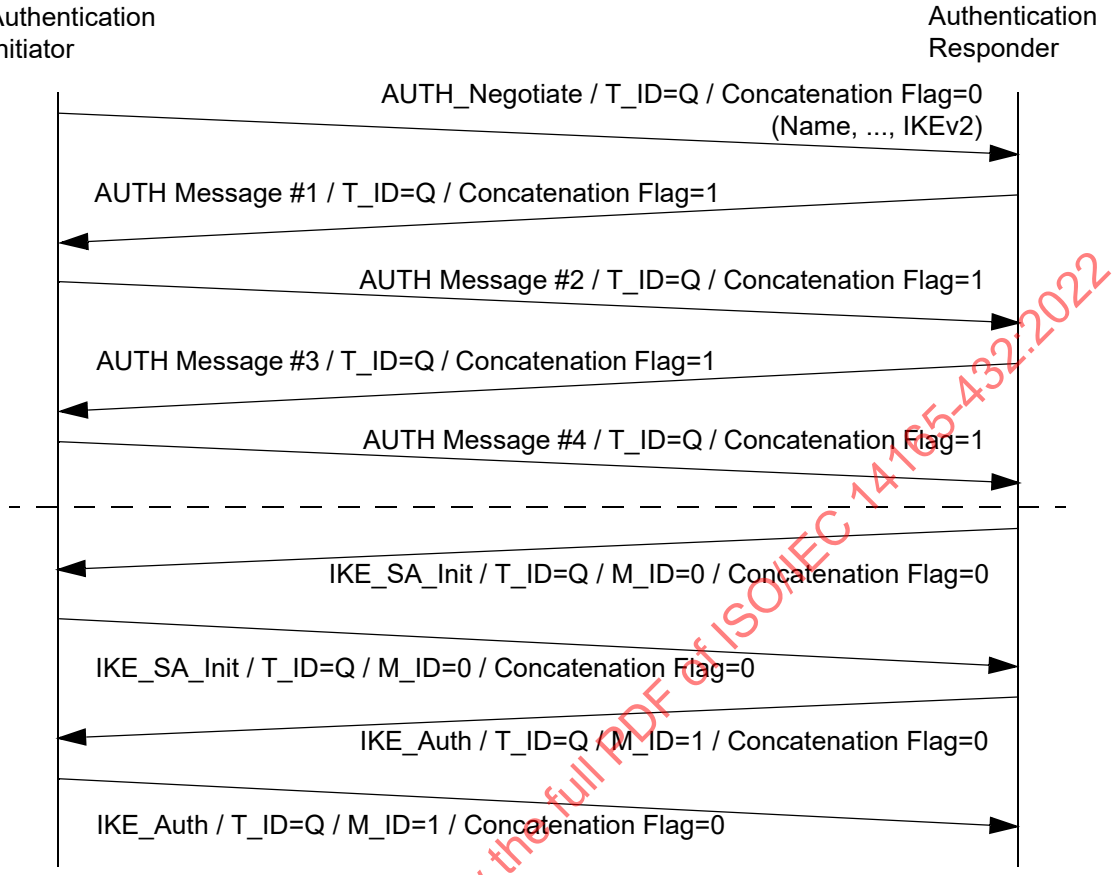

Figure 24 – Concatenation of Authentication and SA Management Transactions.

6.7.3 SA Management Transaction as Authentication Transaction

When an SA Management Transaction is used to perform both functions of authentication and SA management, replacing the Authentication Protocols defined in clause 5, is referred to with the name IKEv2-AUTH protocol. The credentials that shall be used in an IKEv2-AUTH transaction are either strong shared secrets or Certificates. If Certificates are used, the IKE_Auth message shall carry the optional Certificate and Certificate Request Payloads.

The Authentication Initiator shall declare the capability to use an SA Management Transaction for both authentication and SA management by including the IKEv2-AUTH protocol identifier in the list of usable Authentication Protocols contained in the AUTH_Negotiate message (see 5.3.2). The IKEv2-AUTH protocol has no Authentication Protocol Parameters in the AUTH_Negotiate message.

The Authentication Responder that selects IKEv2-AUTH as the Authentication Protocol shall reply with the same sequence of messages used during an SA Management Transaction. The Authentication Responder, that becomes the SA_Initiator, shall then send an IKE_SA_Init message to the Authentication

Initiator, that becomes the SA_Responder. The SA Management Transaction shall proceed as specified in clause 6. Figure 25 shows a typical IKEv2-AUTH transaction.

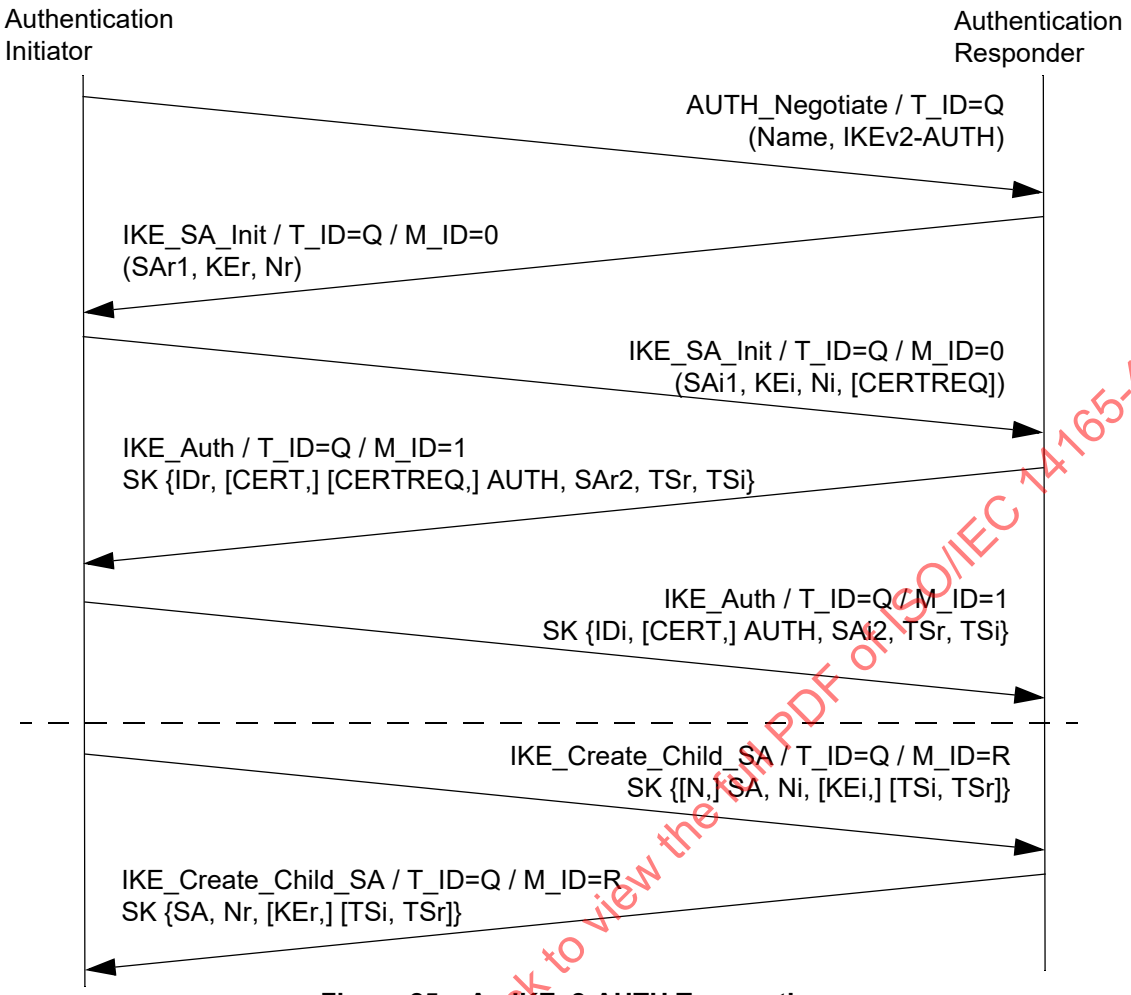

Figure 25 – An IKEv2-AUTH Transaction.

6.8 IKEv2 Protocol Details

6.8.1 Use of Retransmission Timers

See RFC 5996 (i.e., IKEv2), clause 2.1. A robust responder should perform the IKE SA lookup using the whole message, its hash, or the Ni payload in all cases, not only in the single NAT scenario specified in RFC 5996, clause 2.1 (that does not apply to Fibre Channel).

NOTE 35 – Unlike IKEv2, multiple outstanding requests, as defined in IKEv2 with the SET_WINDOW_SIZE mechanism, are not used in this standard.

6.8.2 Use of Sequence Numbers for Message_IDs

See RFC 5996 (i.e., IKEv2), clause 2.2.

6.8.3 Overlapping Requests

An SA Management protocol implementation shall process requests strictly in order and/or wait for a response to one request before issuing another.

NOTE 36 – Unlike IKEv2, multiple outstanding requests, as defined in IKEv2 with the SET_WINDOW_SIZE mechanism, are not used in this standard.

After an IKE_SA is set up, either end may initiate a request. An IKE endpoint shall be prepared to accept and process a request while it has a request outstanding in order to avoid a deadlock in this situation. An IKE endpoint shall wait for a response to each of its messages before sending a subsequent message. Possible requests collisions should be handled as specified in RFC 5996 (i.e., IKEv2), clause 2.25.

6.8.4 State Synchronization and Connection Timeouts

See RFC 5996 (i.e., IKEv2), clause 2.4.

6.8.5 Cookies and Anti-Clogging Protection

The term cookie identifies a pseudo-random unique value used in network protocols to counter attacks that try to exhaust CPU resources. This feature is often referred to as anti-clogging protection.

IKEv2 may include a cookie in the IKE_SA_Init exchange to provide anti-clogging protection. The SA Management protocol does not use cookies but instead relies on the services provided by the underlying FC stack to provide anti-clogging service (see 5.4.8). When state and CPU resources are scarce, the SA_Responder of an IKE_SA_Init message shall reject the message with an LS_RJT or an SW_RJT having a Reason Code of ‘Logical Busy’ (see 5.8.3, 5.9.3 and 5.10.3). This avoids the creation of any state on the SA_Responder, and allows the SA_Initiator to retry the protocol later.

6.8.6 Cryptographic Algorithms Negotiation

The SA Payload indicates a Proposal for a single protocol to protect (e.g., IKE, ESP_Header, or CT_Authentication) for the Security Association as well as cryptographic algorithms (i.e., Transforms) associated with each protocol.

NOTE 37 – Unlike IKEv2, Proposals for more than one protocol are not used in this standard. Each SA Payload carries a single Proposal for IKE, ESP_Header or CT_Authentication.

See also RFC 5996 (i.e., IKEv2), clause 2.7.

6.8.7 Rekeying

See RFC 5996 (i.e., IKEv2), clause 2.8.

6.8.8 Traffic Selector Negotiation

When a frame or a CT_IU received from the FC-4 matches a selector that specifies to protect it in the SADB (Security Associations Database, see 4.7), the frame or CT_IU shall be protected with ESP_Header or CT_Authentication encapsulation according to the type of frame or CT_IU and the policy specified in the matching selector. When no SA exists it is the task of the SA Management protocol to create it. Maintenance of a system's SADB is outside the scope of this standard, though some implementations may update their SADB in connection with the running of the SA Management protocol.

See also RFC 5996 (i.e., IKEv2), clause 2.9.

6.8.9 Nonces

See RFC 5996 (i.e., IKEv2), clause 2.10.

6.8.10 Reuse of Diffie-Hellman Exponential

IKE generates keying material using an ephemeral Diffie-Hellman exchange in order to gain the property of perfect forward secrecy. This means that once an SA pair is closed and its corresponding keys are forgotten, even someone who has recorded all of the data from the SA pair and gets access to all of the long term keys of the two endpoints is unable to reconstruct the keys used to protect the conversation without doing a brute force search of the session key space.

To achieve perfect forward secrecy, each endpoint shall include in the actions taken when an SA pair is closed the discarding of:

a) the keys used by the SA pair (e.g., the secrets used in the Diffie-Hellman calculation); and

b) any information that could be used to recompute those keys (e.g., the state of the random number generator (see Annex C).

Since the computing of Diffie-Hellman exponentials is computationally expensive, an endpoint may find it advantageous to reuse those exponentials for multiple SA pair setups. There are several reasonable strategies for doing this. An endpoint may select a new exponential only periodically though this may result in less-than-perfect forward secrecy if some SA pair lasts for less than the lifetime of the exponential. Alternatively it may keep track of which exponential was used for each SA pair and delete the information associated with the exponential only when some corresponding SA pair was closed. This allows the exponential to be reused without losing perfect forward secrecy at the cost of maintaining more state.

Decisions as to whether and when to reuse Diffie-Hellman exponentials is a private decision in the sense that it does not affect interoperability. An implementation that reuses exponentials may select to remember the exponential used by the other endpoint on past exchanges and if one is reused to avoid the second half of the calculation.

6.8.11 Generating Keying Material

See RFC 5996 (i.e., IKEv2), clause 2.13.

6.8.12 Generating Keying Material for the IKE_SA

See RFC 5996 (i.e., IKEv2), clause 2.14.

6.8.13 Authentication of the IKE_SA

The SA Management protocol may be used to create Security Associations after one of the Authentication Protocols described in clause 5 has been successfully executed, and a session key has been generated, as described in 5.4.6, 5.5.6, and 5.6.6. The session key shall be used as the value of the Shared Secret used to compute the Authentication_Data value in the Authentication Payload (see RFC 5996), and the Auth_Method in the Authentication Payload shall be set to Shared Key Message Integrity Code (i.e., 2).

Alternatively the SA Management protocol may replace the Authentication Protocols described in clause 5 when the interested parties have strong shared secrets or Certificates (see 6.7.3).

See also RFC 5996 (i.e., IKEv2), clause 2.15.

NOTE 38 – Unlike IKEv2, Extended Authentication Protocol (EAP) methods are not used in this standard.

6.8.14 Generating Keying Material for Child_SAs

See RFC 5996 (i.e., IKEv2), clause 2.17.

6.8.15 Rekeying IKE_SAs using the IKE_Create_Child_SA exchange

See RFC 5996 (i.e., IKEv2), clause 2.18.

6.8.16 IKE_Informational Messages outside of an IKE_SA

If a frame arrives with an unrecognized SPI, it may be because the receiving entity has lost state or because of some other system malfunction or attack. If the receiving entity has an active IKE_SA to the source address of a frame or CT_IU, it may send a notification over that IKE_SA. If it does not, it may send an IKE_Informational message without cryptographic protection to the source address to alert it of a possible problem.

6.8.17 Error Handling

See RFC 5996 (i.e., IKEv2), clause 2.21, and subclauses 2.21.1, 2.21.2, 2.21.3.

NOTE 39 – Unlike IKEv2, the Configuration Payload and the Extended Authentication Protocol (EAP) methods are not used in this standard.

6.8.18 Conformance Requirements

IKEv2 is a security protocol, and one of its major functions is to allow only authorized parties to successfully complete establishment of SAs. A particular implementation may be configured with any of a number of restrictions concerning algorithms and trusted authorities that prevents universal interoperability.

To assure interoperability, all implementations shall be capable of parsing all payload types, if only to skip over them, and to ignore payload types that it does not support unless the critical bit is set in the payload header. If the critical bit is set in an unsupported payload header, all implementations shall reject the messages containing those payloads.

Every implementation shall be capable of doing four messages IKE_SA_Init and IKE_Auth exchanges establishing two SAs (i.e., one for IKE, one for ESP_Header or CT_Authentication). Implementations may be initiate-only or respond-only if appropriate for their platform. Every implementation shall be capable of responding to an IKE_Informational exchange, but a minimal implementation may respond to any IKE_Informational message with an empty IKE_Informational reply. A minimal implementation may support the IKE_Create_Child_SA exchange only in so far as to recognize requests and reject them with a Notify payload of type NO_ADDITIONAL_SAS. A minimal implementation need not be able to initiate IKE_Create_Child_SA or IKE_Informational exchanges. When an SA expires, based on locally configured values of either lifetime or bytes passed, an implementation may either try to renew it with an IKE_Create_Child_SA exchange or it may delete (i.e., close) the old SA and create a new one. If the responder rejects the IKE_Create_Child_SA request with a NO_ADDITIONAL_SAS notification, the implementation shall be capable of instead closing the old SA and creating a new one.

For an implementation that supports Certificates, it shall be possible to configure it to accept X.509v3 Certificates containing RSA keys of size 1 024 or 2 048 bits, where the identifier passed as

FC_Name_Identifier shall be reflected by the Subject field or the Subject Alternative Name Extension of the Certificate (see 5.5.3.2).

6.8.19 Rekeying IKE_SAs when Refreshing Authentication

As described in 6.7.2, an SA Management Transaction may be concatenated to an Authentication Transaction. In this case the session key established as a result of the Authentication Transaction is used by the SA Management protocol to authenticate the IKE_SA as a pre-shared key. The IKE_SA may be re-keyed at any time based on this pre-shared key.

An Authentication Transaction may be repeated at any time by either peer in order to re-authenticate (see 5.11).

If the Authentication Transaction is successful, the two peers shall update the entry of the Security Association Database with the new session key resulting from the Authentication Transaction. Then the SA_Initiator shall rekey the corresponding IKE_SA as described in 6.8.15. This procedure causes the new IKE_SA to inherit all of the original IKE_SA's Child_SAs.

If the FC Authentication transaction is not successful, an Authentication failure shall occur, as described in clause 5, and appropriate actions shall be taken to clean up the Security Association Database from any status that is related to the peer that has failed to re-authenticate.

7 Fabric Policies

7.1 Policies Definition

7.1.1 Overview

This clause defines how to manage Fabric policies and extends the Zoning model defined in FC-GS-6 and FC-SW-5 to be used in a secure environment.

NOTE 40 – Examples of Fabric policy implementations prior to this standard are provided in Annex G.

Policies are expressed as a set of related data structures, called Policy Objects. Some Policy Objects are Fabric-wide (i.e., present on each Switch of a Fabric), while others are per Switch (i.e., present only on specific Switches). The main Fabric-wide data structures are the Switch Membership List Object and the Node Membership List Object, listing respectively Switches and Nodes allowed to be part of a Fabric. The per Switch data structures are the Switch Connectivity Objects, listing topology restrictions associated to specific Switches. These and other policy data structures and their relationships are represented in figure 26.

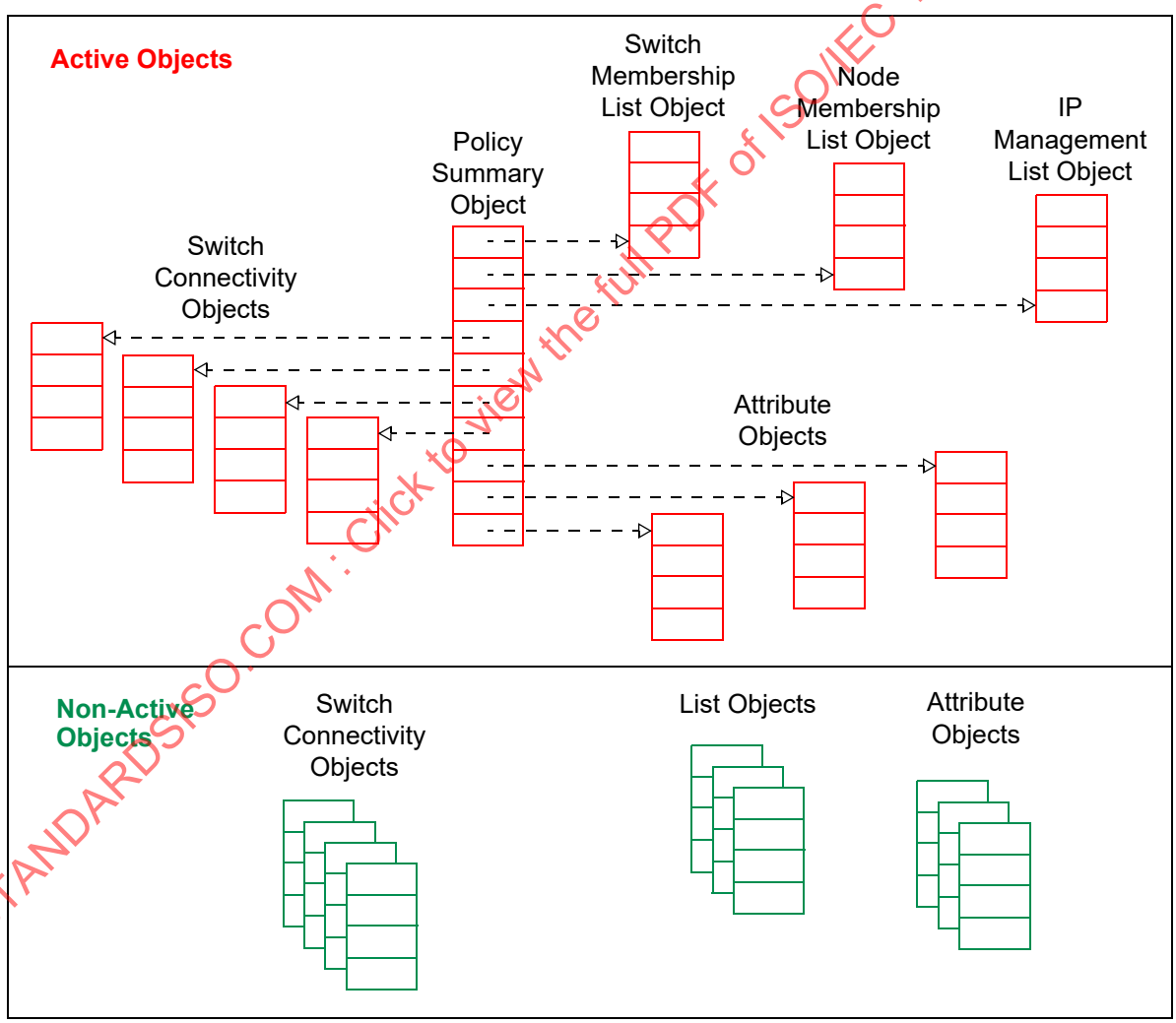

Figure 26 – Policy Data Structures

The set of Policy data structures enforced by a Fabric contains the Policy Objects shown in table 107.

Table 107 – Policy Objects

Object Identifier	Description	Cardinality	Reference
0000 0001h	Policy Summary Object	One per Fabric	7.1.3
0000 0002h	Switch Membership List Object	One per Fabric	7.1.4
0000 0003h	Node Membership List Object	One per Fabric	7.1.5
0000 0004h	Switch Connectivity Object	A set per Fabric	7.1.6
0000 0005h	IP Management List Object	One per Fabric	7.1.7
0000 0006h	Attribute Object	A set per Fabric	7.1.8

The Policy Summary Object is an ordered list of pointers to other Policy Objects, one pointer per each other active Policy Object. Each pointer in a Policy Summary Object is paired with a cryptographic hash of the referenced Policy Object.

The Switch Membership List Object is a Fabric-wide Policy Object that defines which Switches are allowed to be part of a Fabric.

The Node Membership List Object is a Fabric-wide Policy Object that defines which Nodes are allowed to be connected to a Fabric.

The IP Management List Object is a Fabric-wide Policy Object that describes which IP hosts are allowed to manage a Fabric.

If in a Fabric there is no need to express topology restrictions, then a Policy configuration may be composed of the Policy Summary Object, the Switch Membership List Object, the Node Membership List Object and the IP Management List Object. Topology restrictions are expressed as a set of Switch Connectivity Objects.

A Switch Connectivity Object is a per Switch Policy Object that describes topology restrictions associated to a specific Switch.

The Attribute Objects are Fabric-wide Policy Objects that define optional attributes to be associated with Switches or Nodes. They allow the extension of this policy model by defining new attributes as required.

Each of the Policy Objects is identified by a Name, and may be summarized by the value resulting from the computation of a cryptographic hash on that Policy Object. Multiple instances of Policies Objects, some active and some non-active, may exist in a Fabric at a certain time. Each instance is identified by a different Name. Only two instances of a Switch Connectivity Object may exist, one active and one non-active, because a Switch Connectivity Object uses the Switch Node_Name as its Name. The list of Names and hashes of the enforced Policy Objects is encoded in the Policy Summary Object. A non-active Policy Summary Object is not defined by this standard.

This policy model supports three kinds of Switches in a Fabric:

a) Autonomous Switches, that maintain the Fabric-wide Policy Objects, their own Switch Connectivity Object, and a full copy of the FC-SP Zoning Database;

b) Client Switches, that maintain the Fabric-wide Policy Objects, their own Switch Connectivity Object, and a subset of the FC-SP Active Zone Set; and

c) Server Switches, that maintain the Fabric-wide Policy Objects, all the Switch Connectivity Objects, and a full copy of the FC-SP Zoning Database.

Considering a Fabric composed of n Switches and m Nodes, the Switch Membership List Object and the Node Membership List Object have respectively a potential complexity of O(n) and O(m). Instead the set of Switch Connectivity Objects has a potential complexity of $O(n^2)$ to describe the Switch to Switch connections, and a potential complexity of O(n times m) to describe the Switch to Node connections. For these reasons, in order to keep manageable the complexity of the Fabric Policy data structures, this policy model:

a) Allows for the distribution of the Switch Connectivity Objects only where needed (i.e., they are not Fabric-wide information);

b) Allows the association of attributes to Nodes in the Node Membership List Object or to Switches in the Switch Membership List Object; and

c) Does not allow the association of attributes to specific connections (i.e., Switch-to-Switch or Switch-to-Node pairs).

7.1.2 Names used to define Policies

The names used to define policies shall have the format specified in 4.9 with the Name Tag, Name Length, and Name Value content shown in table 108.

Table 108 – Names used to define Policies

Name Tag	Name Length (Bytes)	Name Value content
0002h	8	Node_Name[a]
8002h	8	Restricted Node_Name[a]
0003h	8	Port_Name[a]
8003h	8	Restricted Port_Name[a]
0004h	8	Wildcard
8004h	8	Negated Wildcard
0005h	1 to 64	Alphanumeric Name
0006h	32	IPv6 Address Range
0007h	8	IPv4 Address Range

[a] The IEEE Registered Extended Name_Identifier format (i.e., NAA=6h) shall not be used.

Node_Name Value: a Name_Identifier, as defined by FC-FS-3.

Restricted Node_Name Value: a Name_Identifier, as defined by FC-FS-3.

Port_Name Value: a Name_Identifier, as defined by FC-FS-3.

Restricted Port_Name Value: a Name_Identifier, as defined by FC-FS-3.

The IEEE Registered Extended Name_Identifier format (i.e., NAA=6h) shall not be used.

Wildcard Value: the value 0000 0000 0000 0000h.

Restricted Wildcard Value: the value 0000 0000 0000 0000h.

NOTE 41 – Wildcard Names are used to collectively identify "all others" (e.g., all other members of a Policy Object), not to identify a specific Policy Object.

Alphanumeric Name Value: The Alphanumeric Name Value is right-padded with 00h to be word aligned, while the length field (see table 2) provides the length of the non-padded name. Alphanumeric Names have the following properties:

a) The non-padded name shall be between 1 and 64 characters in length;

b) All characters shall be Printable ASCII characters;

c) The first character of a given name shall be a letter. A letter is defined as either an upper case (i.e., A .. Z) character or a lower case (i.e., a .. z) character; and

d) Any character other than the first character shall be a lower case character (i.e., a .. z), an upper case character (i.e., A .. Z), a number (i.e., 0 .. 9), or one of the following four symbols: dollar-sign ($), dash (-), caret (^), and underscore (_).

I**Pv6 Address Range Value:** two IPv6 addresses in network byte order, the numerically smallest first and the numerically largest second.

I**Pv4 Address Range Value:** two IPv4 addresses in network byte order, the numerically smallest first and the numerically largest second.

7.1.3 Policy Summary Object 7.1.3.1

Format

The Policy Summary Object (see table 109) provides a compact representation of the policies enforced by a Fabric. There is no provision for a non-active instance of the Policy Summary Object.

Table 109 – Policy Summary Object Format

Item	Size (Bytes)
Object Identifier (0000 0001h)	4
Object Flags	4
Object Name[a]	variable
Number of Policy Objects	4
Policy Object Identifier #1	4
Policy Object Pointer #1[b]	variable
Policy Object Hash #1	variable
Policy Object Identifier #2	4
Policy Object Pointer #2[b]	variable
Policy Object Hash #2	variable
...	
Policy Object Identifier #k	4
Policy Object Pointer #k[b]	variable
Policy Object Hash #k	variable

[a] The Name shall be an Alphanumeric Name
[b] The Name shall be an Alphanumeric Name or a Node_Name

Object Identifier: the value 0000 0001h identifies a Policy Summary Object.

Object Flags: the defined Object flags for the Policy Summary Object are shown in table 110.

Table 110 – Object Flags

Bit	Description
31 .. 0	Reserved

Object Name: an Alphanumeric Name identifying the Policy Set Object.

Policy Object Identifier fields: the identifier of a Policy Object (see table 107).

Policy Object Pointer fields: the name (see 7.1.2) of a Policy Object of the type identified by the corresponding Policy Object Identifier.

Policy Object Hash fields: the hash of Policy Object identified by the corresponding Policy Object Pointer. The format of each Hash field is shown in table 111.

Table 111 – Hash Field Format

Item	Size (Bytes)
Hash Tag	4
Hash Length	4
Hash Value	variable

Hash Tag: identifies the cryptographic hash function used to summarize a Policy Object in the Policy Summary Object. Valid Hash types are shown in table 112.

Hash Length: indicates the total length in bytes of the Hash Value. Length values are shown in table 112.

Hash Value: contains the result of the hash computation over a Policy Object.

Table 112 – Hash Formats

Hash Type[a]	Hash Tag	Hash Length (Bytes)
SHA-1[b]	0000 0001h	20
SHA-256[b]	0000 0002h	32

[a] See FIPS PUB 180-4

[b] Support for SHA-1 is mandatory, support for SHA-256 is optional

Each Policy Object has ordering requirements that result in a unique representation of that Object. The one-way hash value of a Policy Object is computed over the sequence of bytes that forms the Object without adding any padding. The computation shall be performed as specified by the standard that defines the Hash function used.

7.1.3.2 Ordering Requirements

In a Policy Summary Object, the Policy Object Identifier, Policy Object pointer, and Policy Object Hash shall be sorted in ascending order using the Policy Object Identifier as the numeric primary key and the Name Value of the Policy Object pointer as either the numeric secondary key for Switch Connectivity Objects, or the alphanumeric secondary key for Objects with any other type of Policy Object Identifier.

7.1.4 Switch Membership List Object

7.1.4.1 Format

The Switch Membership List Object is a Fabric-wide Policy Object that defines which Switches are allowed to be part of a Fabric. The membership may be restricted or unrestricted based on name types (see 7.1.2). A restricted membership means that the Switch is not allowed to be part of the Fabric unless allowed by a specific Switch Connectivity Object. An unrestricted membership means that the Switch is allowed to be part of the Fabric unless disallowed by a specific Switch Connectivity Object. Each Switch is identified by its Node_Name and is associated to a set of basic attributes describing how it shall behave in the Fabric and how it may be directly accessed for management purposes. A Switch entry may point to an Attribute Object to specify additional Switch characteristics. A wildcard entry or negated wildcard entry permits specifying a way to deal with Switches not included in the Switch Membership list, and provides a way to indicate a default Attribute for Switch entries not pointing to a specific Attribute Object.

Default Basic Switch Attributes for Switches not listed in a Switch Membership List Object are specified by the Basic Switch Attributes field of a wildcard or negated wildcard entry in the Switch Membership List Object.

The Switch Membership List Object format is shown in table 113.

Table 113 – Switch Membership List Object Format

Item	Size (Bytes)
Object Identifier (0000 0002h)	4
Object Flags	4
Object Name[a]	variable
Administered Fabric_Name	8
Number of Entries	4
Switch Entry #1	variable
Switch Entry #2	variable
...	
Switch Entry #n	variable

[a] The Name shall be an Alphanumeric Name

Object Identifier: the value 0000 0002h identifies a Switch Membership List Object.

Object Flags: the defined Object flags are shown in table 114.

Table 114 – Object Flags

Bit	Description
31	**Active flag:** This is a status flag, set by the Fabric. If set to one, then the Policy Object is Active (i.e., enforced by the Fabric). If set to zero, then the Policy Object is non-active (i.e., not enforced by the Fabric). The Active flag is used when Policy Objects are returned by the Fabric. When Policy Objects are sent to the Fabric this flag shall be set to zero. The Active flag shall be set to zero before computing the hash of a Policy Object.
30 .. 0	Reserved[a]

[a] In order to generate the hash of a Policy Object, bits 31 .. 16 shall be set to zero before the computation, while bits 15 .. 0 shall be left unchanged.

Object Name: an Alphanumeric Name identifying the Switch Membership List Object.

Administered Fabric_Name: an administratively set Fabric_Name. This parameter is meaningful only when Static Domain_IDs are used in a Fabric (see FC-SW-5). Static Domain_IDs are administratively enabled by setting to one the Static Domain_ID bit in the Switch Flags (see table 117) of each Switch Entry in the Switch Membership List. If Static Domain_IDs are not used the Fabric_Name is dynamically established (see FC-SW-5) and this parameter shall be set to 0000 0000 0000 0000h.

Number of Entries: shall be set to the number of Switch Entries contained in the Switch Membership List Object.

Switch Entry: the format of the Switch Entry field is shown in table 115.

Table 115 – Switch Entry Field Format

Item	Size (Bytes)
Name[a]	12
Basic Switch Attributes	8
Attribute Object Pointer (optional)	variable
[a] The Name shall be a Node_Name or a Wildcard	

Name: an identifier for a Switch. The format of the Name field is described in 7.1.2.

Attribute Object Pointer: an optional Alphanumeric Name pointing to an Attribute Object. The format of the Attribute Object is described in 7.1.8.

Basic Switch Attributes: the format of the Basic Switch Attributes field is shown in table 116

Table 116 – Basic Switch Attributes Format

Item	Size (Bytes)
Switch Flags	4
Reserved	3
Domain_ID	1

Switch Flags: the defined Switch flags are shown in table 117.

Table 117 – Switch Flags

Bit	Description
31	Attribute Object Pointer Present
30 .. 21	Reserved
20	Static Domain_ID
19	Insistent Domain_ID
18	Serial Ports Access
17	Physical Panel Access
16	Manager Role
15 .. 12	Reserved
11 .. 8	Policy Data Role
7 .. 2	Reserved
1 .. 0	Authentication Behavior

Attribute Object Pointer Present: if this bit is set to one then the Attribute Object Pointer field is present in the Switch Entry. If this bit is set to zero then no Attribute Object Pointer field is present in the Switch Entry.

Static Domain_ID: if this bit is set to one the Switch shall use the Static Domain_IDs behavior (see FC-SW-5). In this case the Domain_ID field shall be set to the administratively assigned Domain_ID. When

this bit is set to one, the Insistent Domain_ID bit shall be set to zero. If this bit is set to zero the Switch shall not use the Static Domain_IDs behavior.

The Static Domain_ID bit shall be set for all Switches in the Switch Membership List or for none of them, otherwise the Fabric partitions. When the Static Domain_ID bit is set for all Switches in the Switch Membership List, the Administered Fabric_Name field (see table 113) shall be set to the administratively set Fabric_Name.

Insistent Domain_ID: if this bit is set to one the Switch shall use the Insistent Domain_ID behavior. In this case the Domain_ID field shall be set to the administratively assigned Domain_ID. When this bit is set to one, the Static Domain_ID bit shall be set to zero. If this bit is set to zero the Switch shall not use the Insistent Domain_ID behavior.

A Switch using the Insistent Domain_ID behavior shall join a Fabric if its administratively assigned Domain_ID is granted during the Fabric Initialization procedure. During the Address Distribution phase the Switch shall request to the Principal Switch its administratively assigned Domain_ID in the RDI SW_ILS, and shall isolate the involved Switch Port if the Principal Switch does not grant the requested Domain_ID (see FC-SW-5).

Serial Ports Access: if this bit is set to one, then the Switch shall allow management through serial ports. If this bit is set to zero the Switch shall not allow management through serial ports.

Physical Panel Access: if this bit is set to one, then the Switch shall allow management through the physical panel. If this bit is set to zero the Switch shall not allow management through the physical panel.

Manager Role: if this bit is set to one, then the Switch is able to change the Fabric Policy configuration as defined in 7.3. If this bit is set to zero, then the Switch is not able to change the Fabric Policy configuration. Each Switch shall enforce this bit on receiving any of the EACA, ESFC, EUFC, ACA, SFC, or UFC SW_ILSs. If the originating Switch is a Manager Switch (i.e., it has this bit set to one in its Switch Entry in the currently enforced Switch Membership List), then the SW_ILS may be accepted, otherwise it shall be rejected with a Reason Code ‘Unable to Perform Command Request’ and a Reason Code explanation ‘Not Authorized’ (see FC-SW-5). If FC-SP Zoning (see 7.6) is used, then this bit applies also to changes of the Zoning configuration.

NOTE 42 – Given that the Switch that initiates a Policy Change is also the Switch connected to a management application via the interface defined in FC-GS-6, this bit allows to specify through which Switches the Fabric Policies may be managed. Any number of Switches in a Fabric may have the Manager role.

Policy Data Role: this field defines the role of the Switch in terms of which Policy Data it maintains (see table 118). Each value specifies a different behavior.

Table 118 – Policy Data Role

Value	Description
0000b	The Switch shall operate as a Client Switch. A Client Switch maintains its own Switch Connectivity Object and all Fabric-wide List Objects.[a] If FC-SP Zoning is used, a Client Switch maintains only the subset of the Active Zone Set that it requires to enforce the current Fabric Zoning configuration, and shall implement the client part of the Client-Server protocol (see 7.6.5).
0001b	The Switch shall operate as an Autonomous Switch. An Autonomous Switch maintains its own Switch Connectivity Object and all Fabric-wide List Objects.[a] If FC-SP Zoning is used, an Autonomous Switch maintains a complete copy of the Fabric Zoning Database.
0011b	The Switch shall operate as a Server Switch. A Server Switch maintains all Fabric-wide List Objects and the Switch Connectivity Objects of each Switch in the Fabric. If FC-SP Zoning is used, a Server Switch maintains a complete copy of the Fabric Zoning Database, and shall implement the server part of the Client-Server protocol (see 7.6.5).
all others	Reserved

[a] There is no difference between an Autonomous Switch and a Client Switch if FC-SP Zoning is not used.

A Fabric deploying any Client Switch requires at least one Server Switch to operate properly. If no Server Switches are reachable, a Client Switch is not able to update its FC-SP Zoning configuration when new Nodes are connected to the Fabric.

Authentication Behavior: this field defines the Authentication behavior of the Switch (see table 119).

Table 119 – Authentication Behavior

Value	Description
00b	Connections between this Switch and the neighbor Switches may be authenticated. The rejection of an AUTH_Negotiate message shall be considered as an authentication failure by this Switch.
01b	Any connection between this Switch and a neighbor Switch shall be authenticated. This Switch should initiate the Authentication Transaction. The rejection of an AUTH_Negotiate message shall be considered as an authentication failure by this Switch.
10b	Connections between this Switch and the neighbor Switches may be authenticated. The rejection of an AUTH_Negotiate message shall not be considered as an authentication failure by this Switch.
11b	Any connection between this Switch and a neighbor Switch shall be authenticated. This Switch should initiate the Authentication Transaction. The rejection of an AUTH_Negotiate message shall not be considered as an authentication failure by this Switch.[a]

[a] This behavior allows the gradual deployment of authentication in a Fabric where some Switches support authentication and some do not.

7.1.4.2 Ordering Requirements

In a Switch Membership List Object, the Switch entries shall be sorted in ascending order using the Name Tag as the numeric primary key and the Name Value as the numeric secondary key.

7.1.5 Node Membership List Object

7.1.5.1 Format

The Node Membership List Object is a Fabric-wide Policy Object that defines which Nodes are allowed to be connected to a Fabric. The membership may be restricted or unrestricted based on name types (see 7.1.2). A restricted membership means that the Node is not allowed to be connected to the Fabric unless allowed by a specific Switch Connectivity Object. An unrestricted membership means that the Node is allowed to be connected to the Fabric unless disallowed by a specific Switch Connectivity Object. Each Node is identified by its Node_Name or by one or more of its Port_Names, and is associated to a set of basic attributes describing how it shall behave in the Fabric and how it may access the Fabric for management purposes. The identification via Port_Name permits specifying different behaviors for different ports of the same Node. A Node entry may point to an Attribute Object to specify additional Node characteristics. A wildcard entry or negated wildcard entry permits specifying a way to deal with Nodes not included in the Node Membership list, and provides a way to indicate a default Attribute for Node entries not pointing to a specific Attribute Object.

Default Basic Node Attributes for Nodes not listed in a Node Membership List Object are specified by the Basic Node Attributes field of a wildcard or negated wildcard entry in the Node Membership List Object.

The Node Membership List Object format is shown in table 120.

Table 120 – Node Membership List Object Format

Item	Size (Bytes)
Object Identifier (0000 0003h)	4
Object Flags	4
Object Name[a]	variable
Number of Entries	4
Node Entry #1	variable
Node Entry #2	variable
...	
Node Entry #n	variable

[a] The Name shall be an Alphanumeric Name

Object Identifier: the value 0000 0003h identifies a Node Membership List Object.

Object Flags: the defined Object flags are shown in table 114.

Object Name: an Alphanumeric Name identifying the Node Membership List Object.

Number of Entries: shall be set to the number of Node Entries contained in the Node Membership List Object.

Node Entry: the format of the Node Entry field is shown in table 121.

Table 121 – Node Entry Field Format

Item	Size (Bytes)
Name[a]	12
Basic Node Attributes	variable
Attribute Object Pointer (optional)	variable
[a] The Name shall be a Node_Name or Port_Name or a Wildcard	

Name: an identifier for a Node. The format of the Name field is described in 7.1.2.

Attribute Object Pointer: an optional Alphanumeric Name pointing to an Attribute Object. The format of the Attribute Object is described in 7.1.8.

Basic Node Attributes: the format of the Basic Node Attributes field is shown in table 122.

Table 122 – Basic Node Attribute Format

Item	Size (Bytes)
Node Flags	4
Common Transport Access Specifier	variable

Node Flags: The defined Node flags are shown in table 123.

Table 123 – Node Flags

Bit	Description
31	Attribute Object Pointer Present
30 .. 10	Reserved
9	SCSI Enclosure Services (SES) Access
8	Common Transport Access
7 .. 1	Reserved
0	Authentication Required

Attribute Object Pointer Present: if this bit is set to one, then the Attribute Object Pointer field is present in the Node Entry. If this bit is set to zero, then no Attribute Object Pointer field is present in the Node Entry.

SCSI Enclosure Services (SES) Access: if this bit is set to one, then this Node is allowed to control any Switch through SCSI Enclosure Services. If this bit is set to zero, then this Node is not allowed to control any Switch through SCSI Enclosure Services. This bit is ignored if a Switch does not support SCSI Enclosure Services (see SES).

Authentication Required: if this bit is set to one, then this Node shall authenticate itself to any Switch to which it is connected. If this bit is set to zero, then this Node is not required to authenticate itself to any Switch. Each Switch shall enforce this bit by requiring authentication of any Node with this bit set to one in its Node Entry in the currently enforced Node Membership List. Authentication is required by setting to one the Security Bit in the FLOGI LS_ACC (see FC-LS-2).

Common Transport Access: if this bit is set to one, then the access by this Node to Generic Services via Common Transport is limited by the content of the Common Transport Access Specifier field. If this bit is set to zero, then the access by this Node to Generic Services is not limited and the Common Transport Access Specifier field is not present. Each Switch shall enforce this bit each time a Node accesses a Generic Service over the Switch using Common Transport by checking the correspondent Node Entry in the currently enforced Node Membership List.

Common Transport Access Specifier: a variable length field having the format shown in table 124.

Table 124 – Common Transport Access Specifier Format

Item	Size (Bytes)
CT Access Descriptor #1	4
CT Access Descriptor #2	4
...	
CT Access Descriptor #k	4

The list of CT Access Descriptors determines if a Node is allowed to access a Generic Service or Sub-Server. The format of a CT Access Descriptor is shown in table 125.

Table 125 – CT Access Descriptor Format

Item	Size (Bytes)
CT Access Flags	1
Reserved	1
GS_Type	1
GS_Subtype	1

GS_Type: indicates the GS_Type of the Generic Service subject to access control (e.g., Management Service).

GS_Subtype: indicates the GS_Subtype of the specific GS Server subject to access control (e.g., Fabric Zone Server or Security Policy Server).

CT Access Flags: The defined CT Access flags are shown in table 126.

Table 126 – CT Access Flags

Bit	Description
7	No More Words
6	Allow/Deny
5	GS_Type Wildcard
4	GS_Subtype Wildcard
3	Read Only
2 .. 0	Reserved

No More Words: if this bit is set to zero, then an additional CT Access Descriptor is present in the Common Transport Access Specifier. If this bit is set to one, then this is the last CT Access Descriptor of the Common Transport Access Specifier.

Allow/Deny: if this bit is set to zero, then access to the specified Service and Server is allowed to this Node. If this bit is set to one, then access to the specified Service and Server is not allowed to this Node.

GS_Type Wildcard: if this bit is set to zero, then access restrictions apply to the Generic Service selected by the GS_Type field. If this bit is set to one, then the GS_Type field is ignored, and access restrictions apply to any Generic Service. In this case the GS_Subtype field is ignored.

GS_Subtype Wildcard: if this bit is set to zero, then access restrictions apply to the Fabric Server selected by the GS_Subtype field. If this bit is set to one, then the GS_Subtype field is ignored, and access restrictions apply to any Fabric Server.

Read Only: if this bit is set to zero, then the granted access to the specified Service and Server is for both reading and writing. If this bit is set to one, then the granted access to the specified Service and Server is for reading only (i.e., only GET Requests may be accepted).

When a Node attempts to access a Generic Service through a Switch, the Switch shall check the GS_Type and GS_Subtype with which the access is attempted against the Common Transport Access Specifier associated with that Node, and grant or deny access accordingly. Access is implicitly denied to any GS_Type and GS_Subtype not explicitly listed in a Common Transport Access Specifier (see 7.2.3).

Table 127 shows some common access policies.

Table 127 – Examples of Common Transport Access Specifiers

Item	Example 1[a]	Example 2[b]	Example 3[c,d]
CT Access Flags	01010000b	10010000b	10110000b
Reserved	00h	00h	00h
GS_Type	FAh	FCh	00h
GS_Subtype	00h	00h	00h
CT Access Flags	10110000b		
Reserved	00h		
GS_Type	00h		
GS_Subtype	00h		

[a] The Common Transport Access Specifier to allow a Node to access any Generic Service except the Management Service.

[b] The Common Transport Access Specifier to allow a Node to access only the Directory Service.

[c] The Common Transport Access Specifier to allow a Node to access any Generic Service.

[d] The minimum allowed Common Transport Access Specifier.

7.1.5.2 Ordering Requirements

In a Node Membership List Object, the Node entries shall be sorted in ascending order using the Name Tag as the numeric primary key and the Name Value as the numeric secondary key. The CT Access

Descriptors of the Common Transport Access Specifier shall be sorted in ascending order, using each CT Access Descriptor as numeric primary key.

7.1.6 Switch Connectivity Object

7.1.6.1 Format

A Switch Connectivity Object is a per Switch Policy Object that describes topology restrictions associated to a specific Switch. A Switch Connectivity Object defines to which other Switches or Nodes the considered Switch may be connected at the Node level and/or at the Port level. The identification of specific ports, Nodes, or Switches is accomplished by using Port_Names or Node_Names. Each connection may be restricted or unrestricted, based on name types (see 7.1.2). A restricted connection is a connection that is not allowed. An unrestricted connection is a connection that is allowed. Wildcard Names permits specifying a way to deal with entities not explicitly named.

The Switch Connectivity Object format is shown in table 128.

Table 128 – Switch Connectivity Object Format

Item	Size (Bytes)
Object Identifier (0000 0004h)	4
Object Flags	4
Switch Node_Name	12
Number of Allowed Switches	4
Name[a] #1	12
Name[a] #2	12
...	
Name[a] #k	12
Number of Allowed Nodes	4
Name[a] #1	12
Name[a] #2	12
...	
Name[a] #h	12
Number of Port Connectivity Entries	4
Port Connectivity Entry #1	variable
Port Connectivity Entry #2	variable
...	
Port Connectivity Entry #j	variable
[a] The Name shall be a Node_Name or Port_Name or a Wildcard	

Object Identifier: the value 0000 0004h identifies a Switch Connectivity Object.

Object Flags: the defined Object flags are shown in table 114.

Switch Node_Name: the Node_Name of the Switch to which the Connectivity Object is referred to, formatted as specified in 7.1.2.

Number of Allowed Switches: shall be set to the number of Names describing Switch-to-Switch connectivity. These Names shall be sorted in ascending order using the Name Tag as the numeric primary key, and the Name Value as the numeric secondary key.

Number of Allowed Nodes: shall be set to the number of Names describing Switch-to-Node connectivity. These Names shall be sorted in ascending order using the Name Tag as the numeric primary key, and the Name Value as the numeric secondary key.

Port Connectivity Entry: the Port Connectivity Entries shall be sorted in ascending order using the Switch Port_Name to which they refer to as the numeric key. The Port Connectivity Entry format is shown in table 129.

Table 129 – Port Connectivity Entry Format

Item	Size (Bytes)
Switch Port_Name	8
Number of Allowed Switches	4
Name[a] #1	12
Name[a] #2	12
...	
Name[a] #g	12
Number of Allowed Nodes	4
Name[a] #1	12
Name[a] #2	12
...	
Name[a] #f	12

[a] The Name shall be a Node_Name or Port_Name or a Wildcard

Switch Port_Name: the Port_Name of the Switch to which the Port Connectivity Entry is referred to.

Number of Allowed Switches: shall be set to the number of Names describing Switch-to-Switch connectivity. These Names shall be sorted in ascending order using the Name Tag as the numeric primary key, and the Name Value as the numeric secondary key.

Number of Allowed Nodes: shall be set to the number of Names describing Switch-to-Node connectivity. These Names shall be sorted in ascending order using the Name Tag as the numeric primary key, and the Name Value as the numeric secondary key.

7.1.6.2 Ordering Requirements

In a Switch Connectivity Object, the Names shall be sorted in ascending order using the Name Tag as the numeric primary key and the Name Value as the numeric secondary key.

7.1.7 IP Management List Object

7.1.7.1 Format

The IP Management List Object is a Fabric-wide Policy Object that describes which IP hosts are allowed to manage a Fabric. Each IP host is identified by its IP address and is associated to a set of basic attributes

describing the IP based protocols through which the IP hosts may access the Fabric for management purposes. An IP host entry may point to an Attribute Object to specify additional IP host characteristics.

The IP Management List Object format is shown in table 130.

Table 130 – IP Management List Object Format

Item	Size (Bytes)
Object Identifier (0000 0005h)	4
Object Flags	4
Object Name[a]	variable
Number of IP Management Entries	4
IP Management Entry #1	variable
IP Management Entry #2	variable
...	
IP Management Entry #k	variable
[a] The Name shall be an Alphanumeric Name	

Object Identifier: the value 0000 0005h identifies an IP Management List Object.

Object Flags: the defined Object flags are shown in table 114.

Object Name: an Alphanumeric Name identifying the IP Management List Object.

Number of IP Management Entries: shall be set to the number of IP Management Entries contained in the IP Management List Object.

IP Management Entry: the format of the IP Management Entry is shown in table 131.

Table 131 – IP Management Entry Format

Item	Size (Bytes)
Name[a]	36 or 12
Basic IP Management Attributes	variable
Attribute Object Pointer (optional)	variable
[a] The Name shall be an IPv6 Address Range or an IPv4 Address Range	

Name: an identifier for one or more IP hosts. The format of the Name field is described in 7.1.2.

Attribute Object Pointer: an optional Alphanumeric Name pointing to an Attribute Object. The format of the Attribute Object is described in 7.1.8.

Basic IP Management Attributes: the format of the Basic IP Management Attributes field is shown in table 132.

Table 132 – Basic IP Management Attributes Format

Item	Size (Bytes)
IP Management Flags	4
Well Known Protocols Access Specifier	variable

IP Management Flags: The defined IP Management flags are shown in table 133.

Table 133 – IP Management Flags

Bit	Description
31	Attribute Object Pointer Present
30 .. 0	Reserved

Attribute Object Pointer Present: if this bit is set to one, then the Attribute Object Pointer field is present in the IP Management Entry. If this bit is set to zero, then no Attribute Object Pointer field is present in the IP Management Entry.

Well Known Protocols Access Specifier: a variable length field having the format shown in table 134.

Table 134 – Well Known Protocols Access Specifier Format

Item	Size (Bytes)
WKP Access Descriptor #1	4
WKP Access Descriptor #2	4
...	
WKP Access Descriptor #k	4

The list of WKP Access Descriptors determines if an IP host may or may not access the Fabric using the specified protocols for management purposes. The format of a WKP Access Descriptor is shown in table 135.

Table 135 – WKP Access Descriptor Format

Item	Size (Bytes)
WKP Access Flags	1
Well Known Protocol Number	1
Destination Port Number	2

The Well Known Protocol Numbers are defined in http://www.iana.org/assignments/protocol-numbers. The Destination Port Numbers are defined in http://www.iana.org/assignments/port-numbers, but protocols may use other ports.

The well known protocol for TCP is 7, for UDP is 17. The Destination Port Number parameter is significative when those protocols are used. ICMP (protocol number 1) should be allowed in all cases.

NOTE 43 – Particular care should be used when managing the Well Known Protocol Numbers, otherwise a management application may lock itself outside of the Fabric.

WKP Access Flags: The defined WKP Access flags are shown in table 136.

Table 136 – WKP Access Flags

Bit	Description
7	No More Words
6	Allow/Deny
5	Well Known Protocol Number Wildcard
4	Destination Port Number Wildcard
3	Read Only
2 .. 0	Reserved

No More Words: if this bit is set to zero, then an additional WKP Access Descriptor is present in the Well Known Protocol Access Specifier. If this bit is set to one, then this is the last WKP Access Descriptor of the Well Known Protocol Access Specifier.

Allow/Deny: if this bit is set to zero, then management access using the specified protocol is allowed to this IP management entity. If this bit is set to one, then management access using the specified protocol is not allowed to this IP management entity.

Well Known Protocol Number Wildcard: if this bit is set to zero, then access restrictions apply to the IP Protocol selected by the Protocol Number field. If this bit is set to one, then the Protocol Number field is ignored, and access restrictions apply to any IP Protocol.

Destination Port Number Wildcard: if this bit is set to zero, then access restrictions apply to the TCP or UDP port selected by the Port Number field. If this bit is set to one, then the Port Number field is ignored, and access restrictions apply to any TCP or UDP port.

Read Only: if this bit is set to zero, then the granted management access using the specified protocol is for both reading and writing. If this bit is set to one, then the granted management access using the specified protocol is for reading only.

When an IP entity attempts to access a Switch, the Switch shall check the application protocol with which the access is attempted against the Well Known Protocols Access Specifier associated with that IP entity, and grant or deny access accordingly. Access is implicitly denied to any Protocol Number and Port Number combination not listed in a Well Known Protocols Access Specifier (see 7.2.5).

Table 137 shows some common access policies.

Table 137 – Examples of Well Known Protocols Access Specifiers

Item	Example 1[a]	Example 2[b]	Example 3[c]	Example 4[d]
WKP Access Flags	10000000b	10001000b	00000000b	10110000b
Well Known Protocol Number	11h	11h	06h	00h
Destination Port Number	00A1h	00A1h	0080h	0000h
WKP Access Flags			00000000b	
Well Known Protocol Number			06h	
Destination Port Number			00A1h	
WKP Access Flags			10000000b	
Well Known Protocol Number			11h	
Destination Port Number			00A1h	

[a] The Well Known Protocols Access Specifier to allow an IP management entity to manage the Fabric only through SNMP over UDP.

[b] The Well Known Protocols Access Specifier to allow an IP management entity to manage the Fabric only through read only SNMP over UDP.

[c] The Well Known Protocols Access Specifier to allow an IP management entity to manage the Fabric only through SNMP over UDP or TCP, and HTTP over TCP.

[d] The Well Known Protocols Access Specifier to allow an IP management entity to manage the Fabric with no restrictions.

7.1.7.2 Ordering Requirements

In an IP Management List Object, the IP Management entries shall be sorted in ascending order using the IPv6 Address Range or the IPv4 Address Range as the numeric primary key. IPv6 Address Ranges shall precede IPv4 Address Ranges. The WKP Access Descriptors of the Well Known Protocols Access Specifier shall be sorted in ascending order, using each WKP Access Descriptor as the numeric primary key.

7.1.8 Attribute Object

7.1.8.1 Format

The Attribute Objects are Fabric-wide Policy Objects that define optional attributes to be associated with Switches or Nodes. They allow to extend this policy model by defining new attributes as required.

A default attribute for a Switch or a Node is defined by specifying an Attribute Object Pointer in a wildcard or negated wildcard entry (see 7.1.2) respectively in the Switch Membership List Object or in the Node Membership List Object.

The format of the Attribute Object is shown in table 138.

Table 138 – Attribute Object Format

Item	Size (Bytes)
Object Identifier (0000 0006h)	4
Object Flags	4
Attribute Object Name[a]	variable
Number of Attribute Entries	4
Attribute Entry #1	variable
Attribute Entry #2	variable
...	
Attribute Entry #k	variable

[a] The Name shall be an Alphanumeric Name

Object Identifier: the value 0000 0006h identifies an Attribute Object.

Object Flags: the defined Object flags are shown in table 114.

Object Name: an Alphanumeric Name identifying the Attribute Object.

Number of Attribute Entries: shall be set to the number of Attribute Entries contained in the Attribute Object.

Attribute Entry: the format of the Attribute Entry is shown in table 139.

Table 139 – Attribute Entry Format

Item	Size (Bytes)
Attribute Tag	4
Attribute Length	4
Attribute Value	variable

Attribute Tag: identifies the type of the attribute. Attribute Tags are described in table 140.

Attribute Length: indicates the total length in bytes of the Attribute Value.

Attribute Value: contains the attribute value.

Table 140 – Attribute Formats

Attribute	Attribute Tag	Attribute Value
Authentication Parameters	0000 0001h	The AUTH_Negotiate Message Payload (see table 10)

When used in a Switch entry of a Switch Membership List Object, the Authentication Parameters Attribute specifies what the Switch shall send in an AUTH_Negotiate message and what the Switch may accept in a received AUTH_Negotiate message.

When used in a Node entry of a Node Membership List Object, the Authentication Parameters Attribute specifies what a Switch may accept in an AUTH_Negotiate message from that Node.

To determine what to accept in a received AUTH_Negotiate message, each instance of an Authentication Protocol Identifier and Parameters from the received AUTH_Negotiate message is compared with each instance in this attribute's value. If none match, then the AUTH_Negotiate message is rejected; if one matches, that one is accepted; if multiple match, one of the ones in the AUTH_Negotiate message is accepted, selected in accord with applicable policy. This policy may require that the preference expressed by the Authentication Initiator in the AUTH_Negotiate message be honored.

7.1.8.2 Ordering Requirements

In an Attribute Object, the Attribute Entries shall be sorted in ascending order using the Attribute Tag as the numeric primary key.

7.2 Policies Enforcement

7.2.1 Overview

Policy enforcement (i.e., authorization) occurs whenever:

a) a Switch-to-Switch connection is attempted;

b) a Switch-to-Node connection is attempted;

c) a management application attempts to access the fabric in-band; or

d) a management application attempts to access the fabric out-of-band.

For each of these cases the appropriate Policy Objects need to be checked to determine whether the requested connection or access is to be allowed or denied. In addition, policy enforcement occurs whenever a new policy is activated. In this case, Authentication Transactions may be triggered, connections previously allowed may be disallowed, or connections previously disallowed may be allowed. The policy enforcement rules for each of these situations are defined in 7.2 using the notation shown in table 141.

Table 141 – Notation for Policy Enforcement

Symbols	Description
$N(\alpha)$, $[N(\alpha)]$	Respectively Node_Name and Restricted Node_Name of Switch α
$P_n(\alpha)$, $[P_n(\alpha)]$	Respectively Port_Name and Restricted Port_Name of the Nth port of Switch α
N(A), [N(A)]	Respectively Node_Name and Restricted Node_Name of Node A
$P_k(A)$, $[P_k(A)]$	Respectively Port_Name and Restricted Port_Name of the Kth port of Node A

7.2.2 Switch-to-Switch Connections

Whenever Switch β attempts to connect to Switch α, both α's and β's Switch Connectivity Objects plus the Switch Membership List Object shall be checked to determine whether the connection is to be allowed or denied. The same checks shall be performed when a new policy is activated.

For all checks described in this subclause, a NULL Policy data structure is equivalent to a non-NULL Policy data structure containing only a Wildcard entry.

When Switch β attempts to connect via its Kth port (i.e., Node_Name = N(β), Port_Name = P_k(β)) to the Nth port of Switch α (i.e., Node_Name = N(α), Port_Name = P_n(α)), Switch α shall perform the following authorization checks:

1) Check the α's Switch Connectivity Object for port-connection restrictions (see 7.1.6). If there is no Port Connectivity Entry for P_n(α), then go to step 2. If there is a Port Connectivity Entry for P_n(α), then:

   A) If the list of Allowed Switches of the P_n(α) Port Connectivity Entry contains N(β) or P_k(β) the connection is allowed, or if it contains [N(α)] or [P_k(β)] the connection is denied. No further authorization checks are performed; or

   B) If the list of Allowed Switches of the P_n(α) Port Connectivity Entry does not contain N(β), P_k(β), [N(β)], [P_k(β)], but contains a Wildcard the connection is allowed, or if it contains a Negated Wildcard the connection is denied. No further authorization checks are performed;

2) Check the α's Switch Connectivity Object for Switch-connection restrictions (see 7.1.6). If there are no Switch-connection restrictions, then go to step 3. If there are Switch-connection restriction, then:

   A) If the list of Allowed Switches in the Switch Connectivity Object contains N(β) or P_k(β) the connection is allowed, or if it contains [N(β)] or [P_k(β)] the connection is denied. No further authorization checks are performed; or

   B) If the list of Allowed Switches in the Switch Connectivity Object does not contain N(β), P_k(β), [N(β)], [P_k(β)], but contains a Wildcard the connection is allowed, or if it contains a Negated Wildcard the connection is denied. No further authorization checks are performed; and

3) Check the Switch Membership List Object for Fabric-wide Switch restrictions. If there are no Fabric-wide Switch restrictions the connection is allowed and no further authorization checks are performed. If there are Fabric-wide Switch restrictions, then:

   A) If the Switch Membership List Object contains N(β) the connection is allowed, or if it contains [N(β)] the connection is denied. No further authorization checks are performed; or

   B) If the Switch Membership List Object does not contain N(β), [N(β)], but contains a Wildcard the connection is allowed, or if it contains a Negated Wildcard the connection is denied. No further authorization checks are performed.

Switch β shall perform the complementary checks.

7.2.3 Switch-to-Node Connections

Whenever Node A attempts to connect to Switch α, α's Switch Connectivity Object and the Node Membership List Object shall be checked to determine whether the connection is to be allowed or denied. The same checks shall be performed also when a new policy is activated. The enforcement rules for the Switch side of an attempted connection are defined. The definition of the enforcement rules for the Node side of an attempted connection is outside the scope of this standard.

For all checks described in this subclause, a NULL Policy data structure is equivalent to a non-NULL Policy data structure containing only a Wildcard entry.

When Node A attempts to connect via its Kth port (i.e., Node_Name = N(A), Port_Name = P_k(A)) to the Nth port of Switch α (i.e., Node_Name = N(α), Port_Name = P_n(α)), Switch α shall perform the following authorization checks:

1) Check the α's Switch Connectivity Object for port-connection restrictions (see 7.1.6). If there is no Port Connectivity Entry for $P_n(\alpha)$, then go to step 2. If there is a Port Connectivity Entry for $P_n(\alpha)$, then:

   A) If the list of Allowed Nodes of the $P_n(\alpha)$ Port Connectivity Entry contains N(A) or P_k(A) the connection is allowed, or if it contains [N(A)] or [P_k(A)] the connection is denied. No further authorization checks are performed; or

   B) If the list of Allowed Nodes of the $P_n(\alpha)$ Port Connectivity Entry does not contain N(A), P_k(A), [N(A)], [P_k(A)], but contains a Wildcard the connection is allowed, or if it contains a Negated Wildcard the connection is denied. No further authorization checks are performed;

2) Check the α's Switch Connectivity Object for Node-connection restrictions (see 7.1.6). If there are no Node-connection restrictions, then go to step 3. If there are Node-connection restriction, then:

   A) If the list of Allowed Nodes in the Switch Connectivity Object contains N(A) or P_k(A) the connection is allowed, or if it contains [N(A)] or [P_k(A)] the connection is denied. No further authorization checks are performed; or

   B) If the list of Allowed Nodes in the Switch Connectivity Object does not contain N(A), P_k(A), [N(A)], [P_k(A)], but contains a Wildcard the connection is allowed, or if it contains a Negated Wildcard the connection is denied. No further authorization checks are performed; and

3) Check the Node Membership List Object for Fabric-wide Node restrictions. If there are no Fabric-wide Node restrictions the connection is allowed and no further authorization checks are performed. If there are Fabric-wide Node restrictions, then:

   A) If the Node Membership List Object contains N(A) or P_k(A) the connection is allowed, or if it contains [N(A)] or [P_k(A)] the connection is denied. No further authorization checks are performed; or

   B) If the Node Membership List Object does not contain N(A), P_k(A), [N(A)], [P_k(A)], but contains a Wildcard the connection is allowed, or if it contains a Negated Wildcard the connection is denied. No further authorization checks are performed.

7.2.4 In-Band Management Access to a Switch

Whenever a management application attempts to access a Switch α via an in-band access method, the Node Membership List Object shall be checked to determine whether the access is to be allowed or denied. The same checks shall be performed when a new policy is activated. In-band management access control applies to Common Transport (CT) access and SCSI Enclosure Services (see SES) access. The enforcement rules for the Switch side of an attempted access are defined. The definition of the enforcement rules for the Node side of an attempted access is outside the scope of this standard.

Considering a management application running on Node A attempting to access Switch α via the Node's Kth port (Node_Name = N(A), Port_Name = P_k(A)) with a certain access method, Switch α shall perform the following authorization checks:

1) If there is no Node Membership List Object then there are no access restrictions and the access is allowed. No authorization checks are performed;

2) If there is a Node Membership List Object, then:

A) If the Node Membership List Object contains an entry for N(A), P_k(A), [N(A)] or [P_k(A)], the correspondent Node Flags shall be checked. If the Node Flag associated with CT is set to one then go to step 3. If the Node Flag associated with SES is set to one then the access is allowed and no further authorization checks are performed. If the Node Flag associated with the considered access method is set to zero the access is denied and no further authorization checks are performed; or

B) If the Node Membership List Object does not contain an entry for N(A), P_k(A), [N(A)], [P_k(A)], but contains a Wildcard or a Negated Wildcard entry, the correspondent Node Flags shall be checked. If the Node Flag associated with CT is set to one then go to step 3. If the Node Flag associated with SES is set to one then the access is allowed and no further authorization checks are performed. If the Node Flag associated with the considered access method is set to zero the access is denied and no further authorization checks are performed; and

3) Check the selected Common Transport Access Specifier as follows:

A) If both the GS_Type and GS_Subtype fields in the received CT_IU match the GS_Type and GS_Subtype fields in a CT Access Descriptor, then the Allow/Deny bit in the matched CT Access Descriptor indicates if the access is allowed or denied. If the Allow/Deny bit in the matched CT Access Descriptor is set to one the access is allowed. If the Allow/Deny bit in the matched CT Access Descriptor is set to zero the access is denied. No further authorization checks are performed;

B) If the GS_Type field in the received CT_IU matches the GS_Type field in a CT Access Descriptor having the GS_Subtype Wildcard bit set to one, then the Allow/Deny bit in the matched CT Access Descriptor indicates if the access is allowed or denied. If the Allow/Deny bit in the matched CT Access Descriptor is set to one the access is allowed. If the Allow/Deny bit in the matched CT Access Descriptor is set to zero the access is denied. No further authorization checks are performed;

C) If there is a CT Access Descriptor having both the GS_Type and GS_Subtype Wildcard bits set to one, then the Allow/Deny bit in this CT Access Descriptor indicates if the access is allowed or denied. If the Allow/Deny bit in this CT Access Descriptor is set to one the access is allowed. If the Allow/Deny bit in this CT Access Descriptor is set to zero the access is denied. No further authorization checks are performed; or

D) If any of the conditions listed in A, B or C do not apply, then the access is denied. No further authorization checks are performed.

7.2.5 IP Management Access to a Switch

Whenever a management application attempts to access a Switch α via an IP based access method, the IP Management List Object shall be checked to determine whether the access is to be allowed or denied. The same checks shall be performed when a new policy is activated. IP management access control applies to IP based protocols. IP management access is from a management application running on a IP host. The enforcement rules for the Switch side of an attempted access are defined. The definition of the enforcement rules for the IP host side of an attempted access is outside the scope of this standard.

Considering a management application running on an IP host A attempting to access Switch α using a certain IP Protocol and Port, Switch α shall perform the following authorization checks:

1) If there is no IP Management List Object then there are no access restrictions and the access is allowed. No authorization checks are performed;

2) If there is a IP Management List Object, then:

   A) If the IP Management List Object contains an entry that matches the IP address of IP host A, the correspondent Well Known Protocol Access Specifier needs to be checked, as specified in step 3; or

   B) If the IP Management List Object does not contain an entry that matches the IP address of IP host A, the access is denied and no further authorization checks are performed; and

3) Check the selected Well Known Protocol Access Specifier as follows:

   A) If both the IP Protocol and Port host A is attempting to use match the Well Known Protocol Number and Destination Port Number fields in a WKP Access Descriptor, then the Allow/Deny bit in the matched WKP Access Descriptor indicates if the access is allowed or denied. If the Allow/Deny bit in the matched WKP Access Descriptor is set to one the access is allowed. If the Allow/Deny bit in the matched WKP Access Descriptor is set to zero the access is denied. No further authorization checks are performed;

   B) If the IP Protocol host A is attempting to use matches the Well Known Protocol Number field in a WKP Access Descriptor having the Well Known Protocol Number Wildcard bit set to one, then the Allow/Deny bit in the matched WKP Access Descriptor indicates if the access is allowed or denied. If the Allow/Deny bit in the matched WKP Access Descriptor is set to one the access is allowed. If the Allow/Deny bit in the matched WKP Access Descriptor is set to zero the access is denied. No further authorization checks are performed;

   C) If there is a WKP Access Descriptor having both the Well Known Protocol Number and Destination Port Number Wildcard bits set to one, then the Allow/Deny bit in this WKP Access Descriptor indicates if the access is allowed or denied. If the Allow/Deny bit in this WKP Access Descriptor is set to one the access is allowed. If the Allow/Deny bit in this WKP Access Descriptor is set to zero the access is denied. No further authorization checks are performed; or

   D) If any of the conditions listed in A, B or C do not apply, then the access is denied. No further authorization checks are performed.

7.2.6 Direct Management Access to a Switch

When a management application attempts to access a Switch α via a direct access method, the Switch Membership List Object shall be checked to determine whether the access is to be allowed or denied. The same checks shall be performed when a new policy is activated. Direct management access control applies to Switch serial ports access and physical panel access. Management access from the serial ports or the physical panel of a Switch follows the same enforcement rules.

Switch α shall perform the following authorization checks:

1) If there is no Switch Membership List Object then there are no access restrictions and the access is allowed. No authorization checks are performed; and

2) If there is a Switch Membership List Object, then:

   A) If the Switch Membership List Object contains an entry for N(α) or [N(α)], then the Switch Flag associated with the considered access method in the matched entry indicates if the access is allowed or denied. If the Switch Flag associated with the considered access method in the matched entry is set to zero the access is denied. If the Switch Flag associated with the

considered access method in the matched entry is set to one the access is allowed. No further authorization checks are performed; or

B) If the Switch Membership List Object does not contain an entry for N(α) or [N(α)], but contains a Wildcard or a Negated Wildcard entry, then the Switch Flag associated with the considered access method in the matched entry indicates if the access is allowed or denied. If the Switch Flag associated with the considered access method in the matched entry is set to zero the access is denied. If the Switch Flag associated with the considered access method in the matched entry is set to one the access is allowed. No further authorization checks are performed.

7.2.7 Authentication Enforcement

When Switch β attempts to connect to Switch α, the Switch Membership List Object shall be checked to determine whether the connection, if allowed, is to be authenticated. The same checks shall be performed when a new policy is activated. If either of the Switch entries for Switch α or β has the Authentication Behavior requiring authentication, then an Authentication Transaction shall be performed over the connection, and the Switch with the bit set to one should initiate the Authentication Transaction. A rejection of the AUTH_Negotiate message shall be handled as specified by the Authentication Behavior (see table 119).

When Node A attempts to connect to Switch α, the Node Membership List Object shall be checked to determine whether the connection, if allowed, is to be authenticated. If the Node entry for Node A has the Authentication Required bit set to one, then Switch α shall request authentication to Node A, by setting the Security Bit to one in the FLOGI LS_ACC if Node A supports authentication, or by rejecting the FLOGI if Node A does not support authentication (see 5.10.5). The same checks shall be performed when a new policy is activated, by performing a link initialization if the policy for a certain Node changes.

7.3 Policies Management

7.3.1 Management Interface

Policy changes occur in a server session with the Security Policy Server. Any write access to the Security Policy Server shall occur within a server session, while read access to the Security Policy Server may occur at any time. However, the consistency of the returned data is guaranteed only inside a server session.

The Enhanced Commit Service (see FC-SW-5) is used to perform Fabric operations, when needed (see table 149). For FC-SP Fabric Policies use:

a) the Enhanced Commit Service (ECS) shall be used in assisted mode (see FC-SW-5);

b) the Application_ID in the Transaction Identifier of ECS shall be set to 01h; and

c) the ECS Switch List shall contain all reachable Switches belonging to the Fabric. The reachable Switches belonging to the Fabric shall be obtained from the FSPF topology database.

Each server session begins with a SSB request and completes with a SSE request sent to the Security Policy Server. In the Fabric, the SSB request locks the Fabric with the EACA SW_ILS, while the SSE request releases the lock with the ERCA SW_ILS (see FC-SW-5).

Active and non-active Policy Objects are persistent in the Fabric (i.e., they survive after the end of a server session). A specific management request is provided to allow the removal of non-active Policy Objects. Figure 27 shows the policy management model.

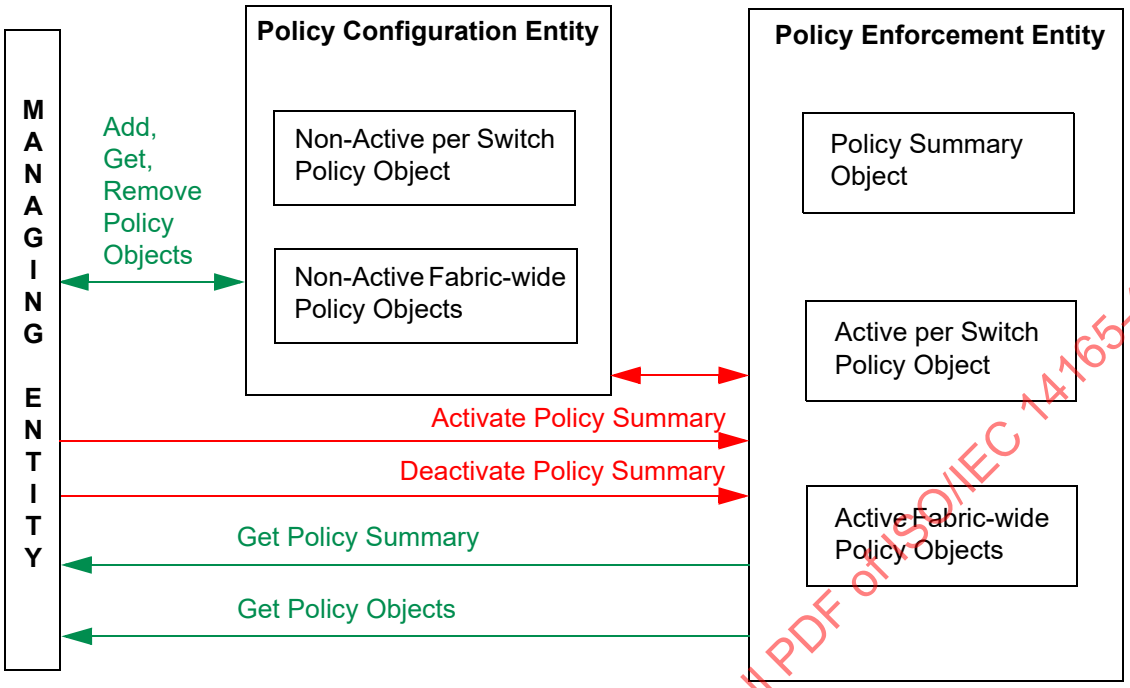

Figure 27 – Policy Management Model

Policy management is done through the Security Policy Server (see FC-GS-6) management requests shown in table 142.

Table 142 – Security Policy Server – Request Command Codes (part 1 of 2)

Code	Mnemonic & Description	Attribute(s) in Request CT_IU	Attribute(s) in Accept CT_IU
0100h	GPS Get Policy Summary	none	Policy Summary Object
0200h	APS Activate Policy Summary	Policy Summary Object	none
0300h	DPS Deactivate Policy Summary	Policy Summary Object Name	none
0110h	GPO Get Policy Object	Policy Object Name	Policy Object
0111h	GALN Get All Lists Names	none	List of List Objects Types, Names

Table 142 – Security Policy Server – Request Command Codes (part 2 of 2)

0112h	GAAO Get All Attribute Objects Names	none	List of Attribute Object Names
0210h	APO Add Policy Object	Policy Object, Hash	none
0310h	RPO Remove Policy Object	Policy Object Type, Name, Hash	none
0311h	RANA Remove All Non-Active Policy Objects	none	none
01F0h	GPOS Get Policy Objects Support	none	Fabric Policy Object Support Flags, Switches Policy Object Support Flags

7.3.2 Fabric Distribution

Only one active and possibly one updating Switch Connectivity Objects may exist on a Switch. Only the active Policy Summary Object may exist in a Fabric. Multiple instances of the other object types, each with a different name, may exist in a Fabric.

On the Fabric, policy data shall be distributed using the ESFC SW_ILS with Application_ID set to 01h and the operation codes shown in table 143.

Table 143 – ESFC Operations for Fabric Policies

Operation Code	Description
01h	Activate Policy Summary
02h	Deactivate Policy Summary
03h	Add Policy Object
04h	Remove Policy Object
05h	Remove All Non-Active Policy Objects
all others	Reserved

The ESFC Application Data payload for operation ‘Activate Policy Summary’ is shown in table 144.

Table 144 – ESFC Payload for Operation ‘Activate Policy Summary’

Item	Size (Bytes)
Operation Code = 01h	1
Reserved	3
Flags	4
Policy Summary Object	variable
Optional Data	variable

Flags: reserved.

Policy Summary Object: see 7.1.3.

Optional Data: see 7.3.5.

The ESFC Application Data payload for operation ‘Deactivate Policy Summary’ is shown in table 145.

Table 145 – ESFC Payload for Operation ‘Deactivate Policy Summary’

Item	Size (Bytes)
Operation Code = 02h	1
Reserved	3
Flags	4
Policy Summary Object Name	variable
Optional Data	variable

Flags: reserved.

Policy Summary Object Name: the Alphanumeric Name of the Policy Summary Object (see 7.1.2).

Optional Data: see 7.3.5.

The ESFC Application Data payload for operation ‘Add Policy Object’ is shown in table 146.

Table 146 – ESFC Payload for Operation ‘Add Policy Object’

Item	Size (Bytes)
Operation Code = 03h	1
Reserved	3
Flags	4
Policy Object	variable
Policy Object Hash	variable
Optional Data	variable

Flags: reserved.

Policy Object: the Policy Object being distributed (see 7.1).

Policy Object Hash: the hash of the Policy Object being distributed.

Optional Data: see 7.3.5.

The ESFC Application Data payload for operation ‘Remove Policy Object’ is shown in table 147.

Table 147 – ESFC Payload for Operation ‘Remove Policy Object’

Item	Size (Bytes)
Operation Code = 04h	1
Reserved	3
Flags	4
Policy Object Type	4
Policy Object Name	variable
Policy Object Hash	variable
Optional Data	variable

Flags: reserved.

Policy Object Type: the type of the Policy Object being removed.

Policy Object Name: the Name of the Policy Object being removed.

Policy Object Hash: the hash of the Policy Object being removed.

Optional Data: see 7.3.5.

NOTE 44 – Type, Name and Hash are sent by a management entity (e.g., via the Security Policy Server request RPO, see 7.3.6.8). Type and Name uniquely identify the policy object to be removed. The hash provides an additional check to detect if the identified object is not what was intended to remove (e.g., a hash mismatch indicates that stale data exist in the management entity).

The ESFC Application Data payload for operation ‘Remove All Non-Active Policy Objects’ is shown in table 148.

Table 148 – ESFC Payload for Operation ‘Remove All Non-Active Policy Objects’

Item	Size (Bytes)
Operation Code = 05h	1
Reserved	3
Flags	4
Policy Summary Object	variable
Optional Data	variable

Flags: reserved.

Policy Summary Object: see 7.1.3.

Optional Data: see 7.3.5.

7.3.3 Relationship between Security Policy Server Requests and Fabric Actions

All data structures are Fabric-wide, except the Switch Connectivity Objects, that are per Switch. Table 149 shows how the Security Policy Server Requests are mapped in Fabric actions.

Table 149 – Security Policy Server CT Requests and Fabric Actions

Security Policy Server CT Requests	Fabric Action
SSB	EACA
Get Requests on Fabric-wide Policies (GPS, GPO, GALN, GAAO)	None (i.e., handled locally by the attached Switch)
Add Request on Fabric-wide Policies (APO)	ESFC 'Add Policy Object'; EUFC
Remove Request on Fabric-wide Policies (RPO)	ESFC 'Remove Policy Object'; EUFC
Get Request on Switch Connectivity Object (GPO)	Autonomous or Client Switch: forward the CT request to the named Switch Server Switch: none (i.e., handled locally by the attached Switch)
Add Request on Switch Connectivity Object (APO)	Forward the CT request to the named Switch and to all Server Switches
Remove Request on Switch Connectivity Object (RPO)	Forward the CT request to the named Switch and to all Server Switches
Activate Policy Summary (APS)	ESFC 'Activate Policy Summary'; EUFC
Deactivate Policy Summary (DPS)	ESFC 'Deactivate Policy Summary'; EUFC
Remove All Non-Active Policy Objects (RANA)	ESFC 'Remove All Non-Active Policy Objects'; EUFC
SSE	ERCA

In order to change the policies enforced by a Fabric, a management application should first distribute new instances of Policy Objects by using the 'Add Policy Object' Request, then activate a new Policy Summary Object pointing to the new instances of Policy Objects, and optionally remove the old Policy Object instances by using the 'Remove Policy Object' Request. The 'Remove All Non-Active Policy Objects' request may be used to remove all non-active Policy Objects in the Fabric.

7.3.4 Policy Objects Support

7.3.4.1 Get Policy Objects Support (GPOS)

The Get Policy Objects Support request allows a management application to discover the level of support for individual Policy Object types provided by the Switches of a Fabric. The Security Policy Server shall, when it receives a GPOS operation request, return a summary of the Policy Objects support level for the Fabric and for each Switch of the Fabric.

The format of the GPOS CT_IU Request is shown in table 150.

Table 150 – GPOS Request CT_IU

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Optional Data	variable

Flags: reserved.

Optional Data: see 7.3.5.

The format of the Accept CT_IU to a GPOS Request is shown in table 151.

Table 151 – Accept CT_IU to a GPOS Request

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Fabric Policy Objects Support Flags	4
Reserved	3
Number of Switch Policy Objects Support Entry	1
Switch #1 Policy Objects Support Entry	12
Switch #2 Policy Objects Support Entry	12
...	
Switch #k Policy Objects Support Entry	12
Optional Data	variable

Flags: reserved.

Optional Data: see 7.3.5.

Fabric Policy Objects Support Flags: the defined flags are shown in table 152

Table 152 – Fabric Policy Objects Support Flags

Bit	Description
31 .. 7	Reserved
6	Attribute Object Supported. When this bit is set to one, all Switches of the Fabric support the Attribute Object. When this bit is set to zero, at least one Switch in the Fabric does not support the Attribute Object.
5	IP Management List Object Supported. When this bit is set to one, all Switches of the Fabric support the IP Management List Object. When this bit is set to zero, at least one Switch in the Fabric does not support the IP Management List Object.
4	Switch Connectivity Object Supported. When this bit is set to one, all Switches of the Fabric support the Switch Connectivity Object. When this bit is set to zero, at least one Switch in the Fabric does not support the Switch Connectivity Object.
3	Node Membership List Object Supported. When this bit is set to one, all Switches of the Fabric support the Node Membership List Object. When this bit is set to zero, at least one Switch in the Fabric does not support the Node Membership List Object.
2	Switch Membership List Object Supported. When this bit is set to one, all Switches of the Fabric support the Switch Membership List Object. When this bit is set to zero, at least one Switch in the Fabric does not support the Switch Membership List Object.
1	Policy Summary Object Supported. When this bit is set to one, all Switches of the Fabric support the Policy Summary Object. When this bit is set to zero, at least one Switch in the Fabric does not support the Policy Summary Object.
0	Reserved

The format of the Switch Policy Objects Support Entry is shown in table 153.

Table 153 – Switch Policy Objects Support Entry Format

Item	Size (Bytes)
Switch_Name	8
Switch Policy Objects Support Flags	4

Switch_Name: shall be set to the Switch_Name of the Switch for which Policy Objects Support Flags are being reported.

Switch Policy Objects Support Flags: the defined flags are shown in table 154

Table 154 – Switch Policy Objects Support Flags

Bit	Description
31 .. 7	Reserved
6	Attribute Object Supported. When this bit is set to one, this Switch supports the Attribute Object. When this bit is set to zero, this Switch does not support the Attribute Object.
5	IP Management List Object Supported. When this bit is set to one, this Switch supports the IP Management List Object. When this bit is set to zero, this Switch does not support the IP Management List Object.
4	Switch Connectivity Object Supported. When this bit is set to one, this Switch supports the Switch Connectivity Object. When this bit is set to zero, this Switch does not support the Switch Connectivity Object.
3	Node Membership List Object Supported. When this bit is set to one, this Switch supports the Node Membership List Object. When this bit is set to zero, this Switch does not support the Node Membership List Object.
2	Switch Membership List Object Supported. When this bit is set to one, this Switch supports the Switch Membership List Object. When this bit is set to zero, this Switch does not support the Switch Membership List Object.
1	Policy Summary Object Supported. When this bit is set to one, this Switch supports the Policy Summary Object. When this bit is set to zero, this Switch does not support the Policy Summary Object.
0	Reserved

7.3.4.2 ESS Security Policy Server Capability Object

The ESS SW_ILS (see FC-SW-5) allows a Switch to discover the level of support for individual Policy Object types provided by other Switches of a Fabric. A Security Policy Server Capability Object is defined for the ESS SW_ILS. The WKA Type shall be set to FAh and the WKA Subtype shall be set to 06h. The ESS Security Policy Server Capability Object format is shown in table 155.

Table 155 – ESS Security Policy Server Capability Object Format

Item	Size (Bytes)
Switch Policy Objects Support Flags	4
Reserved	4

Switch Policy Objects Support Flags: the defined flags are shown in table 154.

7.3.5 Optional Data

7.3.5.1 Overview

Any Payload carrying policy data structures may also carry some optional Security Objects in the Optional Data field. Optional Data are optional to send and may be safely ignored on reception. The Optional Data field has the structure shown in table 156.

Table 156 – Optional Data Field Format

Item	Size (Bytes)
Total Length of Security Objects	4
Security Object #1	variable
Security Object #2	variable
...	
Security Object #k	variable

Total Length of Security Objects: shall be set to the total length in bytes of the Security Objects contained in the Optional Data Field. The length shall be multiple of four. This field shall always be present.

Security Object: each optional Security Object is encoded as shown in table 157.

Table 157 – Security Object Format

Item	Size (Bytes)
Security Object Tag	4
Security Object Length	4
Security Object Payload	variable

Security Object Tag: the defined Security Object Tags are shown in table 158.

Table 158 – Security Object Tags

Tag Value	Security Object
FFFF FF00h : FFFF FFFEh	Vendor Specific
all others	Reserved

Security Object Length: shall be set to the length in bytes of the Security Object Payload.

Security Object Payload: contains the Security Object.

7.3.5.2 Vendor Specific Security Object

The Security Object Payload of Vendor Specific Security Objects shall have the format shown in table 159.

Table 159 – Vendor Specific Security Object Payload Format

Item	Size (Bytes)
Vendor_ID	8
Vendor Specific Information	variable

Vendor_ID: This field shall contain the vendor's T10 Vendor ID.

7.3.6 Detailed Management Specification

7.3.6.1 Get Policy Summary (GPS)

The Security Policy Server shall, when it receives a GPS operation request, return the Policy Summary Object representing the Policy configuration currently enforced by the Fabric.

Fabric action: none (i.e., the request is handled locally by the attached Switch).

The format of the GPS CT_IU Request is shown in table 160.

Table 160 – GPS Request CT_IU

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Optional Data	variable

Flags: reserved.

Optional Data: see 7.3.5.

The format of the Accept CT_IU to a GPS Request is shown in table 161.

Table 161 – Accept CT_IU to a GPS Request

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Policy Summary Object	variable
Optional Data	variable

Flags: reserved.

Policy Summary Object: the Policy Summary Object currently enforced by the Fabric (see 7.1.3).

Optional Data: see 7.3.5.

7.3.6.2 Activate Policy Summary (APS)

The Security Policy Server shall, when it receives an APS operation request, activate the Policy configuration represented in the received Policy Summary Object.

Fabric action: the received Policy Summary Object is distributed to all Switches of the Fabric by using the ESFC SW_ILS with operation code 'Activate Policy Summary'. If all Switches accept the ESFC request, the Policy Summary Object is committed and put in enforcement by sending to all Switches a EUFC SW_ILS. Then the APS CT_IU Accept is sent to the managing entity.

The format of the APS CT_IU Request is shown in table 162.

Table 162 – APS Request CT_IU

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Policy Summary Object	variable
Optional Data	variable

Flags: reserved.

Policy Summary Object: the Policy Summary Object to activate (see 7.4.3).

Optional Data: see 7.3.5.

The format of the Accept CT_IU to an APS Request is shown in table 163.

Table 163 – Accept CT_IU to an APS Request

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Optional Data	variable

Flags: reserved.

Optional Data: see 7.3.5.

7.3.6.3 Deactivate Policy Summary (DPS)

The Security Policy Server shall, when it receives a DPS operation request, deactivate the currently enforced Policy configuration, represented by the Policy Summary Object named in the request.

Fabric action: the received Policy Summary Object Name is distributed to all Switches of the Fabric by using the ESFC SW_ILS with operation code 'Deactivate Policy Summary'. If all Switches accept the ESFC request, the Policy enforcement is disabled by sending to all Switches a EUFC SW_ILS. Then the DPS CT_IU Accept is sent to the managing entity.

The format of the DPS CT_IU Request is shown in table 164.

Table 164 – DPS Request CT_IU

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Policy Summary Object Name	variable
Optional Data	variable

Flags: reserved.

Policy Summary Object Name: the Name of the Policy Summary Object enforced by the Fabric (see 7.1.3).

Optional Data: see 7.3.5.

The format of the Accept CT_IU to a DPS Request is shown in table 165.

Table 165 – Accept CT_IU to a DPS Request

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Optional Data	variable

Flags: reserved.

Optional Data: see 7.3.5.

7.3.6.4 Get Policy Object (GPO)

The Security Policy Server shall, when it receives a GPO operation request, return the requested Policy Object.

Fabric action: none (i.e., the request is handled locally by the attached Switch) if the Switch is a Server Switch, or if the requested Policy Object is a Fabric-wide Policy Object. If the Switch is a Client Switch or an Autonomous Switch, and the Policy Object is a Switch Connectivity Object, then the GPO CT_IU Request is forwarded to the named Switch, and the received reply is forwarded to the managing entity.

The format of the GPO CT_IU Request is shown in table 166.

Table 166 – GPO Request CT_IU

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Policy Object Name	variable
Optional Data	variable

Flags: reserved.

Policy Object Name: the Name of the requested Policy Object.

Optional Data: see 7.3.5.

The format of the Accept CT_IU to a GPO Request is shown in table 167.

Table 167 – Accept CT_IU to a GPO Request

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Policy Object	variable
Optional Data	variable

Flags: reserved.

Policy Object: the requested Policy Object.

Optional Data: see 7.3.5.

7.3.6.5 Get All Lists Names (GALN)

The Security Policy Server shall, when it receives a GALN operation request, return the names of all List Objects (i.e., Switch Membership Lists, Node Membership Lists, IP Management Lists) belonging to the Fabric Policy configuration.

Fabric action: none (i.e., the request is handled locally by the attached Switch).

The format of the GALN CT_IU Request is shown in table 168.

Table 168 – GALN Request CT_IU

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Optional Data	variable

Flags: reserved.

Optional Data: see 7.3.5.

The format of the Accept CT_IU to a GALN Request is shown in table 169.

Table 169 – Accept CT_IU to a GALN Request

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Number of List Objects Names	4
List Object Type #1	4
List Object Name #1	variable
List Object Type #2	4
List Object Name #2	variable
...	
List Object Type #k	4
List Object Name #k	variable
Optional Data	variable

Flags: reserved.

Number of List Objects Names: the number of returned List Objects Names.

List Object Type: the type of the List Object for which name is returned.

List Object Name: the returned List Object Name.

Optional Data: see 7.3.5.

7.3.6.6 Get All Attribute Objects Names (GAAO)

The Security Policy Server shall, when it receives a GAAO operation request, return the names of all Attribute Objects belonging to the Fabric Policy configuration.

Fabric action: none (i.e., the request is handled locally by the attached Switch).

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

Table 170 – GAAO Request CT_IU

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Optional Data	variable

Flags: reserved.

Optional Data: see 7.3.5.

The format of the Accept CT_IU to a GAAO Request is shown in table 171.

Table 171 – Accept CT_IU to a GAAO Request

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Number of Attribute Object Names	4
Attribute Object Name #1	variable
Attribute Object Name #2	variable
...	
Attribute Object Name #h	variable
Optional Data	variable

Flags: reserved.

Number of Attribute Object Names: the number of returned Attribute Object Names.

Attribute Object Name: the returned Attribute Object Name.

Optional Data: see 7.3.5.

7.3.6.7 Add Policy Object (APO)

The Security Policy Server shall, when it receives an APO operation request, add the received Policy Object to the Fabric Policy configuration.

Fabric action: if the received Policy Object is a Fabric-wide Policy Object, then it is distributed to all Switches of the Fabric by using the ESFC SW_ILS with operation code 'Add Policy Object'. If all Switches accept the ESFC request, the addition of the received Policy Object is committed by sending to all Switches a EUFC SW_ILS. Then the APO CT_IU Accept is sent to the managing entity. If the received Policy Object is a Switch Connectivity Object, then the APO CT_IU Request is forwarded to the named Switch and to all Server Switches in the Fabric. If these Switches accept the APO CT_IU Request, then the APO CT_IU Accept is sent to the managing entity.

The APO request does not affect the current Policy Summary Object. The format of the APO CT_IU Request is shown in table 172.

Table 172 – APO Request CT_IU

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Policy Object	variable
Policy Object Hash	variable
Optional Data	variable

Flags: reserved.

Policy Object: the Policy Object being added to the Fabric Policy configuration.

Policy Object Hash: the hash of the Policy Object being added to the Fabric Policy configuration.

Optional Data: see 7.3.5.

The format of the Accept CT_IU to an APO Request is shown in table 173.

Table 173 – Accept CT_IU to an APO Request

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Optional Data	variable

Flags: reserved.

Optional Data: see 7.3.5.

7.3.6.8 Remove Policy Object (RPO)

The Security Policy Server shall, when it receives an RPO operation request, remove the named Policy Object from the Fabric Policy configuration.

Fabric action: if the requested Policy Object is a Fabric-wide Policy Object, then its type, Name and hash are distributed to all Switches of the Fabric by using the ESFC SW_ILS with operation code 'Remove Policy Object'. If all Switches accept the ESFC request, the removal of the named Policy Object is committed by sending to all Switches a EUFC SW_ILS. Then the RPO CT_IU Accept is sent to the managing entity. If the received Policy Object is a Switch Connectivity Object, then the RPO CT_IU Request is forwarded to the named Switch and to all Server Switches in the Fabric. If these Switches accept the RPO CT_IU Request, then the RPO CT_IU Accept is sent to the managing entity.

The RPO request does not affect the current Policy Summary Object. The format of the RPO CT_IU Request is shown in table 174.

Table 174 – RPO Request CT_IU

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Policy Object Type	4
Policy Object Name	variable
Policy Object Hash	variable
Optional Data	variable

Flags: reserved.

Policy Object Type: the type of the Policy Object being removed.

Policy Object Name: the Name of the Policy Object being removed.

Policy Object Hash: the hash of the Policy Object being removed.

Optional Data: see 7.3.5.

NOTE 45 – Type and Name uniquely identify the policy object to be removed. The hash provides an additional check to detect if the identified object is not what was intended to remove (e.g., a hash mismatch indicates that stale data exist in the management entity).

The format of the Accept CT_IU to a RPO Request is shown in table 175.

Table 175 – Accept CT_IU to a RPO Request

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Optional Data	variable

Flags: reserved.

Optional Data: see 7.3.5.

7.3.6.9 Remove All Non-Active Policy Objects (RANA)

The Security Policy Server shall, when it receives a RANA operation request, remove all the non-active Policy Objects from the Fabric Policy configuration.

Fabric action: the Policy Summary Object is distributed to all Switches of the Fabric by using the ESFC SW_ILS with operation code ‘Remove All Non-Active Policy Objects’. If all Switches accept the ESFC request, the removal of all non-active Policy Objects is committed by sending to all Switches a EUFC SW_ILS. Then the RANA CT_IU Accept is sent to the managing entity.

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

Table 176 – RANA Request CT_IU

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Optional Data	variable

Flags: reserved.

Optional Data: see 7.3.5.

The format of the Accept CT_IU to a RANA Request is shown in table 177.

Table 177 – Accept CT_IU to a RANA Request

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Optional Data	variable

Flags: reserved.

Optional Data: see 7.3.5.

7.4 Policies Check

7.4.1 Overview

Two connected Switches shall check the Policy Summary Objects to ensure the two Fabrics have the same Policies. A join between two Fabrics is successful only if the two Fabrics have the same policies (i.e., they are enforcing the same Policy Summary Object). If the policies are not equal the two communicating E_Ports shall enter the Isolated state.

The check of the Fabric Policies is performed with the Check Policy Summary (CPS) SW_ILS. Each E_Port shall send a CPS Request carrying the Policy Summary Object of the Fabric to which the Switch belongs. On receiving a CPS Request, a Switch shall verify if the received Policy Summary Object is equal to the enforced Policy Summary Object. If yes, an SW_ILS Accept is sent in reply. If not, an SW_RJT with Reason Code 'Logical Error' and Reason Code Explanation 'Policy Summary not Equal' is sent in reply, and the E_Port shall enter the Isolated state.

7.4.2 CPS Request Sequence

Protocol: CPS (Check Policy Summary) SW_ILS Request Sequence

Addressing: The S_ID field shall be set to FFFFFDh, indicating the Fabric Controller of the originating Switch, and the D_ID field shall be set to FFFFFDh, indicating the Fabric Controller of the destination Switch.

Payload: The format of the CPS SW_ILS Request Sequence Payload is shown in table 178.

Table 178 – Check Policy Summary SW_ILS Request Payload

Item	Size (Bytes)
4201 0000h	4
Flags	4
Policy Summary Object	variable
Optional Data	variable

Flags: reserved.

Policy Summary Object: see 7.1.3.

Optional Data: see 7.3.5.

7.4.3 CPS Reply Sequence

SW_RJT: SW_RJT shall be sent as a reply to signify the rejection of the CPS Request Sequence for reasons shown in table 179.

Table 179 – Check Policy Summary SW_RJT Reasons

Reason	Reason Code	Reason Code Explanation
Policy Summary not Equal	03h	35h
CPS not supported	0Bh	2Ch
Logical Busy	05h	00h

SW_ACC: SW_ACC shall be sent as a reply to signify the acceptance of the CPS Request Sequence for processing. The format of the CPS SW_ACC Payload is shown in table 180.

Table 180 – Check Policy Summary SW_ACC Payload

Item	Size (Bytes)
0200 0000h	4
Flags	4
Policy Summary Object	variable
Optional Data	variable

Flags: reserved.

Policy Summary Object: see 7.1.3.

Optional Data: see 7.3.5.

7.5 Policy Summation ELSs

7.5.1 Overview

An Nx_Port may obtain a summary of selected Fabric-wide policies and register for notifications of relevant policy changes with the Query Security Attributes (QSA) ELS (see FC-LS-3). The Fabric may notify registered Nx_Ports of relevant changes in selected Fabric-wide policies with the Registered Fabric Change Notification (RFCN) ELS (see FC-LS-3).

7.5.2 Fabric Change Notification Specification

When a QSA Request Sequence from a given N_Port_ID has been accepted by a Fabric Controller and the N_Port_ID registered for change notification for a security attribute, then if a subsequent policy change modifies the state of that security attribute, the following steps shall be performed:

1) The Fabric Controller shall reply to all QSA Request Sequences from all N_Port_IDs with a LS_RJT having a ‘Logical Busy’ Reason Code, and a Reason Code Explanation of ‘No Additional Explanation’;

2) The Fabric Controller shall send a Registered Fabric Change Notification (RFCN) Request Sequence to the registered N_Port_ID, and shall implicitly log out the N_Port_ID with the Fabric;

3) The Fabric Controller shall delay for a maximum of 2 times RA_TOV, to allow all frames received from the N_Port_ID prior to Fabric logout to reach their destinations;

4) The Fabric Controller shall put into effect the change in the security attribute; and

5) The Fabric Controller shall accept subsequent QSA Request Sequences from all N_Port_IDs.

7.6 Zoning Policies

7.6.1 Overview

In order to preserve backward compatibility with existing Zoning definitions and implementations, 7.6 describes a variant of the Enhanced Zoning model defined in FC-SW-5 and FC-GS-6 that follows the general concepts of the Policy model, but keeps Zoning management and enforcement completely independent from other policies management and enforcement. This variant of Zoning is denoted as FC-SP Zoning.

FC-SP Zoning allows some Switches to not maintain a complete replicated copy of the Zoning Database, as follows:

a) Server Switches maintain the policies data structures for all Switches in the Fabric plus a replica of the Zoning data structures;

b) Autonomous Switches maintain only the subset of policies data structures relevant for their operations plus a replica of the Zoning Database; and

c) Client Switches maintain only the subset of policies data structures and the subset of the Active Zone Set relevant for their operations.

When Client Switches are deployed in a Fabric, at least one Server Switch shall be also deployed in the same Fabric. A client-server protocol allows Client Switches to dynamically retrieve the Zoning information they may require from the Server Switches.

A management application manages the Fabric Zoning configuration through the Fabric Zone Server, while other policies are managed through the Security Policy Server. A Zoning Check Protocol, replacing the Zone Merge Protocol (see FC-SW-5), is defined, along with new command codes for the SFC SW_ILS to distribute the FC-SP Zoning configuration on a Fabric. The Zoning definitions shall be ordered, to compute a hash of the Active Zone Set and a hash of the Zone Set Database, plus other optional security data (e.g., for integrity protection of Zoning information).

7.6.2 Management Requests

7.6.2.1 Overview

FC-SP Zoning support by a Fabric is discovered with the GFEZ request (see FC-GS-6) extended by 7.6.2. If all Switches of a Fabric support FC-SP Zoning, then it may be enabled with the SFEZ request (see FC-GS-6). The Active Zone Set and Zone Set Database hashes are carried in an enhanced version of the CMIT request (see FC-GS-6), called SPCMIT (see 7.6.2.4).

7.6.2.2 Get Fabric Enhanced Zoning Support (GFEZ) Additions

To support FC-SP Zoning the FC-SP Zoning Supported bit and the FC-SP Zoning Enabled bit are defined in the Fabric Enhanced Zoning Support Flags and in the Switch Enhanced Zoning Support Flags of the GFEZ payload (see FC-GS-6), as shown in table 181 and table 182.

Table 181 – Fabric Enhanced Zoning Support Flags Additions

Bit Position	Description
8	**FC-SP Zoning Supported:** When this bit is one, the Fabric is able to work in FC-SP Zoning mode. When this bit is zero, the Fabric is not able to work in FC-SP Zoning mode.
9	**FC-SP Zoning Enabled:** When this bit is one, the Fabric is working in FC-SP Zoning mode. When this bit is zero, the Fabric is not working in FC-SP Zoning mode.
all others	See FC-GS-6

Table 182 – Switch Enhanced Zoning Support Flags Additions

Bit Position	Description
8	**FC-SP Zoning Supported:** When this bit is one, this Switch is able to work in FC-SP Zoning mode. When this bit is zero, this Switch is not able to work in FC-SP Zoning mode.
9	**FC-SP Zoning Enabled:** When this bit is one, this Switch is working in FC-SP Zoning mode. When this bit is zero, this Switch is not working in FC-SP Zoning mode.
all others	See FC-GS-6

7.6.2.3 Set Fabric Enhanced Zoning Support (SFEZ) Additions

To support FC-SP Zoning the FC-SP Zoning Enable bit is defined in the Fabric Enhanced Zoning Request Flags of the SFEZ payload (see FC-GS-6), as shown in table 183.

Table 183 – Fabric Enhanced Zoning Request Flags Additions

Bit Position	Description
4	**FC-SP Zoning Enable:** When this bit is one, the management application is requesting the Fabric to set its Zoning operational mode as FC-SP. When this bit is zero, the management application is requesting the Fabric to not change its Zoning operational mode.
all others	See FC-GS-6

The Fabric Zone Server shall, when it receives a SFEZ request, configure the Fabric to operate in the requested Zoning mode. If the Fabric is not capable of operating in FC-SP Zoning mode, then the Fabric Zone Server shall reject the request with an Reason Code Explanation ‘Fabric not able to operate in FC-SP Mode’. When the Fabric is operating in FC-SP mode the SFEZ request may be used to modify the Fabric Zoning policies. When the Fabric is operating in FC-SP mode it is not possible to revert to Basic mode.

7.6.2.4 SP Commit Zone Changes (SPCMIT)

The Fabric Zone Server shall, when it receives a SPCMIT operation request, commit all outstanding modifications made by the issuing client to the Zone Set Database.

The SPCMIT processing may persist longer than the Common Transport timeout (i.e., 3 times R_A_TOV). The Fabric Zone Server shall reply within the Common Transport timeout. The reply shall be a Reject CT_IU having Reason Code 'Logical Busy' and Reason Code Explanation 'Request in Process' until the SPCMIT processing completes successfully or unsuccessfully. The Fabric Zone Server shall return a response to the SPCMIT Request other than a Reject CT_IU with a 'Logical Busy' Reason Code and a 'Request in Process' Reason Code Explanation when the SPCMIT processing completes. The Fabric Zone Server shall respond to any other Fabric Zone Server Requests with a Reject CT_IU with a 'Logical Busy' Reason Code and a 'SPCMIT in Process' Reason Code Explanation until a response to the SPCMIT Request other than a Reject CT_IU with a 'Logical Busy' Reason Code and a 'Request in Process' Reason Code Explanation has been sent.

NOTE 46 – The management application should retransmit the SPCMIT Request until an Accept CT_IU or a Reject CT_IU with a Reason Code other than 'Logical Busy' and Reason Code Explanation other than 'Request in Process' is received, or until the application gets tired of it. When the Fabric is processing a SPCMIT Request, any subsequently received SPCMIT Requests do not interrupt or restart the processing in progress. Instead, subsequent SPCMIT Requests are a way for the management application to know when and how the SPCMIT processing completes.

The SPCMIT Request shall not be used in Basic Zoning or Enhanced Zoning.

The format of the SPCMIT Request payload is shown in table 184.

Table 184 – SPCMIT Request Payload

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Active Zone Set Hash	variable
Zone Set Database Hash	variable
Optional Data	variable

Flags: reserved.

Active Zone Set Hash: the hash of the Active Zone Set being committed. The hash format is as specified in table 111.

Zone Set Database Hash: the hash of the Zone Set Database being committed. The hash format is as specified in table 111.

Optional Data: see 7.3.5.

The format of the SPCMIT Accept payload is shown in table 185.

Table 185 – SPCMIT Accept Payload

Item	Size (Bytes)
CT_IU Preamble	see FC-GS-6
Flags	4
Optional Data	variable

Flags: reserved.

Optional Data: see 7.3.5.

7.6.3 Fabric Operations

7.6.3.1 Overview

FC-SP Zoning support by a Switch is discovered with the ESS SW_ILS (see FC-SW-5) extended by 7.6.3. If all Switches of a Fabric support FC-SP Zoning, then the Fabric is able to operate in FC-SP Zoning mode. When FC-SP Zoning is enabled, the Zone Merge Protocol is not used, and is replaced by the Zoning Check Protocol, while new SFC operation codes are used to distribute Zoning configuration information in a Fabric. In addition, a Fabric may be composed by Server Switches, Autonomous Switches, and Client Switches. A client-server protocol (see 7.6.5) is used between Clients and Server Switches.

7.6.3.2 ESS Enhanced Zone Server Capability Object Additions

To support FC-SP Zoning the FC-SP Zoning Supported bit and the FC-SP Zoning Enabled bit are defined in the Zone Server Support Flags of the ESS SW_ILS payload (see FC-SW-5), as shown in table 186.

Table 186 – ESS Zone Server Support Flags Additions

Bit Position	Description
8	**FC-SP Zoning Supported:** When this bit is one, this Switch is able to work in FC-SP Zoning mode. When this bit is zero, this Switch is not able to work in FC-SP Zoning mode.
9	**FC-SP Zoning Enabled:** When this bit is one, this Switch is working in FC-SP Zoning mode. When this bit is zero, this Switch is not working in FC-SP Zoning mode.
all others	See FC-GS-6

7.6.3.3 The Zoning Check Protocol

7.6.3.3.1 Overview

The Active Zone Set and Zone Set Database hashes are checked to ensure the two Fabrics have the same Zoning configuration. A join between two Fabrics is successful only if the two Fabric have the same Zoning configurations (i.e., the Active Zone Set and Zone Set Database hashes match). If the Zoning configurations are not equal, the two communicating E_Ports shall go to the Isolated state.

The check of the Fabric Zoning configurations is performed with the Zoning Check Protocol (ZCP) SW_ILS. Each E_Port shall send a ZCP Request carrying the Zoning hashes of the Fabric to which the Switch belongs. On receiving a ZCP Request, a Switch shall verify if the received Zoning hashes are equal

to the hashes of its Zoning configuration. If they are equal an SW_ACC is sent in reply. If they are not equal an SW_RJT with Reason Code 'Logical Error' and Reason Code Explanation 'FC-SP Zoning Summary Not Equal' is sent in reply, and the E_Port shall go to the Isolated state.

7.6.3.3.2 ZCP Request Sequence

Protocol: ZCP (Zoning Check Protocol) SW_ILS Request Sequence

Addressing: The S_ID field shall be set to FFFFFDh, indicating the Fabric Controller of the originating Switch, and the D_ID field shall be set to FFFFFDh, indicating the Fabric Controller of the destination Switch.

Payload: The format of the ZCP SW_ILS Request Sequence Payload is shown in table 187.

Table 187 – Zoning Check Protocol SW_ILS Request Payload

Item	Size (Bytes)
4202 0000h	4
Flags	4
Active Zone Set Hash	variable
Zone Set Database Hash	variable
Optional Data	variable

Flags: reserved.

Active Zone Set Hash: the hash of the Active Zone Set of the sending Switch. The hash format is as specified in table 111.

Zone Set Database Hash: the hash of the Zone Set Database of the sending Switch. The hash format is as specified in table 111.

Optional Data: see 7.3.5.

7.6.3.3.3 ZCP Reply Sequence

SW_RJT: SW_RJT shall be sent as a reply to signify the rejection of the ZCP Request Sequence for reasons shown in table 188.

Table 188 – Zoning Check Protocol SW_RJT Reasons

Reason	Reason Code	Reason Code Explanation
FC-SP Zoning Summary Not Equal	03h	36h
ZCP Not Supported	0Bh	2Ch
Logical Busy	05h	00h

SW_ACC: SW_ACC shall be sent as a reply to signify the acceptance of the ZCP Request Sequence for processing. The format of the ZCP SW_ACC Payload is shown in table 189.

Table 189 – Zoning Check Protocol SW_ACC Payload

Item	Size (Bytes)
0200 0000h	4
Flags	4
Active Zone Set Hash	variable
Zone Set Database Hash	variable
Optional Data	variable

The fields are the same of the ZCP Request Sequence Payload.

7.6.3.4 Additional SFC Operation Request Codes

7.6.3.4.1 Overview

The additional SFC SW_ILS (see FC-SW-5) operation request codes shown in table 190 are defined to support FC-SP Zoning.

Table 190 – Additional SFC Operation Request Codes

Value	Operation	Reference
05h .. 07h	Reserved for legacy implementations	
20h .. 27h	Reserved	
28h	FC-SP Activate Zone Set Enhanced	7.6.3.4.2
29h	FC-SP Deactivate Zone Set Enhanced	7.6.3.4.3
2Ah	FC-SP Distribute Zone Set Database	7.6.3.4.4
2Bh	FC-SP Activate Zone Set by Name	7.6.3.4.5
2Ch	FC-SP Set Zoning Policies	7.6.3.4.6
2Dh .. 3Fh	Reserved	

7.6.3.4.2 Operation Request 'FC-SP Activate Zone Set Enhanced'

Operation Request 'FC-SP Activate Zone Set Enhanced' is used in FC-SP Zoning to activate a Zone Set distributing its definition across the Fabric. Together with the Zone Set to be activated, also the entire Zone Set Database may be distributed. Table 191 shows the payload format.

Table 191 – Payload for the Operation Request 'FC-SP Activate Zone Set Enhanced'

Item	Size (Bytes)
Reserved	2
Flags	4
Active Zone Set Length	4
Active Zone Set Object List	variable
Zone Set Database Length	4
Zone Set Database Object List	variable
Active Zone Set Hash	variable
Zone Set Database Hash	variable
Optional Data	variable

Flags: reserved.

Active Zone Set Length: as defined in FC-SW-5.

Active Zone Set Object List: as defined in FC-SW-5.

Zone Set Database Length: as defined in FC-SW-5.

Zone Set Database Object List: as defined in FC-SW-5.

Active Zone Set Hash: the hash of the Active Zone Set being activated. The hash format is as specified in table 111.

Zone Set Database Hash: the hash of the Zone Set Database being distributed. The hash format is as specified in table 111.

Optional Data: see 7.3.5.

7.6.3.4.3 Operation Request 'FC-SP Deactivate Zone Set Enhanced'

Operation Request 'FC-SP Activate Zone Set Enhanced' is used in FC-SP Zoning to deactivate the current Active Zone Set. Table 192 shows the payload format.

Table 192 – Payload for the Operation Request 'FC-SP Deactivate Zone Set Enhanced'

Item	Size (Bytes)
Reserved	2
Flags	4
Optional Data	variable

Flags: reserved.

Optional Data: see 7.3.5.

7.6.3.4.4 Operation Request 'FC-SP Distribute Zone Set Database'

Operation Request 'FC-SP Distribute Zone Set Database' applies to the Zone Set Database. Its purpose is to distribute in the Fabric a new definition of the Zone Set Database, without affecting the Active Zone Set. Table 193 shows the payload format.

Table 193 – Payload for the Operation Request 'FC-SP Distribute Zone Set Database'

Item	Size (Bytes)
Reserved	2
Flags	4
Zone Set Database Length	4
Zone Set Database Object List	variable
Zone Set Database Hash	variable
Optional Data	variable

Flags: reserved.

Zone Set Database Length: as defined in FC-SW-5.

Zone Set Database Object List: as defined in FC-SW-5.

Zone Set Database Hash: the hash of the Zone Set Database being distributed. The hash format is as specified in table 111.

Optional Data: see 7.3.5.

7.6.3.4.5 Operation Request 'FC-SP Activate Zone Set by Name'

Operation Request 'FC-SP Activate Zone Set by Name' applies to both Active Zone Set and Zone Set Database. Its purpose is to activate a Zone Set defined in the Zone Set Database without having to transmit over the Fabric its definition. Table 194 shows the payload format.

Table 194 – Payload for the Operation Request 'FC-SP Activate Zone Set by Name'

Item	Size (Bytes)
Reserved	2
Flags	4
Zone Set Name	variable
Zone Set Database Hash	variable
Optional Data	variable

Flags: reserved.

Zone Set Name: as defined in FC-SW-5.

Zone Set Database Hash: the hash of the Zone Set Database.

Optional Data: see 7.3.5.

7.6.3.4.6 Operation Request 'FC-SP Set Zoning Policies'

Operation Request 'FC-SP Set Zoning Policies' is used in FC-SP Zoning to establish some Zoning specific behaviors, such as the Default Zone behavior. Table 195 shows the payload format.

Table 195 – Payload for the Operation Request 'FC-SP Set Zoning Policies'

Item	Size (Bytes)
Reserved	2
Flags	4
Enhanced Zoning Flags	4
Optional Data	variable

Flags: reserved.

Enhanced Zoning Flags: as defined in FC-SW-5. Among the defined Enhanced Zoning Flags only the Default Zone Setting flag is significant in FC-SP Zoning (i.e., the Merge Control Setting flag does not apply).

Optional Data: see 7.3.5.

7.6.3.5 Fabric Behavior to Handle the CT SFEZ Request

If the Fabric is operating in Basic mode, and the SFEZ command is requesting to change the Zoning operational mode of the Fabric to FC-SP, the Switch handling the CT SFEZ request shall initiate a Fabric Management Session, and redistribute an ordered version of the existing Zoning Database to the other Switches of the Fabric by using the operation request value 'FC-SP Activate Zone Set Enhanced' in a first

Stage Fabric Configuration (SFC) SW_ILS. In this manner the Zoning Database is distributed using the Enhanced Zoning Data structures. If this step is successful, then the Zoning Policy Flags requested by the SFEZ command shall be propagated to the other Switches of the Fabric by using the operation request value 'Set Zoning Policies' in a second Stage Fabric Configuration (SFC) SW_ILS. When this step is successful the UFC SW_ILS is used to apply the new configuration, changing the Zoning operational mode of all Switches in the Fabric in FC-SP mode. Then the Fabric Management Session shall be released.

If the Fabric is operating in Enhanced or FC-SP mode, then the SFEZ command may only change the Zoning Policies, and so it shall be translated in a standalone Fabric Management Session in which the operation request value 'Set Zoning Policies' shall be used in the SFC SW_ILS. If a Fabric Management Session is already active, then the SFEZ command shall generate an immediate SFC with operation request value 'Set Zoning Policies' and a UFC SW_ILSs, keeping the existing Fabric Management Session.

7.6.4 Zoning Ordering Rules

7.6.4.1 Active Zone Set

In the Active Zone Set, Zones shall be sorted alphabetically in ascending order, using the Zone Name as alphabetic primary key.

In each Zone, the Zone Attribute entries, if present, shall be sorted in ascending order, using the Zone Attribute Type as numeric primary key. If multiple Protocol Attribute entries are present, they shall be sorted in ascending order, using the FC-4 type as numeric secondary key. If multiple Vendor Specified Attribute entries are present, they shall be sorted among themselves alphabetically in ascending order, using the Vendor Identifier string as alphabetic primary key and the Vendor Specified Value as alphabetic secondary key.

In each Zone, the Zone Members shall be sorted in ascending order, using the Zone Member Type as numeric primary key and the Zone Member Value as numeric secondary key. If multiple Vendor Specified Member entries are present, they shall be sorted among themselves alphabetically in ascending order, using the Vendor Identifier string as alphabetic primary key and the Vendor Specified Value as alphabetic secondary key.

7.6.4.2 Zone Set Database

In the Zone Set Database, Zoning Objects shall be sorted in ascending order, using the Zoning Object Type as numeric primary key and the Zoning Object Name as alphabetic secondary key.

In each Zone Set Object, Zone References shall be sorted in ascending order, using the referenced Zone Name as alphabetic primary key.

In each Zone Object, the Zone Members shall be sorted in ascending order, using the Zone Member Type as numeric primary key and the Zone Member Value as numeric secondary key (alphabetic if the Zone Member type is Alias Zone Member). If multiple Vendor Specified Member entries are present, they shall be sorted among themselves alphabetically in ascending order, using the Vendor Identifier string as alphabetic primary key and the Vendor Specified Value as alphabetic secondary key.

In each Zone Attribute Object, the Zone Attribute entries shall be sorted in ascending order, using the Zone Attribute Type as numeric primary key. If multiple Protocol Attribute entries are present, they shall be sorted in ascending order, using the FC-4 type as numeric secondary key. If multiple Vendor Specified Attribute entries are present, they shall be sorted among themselves alphabetically in ascending order,

using the Vendor Identifier string as alphabetic primary key and the Vendor Specified Value as alphabetic secondary key.

In each Zone Alias Object, the Zone Members shall be sorted in ascending order, using the Zone Member Type as numeric primary key and the Zone Member Value as numeric secondary key. If multiple Vendor Specified Member entries are present, they shall be sorted among themselves alphabetically in ascending order, using the Vendor Identifier string as alphabetic primary key and the Vendor Specified Value as alphabetic secondary key.

7.6.5 The Client-Server Protocol

7.6.5.1 Overview

FC-SP Zoning allows some Switches (i.e., Client Switches) to not maintain a complete replicated copy of the Zoning Database. Client Switches are not required to implement the Zone Set Database, nor to maintain a complete copy of the Active Zone Set. They need to maintain only the subset of the Active Zone Set required to enforce the current Zoning configuration for the Nx_Ports directly connected to them, and the hashes for both Active Zone Set and the Zone Set Database.

When a Client Switch joins a Fabric, it shall check if its Zoning configuration is equal to the Zoning configuration of the Fabric to which is joining with the ZCP SW_ILS. If the hashes are equal then the join is able to succeed, otherwise the join shall fail.

When a management application changes the Zoning configuration of a Fabric, using the method described in 7.6.3.4, each Client Switch receives a complete copy of the Zoning configuration with the associated hashes. A Client Switch shall maintain these hashes, and may maintain only the subset of the Active Zone Set required to enforce the Zoning configuration for the Nx_Ports directly connected to it.

When some Nx_Ports connect to a Client Switch, if the Switch has no Zoning information relative to those Nx_Ports, it shall request to a Server Switch the missing Zoning information by sending a ZIR SW_ILS request. The Server Switch shall reply with an SW_ACC containing the list of Zones in the Active Zone Set containing the requested members. As a result a Fabric deploying Client Switches shall also deploy at least one Server Switch. While this process is in progress the Client Switch may send OLS (see FC-FS-3) to the connecting Nx_Ports, until the requested Zoning information is received.

The Client Switch should request the missing Zoning information from the nearest Server Switch, according to the FSPF shortest path (see FC-SW-5). This allows an effective spread of the load of the queries over the available Server Switches. Each Client Switch is able to keep the information about which zone member types are used in the Active Zone Set. This allows the Client Switch to optimize the requests to Server Switches by using in a Zoning Information Request (see 7.6.5.2) only the zone member types currently used by the Fabric to identify a specific Nx_Port, and not all the possible ones.

7.6.5.2 Zone Information Request (ZIR)

7.6.5.2.1 Overview

Client Switches and Server Switches using FC-SP Zoning shall implement the ZIR SW_ILS. Client Switches shall implement the requesting part of the protocol and Server Switches shall implement the responding part.

Whenever a Client Switch needs to acquire the Zoning information for a specific Nx_Port, it shall query the nearest Server Switch by using the zone member types used by the current Active Zone Set to identify that Nx_Port. The nearest Server Switch is selected among the set of Server Switches by using the FSPF distance as metric.

A Server Switch receiving a ZIR request shall determine which Zones in the Active Zone Set contain the requested zone members, and reply to the requestor with an SW_ACC containing the list of these Zones.

7.6.5.2.2 ZIR Request Sequence

Protocol: ZIR (Zone Information Request) SW_ILS Request Sequence

Addressing: the S_ID field shall be set to FFFCxxh, indicating the Domain_Controller of the originating Switch, and the D_ID field shall be set to FFFCyyh, indicating the Domain_Controller of the destination Switch.

Payload: The format of the ZIR SW_ILS Request Sequence Payload is shown in table 196.

Table 196 – Zone Information Request SW_ILS Request Payload

Item	Size (Bytes)
4203 0000h	4
Flags	4
Active Zone Set Hash	variable
Zone Member Identifier Entry Length	4
Number of Zone Member Identifier Entries	4
Zone Member Identifier Entry #1	variable
Zone Member Identifier Entry #2	variable
...	
Zone Member Identifier Entry #k	variable
Optional Data	variable

Flags: reserved.

Active Zone Set Hash: the hash of the Active Zone Set.

Zone Member Identifier Entry Length: contains the length in bytes of each Zone Member Entry. This length depends on the zone member types currently used in the Active Zone Set.

Number of Zone Member Identifier Entries: contains the number Zone Member Identifier Entries contained in the payload.

Zone Member Identifier Entry: each Zone Member Identifier Entry is a list of zone members identifying a specific Nx_Port. The zone member format shall be as specified in FC-SW-5. The list shall be sorted in ascending order using the Zone Member type as numeric primary key.

Optional Data: see 7.3.5.

7.6.5.2.3 ZIR Reply Sequence

SW_RJT: SW_RJT shall be sent as a reply to signify the rejection of the ZIR Request Sequence for reasons shown in table 197.

Table 197 – Zone Information Request SW_RJT Reasons

Reason	Reason Code	Reason Code Explanation
ZIR not supported	0Bh	2Ch
Logical Busy	05h	00h

SW_ACC: SW_ACC shall be sent as a reply to signify the acceptance of the ZIR Request Sequence for processing. The format of the ZIR SW_ACC Payload is shown in table 198.

Table 198 – Zone Information Request SW_ACC Payload

Item	Size (Bytes)
0200 0000h	4
Flags	4
Active Zone Set Hash	variable
Zones List Length	4
Zones List	variable
Optional Data	variable

Flags: reserved.

Active Zone Set Hash: the hash of the Active Zone Set.

Zones List Length: contains the length in bytes of the Zones List.

Zones List: contains the subset of the Active Zone Set providing the requested Zoning information. Each Zone shall be formatted as described in FC-SW-5 for the Active Zone Set in Enhanced Zoning.

Optional Data: see 7.3.5.

8 Combinations of Security Protocols

8.1 Entity Authentication Overview

This clause describes how to combine the protocols defined in the previous clauses and in other standards to allow a pair of Fibre Channel entities to negotiate, conduct, and terminate mutually acceptable entity authentication. In this context an entity identifies itself using either:

a) a Name_Identifier; or

b) a Well-known address at which a Generic Service is operating.

The combination of protocols described in this clause is referred to as “entity authentication”.

Upon successful authentication, entity authentication has established the following security relationships for the entities involved:

a) the requirement for using an Authentication Protocol;

b) completion of an Authentication Protocol if required; and

c) other security relationships as established by the Authentication Protocol, that may include:

   A) the requirements for Security Association management;

   B) optionally, a shared secret; and

   C) optionally, an initial Security Association.

Entity authentication also includes process for non-disruptively renewing authentication and security relationships, and for negotiated or unilateral termination of security relationships.

The determination of an acceptable set of security relationships is based on a security policy maintained for each entity. The content of such policies and how they are maintained may be a combination of vendor specific rules and rules specified by this standard.

8.2 Terminology

In the description of entity authentication within this clause, the special terminology introduced here is used.

A Fabric entity is a Fabric, a Switch, or an Fx_Port acting as a participant in entity authentication.

An E_Port entity is an E_Port acting as a participant in entity authentication.

An Nx_Port entity is either an entity that identifies itself using a Name_Identifier associated with an Nx_Port or a Generic Service operating at a Well-known address.

This standard specifies the entity authentication protocol from the perspective of one entity in communication with another entity (see figure 28). Within this standard, the entity for which the protocol is described is called the local entity and the other entity with which the local entity is communicating is called the remote entity. Similarly, the terms local FC_Port, local E_Port, local Nx_Port, and local N_Port_ID refer to objects associated with the local entity, and the terms remote FC_Port, remote E_Port, remote Nx_Port,

and remote N_Port_ID refer to objects associated with the remote entity. Local Fx_Port and remote Fx_Port are not used in this sense because these terms are already otherwise defined in FC-FS-3.

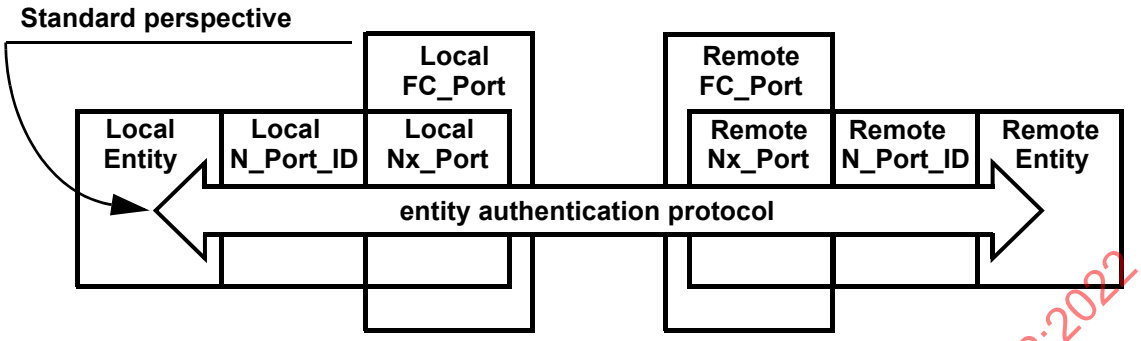

Figure 28 – Entity Authentication Standard Perspective

8.3 Scope of Security Relationships

8.3.1 N_Port_ID Virtualization

A Name_Identifier may be associated with an N_Port by FLOGI or by some forms of FDISC (see FC-LS-2). Either of these actions creates an Nx_Port entity that shall independently exercise entity authentication. Security relationships established for an Nx_Port entity shall not apply to any other Nx_Port entity, regardless of whether the Nx_Port entities are associated with the same Nx_Port.

8.3.2 Nx_Port Entity to a Fabric Entity

Entity authentication between an Nx_Port entity and a Fabric entity shall be conducted between the Nx_Port entity and Well-known address FFFFFEh (i.e., the F_Port Controller). For purposes of entity authentication between an Nx_Port entity and a Fabric entity, the local Fx_Port acts as a proxy for the whole Fabric (i.e., any Fabric action that does not require N_Port Login does not require any further authentication).

A Fabric that requires or allows certain security relationships (e.g., Fabric Login not required to be followed by authentication) for Fabric Login of an Nx_Port entity when connected to an Fx_Port may require or accept different security relationships for the Nx_Port entity when connected to a different Fx_Port.

The security relationships established for an Nx_Port entity by entity authentication with a Fabric entity shall apply to any communication between the Nx_Port entity and the Fabric that is not within the scope of N_Port login. This shall include any ELSs sent without N_Port login and for which either the source or destination is a Well-known address (e.g., a Request Node Identification (RNID) ELS sent to the Fabric Controller (FFFFFDh)).

The security relationships established for an Nx_Port entity by entity authentication with a Fabric entity shall apply to that Nx_Port entity only, even if other Nx_Port entities share the same Nx_Port (i.e., a Fabric that supports multiple N_Port_IDs (see FC-LS-2) requires each Nx_Port to independently maintain security relationships for each N_Port_Name that has been assigned an N_Port_ID at that Nx_Port).

The security relationships established for an Nx_Port entity by entity authentication with a Fabric entity may not apply to communication between the Nx_Port entity and any Generic Service that is provided at a

Well-known Address. A Generic Service provided at a Well-known Address and an Nx_Port entity may conduct authentication independently, or may mutually rely on authentication established at Fabric Login.

NOTE 47 – It is possible to configure a fabric so that an N_Port_Name is required to be authenticated for Fabric Login at some Fx_Ports but not at others. Such configurations are open to denial of service attacks against an Nx_Port entity that has authenticated using the N_Port_Name unless other means of security (e.g., physical enclosure) are provided for the Fx_Ports that allow unauthenticated access, and therefore are discouraged.

8.3.3 Nx_Port Entity to Nx_Port Entity

The security relationships established for an Nx_Port entity by entity authentication to another Nx_Port entity, including a Generic Service, shall apply to all communication between that pair of Nx_Port entities. This shall include all ELSs between the pair of Nx_Port entities while they remain logged in.

The security relationships established for an Nx_Port entity by entity authentication with another Nx_Port entity shall not apply to any other Nx_Port entity, even if the other Nx_Port entity shares the same Nx_Port as either of the Nx_Port entities that conducted entity authentication (i.e., an Nx_Port that supports multiple N_Port_IDs (see FC-LS-2) independently maintains security relationships for each N_Port_Name that has been assigned an N_Port_ID at that Nx_Port).

8.4 Entity Authentication Model

The state machines described in this clause are presented as a model for behavior, not for implementation. An implementation of entity authentication shall display behavior at its external interfaces (i.e., below the FC_Port box in figure 29) equivalent to the state machine set specified here. However, an implementation of entity authentication may not explicitly realize the interfaces, states, events, and transitions of this state machine set.

NOTE 48 – The ordered lists of actions in the state transition definitions in this clause produce the modeled externally observable behavior. Other orderings may produce equivalent behavior and such orderings are not prohibited by this standard.

Clients are abstractions of other entities that have dependencies on services provided by entity authentication. The single client for entity authentication services is referred to as authentication client.

Abstract services (see 8.5) describe interfaces of entity authentication to functions specified elsewhere in this standard or in other standards.

Figure 29 represents a model of entity authentication for an Nx_Port with its clients and services, and with some further components of a Fibre Channel node.

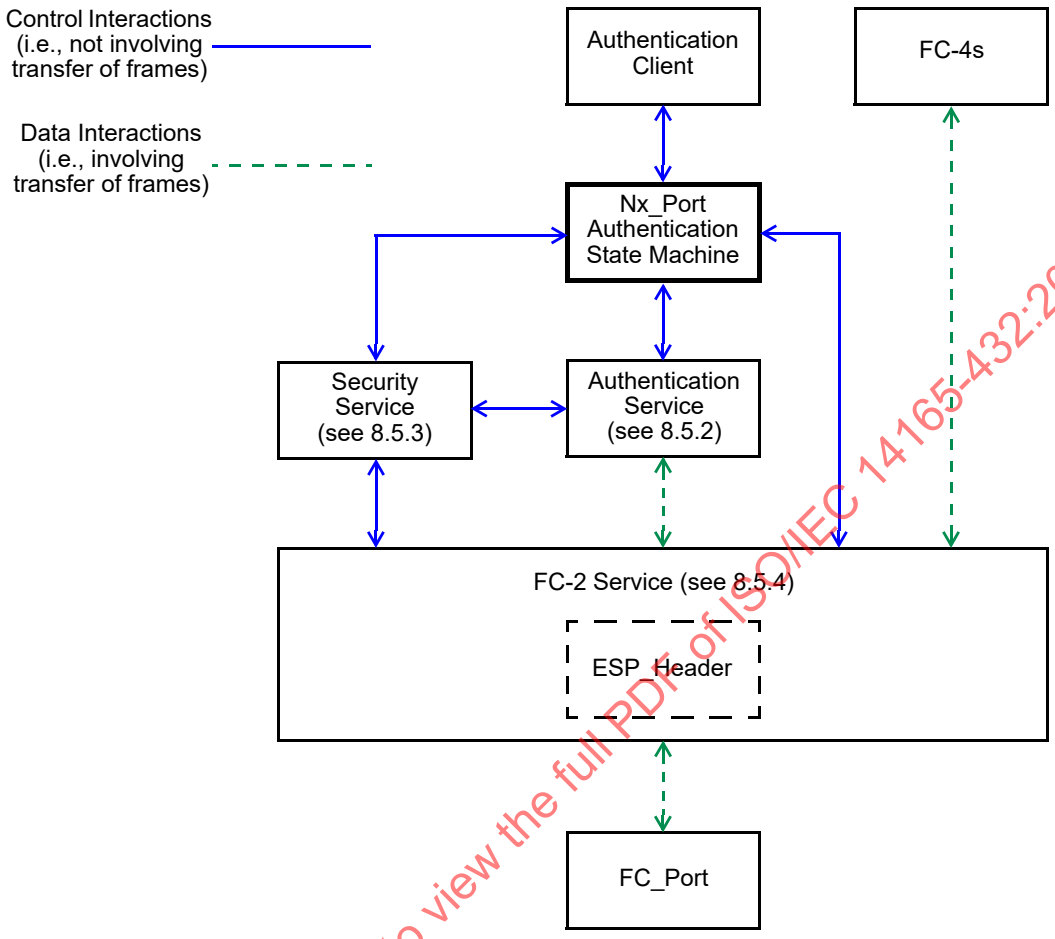

Figure 29 – Entity Authentication Model for an Nx_Port (Informative)

The state machines that specify entity authentication are:

a) Nx_Port to fabric authentication state machine (NFA, see 8.6);

b) fabric from Nx_Port authentication state machine (FNA, see 8.7);

c) Nx_Port to Nx_Port authentication state machine (NNA, see 8.8); and

d) E_Port to E_Port authentication state machine (see 8.9.1).

8.5 Abstract Services for Entity Authentication

8.5.1 Overview

This subclause specifies no formal interfaces. The service interfaces here are abstractions. They are normative only to the extent that as referenced from the state machines for entity authentication, they imply a sequence of actions and events involving an FC-FS-3 FC_Port state machine and/or FC-SW-5 E_Port state machine and the state machines and services of this standard.

8.5.2 Authentication Service

8.5.2.1 Authentication Request

The authentication request shall cause the authentication service to conduct the protocol of AUTH ELSs or SW_ILSs necessary to authenticate a local entity and a remote entity in accord with standard and/or vendor-specific security policy applicable to the pairing of the local entity and the remote entity. The request for authentication identifies whether the local entity is to initiate AUTH_Negotiate or begin an authentication timeout pending arrival of AUTH_Negotiate. On completion of the protocol, the authentication service reports whether authentication has succeeded, has failed for reasons suggesting a threat, or has failed for reasons not suggesting a threat.

8.5.2.2 Abandon Authentication Request

The abandon authentication request shall cause the authentication service to:

a) terminate any communication caused by authentication in progress; and

b) set any internal state to reflect no authentication is in progress,

between the local entity and the remote entity of the entity authentication state machine that made the request.

8.5.2.3 Reauthentication

The authentication service shall conduct reauthentication if:

a) an authentication request (see 8.5.2.1) is made by the state machine for the local entity and the remote entity while it is in the Normal Operation state;

b) an AUTH_Reject message requesting authentication restart arrives while the state machine for the local entity and the remote entity is in the Normal Operation state; or

c) an AUTH_Negotiate message arrives while the state machine for the local entity and the remote entity is in the Normal Operation state.

On completion of the protocol, the authentication service reports whether authentication has succeeded, has failed for reasons suggesting a threat, or has failed for reasons not suggesting a threat.

8.5.2.4 Spurious Traffic

The authentication service may detect a spurious traffic event if any authentication ELS other than an AUTH_Negotiate message or an AUTH_Reject message requesting authentication restart arrives outside of an ongoing authentication transaction. An authentication service that detects a spurious traffic event may send a spurious traffic event for the port pair identified by the S_ID and D_ID of the frame to the entity

authentication state machine, and may log the spurious traffic event or a synopsis of the history of spurious traffic events in a vendor specific way.

8.5.3 Security Service

8.5.3.1 Maintain Security Policy

The security service maintains the policies that determine security behaviors applicable to any pairing of a local entity with a remote entity. These policies may be standard and/or vendor-specific and may be configurable per entity pair by means subject to this standard and/or not subject to this standard.

8.5.3.2 Clear Security Relationships

The clear security relationships request shall cause the security service to terminate all the established security relationships relevant to communication between the pair of entities associated with the calling state machine.

8.5.3.3 IKEv2 Dead Peer

If the Security service detects an IKEv2 dead peer condition (see 6.8.4) between a local Nx_Port entity and a remote Nx_Port entity, it shall report an IKEv2 dead peer event to the state machine.

8.5.4 FC-2 Service

8.5.4.1 Maintain ELS Buffer Condition Requirements

The FC-2 service maintains any ELS buffer condition requirements applicable to any pairing of a local entity with a remote entity.

8.5.4.2 N_Port_ID Assignment Request

The N_Port_ID assignment request shall cause the FC-2 service to attempt fabric login if the local Nx_Port is not logged in with the fabric or shall cause the FC-2 service to request an additional N_Port_ID using FDISC if the local Nx_Port is already logged in with the fabric.

8.5.4.3 N_Port Login Request

The N_Port login request shall cause the FC-2 service to attempt N_Port login of the local Nx_Port entity with the remote Nx_Port entity.

8.5.4.4 Negotiate ELS Buffer Conditions Request

The negotiate ELS buffer conditions request shall cause the FC-2 service to negotiate ELS buffer conditions using the RPBC ELS (see FC-LS-2) for the local entity of the entity authentication state machine that made the request with the remote entity of the entity authentication state machine that made the request. The request identifies whether to initiate or await the RPBC ELS.

8.5.4.5 Explicit Logout Request

The explicit logout request shall cause the FC-2 service to conduct a LOGO ELS protocol for the local entity of the entity authentication state machine that made the request to the remote entity of that entity authentication state machine.

A Fabric may send a LOGO to an Nx_Port (i.e., explicitly logout). An Nx_Port that supports receiving LOGO from the Fabric is able to more quickly recover from security events that are reported in this manner.

After a state machine has requested explicit logout, an authentication client should not request a new login or N_Port_ID assignment from the state machine until the state machine has entered the Noncommunicating state and the authentication client has determined the need to communicate with the remote entity of the state machine. If there is no traffic to send, the login should be delayed in order to reduce potential load on the fabric and/or destination node.

8.5.4.6 Implicit Logout Request

The Implicit logout request shall cause the FC-2 service to implicitly log out the local entity of the entity authentication state machine that made the request from the remote entity of that entity authentication state machine.

After a state machine has requested implicit logout, an authentication client should not request a new login or N_Port_ID assignment from the state machine until the state machine has entered the Noncommunicating state and the authentication client has determined the need to communicate with the remote entity of the state machine. If there is no traffic to send, the login should be delayed in order to reduce potential load on the fabric and/or destination node.

8.5.4.7 Terminate All Communication Request

The terminate all communication request shall cause the FC-2 service to abort all open Exchanges between the local entity and the remote entity of the entity authentication state machine that made the request.

8.5.4.8 Link Initialization Request

The link initialization request shall cause the FC-2 service to perform link initialization for the local FC_Port in accord with FC-FS-3. This in turn shall lead to implicit fabric logout of all the N_Port_IDs assigned to or by the local FC_Port.

8.5.4.9 Disable Request

The Disable request shall cause the FC-2 service to cause the local Nx_Port to be unable to communicate with the remote Nx_Port. A disabled Nx_Port should remain disabled pending vendor-specific outside intervention.

If:

a) either the local FC_Port or the remote FC_Port is an Fx_Port;

b) the local FC_Port is not operating as an L_Port;

c) the local FC_Port does not support additional N_Port_ID assignment; and

d) the local FC_Port does not support Virtual Fabric Tagging,

the local Nx_Port may be disabled by holding a receiver reset condition on the receiver (see FC-FS-3) and holding the transmitter in the not-enabled state (see FC-PI-2), or by refusing to initiate or respond to FLOGI (see FC-FS-3). Otherwise, the local Nx_Port may be disabled by refusing to either initiate or

respond to FLOGI or FDISC or PLOGI with the remote Nx_Port, but shall not be disabled by holding a receiver reset condition on the receiver or holding the transmitter in the not-enabled state.

8.5.4.10 PLOGI Arrival

If the FC-2 service receives a PLOGI from a remote Nx_Port entity for a local Nx_Port entity, the FC-2 service shall report PLOGI arrival to the local entity authentication state machine for the local and remote entity.

8.5.4.11 Login Complete

If the FC-2 service accepts or receives acceptance of a PLOGI between a remote entity and a local entity, the FC-2 service shall report login complete to the local entity authentication state machine for the local and remote entity.

8.5.4.12 N_Port_ID Assignment Complete

If the FC-2 service accepts or receives acceptance of an FLOGI or FDISC between a remote entity and a local entity, the FC-2 service shall report N_Port_ID assignment complete to the local entity authentication state machine for the local and remote entity.

8.5.4.13 Explicit Logout Complete

If the FC-2 service sends an LS_ACC for a LOGO received from a remote entity to a local entity or receives an LS_ACC or LS_RJT for a LOGO sent to a remote entity from a local entity or times out waiting for an LS_ACC for a LOGO sent to a remote entity from a local entity, the FC-2 service shall report explicit logout complete to the local entity authentication state machine for the local and remote entity.

8.5.4.14 Port Logout

If the FC-2 service determines that a remote Nx_Port entity has been implicitly logged out from a local Nx_Port entity, the FC-2 service shall report implicit port logout to the local entity authentication state machine for the local and remote entity.

8.5.4.15 Fabric Logout

If the FC-2 service for a local Nx_Port determines that the local Nx_Port has been implicitly logged out from the fabric, the FC-2 service shall report implicit fabric logout to the local entity authentication state machines for all local Nx_Port entities associated with the local Nx_Port.

If the FC-2 service for a Fabric determines that a remote Nx_Port has been implicitly logged out from the Fabric, the FC-2 service shall report implicit Fabric Logout to the Fabric from Nx_Port Authentication state machines associated with every remote Nx_Port entity at the remote Nx_Port.

8.5.4.16 Link Parameter Change

If the FC-2 service for a local Nx_Port determines the need to change link parameters negotiated by PLOGI (see FC-LS-2) on a point-to-point topology (see FC-FS-3), it shall perform link initialization for the local Nx_Port in accord with FC-FS-3.

NOTE 49 – Link Initialization in accord with FC-FS-3 includes implicit Nx_Port Logout of the local Nx_Port to the remote Nx_Port on the link, so it causes the FC-2 service to report an implicit port logout event to the Nx_Port to Nx_Port authentication state machine, bringing both to expect no security for a subsequent login.

8.5.4.17 Security Change

If the FC-2 service for a local FC_Port discovers a change of identifying information (i.e., address_identifier, Port_Name, Node_Name, Fabric_Name, or security flag) for a remote entity by means other than login or N_Port_ID creation (e.g., the change is discovered by PDISC, or FDISC with nonzero S_ID, or the fabric name server), it shall report a security change to any entity authentication state machine that associates a local entity with the remote entity for which the change of identifying information was discovered.

8.5.4.18 Security Enforcement

If the entity authentication state machine for an entity pair is not in the Normal Operation state, the FC-2 service shall:

a) discard all arriving frames and reject all local requests to originate Exchanges, if both entities in the entity pair are Nx_Ports and the local entity is not in Normal Operation state with a fabric entity;

b) reject all local requests to originate Exchanges except for Exchanges caused by requests from the entity authentication state machine;

c) apply ESP_Header processing to departing frames in accord with established Security Associations;

d) discard any arriving frame that has an ESP_header inconsistent with established Security Associations, and optionally treat it as a spurious traffic event;

e) discard any arriving frame that does not have a Routing field indicating Link_Control, Basic Link Service, or Extended Link Service, and optionally treat it as a spurious traffic event;

f) accept any arriving frame that has a Routing field indicating Link_Control, Basic Link Service, or Extended Link Service, and has an ESP_header that is consistent with established Security Associations;

g) handle an ELS that is required by the entity authentication state machines and services in accord with the entity authentication state machines and services;

h) optionally handle an ELS that is permitted by FC-LS-2 without Nx_Port login, if both entities in the entity pair are Nx_Ports and the local entity is in the Normal Operation state with a fabric entity; and

i) discard an ELS that is neither required by the entity authentication state machines and services nor permitted by FC-LS-2 without Nx_Port login, and optionally treat it as a spurious traffic event.

If the entity authentication state machine for an entity pair is in the Normal Operation state, the FC-2 service shall:

a) accept all local requests to originate Exchanges;

b) apply any security processing required by established Security Associations to departing frames that do not belong to an FLOGI ELS Exchange or ELP Exchange;

c) apply no security processing to departing frames that belong to an FLOGI ELS Exchange or ELP Exchange;

d) apply any frame processing required by the frame header to arriving frames;

e) accept any arriving frames that are protected with an ESP_Header in accord with established Security Associations for the entity pair and do not belong to an FLOGI ELS Exchange or ELP Exchange;

f) accept any arriving frames that belong to an FLOGI ELS Exchange or ELP Exchange and are not protected with an ESP_Header;

g) discard any arriving frames that are not protected with an ESP_Header in accord with established Security Associations for the entity pair and do not belong to an FLOGI ELS Exchange or ELP Exchange, and optionally treat them as spurious traffic events;

h) discard any arriving frames that belong to an FLOGI ELS Exchange or ELP Exchange and are protected with an ESP_Header, and optionally treat them as spurious traffic events; and

i) handle any accepted arriving frames as specified by this subclause, FC-FS-3, and the applicable standards for those FC-4 services that are implemented.

An FC-2 service that detects a spurious traffic event for an arriving frame may send a spurious traffic event to the entity authentication state machine for the port pair identified by the S_ID and D_ID of the frame, and may log the spurious traffic event or a synopsis of the history of spurious traffic events in a vendor specific way.

A Switch shall observe the following behaviors with respect to Nx_Port entities associated with any of the Fx_Ports of the Switch that require Authentication:

a) for an Nx_Port entity that is not in the Normal Operation state with a fabric entity, frame routing shall discard any frames from the N_Port_ID of the Nx_Port entity that do not have Well-known address FFFFFEh as D_ID and shall discard any frames to the N_Port_ID of the Nx_Port entity other than those originated by entity authentication state machines on the Switch;

b) for an Nx_Port entity that is in the Normal Operation state with a fabric entity, frame routing shall attempt to route all frames addressed to or from the N_Port_ID of the Nx_Port entity;

c) after the N_Port_ID of an Nx_Port entity has been sent an FLOGI or FDISC LS_ACC with the security flag set to one, the fabric shall not register its attributes with the Name Server until the Nx_Port entity is in the Normal Operation state with a fabric entity;

d) when an Nx_Port entity reaches Normal Operation state with a Fabric entity, the Fabric should implicitly log out any other Nx_Port in the Fabric that has the same N_Port_Name;

e) An N_Port_ID assigned by an Fx_Port shall not be reassigned by another Fx_Port if the Nx_Port_ID:

   A) was assigned by an FLOGI LS_ACC with the security flag set to one or FDISC LS_ACC with the security flag set to one; and

   B) has not subsequently been explicitly or implicitly logged out from the Fabric;

f) after an Nx_Port entity that has been in the Normal Operation state with a fabric entity transitions from logged in with the fabric to not logged in with the fabric, the fabric shall deregister the N_Port_ID of the Nx_Port entity from the Name Server and send any necessary RSCNs; and

g) after any Nx_Port entity that has been in the Normal Operation state with a fabric entity transitions from logged in with the fabric to not logged in with the fabric, the fabric shall not reassign its address_identifier for a period of R_A_TOV.

8.6 Nx_Port to Fabric Authentication (NFA) State Machine

8.6.1 Overview

An NFA state machine shall be associated with each possible pair of a local Nx_Port entity and a remote fabric entity. An Nx_Port that is configured for entity authentication shall maintain one instance of the NFA state machine with the fabric of the Nx_Port for each possible N_Port_ID it may acquire.

Activity in some states in the NFA authentication state machine may use significant computational resource (e.g., compute time or context memory). A system that is either accidentally or deliberately requested to process several such states concurrently may have insufficient resources for timely processing of some or all of the states. An implementation may ameliorate this by rejecting a request that causes entry to a state when the implementation determines that entry to the state would unacceptably impact performance. The method of rejecting a request for this purpose is specific to the type of request and may not be the subject of this standard. The method of determining when a request should be rejected for this purpose is vendor specific. After an implementation rejects a request to protect performance, it shall make an implicit logout request to the FC-2 service and transition to the noncommunicating state. These transitions are not shown in the state machine specifications.

An implementation may limit the impact of accidental or deliberate prolonged authentication attempts by limiting either the time or the number of ELS requests used to accomplish authentication and login. Whether login timers and/or counters are used, how they are maintained, and the values at which they expire are vendor specific. If login timers and/or counters are used, the NFA state machine indicates:

a) the circumstances when login timers and/or counters shall be started and stopped; and

b) the actions that shall be taken on their expirations.

The NFA state machine shall be specified by the states in 8.6.2, the events in 8.6.3, and the transitions in 8.6.4. Combinations of states and events not described in 8.6.4 shall be handled in accord with any other relevant standards and subclauses of this standard, and the NFA state machine shall cause no action or state change.

Figure 30 provides an overview of the NFA state machine.

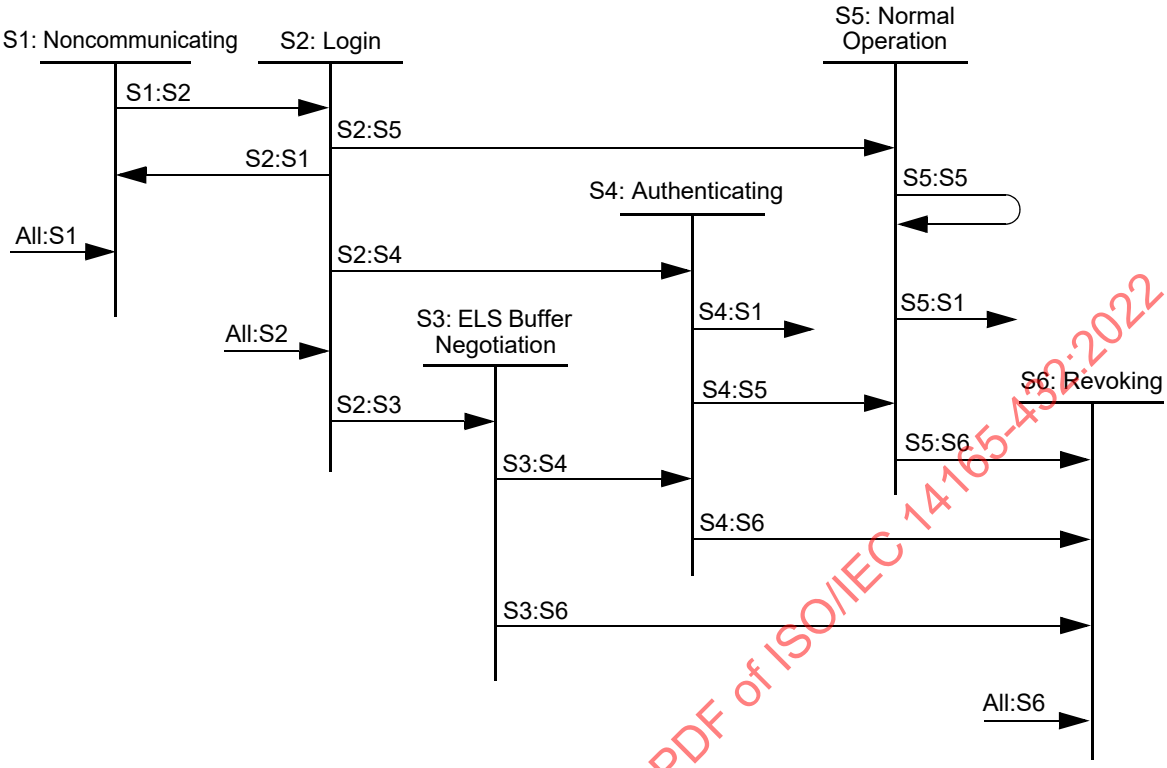

Figure 30 – NFA State Machine

8.6.2 NFA States

S1: Noncommunicating. The local Nx_Port entity is not logged in with the fabric and has no security relationships established with a fabric entity. This state shall be presumed if the local Nx_Port entity has no context concerning a fabric entity.

S2: Login. The FC-2 service is getting an N_Port_ID by either FLOGI or FDISC.

S3: ELS Buffer Negotiation. The FC-2 service is negotiating ELS buffer conditions. This state is not entered for VFT-capable N_Ports because they are expected to have no ELS buffer restrictions.

S4: Authenticating. The authentication service is authenticating the local Nx_Port entity with a fabric entity.

S5: Normal Operation. The local Nx_Port entity has logged in and completed all required authentication with a fabric entity. The Authentication service may conduct an authentication transaction while the NFA state machine remains in the Normal Operation state (i.e., reauthentication), and therefore continues normal communication. If the authentication transaction does not complete successfully, the NFA state machine leaves the Normal Operation state and normal communication may be interrupted. If the local Nx_Port is VFT-capable and the FLOGI ACC sent to the local Nx_Port that led to entry to the Normal Operation state had the Virtual Fabrics bit set to one, the Normal Operation state corresponds to the EVFP state (see FC-LS-2).

S6: Revoking. The local Nx_Port entity is revoking authentication with a fabric entity.

8.6.3 NFA Events

E1: A request for login is received from an authentication client.

E2: Completion of N_Port_ID assignment is reported by the FC-2 service.

E3: Completion of ELS buffer negotiation is reported by the FC-2 service.

E4: Completion of authentication is reported by the authentication service.

E5: A request for explicit logout is received from an authentication client.

E6: Completion of explicit logout is reported by the FC-2 service (i.e., sending or receiving LOGO LS_ACC).

E7: A request for reauthentication is received from an authentication client.

E8: An implicit fabric logout for the local Nx_Port is reported by the FC-2 service.

E9: A security change for the fabric entity is reported by the FC-2 service.

E10: Spurious traffic for the local Nx_Port entity from the fabric entity is reported by the FC-2 service or by the authentication service.

E11: A timeout or counter associated with login expires.

E12: IKEv2 dead peer is reported by the security service.

8.6.4 NFA Transitions

8.6.4.1 All:S1

There are three transitions from any state to state S1: IKEv2 dead peer, logout complete, and violation.

An IKEv2 dead peer transition occurs when in any state except state S1 (i.e., Noncommunicating) or S2 (i.e., Login), event E12 (i.e., IKEv2 dead peer is reported by the security service) occurs. When an IKEv2 dead peer transition occurs, the NFA state machine shall:

1) cancel all timeouts and counters;
2) request the authentication service to abandon any authentication in progress;
3) request the FC-2 service to terminate all communication;
4) request the security service to clear security relationships; and
5) transition to state S1 (i.e., Noncommunicating).

A logout complete transition occurs when in any state except state S1 (i.e., Noncommunicating), if event E6 (i.e., completion of explicit logout is reported by the FC-2 service) or event E8 (i.e., an implicit fabric logout for the local Nx_Port is reported by the FC-2 service) occurs. When a logout complete transition occurs, the NFA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the authentication service to abandon any authentication in progress;

3) request the FC-2 service to terminate all communication;

4) request the security service to clear security relationships; and

5) transition to state S1 (i.e., Noncommunicating).

A violation transition occurs when in any state, event E10 (i.e., spurious traffic for the local Nx_Port entity from the fabric entity is reported by the FC-2 service or by the authentication service) or event E11 (i.e., a timeout or counter associated with login expires) occurs. When a violation transition occurs, the NFA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the fabric;

5) optionally log a possible Nx_Port security policy violation;

6) optionally request the FC-2 service to initialize the link;

7) optionally request the FC-2 service to disable the local Nx_Port; and

8) transition to state S1 (i.e., Noncommunicating).

8.6.4.2 All:S2

Except in state S1 (i.e., Noncommunicating) or state S2 (i.e., Login), if event E1 (i.e., a request for login is received from an authentication client) occurs, then the NFA state machine shall:

1) request the security service to clear security relationships;

2) request the authentication service to abandon authentication;

3) request the FC-2 service to initiate N_Port_ID assignment;

4) if a login timeout or counter is used and not running, start it; and

5) transition to state S2 (i.e., Login).

8.6.4.3 All:S6

Except in state S1 (i.e., Noncommunicating) or state S6 (i.e., Revoking), if event E5 (i.e., a request for explicit logout is received from an authentication client) or event E9 (i.e., a security change for the fabric entity is reported by the FC-2 service) occurs, then the NFA state machine shall

1) request the FC-2 service to initiate explicit fabric logout; and

2) transition to state S6 (i.e., Revoking).

8.6.4.4 S1:S2

In state S1 (i.e., Noncommunicating), if event E1 (i.e., a request for login is received from an authentication client) occurs, then the NFA state machine shall:

1) request the FC-2 service to initiate N_Port_ID assignment;

2) if a login timeout or counter is used, start it; and

3) transition to state S2 (i.e., Login).

8.6.4.5 S2:S1

There are two transitions from state S2 to state S1: login error and violation.

A login error transition occurs when in state S2 (i.e., Login), if event E2 (i.e., completion of N_Port_ID assignment is reported by the FC-2 service) occurs and the N_Port_ID assignment completed with an error indication. When a login error transition occurs, the NFA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the security service to clear security relationships; and

3) transition to state S1 (i.e., Noncommunicating).

A violation transition occurs when in state S2 (i.e., Login), event E2 (i.e., completion of N_Port_ID assignment is reported by the FC-2 service) occurs and:

a) the N_Port_ID assignment completed normally; and

b) the N_Port_ID assignment negotiated to use or not use of authentication contradictory to the Nx_Port security policy determined by the security service.

When a violation transition occurs, then the NFA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the fabric;

5) optionally log a possible Nx_Port security policy violation;

6) optionally request the FC-2 service to initialize the link;

7) optionally request the FC-2 service to disable the local Nx_Port; and

8) transition to state S1 (i.e., Noncommunicating).

8.6.4.6 S2:S3

In state S2 (i.e., Login), if event E2 (i.e., completion of N_Port_ID assignment is reported by the FC-2 service) occurs and:

a) the N_Port_ID assignment completed normally;

b) the N_Port_ID assignment negotiated to operate with authentication;

c) the Nx_Port security policy determined by the security service allows the local Nx_Port entity to authenticate with the fabric entity;

d) the N_Port_ID assignment caused FLOGI; and

e) the FLOGI indicated ELS buffer conditions,

then the NFA state machine shall:

1) request the FC-2 service to initiate ELS buffer negotiation; and

2) transition to state S3 (i.e., ELS Buffer Negotiation).

8.6.4.7 S2:S4

In state S2 (i.e., Login), if event E2 (i.e., completion of N_Port_ID assignment is reported by the FC-2 service) occurs and:

a) the N_Port_ID assignment completed normally;

b) the N_Port_ID assignment negotiated to operate with authentication;

c) the Nx_Port security policy determined by the security service allows the local Nx_Port entity to authenticate with the fabric entity; and

d) either:

A) the N_Port_ID assignment caused FLOGI and the FLOGI indicated no ELS buffer conditions; or

B) the N_Port_ID assignment caused FDISC,

then the NFA state machine shall:

1) request the authentication service to initiate authentication; and

2) transition to state S4 (i.e., Authenticating).

8.6.4.8 S2:S5

In state S2 (i.e., Login), if event E2 (i.e., completion of N_Port_ID assignment is reported by the FC-2 service) occurs and:

a) the N_Port_ID assignment completed normally;

b) the N_Port_ID assignment negotiated to operate without authentication; and

c) the Nx_Port security policy determined by the security service allows the local Nx_Port entity to communicate with the fabric without authentication,

then the NFA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the security service to clear security relationships; and

3) transition to state S5 (i.e., Normal Operation).

8.6.4.9 S3:S4

In state S3 (i.e., ELS Buffer Negotiation), if event E3 (i.e., completion of ELS buffer negotiation is reported by the FC-2 service) occurs and the ELS buffer condition negotiation completed successfully, then the NFA state machine shall:

1) request the authentication service to initiate authentication; and

2) transition to state S4 (i.e., Authenticating).

8.6.4.10 S3:S6

In state S3 (i.e., ELS Buffer Negotiation), if event E3 (i.e., completion of ELS buffer negotiation is reported by the FC-2 service) occurs and the ELS buffer condition negotiation completed with an error indication, then the NFA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to initiate explicit fabric logout; and

3) transition to state S6 (i.e., Revoking).

8.6.4.11 S4:S1

In state S4 (i.e., Authenticating), if event E4 (i.e., completion of authentication is reported by the authentication service) occurs and authentication failed for reasons suggesting a threat, then the NFA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the fabric;

5) optionally log a possible Nx_Port security policy violation;

6) optionally request the FC-2 service to initialize the link;

7) optionally request the FC-2 service to disable the local Nx_Port; and

8) transition to state S1 (i.e., Noncommunicating).

8.6.4.12 S4:S5

In state S4 (i.e., Authenticating), if event E4 (i.e., completion of authentication is reported by the authentication service) occurs and authentication succeeded, then the NFA state machine shall transition to state S5 (i.e., Normal Operation).

8.6.4.13 S4:S6

In state S4 (i.e., Authenticating), if event E4 (i.e., completion of authentication is reported by the authentication service) occurs and authentication failed for reasons not suggesting a threat, then the NFA state machine shall:

1) request the FC-2 service to initiate explicit fabric logout; and

2) transition to state S6 (i.e., Revoking).

8.6.4.14 S5:S1

In state S5 (i.e., Normal Operation), if event E4 (i.e., completion of authentication is reported by the authentication service) occurs and authentication failed for reasons suggesting a threat, then the NFA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the fabric;

5) optionally log a possible Nx_Port security policy violation;

6) optionally request the FC-2 service to initialize the link;

7) optionally request the FC-2 service to disable the local Nx_Port; and

8) transition to state S1 (i.e., Noncommunicating).

8.6.4.15 S5:S5

There are two transitions from state S5 (i.e., Normal Operation) to itself: successful reauthentication and start reauthentication.

A successful reauthentication transition occurs when in state S5 (i.e., Normal Operation), event E4 (i.e., completion of authentication is reported by the authentication service) occurs and authentication succeeded. When a successful reauthentication transition occurs, then the NFA state machine shall remain in state S5 (i.e., Normal Operation).

A start reauthentication transition occurs when in state S5 (i.e., Normal Operation), event E7 (i.e., a request for reauthentication is received from an authentication client) occurs. When a start reauthentication transition occurs, the NFA state machine shall:

1) request the authentication service to initiate authentication; and

2) remain in state S5 (i.e., Normal Operation).

8.6.4.16 S5:S6

In state S5 (i.e., Normal Operation), if event E4 (i.e., completion of authentication is reported by the authentication service) occurs and authentication failed for reasons not suggesting a threat, then the NFA state machine shall:

1) request the FC-2 service to initiate explicit fabric logout; and

2) transition to state S6 (i.e., Revoking).

8.7 Fabric from Nx_Port Authentication (FNA) State Machine

8.7.1 Overview

An FNA state machine shall be associated with each possible pair of a local fabric entity and a remote Nx_Port entity. A fabric that is configured for entity authentication shall maintain one instance of the FNA state machine with each possible N_Port_ID it may assign.

NOTE 50 – Although this requires a number of FNA abstract state machines in a fabric equal to the number of N_Port_IDs that may be assigned, context is not necessary for any N_Port_IDs that have not initiated N_Port_ID assignment (i.e., FLOGI or FDISC).

Activity in some states in the FNA authentication state machine may use significant computational resource (e.g., compute time or context memory). A system that is either accidentally or deliberately requested to process several such states concurrently may have insufficient resources for timely processing of some or all of the states. An implementation may ameliorate this by rejecting a request that causes entry to a state when the implementation determines that entry to the state would unacceptably impact performance. The method of rejecting a request for this purpose is specific to the type of request and may not be the subject of this standard. The method of determining when a request should be rejected for this purpose is vendor specific. After an implementation rejects a request to protect performance, it shall make an implicit logout request to the FC-2 service and transition to the noncommunicating state. These transitions are not shown in the state machine specifications.

An implementation may limit the impact of accidental or deliberate prolonged authentication attempts by limiting either the time or the number of ELS requests used to accomplish authentication and login. Whether login timers and/or counters are used, how they are maintained, and the values at which they expire are vendor specific. If login timers and/or counters are used, the FNA state machine indicates:

a) the circumstances when login timers and/or counters shall be started and stopped; and

b) the actions that shall be taken on their expirations.

The FNA state machine shall be specified by the states in 8.7.2, the events in 8.7.3, and the transitions in 8.7.4. Combinations of states and events not described in 8.7.4 shall be handled in accord with any other relevant standards and subclauses of this standard, and the FNA state machine shall cause no action or state change.

Figure 31 provides an overview of the FNA state machine.

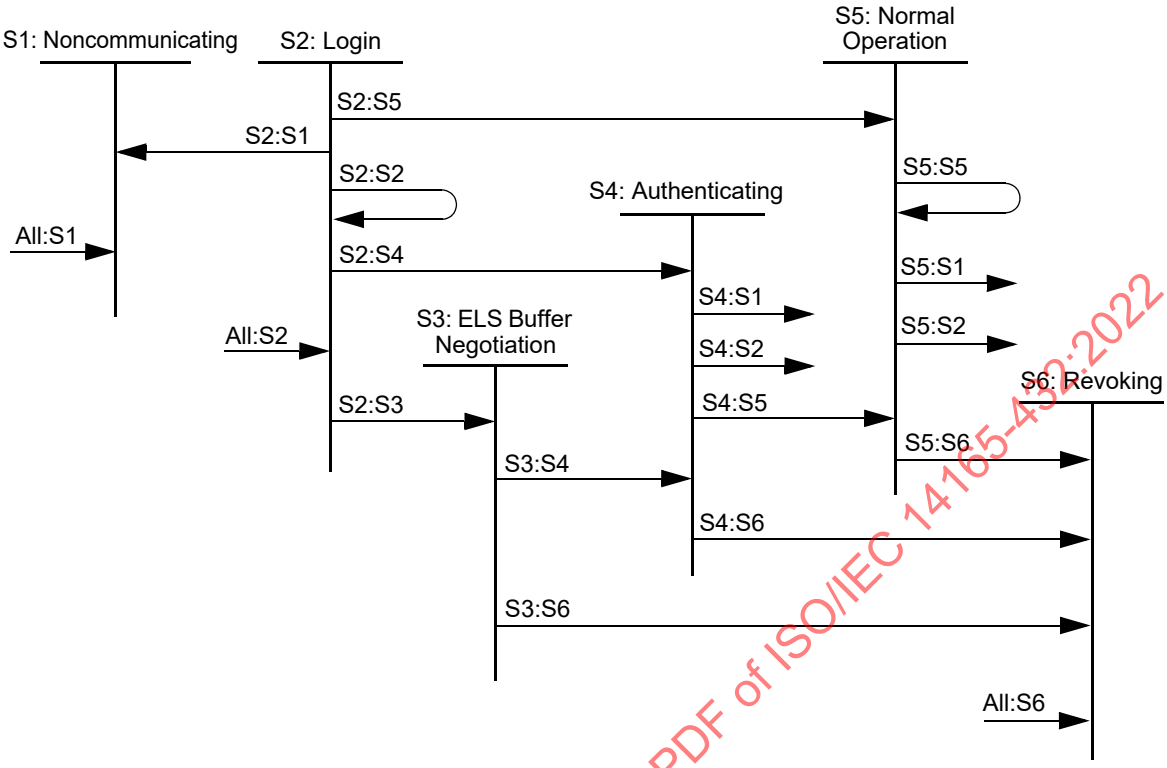

Figure 31 – FNA State Machine

8.7.2 FNA States

S1: Noncommunicating. The Fx_Port is not initialized. This state shall be presumed if the fabric has no context concerning the remote Nx_Port entity.

S2: Login. The FC-2 service is waiting for N_Port_ID assignment completion for the remote Nx_Port entity.

S3: ELS Buffer Negotiation. The FC-2 service is negotiating ELS buffer conditions. This state is not entered for VFT-capable F_Ports because they are expected to have no ELS buffer restrictions.

S4: Authenticating. The authentication service is authenticating the remote Nx_Port entity with the fabric entity.

S5: Normal Operation. The remote Nx_Port entity has acquired its N_Port_ID and completed all required authentication with the fabric entity. The Authentication service may conduct an authentication transaction while the FNA state machine remains in the Normal Operation state (i.e., reauthentication), and therefore continues normal communication. If the authentication transaction does not complete successfully, the FNA state machine leaves the Normal Operation state and normal communication may be interrupted.

S6: Revoking. The fabric entity is revoking authentication with the remote Nx_Port entity.

8.7.3 FNA Events

E1: Switch port initialization (see FC-SW-5) has reached the point where the Switch port is required to process a received FLOGI (see FC-FS-3 and FC-LS-2).

E2: Completion of N_Port_ID assignment for the remote Nx_Port entity is reported by the FC-2 service.

E3: Completion of ELS buffer negotiation with the remote Nx_Port entity is reported by the FC-2 service.

E4: Completion of authentication of the remote Nx_Port entity is reported by the authentication service.

E5: A request for explicit logout of the remote Nx_Port entity is received from an authentication client.

E6: Completion of explicit logout is reported by the FC-2 service.

E7: A request for reauthentication of the remote Nx_Port entity is received from an authentication client.

E8: an implicit fabric logout of the remote Nx_Port entity is reported by the FC-2 service.

E9: A security change for the remote Nx_Port entity is reported by the FC-2 service.

E10: Spurious traffic from the remote Nx_Port entity is reported by the FC-2 service or by the authentication service.

E11: A timeout or counter associated with login expires.

E12: IKEv2 dead peer is reported by the security service.

8.7.4 FNA Transitions

8.7.4.1 All:S1

If event E10 (i.e., spurious traffic from the remote Nx_Port entity is reported by the FC-2 service or by the authentication service) or event E11 (i.e., a timeout or counter associated with login expires) occurs, and Nx_Port security policy specifies that the link is to be initialized upon detection of a threat, then the FNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;
2) request the FC-2 service to terminate all communication;
3) request the security service to clear security relationships;
4) request the FC-2 service to implicitly log out the remote Nx_Port entity;
5) optionally log a possible Nx_Port security policy violation;
6) request the FC-2 service to initialize the link;
7) optionally request the FC-2 service to disable the Fx_Port; and
8) transition to state S1 (i.e., Noncommunicating).

8.7.4.2 All:S2

There are four transitions from all states to state S2: violation, IKEv2 dead peer, link initialized, and logout complete.

A violation transition occurs when in any state, event E10 (i.e., spurious traffic from the remote Nx_Port entity is reported by the FC-2 service or by the authentication service) or event E11 (i.e., a timeout or counter associated with login expires) occurs, and Nx_Port security policy specifies that the link is not to be initialized upon detection of a threat. When a violation transition occurs, the FNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the remote Nx_Port entity;

5) optionally log a possible Nx_Port security policy violation;

6) optionally request the FC-2 service to disable the Fx_Port; and

7) transition to state S2 (i.e., Login).

An IKEv2 dead peer transition occurs when in any state except state S1 (i.e., Noncommunicating) or S2 (i.e., Login), event E12 (i.e., IKEv2 dead peer is reported by the security service) occurs. When an IKEv2 dead peer transition occurs, the FNA state machine shall:

1) cancel all timeouts and counters;

2) request the authentication service to abandon any authentication in progress;

3) request the FC-2 service to terminate all communication;

4) request the security service to clear security relationships; and

5) transition to state S2 (i.e., Login).

A link initialized transition occurs when in any state, event E1 (i.e., ELP completed as an Fx_Port) occurs. When a link initialized transition occurs, the FNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the remote Nx_Port entity;

5) request the FC-2 service to accept N_Port_ID assignment; and

6) transition to state S2 (i.e., Login).

A logout transition occurs when in any state except state S1 (i.e., Noncommunicating) or state S2 (i.e., Login), event E6 (i.e., completion of explicit logout is reported by the FC-2 service) or event E8 (i.e., an implicit fabric logout of the remote Nx_Port entity is reported by the FC-2 service) occurs. When logout transition occurs, the FNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the authentication service to abandon any authentication in progress;

3) request the FC-2 service to terminate all communication;

4) request the security service to clear security relationships;

5) request the FC-2 service to accept N_Port_ID assignment; and

6) transition to state S2 (i.e., Login).

8.7.4.3 All:S6

Except in state S1 (i.e., Noncommunicating) or state S2 (i.e., Login) or state S6 (i.e., Revoking), if event E5 (i.e., a request for explicit logout of the remote Nx_Port entity is received from an authentication client) or event E9 (i.e., a security change for the remote Nx_Port entity is reported by the FC-2 service) occurs, then the FNA state machine shall:

1) request the FC-2 service to initiate explicit logout of the remote Nx_Port entity; and

2) transition to state S6 (i.e., Revoking).

8.7.4.4 S2:S1

In state S2 (i.e., Login), if event E2 (i.e., completion of N_Port_ID assignment for the remote Nx_Port entity is reported by the FC-2 service) occurs and:

a) the N_Port_ID assignment completed normally;

b) the N_Port_ID assignment negotiated to use or not use of authentication contradictory to the Nx-_Port security policy determined by the security service; and

c) Nx_Port security policy specifies that the link is to be initialized upon detection of a threat,

then the FNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the remote Nx_Port entity;

5) optionally log a possible Nx_Port security policy violation;

6) request the FC-2 service to initialize the link;

7) optionally request the FC-2 service to disable the Fx_Port; and

8) transition to state S1 (i.e., Noncommunicating).

8.7.4.5 S2:S2

There are two transitions from state S2 to itself: violation and login error.

A violation transition occurs when in state S2 (i.e., Login), event E2 (i.e., completion of N_Port_ID assignment for the remote Nx_Port entity is reported by the FC-2 service) occurs and:

a) the N_Port_ID assignment completed normally;

b) the N_Port_ID assignment negotiated to use or not use of authentication contradictory to the Nx_Port security policy determined by the security service; and

c) Nx_Port security policy specifies that the link is not to be initialized upon detection of a threat.

When a violation transition occurs, the FNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the remote Nx_Port entity;

5) optionally log a possible Nx_Port security policy violation;

6) optionally request the FC-2 service to disable the Fx_Port; and

7) transition to state S2 (i.e., Login).

A login error transition occurs when in state S2 (i.e., Login), event E2 (i.e., completion of N_Port_ID assignment for the remote Nx_Port entity is reported by the FC-2 service) occurs and the N_Port_ID assignment completed with an error indication. When a login error transition occurs, the FNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the security service to clear security relationships; and

3) remain in state S2 (i.e., Login).

8.7.4.6 S2:S3

In state S2 (i.e., Login), if event E2 (i.e., completion of N_Port_ID assignment for the remote Nx_Port entity is reported by the FC-2 service) occurs and:

a) the N_Port_ID assignment completed normally;

b) the N_Port_ID assignment negotiated to operate with authentication;

c) the Nx_Port security policy determined by the security service allows the remote Nx_Port entity to authenticate with the fabric entity;

d) the N_Port_ID assignment was requested by FLOGI; and

e) the FLOGI indicated ELS buffer conditions,

then the FNA state machine shall:

1) request the FC-2 service to wait for ELS buffer negotiation; and

2) transition to state S3 (i.e., ELS Buffer Negotiation).

8.7.4.7 S2:S4

In state S2 (i.e., Login), if event E2 (i.e., completion of N_Port_ID assignment for the remote Nx_Port entity is reported by the FC-2 service) occurs and:

a) the N_Port_ID assignment completed normally;

b) the N_Port_ID assignment negotiated to operate with authentication;

c) the Nx_Port security policy determined by the security service allows the remote Nx_Port entity to authenticate with the fabric entity; and

d) either:

A) the N_Port_ID assignment was requested by FLOGI and the FLOGI indicated no ELS buffer conditions; or

B) the N_Port_ID assignment was requested by FDISC,

then the FNA state machine shall:

1) request the authentication service to accept authentication; and

2) transition to state S4 (i.e., Authenticating).

8.7.4.8 S2:S5

In state S2 (i.e., Login), if event E2 (i.e., completion of N_Port_ID assignment for the remote Nx_Port entity is reported by the FC-2 service) occurs and:

a) the N_Port_ID assignment completed normally;

b) the N_Port_ID assignment negotiated to operate without authentication; and

c) the Nx_Port security policy determined by the security service allows the remote Nx_Port entity to communicate with the fabric without authentication,

then the FNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the security service to clear security relationships; and

3) transition to state S5 (i.e., Normal Operation).

8.7.4.9 S3:S4

In state S3 (i.e., ELS Buffer Negotiation), if event E3 (i.e., completion of ELS buffer negotiation with the remote Nx_Port entity is reported by the FC-2 service) occurs and the ELS buffer condition negotiation completed successfully, then the FNA state machine shall:

1) request the authentication service to wait for authentication; and

2) transition to state S4 (i.e., Authenticating).

8.7.4.10 S3:S6

In state S3 (i.e., ELS Buffer Negotiation), if event E3 (i.e., completion of ELS buffer negotiation with the remote Nx_Port entity is reported by the FC-2 service) occurs and the ELS buffer condition negotiation completed with an error indication, then the FNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to initiate explicit logout of the remote Nx_Port entity; and

3) transition to state S6 (i.e., Revoking).

8.7.4.11 S4:S1

In state S4 (i.e., Authenticating), if event E4 (i.e., completion of authentication of the remote Nx_Port entity is reported by the authentication service) occurs and authentication failed for reasons suggesting a threat and Nx_Port security policy specifies that the link is to be initialized upon detection of a threat, then the FNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the remote Nx_Port entity;

5) optionally log a possible Nx_Port security policy violation;

6) request the FC-2 service to initialize the link;

7) optionally request the FC-2 service to disable the Fx_Port; and

8) transition to state S1 (i.e., Noncommunicating).

8.7.4.12 S4:S2

In state S4 (i.e., Authenticating), if event E4 (i.e., completion of authentication of the remote Nx_Port entity is reported by the authentication service) occurs and authentication failed for reasons suggesting a threat and Nx_Port security policy specifies that the link is not to be initialized upon detection of a threat, then the FNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the remote Nx_Port entity;

5) optionally log a possible Nx_Port security policy violation;

6) optionally request the FC-2 service to disable the Fx_Port; and

7) transition to state S2 (i.e., Login).

8.7.4.13 S4:S5

In state S4 (i.e., Authenticating), if event E4 (i.e., completion of authentication of the remote Nx_Port entity is reported by the authentication service) occurs and authentication succeeded, then the FNA state machine shall transition to state S5 (i.e., Normal Operation).

8.7.4.14 S4:S6

In state S4 (i.e., Authenticating), if event E4 (i.e., completion of authentication of the remote Nx_Port entity is reported by the authentication service) occurs and authentication failed for reasons not suggesting a threat, then the FNA state machine shall:

1) request the FC-2 service to initiate explicit logout of the remote Nx_Port entity; and

2) transition to state S6 (i.e., Revoking).

8.7.4.15 S5:S1

In state S5 (i.e., Normal Operation), if event E4 (i.e., completion of authentication of the remote Nx_Port entity is reported by the authentication service) occurs and authentication failed for reasons suggesting a threat and Nx_Port security policy specifies that the link is to be initialized upon detection of a threat, then the FNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the remote Nx_Port entity;

5) optionally log a possible Nx_Port security policy violation;

6) request the FC-2 service to initialize the link;

7) optionally request the FC-2 service to disable the Fx_Port; and

8) transition to state S1 (i.e., Noncommunicating).

8.7.4.16 S5:S2

In state S5 (i.e., Normal Operation), if event E4 (i.e., completion of authentication of the remote Nx_Port entity is reported by the authentication service) occurs and authentication failed for reasons suggesting a threat and Nx_Port security policy specifies that the link is not to be initialized upon detection of a threat, then the FNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the remote Nx_Port entity;

5) optionally log a possible Nx_Port security policy violation;

6) optionally request the FC-2 service to disable the Fx_Port; and

7) transition to state S2 (i.e., Login).

8.7.4.17 S5:S5

There are two transitions from state S5 to itself: successful reauthentication and start reauthentication.

A successful reauthentication transition occurs when in state S5 (i.e., Normal Operation), event E4 (i.e., completion of authentication of the remote Nx_Port entity is reported by the authentication service) occurs and authentication succeeded. When a successful reauthentication transition occurs, the FNA state machine shall remain in state S5 (i.e., Normal Operation).

A start reauthentication transition occurs when in state S5 (i.e., Normal Operation), if event E7 (i.e., a request for reauthentication of the remote Nx_Port entity is received from an authentication client) occurs. When a start reauthentication transition occurs, the FNA state machine shall:

1) request the authentication service to initiate authentication; and

2) remain in state S5 (i.e., Normal Operation).

8.7.4.18 S5:S6

In state S5 (i.e., Normal Operation), if event E4 (i.e., completion of authentication of the remote Nx_Port entity is reported by the authentication service) occurs and authentication failed for reasons not suggesting a threat, then the FNA state machine shall:

1) request the FC-2 service to initiate explicit logout of the remote Nx_Port entity; and

2) transition to state S6 (i.e., Revoking).

8.8 Nx_Port to Nx_Port Authentication (NNA) State Machine

8.8.1 Overview

An NNA state machine shall be associated with each possible pair of a local Nx_Port entity and a remote Nx_Port entity. An Nx_Port that is configured for entity authentication shall maintain one instance of the NNA state machine for each possible N_Port_ID it may acquire with every other possible N_Port_ID in the fabric.

NOTE 51 – Although this requires a number of NNA abstract state machines in the Nx_Ports of a fabric proportional to the square of the number of N_Port_IDs that may be assigned, context is not necessary for any N_Port_ID pair that has not initiated N_Port login.

Activity in some states in the NNA authentication state machine may use significant computational resource (e.g., compute time or context memory). A system that is either accidentally or deliberately requested to process several such states concurrently may have insufficient resources for timely processing of some or all of the states. An implementation may ameliorate this by rejecting a request that causes entry to a state when the implementation determines that entry to the state would unacceptably impact performance. The method of rejecting a request for this purpose is specific to the type of request and may not be the subject of this standard. The method of determining when a request should be rejected for this purpose is vendor specific. After an implementation rejects a request to protect performance, it shall make an implicit logout request to the FC-2 service and transition to the noncommunicating state. These transitions are not shown in the state machine specifications.

An implementation may limit the impact of accidental or deliberate prolonged authentication attempts by limiting either the time or the number of ELS requests used to accomplish authentication and login. Whether login timers and/or counters are used, how they are maintained, and the values at which they expire are vendor specific. If login timers and/or counters are used, the NNA state machine indicates:

a) the circumstances when login timers and/or counters shall be started and stopped; and

b) the actions that shall be taken on their expirations.

The NNA state machine shall be specified by the states in 8.8.2, the events in 8.8.3, and the transitions in 8.8.4. Combinations of states and events not described in 8.8.4 shall be handled in accord with any other relevant standards and subclauses of this standard, and the NNA state machine shall cause no action or state change.

Figure 32 provides an overview of the NNA state machine.

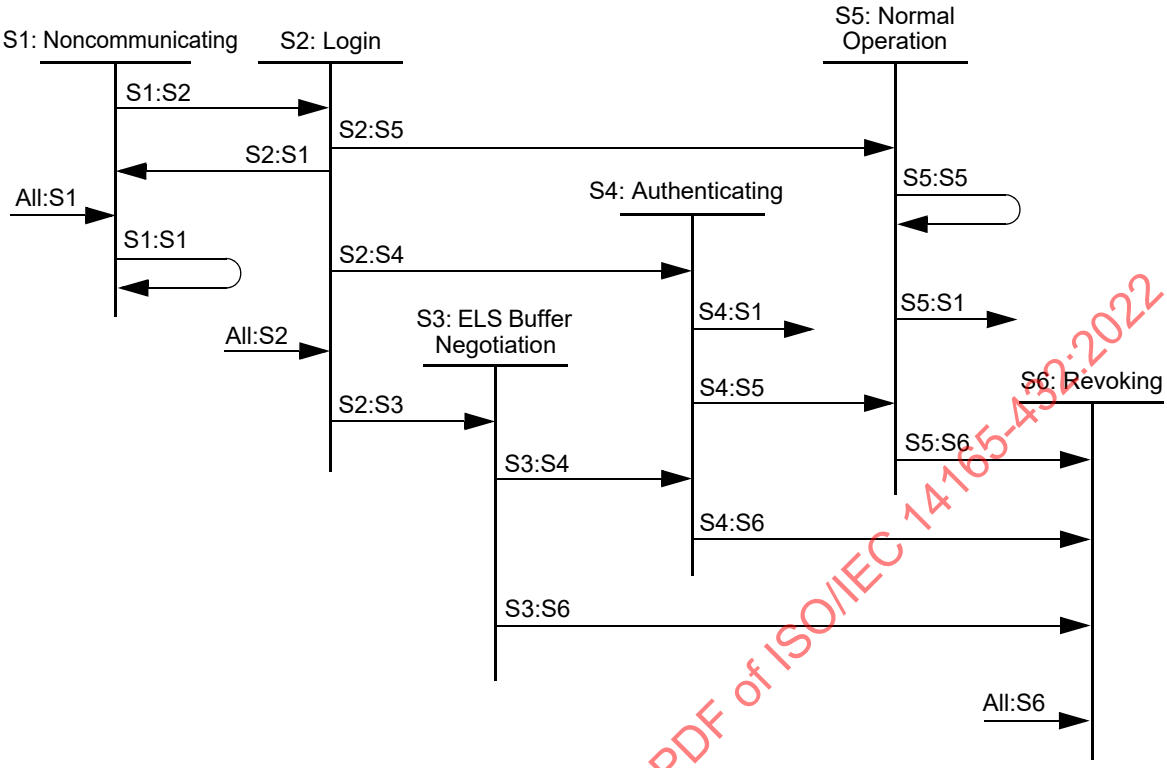

Figure 32 – NNA State Machine

8.8.2 NNA States

S1: Noncommunicating. The local Nx_Port entity is not logged in with the remote Nx_Port entity and has no security relationships established with the remote Nx_Port entity. This state shall be presumed if the local Nx_Port entity has no context concerning the remote Nx_Port entity.

S2: Login. The FC-2 service is waiting for N_Port Login request or reply from the remote Nx_Port entity.

S3: ELS Buffer Negotiation. The FC-2 service is negotiating ELS buffer conditions. This state is not entered for VFT-capable N_Ports because they are expected to have no ELS buffer restrictions.

S4: Authenticating. The authentication service is authenticating the local Nx_Port entity with the remote Nx_Port entity.

S5: Normal Operation. The local Nx_Port entity has logged in and completed all required authentication with the remote Nx_Port entity. The Authentication service may conduct an authentication transaction while the NNA state machine remains in the Normal Operation state (i.e., reauthentication), and therefore continues normal communication. If the authentication transaction does not complete successfully, the NNA state machine leaves the Normal Operation state and normal communication may be interrupted.

S6: Revoking. The local Nx_Port entity is revoking authentication with the remote Nx_Port entity.

8.8.3 NNA Events

E1: A request for login is received from an authentication client.

E2: PLOGI arrival is reported by the FC-2 service.

E3: Completion of N_Port login is reported by the FC-2 service.

E4: Completion of ELS buffer negotiation is reported by the FC-2 service.

E5: Completion of authentication is reported by the authentication service.

E6: A request for explicit logout is received from an authentication client.

E7: Completion of explicit logout is reported by the FC-2 service.

E8: A request for reauthentication is received from an authentication client.

E9: An implicit fabric logout for the local Nx_Port is reported by the FC-2 service.

E10: An implicit logout for the remote Nx_Port is reported by the FC-2 service.

E11: A security change for the remote Nx_Port entity is reported by the FC-2 service.

E12: Spurious traffic for the local Nx_Port entity from the remote Nx_Port entity is reported by the FC-2 service or by the authentication service.

E13: A timeout or counter associated with login expires.

E14: IKEv2 dead peer is reported by the security service.

8.8.4 NNA Transitions

8.8.4.1 All:S1

There are three transitions from all states to state S1: IKEv2 dead peer, logout complete, and violation.

An IKEv2 dead peer transition occurs when in any state except state S1 (i.e., Noncommunicating), event E14 (i.e., IKEv2 dead peer is reported by the security service) occurs. When an IKEv2 dead peer transition occurs, the NNA state machine shall:

1) cancel all timeouts and counters;

2) request the authentication service to abandon any authentication in progress;

3) request the FC-2 service to terminate all communication;

4) request the security service to clear security relationships; and

5) transition to state S1 (i.e., Noncommunicating).

A logout complete transition occurs when in any state except state S1 (i.e., Noncommunicating), event E7 (i.e., completion of explicit logout is reported by the FC-2 service) or event E9 (i.e., an implicit fabric logout for the local Nx_Port is reported by the FC-2 service) or event E10 (i.e., an implicit logout for the remote

Nx_Port is reported by the FC-2 service) occurs. When a logout complete transition occurs, the NNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the authentication service to abandon any authentication in progress;

3) request the FC-2 service to terminate all communication;

4) request the security service to clear security relationships; and

5) transition to state S1 (i.e., Noncommunicating).

A violation transition occurs when in any state except state S1 (i.e., Noncommunicating), event E12 (i.e., spurious traffic for the local Nx_Port entity from the remote Nx_Port entity is reported by the FC-2 service or by the authentication service) or event E13 (i.e., a timeout or counter associated with login expires) occurs. When a violation transition occurs, the NNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the Nx_Port;

5) optionally log a possible Nx_Port security policy violation;

6) optionally request the FC-2 service to initialize the link;

7) optionally request the FC-2 service to disable the local Nx_Port; and

8) transition to state S1 (i.e., Noncommunicating).

8.8.4.2 All:S2

Except in state S1 (i.e., Noncommunicating) or state S2 (i.e., Login), if event E1 (i.e., a request for login is received from an authentication client) or event E2 (i.e., PLOGI arrival is reported by the FC-2 service) occurs, then the NNA state machine shall:

1) request the security service to clear security relationships;

2) request the authentication service to abandon authentication;

3) if login was requested by a local client, request the FC-2 service to initiate N_Port login;

4) If an login timeout or counter is used and not running, start it; and

5) transition to state S2 (i.e., Login).

8.8.4.3 All:S6

Except in state S1 (i.e., Noncommunicating) or state S6 (i.e., Revoking), if event E6 (i.e., a request for explicit logout is received from an authentication client) or event E11 (i.e., a security change for the remote Nx_Port entity is reported by the FC-2 service) occurs, then the NNA state machine shall:

1) request the FC-2 service to initiate explicit Nx_Port logout; and

2) transition to state S6 (i.e., Revoking).

8.8.4.4 S1:S1

In state S1 (i.e., Noncommunicating), if event E12 (i.e., spurious traffic for the local Nx_Port entity from the remote Nx_Port entity is reported by the FC-2 service or by the authentication service) or event E13 (i.e., a timeout or counter associated with login expires) occurs, then the NNA state machine shall:

1) optionally log a possible Nx_Port security policy violation;

2) optionally request the FC-2 service to initialize the link;

3) optionally request the FC-2 service to disable the local Nx_Port; and

4) remain in state S1 (i.e., Noncommunicating).

8.8.4.5 S1:S2

There are two transitions from state S1 to state S2: locally initiated and remotely initiated.

A locally initiated transition occurs when in state S1 (i.e., Noncommunicating), event E1 (i.e., a request for login is received from an authentication client) occurs. When a locally initiated transition occurs, the NNA state machine shall:

1) request the FC-2 service to initiate N_Port login;

2) if a login timeout or counter is used, start it; and

3) transition to state S2 (i.e., Login).

A remotely initiated transition occurs when in state S1 (i.e., Noncommunicating), event E2 (i.e., PLOGI arrival is reported by the FC-2 service) occurs. When a remotely initiated transition occurs, the NNA state machine shall:

1) If a login timeout or counter is used, start it; and

2) transition to state S2 (i.e., Login).

8.8.4.6 S2:S1

There are two transitions from state S2 to state S1: violation and error.

A violation transition occurs when in state S2 (i.e., Login), event E3 (i.e., completion of N_Port login is reported by the FC-2 service) occurs and:

a) the N_Port login completed normally; and

b) the N_Port login negotiated to use or not use authentication in contradiction to the Nx_Port security policy determined by the security service.

When a violation transition occurs, the NNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the Nx_Port;

5) optionally log a possible Nx_Port security policy violation;

6) optionally request the FC-2 service to initialize the link;

7) optionally request the FC-2 service to disable the local Nx_Port; and

8) transition to state S1 (i.e., Noncommunicating).

An error transition occurs when in state S2 (i.e., Login), event E3 (i.e., completion of N_Port login is reported by the FC-2 service) occurs and the N_Port login completed with an error indication. When an error transition occurs, the NNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the security service to clear security relationships; and

3) transition to state S1 (i.e., Noncommunicating).

8.8.4.7 S2:S3

In state S2 (i.e., Login), if event E3 (i.e., completion of N_Port login is reported by the FC-2 service) occurs and:

a) the N_Port login completed normally;

b) the N_Port login negotiated to operate with authentication;

c) the Nx_Port security policy determined by the security service allows the local Nx_Port entity to authenticate with the remote Nx_Port entity; and

d) the PLOGI indicated ELS buffer conditions,

then the NNA state machine shall:

1) if the local entity initiated PLOGI, request the FC-2 service to initiate ELS buffer negotiation;

2) if the remote entity initiated PLOGI, request the FC-2 service to accept ELS buffer negotiation; and

3) transition to state S3 (i.e., ELS Buffer Negotiation).

8.8.4.8 S2:S4

In state S2 (i.e., Login), if event E3 (i.e., completion of N_Port login is reported by the FC-2 service) occurs and:

a) the N_Port login completed normally;

b) the N_Port login negotiated to operate with authentication;

c) the Nx_Port security policy determined by the security service allows the local Nx_Port entity to authenticate with the remote Nx_Port entity; and

d) the PLOGI indicated no ELS buffer conditions,

then the NNA state machine shall:

1) if the local entity initiated PLOGI, request the authentication service to initiate authentication;

2) if the remote entity initiated PLOGI, request the authentication service to accept authentication; and

3) transition to state S4 (i.e., Authenticating).

8.8.4.9 S2:S5

In state S2 (i.e., Login), if event E3 (i.e., completion of N_Port login is reported by the FC-2 service) occurs and:

a) the N_Port login completed normally;

b) the N_Port login negotiated to operate without authentication; and

c) the Nx_Port security policy determined by the security service allows the local Nx_Port entity to communicate with the remote Nx_Port entity without authentication,

then the NNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the security service to clear security relationships; and

3) transition to state S5 (i.e., Normal Operation).

8.8.4.10 S3:S4

In state S3 (i.e., ELS Buffer Negotiation), if event E4 (i.e., completion of ELS buffer negotiation is reported by the FC-2 service) occurs and the ELS buffer condition negotiation completed successfully, then the NNA state machine shall:

1) if the local entity initiated PLOGI, request the authentication service to initiate authentication;

2) if the remote entity initiated PLOGI, request the authentication service to accept authentication; and

3) transition to state S4 (i.e., Authenticating).

8.8.4.11 S3:S6

In state S3 (i.e., ELS Buffer Negotiation), if event E4 (i.e., completion of ELS buffer negotiation is reported by the FC-2 service) occurs and the ELS buffer condition negotiation completed with an error indication, then the NNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to initiate explicit Nx_Port logout; and

3) transition to state S6 (i.e., Revoking).

8.8.4.12 S4:S1

In state S4 (i.e., Authenticating), if event E5 (i.e., completion of authentication is reported by the authentication service) occurs and authentication failed for reasons suggesting a threat, then the NNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the Nx_Port;

5) optionally log a possible Nx_Port security policy violation;

6) optionally request the FC-2 service to initialize the link;

7) optionally request the FC-2 service to disable the local Nx_Port; and

8) transition to state S1 (i.e., Noncommunicating).

8.8.4.13 S4:S5

In state S4 (i.e., Authenticating), if event E5 (i.e., completion of authentication is reported by the authentication service) occurs and authentication succeeded, then the NNA state machine shall transition to state S5 (i.e., Normal Operation).

8.8.4.14 S4:S6

In state S4 (i.e., Authenticating), if event E5 (i.e., completion of authentication is reported by the authentication service) occurs and authentication failed for reasons not suggesting a threat, then the NNA state machine shall:

1) request the FC-2 service to initiate explicit Nx_Port logout; and

2) transition to state S6 (i.e., Revoking).

8.8.4.15 S5:S1

In state S5 (i.e., Normal Operation), if event E5 (i.e., completion of authentication is reported by the authentication service) occurs and authentication failed for reasons suggesting a threat, then the NNA state machine shall:

1) cancel all timers and counters associated with the login and authentication protocols in progress in this state machine;

2) request the FC-2 service to terminate all communication;

3) request the security service to clear security relationships;

4) request the FC-2 service to implicitly log out the Nx_Port;

5) optionally log a possible Nx_Port security policy violation;

6) optionally request the FC-2 service to initialize the link;

7) optionally request the FC-2 service to disable the local Nx_Port; and

8) transition to state S1 (i.e., Noncommunicating).

8.8.4.16 S5:S5

There are two transitions from state 5 to itself: successful reauthentication and start reauthentication.

A successful reauthentication transition occurs when in state S5 (i.e., Normal Operation), event E5 (i.e., completion of authentication is reported by the authentication service) occurs and authentication succeeded. When successful reauthentication transition occurs, the NNA state machine shall remain in state S5 (i.e., Normal Operation).

A start reauthentication transition occurs when in state S5 (i.e., Normal Operation), event E8 (i.e., a request for reauthentication is received from an authentication client) occurs. When start reauthentication transition occurs, the NNA state machine shall:

1) request the authentication service to initiate authentication; and

2) remain in state S5 (i.e., Normal Operation).

8.8.4.17 S5:S6

In state S5 (i.e., Normal Operation), if event E5 (i.e., completion of authentication is reported by the authentication service) occurs and authentication failed for reasons not suggesting a threat, then the NNA state machine shall:

1) request the FC-2 service to initiate explicit Nx_Port logout; and

2) transition to state S6 (i.e., Revoking).

8.9 Additional Security State Machines 8.9.1

E_Port to E_Port Security Checks 8.9.1.1 Overview

FC-SW-5 defines the '**P17:Security Checks**' state in the Switch Port Initialization state machine, state in which E_Port to E_Port security checks are performed. This subclause describes which security checks are performed by a Switch Port while in this state. Figure 33 shows the sub-states within state P17.

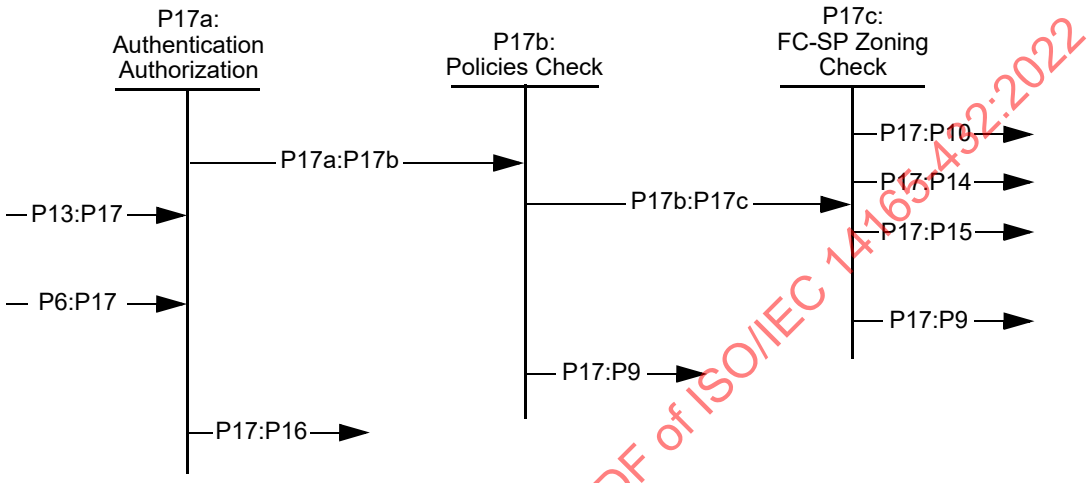

Figure 33 – State P17:Security Checks

FC-SW-5 also defines the state '**P18:Disabled**'. A Switch Port may transition to state P18 from any other state (see FC-SW-5).

8.9.1.2 States

P17a: Authentication and Authorization. In this state an AUTH_ILS transaction (see 5.8) between the two involved Switch Ports, and/or an authorization check to verify if the connection is permitted by the active Fabric Policy (see 7.2.2) may be performed. The identity authenticated and/or authorized shall be the Core Switch_Name or the Switch_Name used during the ELP Exchange (see FC-SW-5).

If Virtual Fabrics are not supported and the AUTH_ILS transaction includes the establishment of a PE_Port to PE_Port Security Association to secure the link traffic via the ESP_Header, then the secured FC frames shall be protected via link-by-link ESP_Header applied to frames with an Enc_Header (see FC-FS-3). If Virtual Fabrics are supported, then the AUTH_ILS transaction should be performed in state P22 (see 8.9.3.1), not in state P17a.

NOTE 52 – If an AUTH_ILS transaction is performed in state P22 (see 8.9.3.1), state P17a is redundant.

P17b: Policies Check. In this state the Check Policy Summary protocol may be processed (see 7.4.2).

P17c: FC-SP Zoning Check. In this state the FC-SP Zoning Check Protocol may be processed (see 7.6.3.3).

NOTE 53 – Each of the above states may also perform no actions.

8.9.1.3 Transitions

P17a:P17b. This transition occurs when the AUTH_ILS transaction or Authorization checks performed in state P17a, if any, completes successfully.

P17b:P17c. This transition occurs when the Check Policy Summary protocol processed in state P17b, if performed, completes successfully.

P6:P17. As defined in FC-SW-5.

P13:P17. As defined in FC-SW-5.

P17:P9. As defined in FC-SW-5.

P17:P10. As defined in FC-SW-5.

P17:P14. As defined in FC-SW-5.

P17:P15. As defined in FC-SW-5.

P17:P16. As defined in FC-SW-5.

8.9.2 B_Port Security Checks

FC-SW-5 defines a state in which to perform E_Port to B_Port Authentication in the Switch Port Initialization state machine. This state is called '**P19:B_Port Security Checks**'. In this state a B_AUTH_ILS transaction (see 5.9) between the involved E_Port and B_Port, and/or an authorization check to verify if the connection is permitted by the active Fabric Policy (see 7.2.2) may be performed. The E_Port identity authenticated and/or authorized shall be the Core Switch_Name or the Switch_Name used during the ELP Exchange (see FC-SW-5).

A B_AUTH_ILS transaction is not expected to establish a Security Association (i.e., it should perform only Authentication and Authorization).

8.9.3 Switch Security Checks with Virtual Fabrics

8.9.3.1 Overview

FC-SW-5 defines a state in which to perform Switch Port to Switch Port Authentication before to perform the EVFP protocol (see FC-SW-5) in the Switch Port Initialization state machine. This state is called '**P22:AUTH_ILS**'. In this state an AUTH_ILS transaction between the involved Switch Ports (see 5.8) shall be performed. The identity authenticated shall be the Core Switch_Name (see FC-SW-5).

If the AUTH_ILS transaction includes the establishment of a PE_Port to PE_Port Security Association to secure the link traffic via the ESP_Header, then the secured FC frames shall be protected via link-by-link ESP_Header applied to frames with a VFT_Header (see FC-FS-3).

FC-SW-5 also defines a state in which to perform VE_Port to VE_Port security checks when Virtual Fabrics are enabled. This state is called '**P24$^{(k)}$:Security Checks**'. This subclause describes which

security checks are performed by a VE_Port while in this state. Figure 34 shows the sub-states within state $P24^{(k)}$.

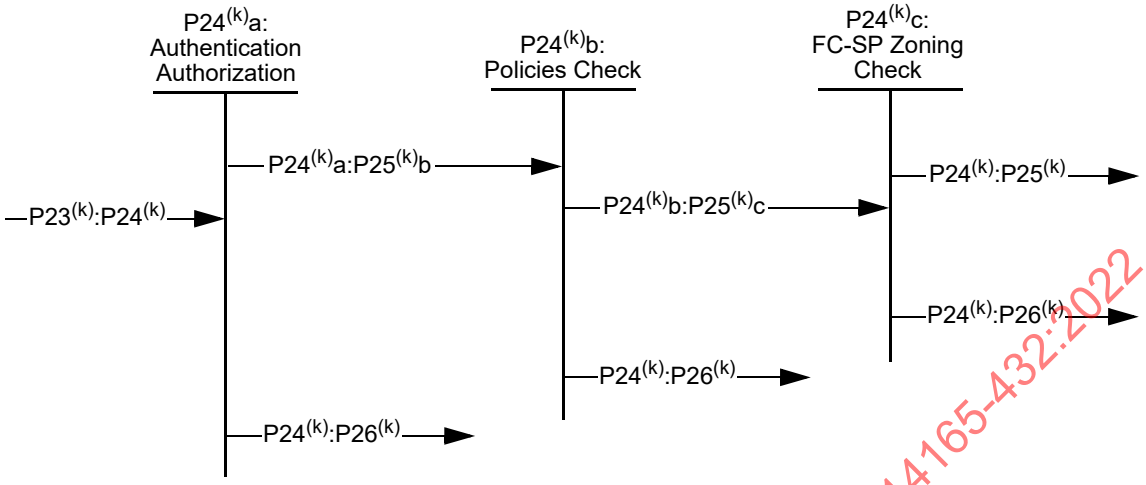

Figure 34 – State $P24^{(k)}$:Security Checks

8.9.3.2 States

$P24^{(k)}$a: Authentication and Authorization. In this state an AUTH_ILS transaction (see 5.8) between the two involved VE_Ports, and/or an authorization check to verify if the connection is permitted by the active Fabric Policy (see 7.2.2) in Virtual Fabric K may be performed. The identity authenticated and/or authorized shall be the Switch_Name of the Virtual Switch associated with Virtual Fabric K.

The AUTH_ILS transaction in this state is not expected to establish a Security Association (i.e., it should perform only Authentication and Authorization).

$P24^{(k)}$b: Policies Check. In this state the Check Policy Summary protocol may be processed (see 7.4.2) in Virtual Fabric K.

$P24^{(k)}$c: FC-SP Zoning Check. In this state the FC-SP Zoning Check Protocol may be processed (see 7.6.3.3) in Virtual Fabric K.

NOTE 54 – Each of the above states may also perform no actions.

8.9.3.3 Transitions

$P24^{(k)}$a:$P24^{(k)}$b. This transition occurs when the AUTH_ILS transaction or Authorization checks performed in state $P24^{(k)}$a, if any, completes successfully.

$P24^{(k)}$b:$P24^{(k)}$c. This transition occurs when the Check Policy Summary protocol processed in state $P24^{(k)}$b, if performed, completes successfully.

$P23^{(k)}$:$P24^{(k)}$. As defined in FC-SW-5.

$P24^{(k)}$:$P25^{(k)}$. As defined in FC-SW-5.

$P24^{(k)}$:$P26^{(k)}$. As defined in FC-SW-5.

8.9.4 N_Port Security Checks with Virtual Fabrics

FC-LS-2 defines a state in which to perform N_Port to F_Port Authentication before the EVFP protocol (see FC-LS-2). This state is called '**P2:AUTH_ELS**'. In this state an AUTH_ELS transaction (see 5.10) between the involved N_Port and F_Port shall be performed. If the two involved FC_Ports negotiated to perform the EVFP protocol (see FC-LS-2) then the identities authenticated shall be the Core Switch_Name for the F_Port (see FC-SW-5) and the Core N_Port_Name or the Node_Name for the N_Port (see FC-FS-3). If the two involved FC_Ports negotiated to not perform the EVFP protocol (see FC-LS-2) then the identities authenticated shall be the Core Switch_Name or the Switch_Name associated with the Port VF_ID for the F_Port (see FC-SW-5) and the Core N_Port_Name or the Node_Name or the N_Port_Name of the VN_Port associated with the Port VF_ID for the N_Port (see FC-FS-3).

If the two involved FC_Ports negotiated to perform the EVFP protocol and the AUTH_ELS transaction includes the establishment of a PN_Port to PF_Port Security Association to secure the link traffic via the ESP_Header, then the secured FC frames shall be protected via link-by-link ESP_Header applied to frames with a VFT_Header (see FC-FS-3). If the two involved FC_Ports negotiated to not perform the EVFP protocol and the AUTH_ELS transaction includes the establishment of a PN_Port to PF_Port Security Association to secure the link traffic via the ESP_Header, then the secured FC frames shall be protected via link-by-link ESP_Header applied to frames with a Enc_Header (see FC-FS-3).

FC-LS-2 also defines a state in which to perform VN_Port to VF_Port Authentication in Virtual Fabric K (see FC-LS-2). This state is called '**P6$^{(k)}$: AUTH_ELS**'. In this state an AUTH_ELS transaction (see 5.10) between the VN_Port and the VF_Port shall be performed. The identities authenticated shall be the Switch_Name associated with Virtual Fabric K for the VF_Port (see FC-LS-2) and the Node_Name or the N_Port_Name of the VN_Port (see FC-FS-3).

The AUTH_ELS transaction in this state is not expected to establish a Security Association (i.e., it should perform only Authentication and Authorization).

8.10 Impact on Other Standards

The security flag value in the Common Service Parameter field of an FDISC and FDISC LS_ACC used to acquire an N_Port_ID for an Nx_Port may differ from the security flag value in the original FLOGI for the Nx_Port and is attended. Attendance to the security flag in the Common Service Parameter field for this purpose takes precedence over the rule in FC-LS-2 that the common service parameters in an FDISC and LS_ACC to an FDISC are ignored.

Entity authentication restricts communication for the local entity during login, authentication, and logout. Unless the local Nx_Port is in the Normal Operation state, only communication pursuant to entity authentication proceeds between the authenticating entities.

NOTE 55 – This constrains communication more tightly than is indicated in the Nx_Port initialization state machines in FC-DA-2.

Annex A: FC-SP-2 Compliance Summary (normative)

A.1 Compliance Elements

A.1.1 Overview

Vendors may implement subsets of this standard. To ensure interoperability, this annex defines some subsets of features as Compliance Elements. Implementations claiming compliance to a certain Compliance Element are expected to interoperate when using the features associated with that Compliance Element. Multiple Compliance Elements may be implemented. Compliance Elements are not security levels.

NOTE 56 – This annex does not cover all functions defined by this standard. Depending on the environment, other functions defined by this standard may be applicable by themselves or in combination with functions for which Compliance Elements are defined.

The features of each Compliance Element are summarized in this annex in the form of Feature Set tables. These tables indicate whether the feature is Required, Allowed or Prohibited for compliance with a specific Compliance Element; or whether a parameter is Required to be a particular value for compliance with a specific Compliance Element. What is not explicitly Required or Prohibited is implicitly Allowed. Table A.1, table A.2 and table A.3 summarize the FC-SP-2 Compliance Elements.

Table A.1 – FC-SP-2 Authentication Compliance Elements

Element	Description	Reference
AUTH-A	DH-CHAP with 2 048 DH algorithm	A.2.1
AUTH-B1	AUTH-A + FCAP	A.2.2
AUTH-B2	AUTH-A + FCPAP	A.2.3
AUTH-B3	AUTH-A + FCEAP	A.2.4

Table A.2 – FC-SP-2 SA Management Compliance Elements

Element	Description	Reference
SA-A	SA Management Protocol	A.3.2
SA-B	AUTH-A + SA Management Protocol	A.3.3
SA-C1	AUTH-B1 + SA Management Protocol	A.3.4
SA-C2	AUTH-B2 + SA Management Protocol	A.3.5
SA-C3	AUTH-B3 + SA Management Protocol	A.3.6

Table A.3 – FC-SP-2 Policy Compliance Elements

Element	Description	Reference
POL-A1	Switch Membership List	A.4.1
POL-A2	IP Management List	A.4.2
POL-A3	Policy Summation ELSs	A.4.3
POL-B3	POL-A1 + POL-A3 + Insistent Domain_IDs	A.4.4

NOTE 57 – In this annex every Compliance Element is described completely, without reference to any other Compliance Element.