



**ANSI C12.22-2008**

***Draft 2008-11-08***

**American National Standard**

**Protocol Specification  
For  
Interfacing to Data Communication Networks**

Secretariat:

**National Electrical Manufacturers Association**

Approved Month DD, 2008

**American National Standards Institute, Inc.**

## **NOTICE AND DISCLAIMER**

The information in this publication was considered technically sound by the consensus of persons engaged in the development and approval of the document at the time it was developed. Consensus does not necessarily mean that there is unanimous agreement among every person participating in the development of this document.

NEMA standards and guideline publications, of which the document contained herein is one, are developed through a voluntary consensus standards development process. This process brings together volunteers and/or seeks out the views of persons who have an interest in the topic covered by this publication. While NEMA administers the process and establishes rules to promote fairness in the development of consensus, it does not write the document and it does not independently test, evaluate, or verify the accuracy or completeness of any information or the soundness of any judgments contained in its standards and guideline publications.

NEMA disclaims liability for any personal injury, property, or other damages of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, application, or reliance on this document. NEMA disclaims and makes no guaranty or warranty, express or implied, as to the accuracy or completeness of any information published herein, and disclaims and makes no warranty that the information in this document will fulfill any of your particular purposes or needs. NEMA does not undertake to guarantee the performance of any individual manufacturer or seller's products or services by virtue of this standard or guide.

In publishing and making this document available, NEMA is not undertaking to render professional or other services for or on behalf of any person or entity, nor is NEMA undertaking to perform any duty owed by any person or entity to someone else. Anyone using this document should rely on his or her own independent judgment or, as appropriate, seek the advice of a competent professional in determining the exercise of reasonable care in any given circumstances. Information and other standards on the topic covered by this publication may be available from other sources, which the user may wish to consult for additional views or information not covered by this publication.

NEMA has no power, nor does it undertake to police or enforce compliance with the contents of this document. NEMA does not certify, test, or inspect products, designs, or installations for safety or health purposes. Any certification or other statement of compliance with any health or safety-related information in this document shall not be attributable to NEMA and is solely the responsibility of the certifier or maker of the statement.

# **AMERICAN NATIONAL STANDARD**

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

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

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

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

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

Published by

**National Electrical Manufacturers Association  
1300 North 17th Street, Rosslyn, VA 22209**

© Copyright 2008 by National Electrical Manufacturers Association

All rights reserved including translation into other languages, reserved under the Universal Copyright Convention, the Berne Convention for the Protection of Literary and Artistic Works, and the International and Pan American Copyright Conventions.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Printed in the United States of America

**This page intentionally left blank.**

## Contents

	Page
<b>1 SCOPE.....</b>	<b>1</b>
<b>2 REFERENCES .....</b>	<b>2</b>
2.1 NORMATIVE.....	2
2.2 OTHERS .....	4
<b>3 DEFINITIONS AND SYNTAX .....</b>	<b>4</b>
3.1 DEFINITIONS .....	4
3.1.1 Absolute UID .....	4
3.1.2 ACSE.....	4
3.1.3 APDU Segment .....	5
3.1.4 Application Association .....	5
3.1.5 Application Context .....	5
3.1.6 Application Entity .....	5
3.1.7 Application Process .....	5
3.1.8 Application Protocol Data Unit (APDU).....	5
3.1.9 ApTitle .....	5
3.1.10 Association.....	5
3.1.11 Bit.....	6
3.1.12 BER.....	6
3.1.13 Byte .....	6
3.1.14 Calling ApTitle.....	6
3.1.15 Called ApTitle.....	6
3.1.16 C12.19 Device.....	6
3.1.17 C12.19 Device Class.....	6
3.1.18 C12.22 Application.....	6
3.1.19 C12.22 Authentication Host.....	6
3.1.20 C12.22 Client.....	7
3.1.21 C12.22 Communication Module .....	7
3.1.22 C12.22 Device.....	7
3.1.23 C12.22 Gateway .....	7
3.1.24 C12.22 Host .....	7
3.1.25 C12.22 Master Relay .....	7
3.1.26 C12.22 Message.....	7
3.1.27 C12.22 Network.....	7
3.1.28 C12.22 Network Segment.....	8
3.1.29 C12.22 Node .....	8
3.1.30 C12.22 Notification Host.....	8
3.1.31 C12.22 Relay .....	8
3.1.32 C12.22 Datagram Segmentation and Reassembly.....	8
3.1.33 C12.22 Server .....	8
3.1.34 Channel.....	8
3.1.35 Cipher .....	8
3.1.36 Cipher, Inverse .....	8
3.1.37 Ciphertext .....	8
3.1.38 Cleartext .....	8
3.1.39 Connection.....	9
3.1.40 Datagram.....	9
3.1.41 EPSEM.....	9
3.1.42 Fragment .....	9

3.1.43	<i>Interface</i> .....	9
3.1.44	<i>Local Port</i> .....	9
3.1.45	<i>Octet</i> .....	9
3.1.46	<i>Other Device</i> .....	9
3.1.47	<i>Plaintext</i> .....	9
3.1.48	<i>PSEM</i> .....	9
3.1.49	<i>Relative UID</i> .....	10
3.1.50	<i>Segment</i> .....	10
3.1.51	<i>Segmentation</i> .....	10
3.1.52	<i>Session</i> .....	10
3.1.53	<i>Transaction</i> .....	10
3.1.54	<i>UID</i> .....	10
3.2	DOCUMENT SYNTAX.....	10
3.3	TABLE SYNTAX.....	11
<b>4</b>	<b>REFERENCE TOPOLOGY</b> .....	<b>11</b>
<b>5</b>	<b>C12.22 NODE TO C12.22 NETWORK SEGMENT DETAILS</b> .....	<b>13</b>
5.1	C12.22 NODE TO C12.22 NETWORK SEGMENT REFERENCE.....	13
5.2	DATA ENCODING RULES.....	13
5.2.1	<i>Data order</i> .....	13
5.2.2	<i>Length Fields Encoding</i> .....	14
5.2.3	<i>Universal Identifiers Encoding</i> .....	14
5.2.4	<i>Universal Identifiers Canonical Encoding</i> .....	15
5.3	LAYER 7 - APPLICATION LAYER.....	16
5.3.1	<i>Data Structure - Utility Industry Data Tables</i> .....	16
5.3.2	<i>EPSEM</i> .....	16
5.3.2.1	Request Codes.....	17
5.3.2.2	Response Codes.....	17
5.3.2.3	Time-out.....	20
5.3.2.3.1	Session Time-out.....	20
5.3.2.3.2	Application Layer Response Time-out.....	20
5.3.2.4	Services.....	21
5.3.2.4.1	Identification Service.....	21
5.3.2.4.2	Read Service.....	23
5.3.2.4.3	Write Service.....	25
5.3.2.4.4	Logon Service.....	27
5.3.2.4.5	Security Service.....	27
5.3.2.4.6	Logoff Service.....	28
5.3.2.4.7	Terminate Service.....	29
5.3.2.4.8	Disconnect Service.....	29
5.3.2.4.9	Wait Service.....	30
5.3.2.4.10	Registration Service.....	31
5.3.2.4.11	Deregistration Service.....	37
5.3.2.4.12	Resolve Service.....	37
5.3.2.4.13	Trace Service.....	39
5.3.2.5	Service sequence state control.....	39
5.3.2.6	Partial Table access using index/element-count Method.....	41
5.3.2.7	Partial Table access using offset/octet-count method.....	43
5.3.3	<i>EPSEM Envelope Structure</i> .....	44
5.3.4	<i>Association Control - Association Control Service Element (ACSE)</i> .....	45
5.3.4.1	Application Context Element (A1 <sub>H</sub> ).....	46
5.3.4.2	Called AP Title Element (A2 <sub>H</sub> ).....	46
5.3.4.3	Calling AP Title Element (A6 <sub>H</sub> ).....	47
5.3.4.4	Universal Identifier of Called and Calling AP Title Element (06 <sub>H</sub> ).....	47
5.3.4.5	Relative Universal Identifier of Called and Calling AP Title Element (80 <sub>H</sub> ).....	47
5.3.4.6	Calling Application Entity Qualifier Element (A7 <sub>H</sub> ).....	47
5.3.4.7	Mechanism Name Element (8B <sub>H</sub> ).....	49

5.3.4.8	Calling Authentication Value Element (AC <sub>H</sub> ).....	49
5.3.4.8.1	C12.22 Security Mechanism (<application-context-oid>.2.1) .....	51
5.3.4.8.2	C12.21 Security Mechanism (<application-context-oid>.2.0) .....	53
5.3.4.8.3	C12.22 Other Security Mechanisms.....	55
5.3.4.9	Called AP Invocation ID Element (A4 <sub>H</sub> ) .....	55
5.3.4.10	Calling AP Invocation ID Element (A8 <sub>H</sub> ).....	56
5.3.4.11	User Information Element (BE <sub>H</sub> ) .....	57
5.3.4.12	Use of Subbranches of a Registered ApTitle.....	59
5.3.4.13	C12.22 Security Mechanism .....	61
5.3.4.13.1	C12.22 Security Mechanism (<application-context-oid>.2.1) .....	62
5.3.5	<i>Application Segmentation Sub-layer</i> .....	68
5.3.5.1	APDU Segmentation.....	69
5.3.5.2	APDU Segment.....	69
5.3.5.2.1	Called AE Qualifier Element (A3 <sub>H</sub> ) .....	69
5.3.5.2.2	Segment User Information Element (BE <sub>H</sub> ).....	70
5.3.5.2.2.1	Segment Association Information Element.....	70
5.3.5.2.2.2	Segment Data Elements .....	70
5.3.5.3	The Segmentation and Reassembly.....	71
5.3.5.3.1	The Segmentation Algorithm .....	71
5.3.5.3.2	The Reassembly Algorithm .....	72
5.4	LAYER 6 - PRESENTATION LAYER.....	73
5.5	LAYER 5 - SESSION LAYER .....	73
5.6	LAYER 4 - TRANSPORT LAYER.....	73
5.7	LAYER 3 - NETWORK LAYER .....	73
5.8	LAYER 2 - DATA LINK LAYER .....	74
5.9	LAYER 1 - PHYSICAL LAYER.....	74
<b>6</b>	<b>PROTOCOL DETAILS: C12.22 DEVICE TO C12.22 COMMUNICATION MODULE INTERFACE</b> .....	<b>75</b>
6.1	INTERFACE ARCHITECTURE .....	75
6.2	INTERFACE DIAGRAM.....	75
6.3	IMPLEMENTATION GUIDELINES.....	76
6.3.1	<i>C12.22 Communication Module</i> .....	76
6.3.2	<i>C12.22 Device</i> .....	77
6.4	LAYER 7 - APPLICATION LAYER.....	77
6.5	LAYER 6 - PRESENTATION LAYER.....	77
6.6	LAYER 5 - SESSION LAYER .....	78
6.7	LAYER 4 - TRANSPORT LAYER.....	78
6.7.1	<i>Negotiate Service</i> .....	78
6.7.2	<i>Get Configuration Service</i> .....	80
6.7.3	<i>Link Control Service</i> .....	82
6.7.4	<i>Send Message Service</i> .....	84
6.7.5	<i>Get Status Service</i> .....	86
6.7.6	<i>Get Registration Status Service</i> .....	87
6.7.7	<i>Service Time Sequence Diagrams</i> .....	88
6.7.8	<i>Service Sequence States</i> .....	91
6.8	LAYER 3 - NETWORK LAYER .....	93
6.9	LAYER 2 - DATA LINK LAYER.....	93
6.9.1	<i>Basic Data Information</i> .....	94
6.9.1.1	Fixed Settings.....	94
6.9.1.2	Variable Settings.....	94
6.9.2	<i>Packet Definition</i> .....	94
6.9.3	<i>CRC Selection</i> .....	96
6.9.4	<i>Acknowledgment</i> .....	97
6.9.5	<i>Retry Attempts</i> .....	97
6.9.6	<i>Timeouts</i> .....	97
6.9.6.1	Traffic Time-out.....	97
6.9.6.2	Inter-character Time-out .....	97

6.9.6.3	Response Time-out .....	97
6.9.7	Turn Around Delay .....	98
6.9.8	Collision .....	98
6.9.9	Duplicate Packets .....	98
6.9.10	Transparency .....	98
6.9.11	Supervision of the Communications Link .....	98
6.9.12	Local Routing .....	99
6.9.13	Service Sequence States .....	100
6.10	LAYER 1 - PHYSICAL LAYER .....	101
6.10.1	Signal Definition .....	101
6.10.2	Electrical Properties of Connection .....	101
6.10.3	Mechanical and Environmental Properties .....	102
6.10.4	Supervision of the Communications Link .....	103
<b>7</b>	<b>LOCAL PORT COMMUNICATION PROTOCOL DETAILS .....</b>	<b>104</b>
7.1	PROTOCOL DEFINITION .....	104
7.1.1	Layer 7 - Application Layer .....	104
7.1.2	Layer 6 - Presentation Layer .....	104
7.1.3	Layer 5 - Session Layer .....	104
7.1.4	Layer 4 - Transport Layer .....	104
7.1.5	Layer 3 - Network Layer .....	105
7.1.6	Layer 2 - Data Link Layer .....	105
7.1.7	Layer 1 - Physical Layer .....	105
7.2	C12.22 LOCAL PORT COMMUNICATION USING A C12.18 OPTICAL PORT .....	105
7.2.1	Establishment of ANSI C12.18 Protocol Compatibility Mode .....	106
7.2.2	Establishment of ANSI C12.22 Protocol Compatibility Mode .....	106
<b>8</b>	<b>BACKWARD COMPATIBILITY .....</b>	<b>107</b>
<b>9</b>	<b>COMPLIANCE .....</b>	<b>108</b>
<b>ANNEX A - RELAYS .....</b>	<b>109</b>	
A.1	HIERARCHICAL TOPOLOGY .....	109
A.2	C12.22 MASTER RELAYS .....	109
A.3	REGISTRATION NOTIFICATION .....	110
A.4	REGISTRATION ALGORITHM DETAILS .....	110
A.5	C12.22 NODE AP TITLE AUTO-ASSIGNMENT .....	110
A.6	C12.22 MASTER RELAY AP TITLE AUTO-ASSIGNMENT .....	111
A.7	OBSOLETE ROUTES .....	111
A.8	MULTIPLE ROUTES .....	111
A.9	APPLICATION LAYER SUPERVISION .....	111
A.10	ROUTING .....	111
<b>ANNEX B - ROUTING EXAMPLES .....</b>	<b>113</b>	
B.1	C12.22 RELAYS WITH A SINGLE SERVICE PROVIDER .....	113
B.2	C12.22 RELAYS SHARED BY MULTIPLE SERVICE PROVIDERS .....	113
<b>ANNEX C - MODIFICATIONS AND EXTENSIONS TO C12.19-1997 .....</b>	<b>115</b>	
C.1	DECADE 12: NODE NETWORK CONTROL TABLES .....	116
	TABLE 120 Dimension Network Table .....	116
	TABLE 121 Actual Network Table .....	120
	TABLE 122 Interface Control Table .....	123
	TABLE 123 Exception Report Configuration Table .....	126
	TABLE 124 Filtering Rules Table .....	128
	TABLE 125 Interface Status Table .....	130



TABLE 126 Registration Status Table.....	135
TABLE 128 Network Statistics Table.....	138
C.2 DECADE 130 - RELAY CONTROL TABLES.....	140
TABLE 130 Dimension Relay Table.....	140
TABLE 131 Actual Relay Table.....	142
TABLE 132 Registration List Table.....	143
TABLE 133 Static Routing Table.....	146
TABLE 134 Host Notification Table.....	148
TABLE 135 Master Relay Assignment Table.....	150
TABLE 136 Dynamic Routing Report Table.....	151
C.3 UNIVERSAL ID PATTERN DESCRIPTION OF AP TITLES .....	152
C.4 ADDITIONS TO TABLE 07 - PROCEDURE INITIATE TABLE .....	153
PROCEDURE 23 Register .....	153
PROCEDURE 24 Deregister.....	153
PROCEDURE 25 Network Interface Control.....	153
PROCEDURE 26 Exception Report.....	154
C.5 TABLE 46: EXTENDED KEY TABLE .....	156
C.6 TABLE 47 HOST ACCESS SECURITY TABLE .....	158
<b>ANNEX D - UNIVERSAL IDENTIFIER.....</b>	<b>162</b>
<b>ANNEX E - ONE-WAY DEVICES.....</b>	<b>164</b>
<b>ANNEX F - APDU RESPONSE TIMEOUT ALGORITHM.....</b>	<b>166</b>
<b>ANNEX G - COMMUNICATION EXAMPLE .....</b>	<b>167</b>
EXAMPLE #1: UNSECURED SESSION.....	167
EXAMPLE #2: UNSECURED SESSIONLESS .....	168
EXAMPLE #3: UNSECURED NOTIFICATION.....	169
EXAMPLE #4: AUTHENTICATED SESSION .....	169
EXAMPLE #5: AUTHENTICATED SESSIONLESS .....	171
EXAMPLE #6: AUTHENTICATED NOTIFICATION .....	172
EXAMPLE #7: ENCRYPTED SESSION .....	173
EXAMPLE #8: ENCRYPTED SESSIONLESS.....	177
EXAMPLE #9: ENCRYPTED NOTIFICATION .....	178
<b>ANNEX H - CRC EXAMPLES.....</b>	<b>180</b>
H.1 TRACE .....	180
H.2 CRC CODE EXAMPLE .....	181
<b>ANNEX I - THE EAX' CRYPTOGRAPHIC MODE .....</b>	<b>182</b>
I.1 EAX' DESCRIPTION .....	182
I.2 JUSTIFICATIONS FOR SELECTION OF EAX RATHER THAN CCM .....	187
I.3 JUSTIFICATIONS FOR THE EAX' OPTIMIZATIONS .....	188
I.4 EAX' C CODE EXAMPLE .....	191
I.4 AES C CODE EXAMPLE .....	195
<b>ANNEX J – CONNECTIONLESS-ACSE-1 EQUIVALENT REDUCED SYNTAX FOR C12.22 MESSAGE TRANSMISSION .....</b>	<b>200</b>

**Foreword** (This Foreword is not part of American National Standard C12.22-2008.)

This Standard is another in the series of communications protocols that describe how to transport Tables (defined in ANSI C12.19, "Utility Industry End Device Data Tables"). Because this Standard describes a protocol that operates over networks, it is necessarily more complex than the simple point-to-point protocols defined in ANSI C12.18 and ANSI C12.21, but the committee has done as much as practical to smooth the transition from those earlier standards.

This Standard describes three different but related uses. One is the operation of the protocol over the network that all C12.22 Nodes implement. The second is an optionally exposed point-to-point interface between a C12.22 Device, e.g., a meter, and, a C12.22 Communications Module, e.g., a network adaptor. The third is the capture, translation and transmission of one way device messages (blurts).

This division was chosen to foster interoperability among communications modules and meters. Suggestions for improvement to this Standard are welcome. They should be sent to:

National Electrical Manufacturers Association  
Vice President of Engineering  
1300 North 17th Street  
Suite 1752  
Rosslyn, VA 22209

This Standard was processed and approved for submittal to ANSI by Accredited Standards Committee for Electricity Metering C12. At the time the committee approved this Standard, the C12 Committee had the following members:

**Tom Nelson, Chairman**  
**Paul Orr, Secretary**

Ed Beroset  
Ron Breschini  
Curt Crittenden  
David Ellis  
Cruz Gomez  
Bob Hughes  
Lawrence Kotewa  
Francis Marta  
John McEvoy  
Herman Millican  
James Mining  
Avygdor Moise  
Tim Morgan  
Roy Moxley  
D. Young Nguyen  
Lauren Pananen  
Aaron Snyder  
Richard Tucker  
John Voisine  
Scott Weikel

Working Group 1 of Subcommittee 17 that developed the Standard consisted of:

**Ed Berozet, Chairman**  
**Richard Tucker, Vice Chairman**  
**Michel Veillette, Editor**

Michael Anderson  
Martin Burns  
Janice Jennings  
Lawrence Kotewa  
Avygdor Moise  
Vuong Nguyen  
Terry Penn  
Bin Qiu  
Chris Schafer  
Aaron Snyder  
Virginia Zinkowski

**This page intentionally left blank.**

## **Protocol Specification For Interfacing To Data Communication Networks**

### **1 Scope**

Initially, communications with electronic devices consisted of transporting memory data via proprietary protocols that were unique to each manufacturer. The desire for interoperability and support for multiple manufacturers by reading and programming systems created a need for standardization of data formats and transport protocols.

The first step was to standardize data formats. Internal data was abstracted as a set of Tables. A set of standard Table contents and formats were defined in ANSI C12.19, "Utility Industry End Device Data Tables".

In the "Protocol Specification for ANSI Type 2 Optical Port" (ANSI C12.18) Standard, a point-to-point protocol was developed to transport table data over an optical connection. The ANSI C12.18 protocol included an application language called Protocol Specification for Electric Metering (PSEM) that allowed applications to read and write Tables. The "Protocol Specification for Telephone Modem Communication" (ANSI C12.21) was then developed to allow devices to use PSEM to transport Tables over telephone modems.

This Standard extends on the concepts of the ANSI C12.18, ANSI C12.19 and the ANSI C12.21 standards to allow transport of Table data over any reliable networking communications system. Note that in this use of the word, "reliable" means that for every message sent, the sender receives a response at its option: either a positive acknowledgement or an error message. That is, messages cannot fail silently in a reliable network (see discussion of Reliable Stream Transport Service in [IPPA : 1995]).

In addition, this Standard describes an optionally exposed point-to-point interface between a C12.22 Device and a C12.22 Communications Module designed to attach to "any" network.

Futhermore, this Standard defines a methodology to capture, translate and transmit one way device messages (blurts).

This Standard defines interfaces between ANSI C12.19 Devices and network protocols.

Specific goals identified by the committee in the creation of this Standard were:

1. Defining a Datagram that may convey ANSI C12.19 data Tables through any network.

This was accomplished by:

- Assuming that the data source is ANSI C12.19 data Tables.
- Defining the Application Layer services (language).

2. Providing a full stack definition for interfacing a C12.22 Device to a C12.22 Communication Module.

This was accomplished by:

- Defining the physical interface requirements between the C12.22 Device and the C12.22 Communication Module.
- Defining the interface lower layers; 4 (transport), 3 (network), 2 (data link) and 1 (physical).

3. Providing a full stack definition for point-to-point communication to be used over local ports such as optical ports, or modems.

This was accomplished by defining a Layer 4 (transport) and Layer 2 (data link).

4. Providing support for efficient one-way messaging (blurts).

This was accomplished by:

- Defining a compact message format that can be easily transformed to a standard ANSI C12.22 Datagram.
  - Assuring that all needed layers defined in this Standard can support one-way messaging
5. Providing network architecture compatible with this protocol. (Some architectural concepts were derived from [HCCS 1: 1987, HCCS 2: 1987, HCCS 3: 1988, DND : 1993, IPPA : 1995, TCPCE : 1997].)

This was accomplished by:

- Defining different type of nodes such as C12.22 Relay, C12.22 Master Relay, C12.22 Host, C12.22 Authentication Host, C12.22 Notification Host, and C12.22 Gateway.
  - Defining the role and responsibilities of each of these C12.22 Nodes.
6. Providing data structure definitions in support of this protocol.

This was accomplished by:

- Defining an ANSI C12.19 Decade to be used by C12.22 Nodes.
- Defining an ANSI C12.19 Decade to be used by C12.22 Relays.
- Defining new procedures in support of this protocol.
- Defining a new Table for enhanced security.

## 2 References

### 2.1 Normative

ANSI C12.18-1996	Protocol Specification for ANSI Type 2 Optical Port.
ANSI C12.19-1997	Utility Industry End Device Data Tables.
ANSI C12.21-1999	Protocol Specification for Telephone Modem Communication.
IEEE C37.90.1-2002	IEEE Standard for Surge Withstand Capability (SWC) Tests for Relays and Relay Systems Associated with Electric Power Apparatus.
IEEE C62.41-2002	IEEE recommended practice on surge voltages in low-voltage AC power circuits.
ISO/IEC 7498-1	Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model.
ISO/IEC 13239:2002	Information Technology - Telecommunications and information exchange between systems - High-level data link control (HDLC) procedures - Frame Structure, Annex A, Explanatory Notes On Implementation of the Frame Checking Sequence.
ANSI INCITS 92	Data Encryption Algorithm.

EAX 2003	Authenticated Encryption with Associated Data (AEAD) algorithm designed to simultaneously protect both authentication and privacy of messages, as described in "A Conventional Authenticated-Encryption Mode", M. Bellare, P. Rogaway and D. Wagner, April 13, 2003, available from <a href="http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/eax/eax-spec.pdf">http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/eax/eax-spec.pdf</a> , and described in [EAX MO 2004].
EAX MO 2004	The EAX Mode of Operation, A Two-Pass Authenticated-Encryption Scheme Optimized for Simplicity and Efficiency, M. BELLARE, P. ROGAWAY, and D. WAGNER, January 18 2004, available from <a href="http://www.cs.ucdavis.edu/~rogaway/papers/eax.pdf">http://www.cs.ucdavis.edu/~rogaway/papers/eax.pdf</a> .
FIPS Pub 197	Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, US Department of Commerce/N.I.S.T, Springfield, Virginia, November 26, 2001. Available from <a href="http://csrc.nist.gov/">http://csrc.nist.gov/</a> .
NIST SP800-38A	Recommendation for Block Cipher Modes of Operation; methods and techniques. NIST Special Publication 800-38A 2001 Edition. US Department of Commerce/N.I.S.T, Springfield, Virginia, December 2001. Available from <a href="http://csrc.nist.gov/publications/nistpubs/800-38A/sp800-38A.pdf">http://csrc.nist.gov/publications/nistpubs/800-38A/sp800-38A.pdf</a> .
NIST SP 800-38B	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. NIST Special Publication 800-38B 2001 Edition. US Department of Commerce/N.I.S.T, Springfield, Virginia, May 2005. Available from <a href="http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf">http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf</a> .
ISO/IEC 8824-1:2002	Information technology - Abstract Syntax Notation One (ASN.1): Specification of basic notation.
ISO/IEC 8824-2:2002	Information technology - Abstract Syntax Notation One (ASN.1): Information Object Specification.
ISO/IEC 8824-3:2002	Information technology - Abstract Syntax Notation One (ASN.1): Constraint specification.
ISO/IEC 8824-4:2002	Information technology Abstract Syntax Notation One (ASN.1): Parameterization of ASN.1 specifications.
ISO/IEC 8825-1:2002	Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER).
ISO/IEC 8650-1:1996	Information Technology – Open Systems Interconnection – Connection-Oriented Protocol for the Association Control Service Element: Protocol Specification.
ISO/IEC 15954:1999	Information technology – Open Systems Interconnection – Connection-mode protocol for the Application Service Object Association Control Service Element.
ISO/IEC 15955:1999	Information technology – Open Systems Interconnection – Connectionless protocol for the application service object Association control service.
ISO/IEC 10035-1:1995	Information Technology – Open Systems Interconnection – Connectionless Protocol for the Association Control Service Element: Protocol Specification

ISO/IEC 646: 1991	ASCII character set.
ATIS T1.667-1999	ATIS T1.667-2002 Intelligent Network (Revision of T1.667-1999): May 2002.
NIST 800-38A -2001	Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation, Methods and Techniques, 2001.

## 2.2 Others

FOLDOC: 2006	Free Online Dictionary of Computing; <a href="http://foldoc.org/">http://foldoc.org/</a> (retrieved on 2 May 2006).
HCCS 1: 1987	Handbook of computer-communications standards; Vol. 1: the open systems interconnection (OSI) model and OSI-related standards, W. Stallings, Macmillan Publishing Co., Inc, 1987. ISBN: 0-02-948071-X.
HCCS 2: 1987	Handbook of computer communications standards, Vol. 2: local network standards, W. Stallings, Macmillan Publishing Co., Inc, 1987. ISBN: 0-02-948070-1.
HCCS 3: 1988	Handbook of computer-communications standards. Vol. 3: Department of Defense (DoD) protocol standards, W. Stallings, Macmillan Publishing Co., Inc, 1988. ISBN: 0-02-948072-8.
DND : 1993	Data Network Design: Packet-Switching Frame Relay 802.6\DQDB SMDS, ATM B-ISDN, SONET, Darren L. Spohn, McGraw-Hill Companies, 1993. ISBN: 0-07-060360-X.
IPPA : 1995	Internetworking with TCP/IP Vol.1: Principles, Protocols, and Architecture, C. Douglas, Prentice Hall, 1995 (3 <sup>rd</sup> edition) ISBN: 0-13-216987-8, 2000 (4 <sup>th</sup> Edition), ISBN: 0-13-018380-6.
OGUSPTO: 1976	Official Gazette of the United States Patent and Trademark Office (9434 O.G. 452 and 949 O.G. 1717), Aug 31, 1976.
TCPCE : 1997	TCP/IP Clearly Explained, Pete Loshin, Academic Press Limited, 1997 (2 <sup>nd</sup> Edition), ISBN: 0-12-455835-6.

## 3 Definitions And Syntax

### 3.1 Definitions

For the purposes of this Standard, the following definitions and terms are used.

#### 3.1.1 Absolute UID

Absolute encoding of a UID according to "Encoding of an object identifier value" clause in ISO/IEC 10035-1.

#### 3.1.2 ACSE

Association Control Service Element encoded per "Connectionless protocol for the association control service element: Protocol specification", ISO/IEC 10035-1. Connectionless ACSE defines the Datagram



encapsulation protocol used in this Standard.

### **3.1.3 APDU Segment**

An Application Protocol Data Unit that is constructed using C12.22 Segmentation as the process of breaking a C12.22 Datagram into smaller units before transmission, see section 3.1.32 "C12.22 Datagram Segmentation and Reassembly".

### **3.1.4 Application Association**

See "Association".

### **3.1.5 Application Context**

A set of service elements and supporting information used on the Application Association. It includes a description of the relationships and dependencies of the C12.22 Application service elements, and a description of the logical structure of information to be exchanged between co-operating Application Entities.

### **3.1.6 Application Entity**

The system-independent application activities that are made available as application services to the application agent; e.g., a set of application service elements that together perform all or part of the communication aspects of an application process. [ATIS T1.667-1999].

### **3.1.7 Application Process**

See "Application Entity".

### **3.1.8 Application Protocol Data Unit (APDU)**

A Datagram that is transferred error-free between network nodes. The C12.22 standard encodes APDUs using ACSE to carry EPSEM services and C12.19 payloads between C12.22 Nodes.

### **3.1.9 ApTitle**

In addition to the addressing constructs (transport address and possibly session and presentation selectors), the communicating application entities have names - application-entity titles (AeTitle). These are carried by ACSE as two fields - the Application-process titles (ApTitle) and the application-entity qualifier (AeQualifier).

C12.22 ApTitles may be encoded absolutely or relatively. Relative UIDs shall be unique only within the context of a C12.22 Network or inside C12.22 Nodes. C12.22 Relays, C12.22 Communication Modules or C12.22 Transport Layers shall map Relative UIDs to other Relative UIDs or Absolute UIDs to ensure the uniqueness of the ApTitle within the context of any network, a C12.22 Network, or a C12.22 Network Segment as needed.

### **3.1.10 Association**

A cooperative relationship among peer (utilizing the same protocol) Application Entities that enables the communication of information and the coordination of their joint operation for an instance of communication. This relationship may be formed by the transfer of application protocol control information during the establishment of a connection, or transitionally, during a single invocation through a connectionless service. Associations can also be predefined and longstanding.

This Standard provides for Application Entities that communicate interactively using connectionless communication, and it provides for state information that is shared between them for the duration of the communication.

### **3.1.11 Bit**

A Binary Digit. The unit of information of a computational quantity that can take on one of two values, such as false and true or 0 and 1. A bit is said to be "set" if its value is true or 1, and "reset" or "clear" if its value is false or 0. One speaks of setting and clearing bits. To toggle or "invert" a bit is to change it, either from 0 to 1 or from 1 to 0.

### **3.1.12 BER**

Basic Encoding Rules as defined by ISO/IEC 8825-1, describing methods used to identify and encode data elements for transport.

### **3.1.13 Byte**

A group of 8 bits of data. When expressed in this Standard, bit 0 is the least significant bit and it is written at the right-most position of a bit sequence. Bit 7 is the most significant bit and it is written at the left-most position of the bit sequence. The actual bit-signal transmission sequence and bit polarity, which represents ones or zeroes, is determined by the characteristics of the appropriate OSI Physical Layer used to transmit the byte.

### **3.1.14 Calling ApTitle**

The Calling ApTitle is encoded as an Absolute UID or Relative UID. It uniquely identifies the source of an ACSE C12.22 Message.

### **3.1.15 Called ApTitle**

The Called ApTitle is encoded as an Absolute UID or Relative UID. It uniquely identifies the target of an ACSE C12.22 Message.

### **3.1.16 C12.19 Device**

A C12.22 Node that contains Tables.

### **3.1.17 C12.19 Device Class**

A Relative UID that uniquely identifies a C12.19 Device Table set per the MANUFACTURER field defined in Table 0 of ANSI C12.19-1997 or the DEVICE\_CLASS field defined by version 2 of ANSI C12.19.

### **3.1.18 C12.22 Application**

An Application Entity that implements a set of services and procedures as defined in this Standard permitting one or more well-defined devices (C12.22 Host, C12.22 Relay, C12.22 Device, C12.22 Communication Module, etc.) to interact within the framework of a C12.22 Network. It may also contain C12.19 Tables.

### **3.1.19 C12.22 Authentication Host**

A C12.22 Host that is an authoritative administrative host for a registering C12.22 Node in the C12.22 Master Relay domain. The C12.22 Authentication Host may be embedded inside a C12.22 Master Relay or it may be a separate C12.22 Node on the network. There may be one or more C12.22 Authentication

Hosts operating under the domain of a single C12.22 Master Relay. Registration with C12.22 Master Relays can only succeed if at least one C12.22 Authentication Host accepts registration on behalf of a C12.22 Node by a C12.22 Master Relay.

### **3.1.20 C12.22 Client**

A C12.22 Node which initiates a Logon service request for the purpose of establishing a session with a C12.22 Server.

### **3.1.21 C12.22 Communication Module**

Hardware module that attaches a C12.22 Device to a C12.22 Network Segment. A C12.22 Communication Module can be physically located inside or outside the C12.22 Device enclosure. However, it is physically and logically distinct from the C12.22 Device. The interface between the C12.22 Communication Module and the C12.22 Device is completely defined by this Standard. The combination of a C12.22 Device and a C12.22 Communication module is a C12.22 Node. If a C12.22 Communication Module contains Tables, it is also a C12.22 Node.

### **3.1.22 C12.22 Device**

A module that hosts C12.22 Application(s) and provides at least one Interface to a C12.22 Communication Module.

### **3.1.23 C12.22 Gateway**

A C12.22 Node that translates the ANSI Standard C12.22 protocol to/from other protocols. Gateways are required when a C12.22 Node needs to communicate with non-C12.22 Nodes. C12.22 Gateways can be attached directly to the non-C12.22 Devices or they can provide their translation services through any network segment.

### **3.1.24 C12.22 Host**

A C12.22 Node that may be a C12.22 Authentication Host or C12.22 Notification Host or both. A Host typically runs on a computer instead of within an embedded system.

### **3.1.25 C12.22 Master Relay**

A C12.22 Relay that operates at the top of a hierarchy of relays. It provides registration services of all devices in its domain. It is also responsible for issuing registration service queries to C12.22 Authentication Hosts and Deregistration service requests and notifications to C12.22 Notification Hosts when registering a C12.22 Node. A C12.22 Master Relay can also act as a C12.22 Host.

### **3.1.26 C12.22 Message**

Any notice, service request, service response or device status sent from one C12.22 Node to another C12.22 Node for the purpose of communication across a C12.22 Network. The detailed encoding of C12.22 Messages is defined by the appropriate encoding rules of the OSI Layer from which they are issued.

### **3.1.27 C12.22 Network**

A C12.22 communication infrastructure that is composed of one or more C12.22 Network Segments. A C12.22 Network shall include at least one C12.22 Master Relay.

### **3.1.28 C12.22 Network Segment**

A collection of C12.22 Nodes that can communicate with each other without forwarding messages through a C12.22 Relay. C12.22 Network Segments are interconnected using C12.22 Relays.

### **3.1.29 C12.22 Node**

A point on the network that attaches to a C12.22 Network Segment. C12.22 Nodes contain one or more C12.22 Applications. Each C12.22 Node shall have a unique ApTitle on a C12.22 Network.

### **3.1.30 C12.22 Notification Host**

A C12.22 Host, which contains an application that needs to be notified when C12.22 Nodes are registered for the first time ("first" here means an actual registration request as contrasted with the reuse of the register service as keep-alive) or deregistered. Each C12.22 Notification Host may add the registered C12.22 Node to its active client list for subsequent processing by the C12.22 Host application.

### **3.1.31 C12.22 Relay**

A C12.22 Node that provides address resolution, Datagram segmentation and optionally message forwarding services to other C12.22 Nodes. Address resolution services consist of mapping Layer 7 addresses (ApTitle) to lower layer addresses (Network Entity Title).

### **3.1.32 C12.22 Datagram Segmentation and Reassembly**

The process of breaking a C12.22 Datagram into smaller units before transmission and then reassembling it into the proper order at the receiving C12.22 Node. C12.22 Datagrams are made smaller specifically because of specified packet size restrictions in a given path across a channel. The transport protocol determines the size of the smallest maximum Protocol Data Unit (PDU) supported by the underlying C12.22 Network Segment for the purpose of transmission to the target C12.22 Node.

### **3.1.33 C12.22 Server**

A C12.22 Node that is a recipient of a Logon service request from a C12.22 Client for the purpose of establishing a session with that client.

### **3.1.34 Channel**

A single path for transmitting signals, usually in distinction from other parallel paths. Multiple channels may coexist on the same physical media. The term channel may signify either a one-way path (providing transmission in one direction only), or a two-way path (providing transmission in two directions).

### **3.1.35 Cipher**

A Cipher is an algorithm for performing encryption.

### **3.1.36 Cipher, Inverse**

A Cipher is an algorithm for performing decryption.

### **3.1.37 Ciphertext**

Ciphertext is the data output from the Cipher or input to the Inverse Cipher.

### **3.1.38 Cleartext**

Cleartext is data sent without transformation even when privacy is being used.

### **3.1.39 Connection**

A logical and physical binding between two or more users of a service.

### **3.1.40 Datagram**

A self-contained, independent entity of application data carrying sufficient information to be routed from the source Application Layer to the destination Application Layer. This Standard encapsulates each Datagram as one or more ACSE PDUs. For more discussion on Datagram, see [HCCS 3: 1988 p3, 8-9, 14-15, 26-52]).

### **3.1.41 EPSEM**

Extended PSEM; Extended structures and services enabling transportation of multiple requests and responses at the same time for use by devices such as gas, water, electricity, and related electronic modules. There are also provisions for response control and C12.19 Device Class identification. EPSEM messages are encapsulated within ACSE PDUs.

### **3.1.42 Fragment**

See "APDU Segment".

### **3.1.43 Interface**

The C12.22 Device hardware components used to manifest a C12.22 Node on a C12.22 Network Segment.

### **3.1.44 Local Port**

A physical interface that is directly attached to the C12.22 Node, or a physical interface that is located in the immediate vicinity of the C12.22 Node and attached to it by means of a dedicated short signal path (e.g. cable). The main purpose of the Local Port is to provide direct access to the application process of the C12.22 Node. The C12.22 Node application process may redirect C12.22 Messages that originate from a Local Port to other Local Ports or other C12.22 Node interfaces. Similarly, the C12.22 Node application process may redirect incoming C12.22 Messages to Local Ports. The C12.22 Communication Module interface of a C12.22 Device is not a Local Port. All Local Ports of a C12.22 Node shall access to the same C12.22 Application. The physical Local Port characteristics (OSI Layer 1) are not defined by this Standard; however, the Data Link through Application Layers (Layers 2-7) are fully defined by this Standard.

### **3.1.45 Octet**

See Byte.

### **3.1.46 Other Device**

A device that does not implement the ANSI C12.22 protocol.

### **3.1.47 Plaintext**

Plaintext is data input to the Cipher or output from the Inverse Cipher. This should not be confused with Cleartext, to which a Cipher is not applied.

### **3.1.48 PSEM**

See ANSI C12.18.

### **3.1.49 Relative UID**

Relative encoding of a UID according to “Encoding of a relative object identifier value” in ISO/IEC 10035-1. The Relative UID is a subset of an Absolute UID, e.g. the absolute object identifier 2.100.3.8571.3.2 contains the relative object identifiers 8571.3.2.

### **3.1.50 Segment**

In the context of a C12.22 Network, a C12.22 Network Segment. In the context of a C12.22 APDU, an APDU Segment.

### **3.1.51 Segmentation**

See 3.1.32 “C12.22 Datagram Segmentation and Reassembly”.

### **3.1.52 Session**

An association context that is maintained while the two C12.22 Nodes are communicating back and forth in a conversation of some duration. Managing a session includes setting up and taking down the connection between two communicating C12.22 Nodes. Some session associations last only long enough to send a message in one direction. However, other session associations may last longer, usually with one or both of the communicating parties able to terminate it.

### **3.1.53 Transaction**

A unit of interaction that occurs individually and coherently.

### **3.1.54 UID**

Universal Identifiers (UIDs) are universally unique identifiers that are encoded using BER. A UID may be formulated as an Absolute UID or a Relative UID and can be used to specify C12.22 object identifiers such as Calling ApTitle, Called ApTitle.

## **3.2 Document Syntax**

Document syntax is identical to that described in ANSI C12.18.

Describing data definitions is usually accomplished within the confines of a given language and the grammar rules of that language. Since the data definitions embodied within this document are meant to be independent of specific language and, hopefully, capable of being implemented within the confines of any language, a method for describing the data definitions must be developed. A modified form of the Backus-Naur Form (BNF) serves as the basis for building the descriptions used to construct the data definitions.

The modified form of BNF has the following definitions:

#### **Symbol Meaning**

- |                  |  |
|------------------|--|
| <b>&lt; &gt;</b> | A string contained inside angle brackets is called a non-terminal. That is, while it may be viewed as a single unit it can and should be redefined as consisting of one or more simpler elements.  |
| <b>::=</b>       | This symbol is read as “is defined as.” The non-terminal that occurs on the left hand side (LHS) of this symbol consists of the elements (non-terminals, terminals, or a combination of the two) found on the right hand side (RHS). A line containing an LHS, ::, and an RHS is known as a production rule. |

- |      The vertical bar is an “OR” symbol. The OR symbol always occurs on the right hand side of a production where the left hand side can be defined in more than one way. The OR bar separates valid alternative right hand sides.
- [ ]    A symbol enclosed in square brackets is optional. The production is valid whether or not it is included.
- \*      The superscript asterisk is known as the Kleene star. A symbol followed by the Kleene star may occur zero or more times without violating the grammar.
- +      The superscript plus sign is known as the Kleene cross. A symbol followed by the Kleene cross must occur one or more times.
- +n     A symbol followed by the Kleene cross and any superscript number “n” represents “n” occurrences of the symbol.
- { }    The curly braces are used to enclose comments within the descriptions. Comments have no impact on the productions.

### 3.3 Table syntax

Table document form syntax is identical to that described in ANSI C12.19 version 2.0.

## 4 Reference Topology

This Standard defines components of a C12.22 Network. Each component is defined with enough flexibility to allow multiple components to be incorporated into one physical device. Figure 4.1 describes the C12.22 Network and protocol topology within which C12.22 Nodes are expected to operate.

This network topology accommodates interconnections among C12.22 Nodes that can be located on the same or on different networks. C12.22 Messages are forwarded between C12.22 Network Segments using C12.22 Relays.

The network topology also accommodates C12.22 Gateways that translate the C12.22 protocol to other protocols. C12.22 Devices and non-C12.22 Devices may be collocated on the same C12.22 Network Segment and optionally provide C12.22 Gateway functionality. The blocks labeled Other Device on the topology diagram are logical constructs and may physically include one or more devices (e.g. a non-C12.22 network of devices).

The interface between a C12.22 Device and C12.22 Communication Module(s) is fully defined in this Standard. However, the Standard general definition of the interface of a C12.22 Node to a C12.22 Network Segment is limited to the Application, Presentation, and Session Layers only (layers 7-5). The interface between a C12.22 Device and a C12.22 Communication Module (Layers 1-4) is shown in Figure 4.1 using triple lines.

In the case that a C12.22 Node is connected to more than one C12.22 Network Segment, communication through those segments shall be to the same C12.22 Application(s). Access to the C12.22 Node through a Local Port shall also be to the same C12.22 Application(s).

When a Local Port is attached to a C12.22 Device, this port provides access to the C12.22 Application(s) of this C12.22 Device and may optionally provide access to the attached C12.22 Communication Modules. Similarly, when a Local Port is attached to a C12.22 Communication Module, this port provides access to the C12.22 Application(s) of this C12.22 Communication Module and may optionally provide access to the attached C12.22 Device.

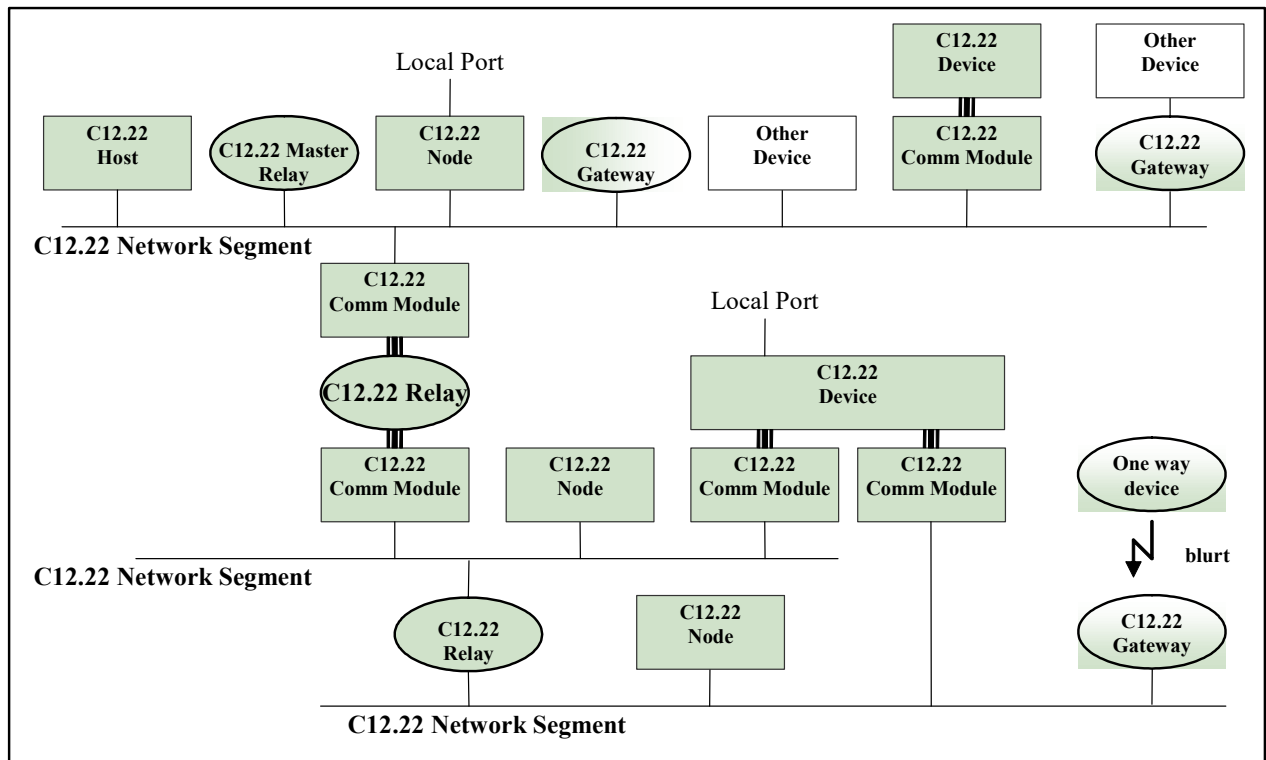


Figure 4.1: Reference Topology

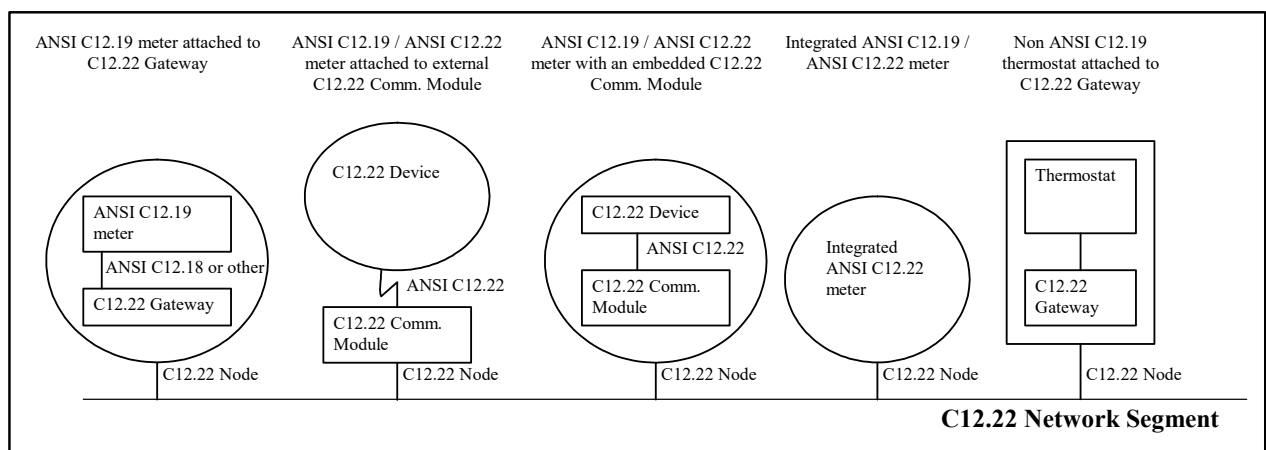


Figure 4.2: C12.22 Node implementation examples



## 5 C12.22 Node to C12.22 Network Segment Details

### 5.1 C12.22 Node to C12.22 Network Segment Reference

Figure 5.1 shows a C12.22 Node that is attached to a C12.22 Network Segment. It also shows relay and gateway interface requirements needed to access remote networks (C12.22 Relay) and the translation services that may exist in these networks (C12.22 Gateway). The figure also shows how a C12.22 Network Segment can be connected to a non-C12.22 network segment.

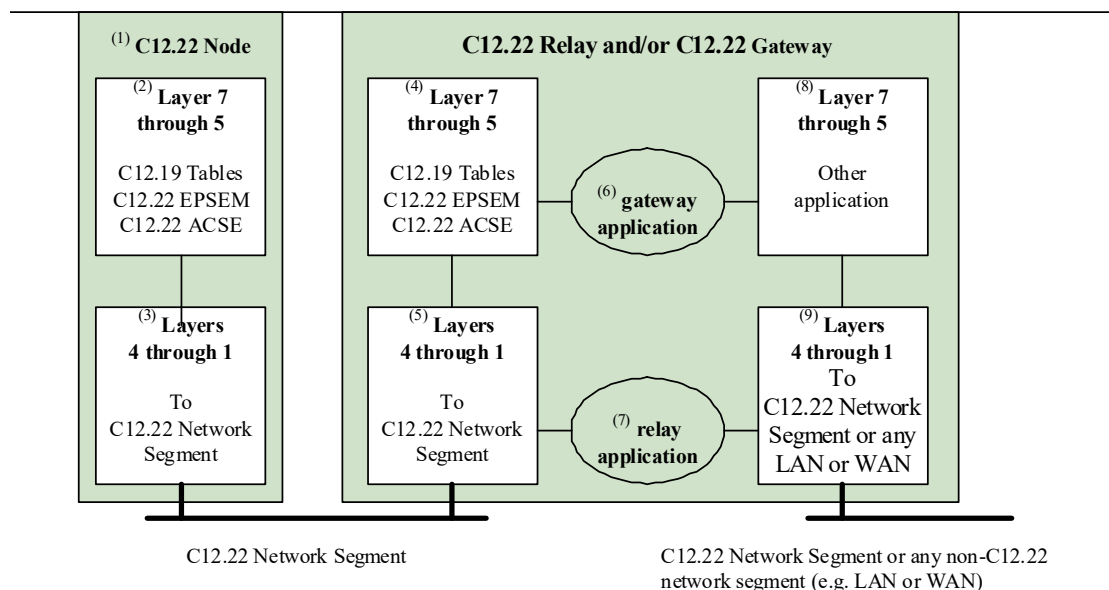


Figure 5.1: C12.22 Reference Network Model

Annotations:

1. C12.22 Node attached to a C12.22 Network Segment
2. C12.22 Application communicating using C12.19 Tables and EPSEM/ACSE encapsulation
3. Layers 4 through 1 interfacing a C12.22 Application to an unspecified native network infrastructure
4. C12.22 Application of a C12.22 Gateway
5. Layers 4 through 1 of a C12.22 Relay
6. C12.22 Gateway performing translation of the C12.22 Application to and from another application
7. C12.22 Relay performing layer 4 through 1 translation
8. Other application of a C12.22 Gateway
9. Remote (other) network interface side of a C12.22 Gateway and/or C12.22 Relay

### 5.2 Data encoding rules

#### 5.2.1 Data order

The data order is identical to that defined in ANSI C12.18 and ANSI C12.19.

Within the syntax definitions, multiple parameters shall be encoded in the order as shown, from left to right.

Unless otherwise noted, parameters in all layers within the protocol definition are encoded most significant byte first. The order of data fields within Tables is dictated by ANSI C12.19.

```

<word24>      ::= <msbyte> <byte> <lsbyte>
<word16>      ::= <msbyte> <lsbyte>

<msbyte>      ::= <byte> {most significant byte}
<lsbyte>      ::= <byte> {least significant byte}

<byte>        ::= {See definition of Byte.}

```

### 5.2.2 Length Fields Encoding

ASN-1/BER field lengths are encoded using the ISO/IEC 8825-1:2002. This encoding is defined as follows:

For values between 0 and 127, length fields are encoded using the short form. This form consists of a single byte representing the length of the subsequent message in octets.

For values greater than 127, length fields are encoded using the long form. This form consists of one byte with bit 7 set to one and bits 0 to 6 set to the number of additional octets, which follow. Additional octets represent the length encoded with most significant octet appearing first.

This is a restricted version of the BER. The BER length is restricted by DER encoding; specifically it shall be encoded as the definite length on the minimum number of octets whether the encoding is in primitive or in constructed form.

Examples:

Short form: 2C<sub>H</sub> represents a length of 2C<sub>H</sub> or 44  
 Long form: 82<sub>H</sub> 01<sub>H</sub> 26<sub>H</sub> represents a length 0126<sub>H</sub> or 294  
 Long form: 81<sub>H</sub> 80 represents a length of 80<sub>H</sub> or 128

### 5.2.3 Universal Identifiers Encoding

This Standard uses Universal Identifiers contained in the <aSO-context>, <mechanism-name>, the <calling-AP-title>, <called-AP-title>, Network and Relay Tables, EPSEM application layer services and C12.22 Communication Module transport layer services. A Universal Identifier, <universal-id>, is an ordered list of sub-identifier values that are concatenated together and represented as follows:

$$\text{value}_1 . \text{value}_2 . \dots . \text{value}_n . \dots . \text{value}_m$$

To guarantee the uniqueness of this identifier over all possible applications, the leading (left most) n values are obtained from an official registration body like the International Organization for Standardization (ISO). Any values following (branches of) this unique prefix (root) can be assigned locally by the owner of this prefix.

Universal identifier is encoded using the Basic Encoding Rules (BER) (ISO/IEC 8825-1:2002) object identifier content encoding. This encoding is defined as follow:

- The first octet has value  $40 \times \text{value}_1 + \text{value}_2$ . (This is unambiguous, since  $\text{value}_1$  is limited to values 0, 1, and 2;  $\text{value}_2$  is limited to the range 0 to 39)
- The following octets, if any, encode  $\text{value}_3, \dots, \text{value}_n$ . Each value is encoded base 128, most significant digit first, with as few digits as possible, and the most significant bit of each octet except the last in the value's encoding set to one.

For efficiency, it is possible to encode a Universal Identifier relative to an ANSI C12 root. In that case, only the branch of the designated ANSI C12 root is included in the Relative Identifier.

The ASN.1 assigned tag for a Universal Identifier (OBJECT IDENTIFIER) is 06<sub>H</sub>. The ASN.1 assigned tag for a Universal Relative Identifier (RELATIVE-UID) is 0D<sub>H</sub>. These tags (06<sub>H</sub> and 0D<sub>H</sub>) may be used as described below only when placing object identifiers in C12.19 Tables, in EPSEM application layer services and in C12.22 Communication Module transport layer services or used explicitly in ASN.1 syntax.

```

<universal-id-element> ::= 06H <universal-id-length> <universal-id>
                                {Absolute encoding of a universal
                                identifier, encoded as described in
                                ISO/IEC 8825-1:2002 [BER].}

<universal-id-length> ::= <byte> {Length of <universal-id>. This value
                                shall range between 00H to 7FH}

<universal-id> ::= <byte>*      {Absolute object identifier content
                                encoded as described in ISO/IEC 8825-
                                1:2002 [BER]. The size of this field shall
                                not exceed 127 bytes.}

<relative-uid-element> ::= 0DH <relative-uid-length> <relative-uid>
                                {Relative encoding of a universal
                                identifier as described in ISO/IEC 8825-
                                1:2002 [BER].}

<relative-uid-length> ::= <byte> {Length of <relative-uid>. This value
                                shall range between 00H to 7FH}

<relative-uid> ::= <byte>*      {Relative object identifier content
                                encoded as described in ISO/IEC 8825-
                                1:2002 [BER].}

```

Note: The Connectionless-mode Association Control Service Element contains elements that encapsulate universal identifiers (such as <aSO-context-element>, <called-AP-title-element>, <calling-AP-title-element>, and <mechanism-name-element>). In this context the ASN.1 tags of the universal identifier may be different from the tags defined in this subsection.

For example,

Let the ApTitle	= <application-context-oid>.0.156.5454 and
let the ANSI C12 root ApTitle	= <application-context-oid>.0

Then the Calling ApTitle element (<calling-AP-title-element>) shall be encoded (in hexadecimal notation) as follows:

Universal Calling ApTitle encoding	= <b>A6</b> 0D <b>06</b> 0C 60 7C 86 F7 54 01 16 00 81 1C AA 4E
Relative Calling ApTitle encoding	= <b>A6</b> 06 <b>80</b> 04 81 1C AA 4E

Similarly for the Called ApTitle element (<calling-AP-title-element>)

Universal Called ApTitle encoding	= <b>A2</b> 0D <b>06</b> 0C 60 7C 86 F7 54 01 16 00 81 1C AA 4E
Relative Calling ApTitle encoding	= <b>A2</b> 06 <b>80</b> 04 81 1C AA 4E

Where **A2<sub>H</sub>** and **A6<sub>H</sub>** are the ACSE assigned tags of the Called and Calling ApTitle elements, respectively; and within each we have the encapsulate tags **06<sub>H</sub>** and **80<sub>H</sub>** that introduce the actual universal and relative identifiers, respectively. Please consult the <acse-pdu> element definitions for more details.

## 5.2.4 Universal Identifiers Canonical Encoding

The C12.22 standard supports two form of ApTitles, Universal (Absolute) and Relative ApTitles. The standard allows for C12.22 Relays to change the ApTitles form when a C12.22 Message is forwarded from one C12.22 Network Segment to another. In the context of C12.22 Message security, modification to ApTitles can be problematic. The called ApTitles and the calling ApTitles (when present), are part of the C12.22 Message authentication process and confidentiality nonce construction. If a message is modified during transmission, the authentication will fail. To resolve this, authentication of C12.22 Messages is always done against the Universal form of the <calling-AP-title> and the <called-AP-title>, regardless of how the message was transmitted. The implication of this is that the receiving C12.22 Node must be able convert a relative ApTitle to a universal ApTitle form upon receipt before enacting the security algorithms.

C12.22 Relays, at their option, may transform relative ApTitles to absolute ApTitles. In order to ensure that relative ApTitles can be unambiguously converted, all C12.22 Message ApTitles that transverse any C12.22 Relay the ApTitles shall always be universal (absolute) or they shall be relative to the <application-context-oid>.0.

When a C12.22 Node sends a C12.22 Message to another C12.22 Node that resides on the same C12.22 Segment (direct messaging), all relative ApTitles shall be relative to <application-context-oid>.0.

**Note 1:** When a C12.22 Message is sent using a proxy C12.22 Relay, the message sender shall indicate this fact by setting the PROXY\_SERVICE\_USED flag of the <epsem-control> of <epsem> to one (1). The recipient of any C12.22 Message that has this flag set to one (1), shall not include the <called-AP-title-element> in the computation of the <mac> (authentication verification code).

**Note 2:** Proxy C12.22 Relays are trusted by the C12.22 Nodes which they serve. Therefore, proxy C12.22 Relays shall validate their content prior to passing C12.22 Message to their client C12.22 Nodes on the return path (from the C12.22 Network to the local C12.22 Network Segment).

### 5.3 Layer 7 - Application Layer

The Application Layer provides a minimal set of services and data structures required to support C12.22 Nodes for purposes of configuration, programming and information retrieval in a networked environment.

This layer is composed of the following four nested components:

- ANSI C12.19 Table data structure
- EPSEM as defined in this section
- ACSE association control as defined by ISO/IEC 15955:1999 and presented in this section

#### 5.3.1 Data Structure - Utility Industry Data Tables

The data structures transported by this protocol are Tables as defined in ANSI C12.19.

#### 5.3.2 EPSEM

This Standard defines thirteen EPSEM services. Each service consists of a request and a response. Each of these requests and responses is described in following sections.

```

<request>      ::= <ident> |      { * Identification Service request }
                  <read> |         { Read Service request }
                  <write> |        { Write Service request }
                  <logon> |        { * Logon Service request }
                  <security> |     { Security Service request }
                  <logout> |       { * Logout Service request }
                  <terminate> |    { * Terminate Service request }
                  <disconnect> |   { * Disconnect Service request }
                  <wait> |         { * Wait Service request }
                  <register> |     { ** Registration Service request }

```

```

<deregister> | {** Deregistration Service}
<resolve> | {** Resolve Service request}
<trace> | {** Trace Service request}

<response> ::= <ident-r> | { * Identification Service response}
               <read-r> | { Read Service response}
               <write-r> | { Write Service response}
               <logon-r> | { * Logon Service response}
               <security-r> | { Security Service response}
               <logoff-r> | { * Logoff Service response}
               <terminate-r> | { * Terminate Service response}
               <disconnect-r> | { * Disconnect Service response}
               <wait-r> | { * Wait Service response}
               <register-r> | {** Registration Service response}
               <deregister-r> | {** Deregistration Service response}
               <resolve-r> | {** Resolve Service response}
               <trace-r> | {** Trace Service response}

```

**Notes:**

- \* Definition or content revised from ANSI C12.18 and/or ANSI C12.21
- \*\* New in ANSI C12.22
- The network management services such as the <register>, <deregister>, <resolve> and <trace> services may be transmitted authenticated but not encrypted.

**5.3.2.1 Request Codes**

EPSEM requests always include a one-byte request code. Code numbers are assigned as follows:

00H-1FH	Codes shall not be used to avoid confusion with response codes
20H-7FH	Codes are available for use within ANSI C12 protocols
80H-FFH	Codes shall be reserved for protocol extensions

**5.3.2.2 Response Codes**

EPSEM responses always include a one-byte response code. These codes are listed below in a suggested order of priority. They represent an extension to the response codes available in ANSI C12.18 and ANSI C12.21. When more than one response code is capable of indicating the error response condition of a C12.22 Node, the response code having the highest priority (from left to right) may be provided as follows:

```

<nok> ::= <sns>|<isss>|<iar>|<sme>|<isc>|<onp>|<bsy>|<dlk>|<dnr>|
          <rno>|<uat>|<netr>|<nett>|<rqt1>|<rst1>|<sgnp>|<sgerr>|<err>

```

For example, if a C12.22 Device with a C12.22 Application contains ANSI C12.19 Tables, and Table 05 of this device is read-only, and it is encoded in non-volatile memory, then a Write Service request to Table 05 would fail. The C12.22 Device may consider the following codes as suitable responses: <err> to indicate an error condition or <dlk> to indicate that the data is locked in memory and cannot be changed, <iar> to indicate that the action requested was not appropriate for this device design or <isc> to indicate that the table access permission are "read-only" under the current security policy. The correct response would be <iar> as it is the highest priority among <iar>, <isc>, <dlk> and <err>. However, if there is a mechanism for providing write access to Table 05, then <iar> should not be considered. Therefore, the response code becomes <isc>.

**Responses**

```

<ok> ::= 00H {Acknowledge
              No problems, request accepted.}

```

## ANSI C12.22-2008

<err>	::= 01 <sub>H</sub>	{Error This code is used to indicate rejection of the received service request. The reason for the rejection is not provided.}
<sns>	::= 02 <sub>H</sub>	{Service Not Supported This Application-level error response will be sent to the device when the requested service is not supported. This error indicates that the message was valid, but the request could not be honored. The <sns> error response has special implications in the context of a response to a Logoff, Terminate or Disconnect service request. Specifically, a C12.22 Node in the Session State shall not issue this error, but it is permitted if the C12.22 Node only supports sessionless communication.}
<isc>	::= 03 <sub>H</sub>	{Insufficient Security Clearance This Application-level error indicates that the current authorization level is insufficient to complete the request.}
<onp>	::= 04 <sub>H</sub>	{Operation Not Possible This Application-level error will be sent to the device that requested an action that is not possible. This error indicates that the message was valid, but the message could not be processed and covers conditions such as invalid <length> or invalid <offset>. It can also be issued if the operation is not possible under the current C12.22 Node configuration.}
<iar>	::= 05 <sub>H</sub>	{Inappropriate Action Requested This Application-level error indicates that the action requested was inappropriate. Covers conditions such as a Write Service request to a read-only Table or an invalid Table identifier.}
<bsy>	::= 06 <sub>H</sub>	{Device Busy This Application-level error indicates that the request was not acted upon because the device was busy doing something else. The operation may be retried at a later time with success, as the data may then be ready for transportation during this active communication.}
<dnr>	::= 07 <sub>H</sub>	{Data Not Ready This Application-level error indicates that request was unsuccessful because the requested data is not ready to be accessed.}
<dlk>	::= 08 <sub>H</sub>	{Data Locked This Application-level error indicates

		that the request was unsuccessful because the data cannot be accessed.}
<rno>	::= 09 <sub>H</sub>	{Renegotiate Request This Application-level error indicates that the responding device wishes to return to the identification or Base State and renegotiate communication parameters.}
<isss>	::= 0A <sub>H</sub>	{Invalid Service Sequence State This Application-level error indicates that the request cannot be accepted at the current service sequence state. For more information on service sequence states, see section 5.3.2.5 "Service sequence state control". This is an indication to the C12.22 Application not to reissue this request at this time because there is a service sequence state problem or an out-of-order operations problem.}
<sme>	::= 0B <sub>H</sub>	{Security Mechanism Error This Application-level error may be returned when a security mechanism error is detected. This code covers errors such as a security mechanism not being supported and invalid encryption key.}
<uat>	::= 0C <sub>H</sub>	{Unknown Application Title This Application-level error may be returned by a C12.22 Relay or the target node when an unknown or invalid <called-AP-title> is received.}
<nett>	::= 0D <sub>H</sub>	{Network Time-out This Application-level error may be returned when a Network Time-out is detected.}
<netr>	::= 0E <sub>H</sub>	{Network Not Reachable This Application-level error may be returned when a node is not reachable.}
<rqt1>	::= 0F <sub>H</sub> <max-request-size>	{Request Too Large This Application-level error may be returned when the request size is too large.}
<max-request-size>	::= <word32>	{Maximum request size in bytes allowed by the target device.}
<rslt1>	::= 10 <sub>H</sub> <max-response-size>	{Response Too Large This Application-level error may be returned when the response size of a response is too large.}
<max-response-size>	::= <word32>	{Maximum response size in bytes allowed by the target device.}

## ANSI C12.22-2008

<sgnp>	::= 11 <sub>H</sub>	{Segmentation not possible This Application-level error may be returned when a C12.22 Node received a segment and does not support the Application Segmentation Sub-layer.}
<sgerr>	::= 12 <sub>H</sub> <segment-byte-offset> <apdu-size>	{Segmentation error This Application-level error may be returned when a C12.22 Node fail to segment or reassemble an <acse-pdu>.}
<segment-byte-offset>	::= <byte>   <word16>   <word24>	{Offset in bytes relative to the beginning of the fully assembled APDU which the first error was detected.}
<apdu-size>	::= <byte>   <word16>   <word24>	{Size of the fully assembled APDU, <acse-pdu>, in bytes. The data encoding format of the <apdu-size> and <segment-byte-offset> shall be identical.}
	13 <sub>H</sub> -1F <sub>H</sub>	{Response codes in this range are not defined by this Standard.}
	20 <sub>H</sub> -7F <sub>H</sub>	{These codes shall not be used to avoid confusion with request codes.}
	80 <sub>H</sub> -FF <sub>H</sub>	{These codes shall be reserved for protocol extensions.}

### 5.3.2.3 Time-out

#### 5.3.2.3.1 Session Time-out

Each session established with a C12.22 Server shall be monitored by the C12.22 Server and shut down when the session becomes inactive. An inactive session is one which does not receive EPSEM messages from the C12.22 Client within an allowable time period (the Session Time-out). An EPSEM message may be a request or a response.

The Session Time-out value is set by the Logon Service request and can be temporarily modified for the next request through the use of the Wait Service.

The Session Time-out interval starts upon transmission by the C12.22 Server of an <ok> response to a Logon Service request that was issued by the C12.22 Client. The timer restarts upon transmission or reception of any byte of an <acse-pdu> on the C12.22 Server side during a session.

The Time-out timer stops when the session ends for any reason.

When multiple concurrent sessions are supported, the Session Time-out for each session is set independently by the Logon Service request that established the session.

The Session Time-out timer is not used in sessionless communication.

#### 5.3.2.3.2 Application Layer Response Time-out

The Application Layer Response Timeout is used by a C12.22 Node that issues service requests to another C12.22 Node and needs to know how long to wait for responses.



A non-recoverable Application Layer Response Timeout shall terminate the associated session if one exists. A non-recoverable Application Layer Response Timeout is the last one, for implementations that allow retries, or the first one in implementations that do not.

An example time-out algorithm is described in “Annex F - APDU Response Timeout Algorithm”.

### 5.3.2.4 Services

#### 5.3.2.4.1 Identification Service

This service is used to obtain information about C12.19 Device functionality. The service returns a code identifying the reference standard, the version and revision of the reference standard implemented, and an optional feature list.

##### Request:

<ident> ::= 20<sub>H</sub>

##### Response:

The responses <isss>, <bsy>, and <err> indicate a problem with the received service request. The response <ok> indicates the Identification Service request was accepted and the standard, version, revision, and optional feature list are included in the response.

```

<ident-r>      ::= <isss> | <bsy> | <err> |
                  <ok> <std> <ver> <rev> <feature>* <end-of-list>

<std>          ::= <byte>          {Code identifying reference standard. The
                                   codes are defined as follows:

                                   00H      = ANSI C12.18
                                   01H      = Reserved
                                   02H      = ANSI C12.21
                                   03H      = ANSI C12.22
                                   04H-FFH = Reserved

                                   This value shall be 03H.}

<ver>          ::= <byte>          {Binary representation of the referenced
                                   standard version number to the left of the
                                   decimal point. This value shall be 01H.}

<rev>          ::= <byte>          {Binary representation of the referenced
                                   standard version number to the right of
                                   the decimal point. This value shall be
                                   00H.}

<feature>      ::= <security-mechanism> |
                  <session-ctrl> |
                  <device-class> |
                  <device-identity> {Features available}

<end-of-list>  ::= 00H            {End of list indicator.}

<security-mechanism> ::= 04H <universal-id-element> |
                        04H <relative-uid-element>
                        {Present in the feature list only if the
                        C12.22 Node supports one or more security

```

mechanisms. This feature element contains the universal id of the security mechanism supported. See the <mechanisms-name-element> in section 5.3.4 "Association Control - Association Control Service Element (ACSE)" for more information.}

<session-ctrl> ::= 05<sub>H</sub> <byte> {Bit 0 to 6: NBR\_SESSION\_SUPPORTED  
If greater than zero, the C12.22 Node supports session-based communication. A Session starts by the reception of the Logon Service and ends by the reception of the Logoff Service or the detection of a Session Time-out.

Bit 7: SESSIONLESS\_SUPPORTED  
If true, the C12.22 Node supports the use of the Read and Write Services outside a session.

By default, when <session-ctrl> field is not included in the identification response, one session and sessionless operations are supported.}

<device-class> ::= 06<sub>H</sub> <absolute-device-class-element> |  
06<sub>H</sub> <relative-device-class-element>  
{A Universal Identifier that uniquely identifies a C12.19 Device Class object for early detection per the value of the MANUFACTURER element as defined in Table 0 of ANSI C12.19-1997 or the DEVICE CLASS element as defined by version 2 of ANSI C12.19. When <device-class> is provided it shall be placed before <feature>s with codes that are greater than 06<sub>H</sub>. The C12.19 Device Class, e.g., <device-class-root-oid>.<device-class>, encoded as described in ISO/IEC 8825-1: 2002 [BER]. The last four bytes of this identifier shall be identical to the values delivered in the C12.19 Table Element MANUFACTURER as defined in Table 00 of ANSI C12.19-1997 or the DEVICE CLASS as defined by version 2 of ANSI C12.19. For example, C12.19 Device Class "<device-class-root-oid>.26.0.0.0" can be encoded relatively as ".26.0.0.0".}

<absolute-device-class-element> ::= <universal-id-element>  
{The absolute encoding of the object identifier encoded as described in ISO/IEC 8825-1:2002 [BER].}

<relative-device-class-element> ::= <relative-uid-element>  
{The relative encoding of the object identifier encoded as described in ISO/IEC 8825-1:2002 [BER].}

<device-identity> ::= 07<sub>H</sub> <identity-length><identity>

{An Identifier that uniquely identifies a C12.19 Device in the application space of the C12.19 Device. This provides for early detection of the device identification element as per IDENTIFICATION of the Device Identification Table (Table 05), or DEVICE\_ID found in the Utility Information Table (Table 06) of ANSI C12.19. The C12.19 <device-identity> feature shall be supplied when the C12.19 Device's Table 05 or Table 06, are readable immediately following the <logon> service. When C12.19 Device Identification is provided it shall not precede <feature>s with codes that are less than 07<sub>H</sub>.}

<identity-length> ::= <byte> {Length of number of bytes that follow in <identity>. This value shall range between 01<sub>H</sub> to 7F<sub>H</sub>}

<identity> ::= <char-format><identification> {Provides for early (pre-logon) disclosure of the C12.19 Device identification.}

<char-format> ::= <byte> {The character encoding format of the bytes which make up <identification>. Its interpretation shall be according to the relevant ANSI C12.19 Standard data model referenced by the C12.19 registered Device Class feature <device-class>. When the <device-class> feature is not supplied in this <ident> response, the value of <char-format> shall be set to 01<sub>H</sub>, and <identification> shall be encoded according to ISO 7-bit coded character set for information interchange, per ISO/IEC 646: 1991.}

<identification> ::= <byte>\* {The C12.19 Device identification string encoded and transmitted according to <char-format>. If the C12.19 Device's ID\_FORM in Table 00, is set to BCD then the BCD digits shall be transmitted as their text equivalent also encoded as per <char-format>.

For example, if the C12.19 Device's GEN\_CONFIG\_TBL.ID\_FORM is BCD and the device's GEN\_CONFIG\_TBL.CHAR\_FORMAT is ISO 7 bit ASCII, as per ISO/IEC 646: 1991, then the BCD digits 00<sub>H</sub> 01<sub>H</sub> 02<sub>H</sub> 03<sub>H</sub> 0A<sub>H</sub> 04<sub>H</sub> 0D<sub>H</sub> 05<sub>H</sub> 06<sub>H</sub> 07<sub>H</sub> shall be transmitted as the character sequence "123-4.567". The C12.19 application shall logically pad this string with trailing spaces, as needed, to fill the identification field width of the C12.19 Device.}

#### 5.3.2.4.2 Read Service

The Read Service is identical to that found in ANSI C12.18 and ANSI C12.21 with the inclusion of

additional error response codes defined by this Standard and the addition of the capability to receive Table data in excess of 65535 bytes.

The Read Service is used to cause a transfer of Table data to the requesting device.

#### Request:

The read request allows both complete and partial Table transfers. The retrieval of a portion of a Table is possible through the use of both index/element-count and offset/octet-count methods. The complete rule set for using these methods is enumerated in section 5.3.2.6 "Partial Table access using index/element-count Method" and section 5.3.2.7 "Partial Table access using offset/octet-count method" respectively. Request codes 30<sub>H</sub>-39<sub>H</sub>, 3E<sub>H</sub> and 3F<sub>H</sub> give access to all possible methods used for Table transfer. Request code 30<sub>H</sub> specifies a complete Table transfer. Request codes 31<sub>H</sub> through 39<sub>H</sub> specify a partial Table transfer using 1 to 9 indices. Request code 3E<sub>H</sub> specifies a default Table transfer. Request code 3F<sub>H</sub> specifies a partial Table transfer using the offset/octet-count method.

```

<read> ::= <full-read> | <pread-index> | <pread-offset> | <pread-
                        default>

<full-read> ::= 30H <tableid>

<pread-index> ::= <read-index-type> <tableid> <index>+ <element-count>

<read-index-type> ::= 31H | {1 <index> included in request}
                      32H | {2 <index> included in request}
                      33H | {3 <index> included in request}
                      34H | {4 <index> included in request}
                      35H | {5 <index> included in request}
                      36H | {6 <index> included in request}
                      37H | {7 <index> included in request}
                      38H | {8 <index> included in request}
                      39H | {9 <index> included in request}

<pread-default> ::= 3EH {Transfer default Table}

<pread-offset> ::= 3FH <tableid> <offset> <octet-count>

<tableid> ::= <word16> {Table identifier as defined in ANSI
                        C12.19.}

<offset> ::= <word24> {Offset into data Table in bytes. Offset
                        0000H is the offset to the first Table
                        element of the Table selected by
                        <tableid>..}

<index> ::= <word16> {Index value used to locate start of data.
                      Index 0000H+ is the index of the first
                      Table element of the Table selected by
                      <tableid>..}

<element-count> ::= <word16> {Number of ANSI C12.19 Table Elements to
                               read starting at the requested <index>..}

<octet-count> ::= <word16> {Length of ANSI C12.19 Table data
                              requested starting at <offset> plus the
                              length of the optional pending header
                              defined by ANSI C12.19, in bytes}

```

#### Response:

Responses of type <nok> indicate a problem with the received service request.

The response <ok> indicates the Read Service was accepted and the <data> is part of the response.

```

<read-r>          ::= <nok> |
                   <ok> <table-data>+

<table-data>      ::= <count> <data> <cksum>

<count>           ::= <word16>          {Length of <data> returned, in bytes, in
                                         this <table-data> element. When <count>
                                         equals to FFFFH it is an indication that
                                         the length of <data> returned in this
                                         <table-data> is 65535 bytes, and that
                                         another <table-data> element shall follow.
                                         The last <table-data> element shall have a
                                         <count> that is less than FFFFH. When the
                                         last <table-data> is exactly 65535 bytes,
                                         it is followed by <table-data> with
                                         <count> equal to 0H. This <count> includes
                                         the optional pending header length as
                                         defined by ANSI C12.19.}

<data>            ::= <byte>*           {The returned C12.19 Table data including
                                         the optional ANSI C12.19 pending header
                                         when requested. The optional pending
                                         header may be placed only in the beginning
                                         of the <data> of the first <table-data>
                                         element of any given read service
                                         response.}

<cksum>           ::= <byte>           {2's complement checksum computed only on
                                         the <data> portion of <table-data>. The
                                         checksum is computed by summing the bytes
                                         (ignoring overflow) and negating the
                                         result.}

```

#### 5.3.2.4.3 Write Service

The Write Service is identical to that found in ANSI C12.18 and ANSI C12.21 with the inclusion of additional error response codes defined by this Standard and the addition of the capability to transmit Table data in excess of 65535 bytes.

The Write Service is issued to transfer Table data to the target device.

##### Request:

The Write Request allows both complete and partial Table transfers. The modification of a portion of a Table is possible through the use of both index/count and offset/count methods. The complete rule set for using these methods is enumerated in 5.3.2.6 and 5.3.2.7 respectively.

Request codes 40<sub>H</sub>-49<sub>H</sub> and 4F<sub>H</sub> give access to all possible methods used for Table transfer. Request code 40<sub>H</sub> specifies a complete table transfer. Request codes 41<sub>H</sub> through 49<sub>H</sub> specify a partial Table transfer using 1 to 9 indices. Request code 4F<sub>H</sub> specifies a partial Table transfer using the offset/count method.

```

<write>           ::= <full-write> | <pwrite-index> | <pwrite-offset>

```

## ANSI C12.22-2008

```

<full-write>      ::= 40H <tableid> <table-data>+

<pwrite-index>    ::= <write-index-type> <tableid> <index>+ <table-data>

<write-index-type> ::=
    41H | {1 <index> included in request}
    42H | {2 <index> included in request}
    43H | {3 <index> included in request}
    44H | {4 <index> included in request}
    45H | {5 <index> included in request}
    46H | {6 <index> included in request}
    47H | {7 <index> included in request}
    48H | {8 <index> included in request}
    49H | {9 <index> included in request}

<pwrite-offset>   ::= 4FH <tableid> <offset> <table-data>

<tableid>         ::= <word16>      {Table identifier as defined in ANSI
                                     Standard C12.19.}

<offset>          ::= <word24>      {Offset into data Table in bytes. Offset
                                     0000H is the offset to the first element
                                     of the Table selected by <tableid>..}

<index>           ::= <word16>      {Index value used to locate start of data.
                                     Index 0000H+ is the index of the first
                                     element of the Table selected by
                                     <tableid>..}

<table-data>      ::= <count> <data> <cksum>

<count>           ::= <word16>      {Length of <data> to be written, in bytes,
                                     from this <table-data> element. When
                                     <count> equals to FFFFH it is an
                                     indication that the length of <data>
                                     contained in this <table-data> is 65535
                                     bytes, and that another <table-data>
                                     element shall follow. The last <table-
                                     data> element shall have a <count> that is
                                     less than FFFFH. When the last <table-
                                     data> is exactly 65535 bytes, it is
                                     followed by <table-data> with <count>
                                     equal to 0H. This <count> includes the
                                     optional pending header length as defined
                                     by ANSI C12.19.}

<data>            ::= <byte>*       {The Table data elements including the
                                     optional pending header as per ANSI C12.19
                                     when requested. The optional pending
                                     header may be placed only in the beginning
                                     of the <data> of the first <table-data>
                                     element of any given read service request}

<cksum>           ::= <byte>        {2's compliment checksum computed only on
                                     the <data> portion of <table-data>. The
                                     checksum is computed by summing the bytes
                                     (ignoring overflow) and negating the
                                     result.}

```

### Response:

Responses of type <nok> indicate a problem with the received service request.

The response <ok> indicates the Write Service was successfully completed and the data was successfully transmitted to the device.

```
<write-r>      ::= <nok> |
                  <ok>
```

#### 5.3.2.4.4 Logon Service

Logon Service establishes a session without establishing access permissions. It provides for immediate transfer to the session state from the idle state. A peer-to-peer association shall be established.

##### Request:

The <user-id> may be inserted in the Event and History Logs as defined in ANSI C12.19, when supported by the C12.22 Node. The <user> field provides the name of the operator requesting the access to the C12.22 Node and may be recorded in the Utility Information Table (Table 06).

The Logon Service request has the following format:

```
<logon>      ::= 50H <user-id> <user> <req-session-idle-timeout>

<user-id>    ::= <word16>      {User identification code}

<user>       ::= <byte>+10     {10 bytes containing user identification}

<req-session-idle-timeout> ::= <word16> {The desired number of seconds a
                                         session may be idle on the C12.22 Server
                                         side before the C12.22 Server may
                                         terminate the session. A value of zero
                                         means a request to keep the session open
                                         forever.}
```

##### Response:

All responses other than <ok> indicate a problem with the received service request and the session and the association shall not be established. The C12.22 Node shall remain in the idle state.

All error responses (including those that are not listed below or extensions on those covered by <nok>) shall be generically interpreted as an indication to the application not to issue another Logon request without first addressing the cause of the indicated error condition.

The response <ok> indicates that the Session was successfully established.

```
<logon-r>      ::= <nok> |
                  <ok> <resp-session-idle-timeout>

<resp-session-idle-timeout> ::= <word16>
                                {The number of seconds a session may be
                                idle on the C12.22 Server before the
                                C12.22 Server may terminate the Session.
                                This value shall be less than or equal to
                                <req-session-idle-timeout>.}
```

#### 5.3.2.4.5 Security Service

The Security Service is provided for setting access permissions.

It should be noted that sending passwords as Cleartext (unencrypted) over the network is a security

concern. Available privacy and authentication encryption services are described in section 5.3.4.13 "C12.22 Security Mechanism".

It should also be noted that the EPSEM Security Service is unrelated to the C12.22 Security Mechanism as described in section 5.3.4.13 "C12.22 Security Mechanism". The Security Service works solely within the context of the EPSEM services. Its purpose is to provide a mechanism to change the C12.19 Tables access privilege or authorization when issuing commands to a C12.22 Node. It does not "secure" the C12.22 Message in any way.

The Security Mechanism uses cryptographic techniques to provide authentication and privacy for C12.22 Messages, regardless of the contents of the C12.22 Message. The Security Mechanism works at the C12.22 Network Application Layer level, and it is described in detail in section 5.3.4.13 "C12.22 Security Mechanism".

#### Request:

A password is used as a means to select access permissions. This service request may only be sent during a session. The <password> field will be compared with those in the Security Table (Standard Table 42) defined in ANSI C12.19, if the password Tables are supported by the C12.22 Node.

```
<security>      ::= 51H <password> [ <user-id> ]

<password>      ::= <byte> +20      {20 byte field containing password}

<user-id>       ::= <word16>      {User identification code . This field
                                   shall be present when this service is
                                   included in a sessionless message. In the
                                   context of a session, this information is
                                   provided by the Logon service and shall
                                   not be included in this service.}
```

#### Response:

The responses <sns>, <isss>, <bsy> and <err> indicate a problem with the received service request.

The response <ok> indicates the security service was successfully completed and the access permissions associated with the password were granted.

```
<security-r>    ::= <sns> | <isss> | <bsy> | <err> |
                   <ok>
```

### **5.3.2.4.6 Logoff Service**

The Logoff Service provides for an orderly termination of the session that was established by the Logon Service. It provides for immediate transfer to the idle state from the session state. The peer-to-peer association shall terminate and all previously negotiated settings shall reset to their default idle-state values.

The Physical Layer signaling parameters of the C12.22 Node shall not be affected by this service.

#### Request:

```
<logoff>        ::= 52H
```

#### Response:



All responses other than <ok> indicate a problem with the received service request and the session and the association shall retain the settings negotiated prior to the issuance of the Logoff Service request unless otherwise specifically indicated by the error code.

All error responses shall be generically interpreted as an indication to the application not to issue another Logoff Service request without first addressing the cause of the indicated error condition.

When a response other than <ok> is understood to be an indication other than the severance of the communication path or association, the application may issue any valid session state service request or choose to terminate the association or it may let the Session Time-out period expire to force the Session to be aborted.

```
<logoff-r>      ::= <nok> |
                  <ok>      {Response code or error codes as per
                              Section 5.3.2.6 "Response Codes".}
```

The response <ok> indicates the successful termination of the Session. This is an indication to the C12.22 Application that the C12.22 Node completed all session-related processing before terminating the session and that it reset its communication parameters to their default settings and entered the idle state. The Association between the peer C12.22 Nodes was terminated.

#### 5.3.2.4.7 Terminate Service

The Terminate Service provides for an orderly abortion of the Session that was established by the Logon Service. It provides for transfer to the idle state from the session state. The peer-to-peer Association shall terminate and all previously negotiated settings shall reset to their default idle-state values.

This application-level service may be used to terminate a Session for reasons such as excessive errors, security issues, internal error conditions, a need to abort session, or other reasons.

The Physical Layer signaling parameters of the C12.22 Node shall not be affected by this service.

##### Request:

```
<terminate>      ::= 21H
```

##### Response:

All responses other than <ok> indicate a problem with the received service request and the session and the association shall retain the settings negotiated prior to the issuance of the Terminate Service request.

All error responses shall be generically interpreted as an indication to the application not to issue another Terminate Service request without first addressing the cause of the indicated error condition.

When a not <nok> response is understood to be an indication other than the severance of the communication path or association, the application may issue any valid session state service request or choose to terminate the Association or it may let the Session timeout to force the session to be aborted.

```
<terminate-r>    ::= <nok> |
                  <ok>
```

The request <ok> indicates that the Session was terminated and that the C12.22 Node aborted all session-related processing and that it reset its communication parameters to their default settings and entered the idle state. The Association between the peer C12.22 Nodes was terminated.

#### 5.3.2.4.8 Disconnect Service

The Disconnect Service is used to remove a C12.22 Node from the C12.22 Network Segment.

The Disconnect Service, when issued within a Session, provides for an orderly abortion of the Session that was established by the Logon Service. It is functionally equivalent to Terminate Service request that is followed by a transition to the off-line state.

When received in the idle state it shall cause the C12.22 Node to enter the off-line state.

All peer-to-peer Associations across the interface of the C12.22 Node on the C12.22 Network segment that processed this request shall terminate. The C12.22 Node's settings shall reset to their default off-line state values for that C12.22 Network Segment.

The Physical Layer signaling parameters of the C12.22 Node may be affected by this service.

For this service request to be successful, the initiator shall have write access permission to Procedure 25 NETWORK\_INTERFACE\_CONTROL\_PROC.

Request:

```
<disconnect>      ::= 22H
```

Response:

All responses other than <ok> indicate a problem with the received service request and the Physical Layer signaling parameters, session and the association shall retain the settings negotiated prior to the issuance of the Disconnect Service request.

All error responses shall be generically interpreted as an indication to the C12.22 Application not to issue another Disconnect Service request without first addressing the cause of the indicated error condition.

When a <nok> response is understood to be an indication other than the severance of the communication path or association, the C12.22 Application may issue any valid service request or choose to terminate the association or it may let the association time-out to force a session to be aborted.

```
<disconnect-r>    ::= <sns>|<iar>|<sme>|<isc>|<onp>|<bsy>|<dlk>|<dnr>|
                    <rno>|<uat>|<netr>|<nett>|<rqt1>|<rst1>|<sgnp>|<sgerr>|
                    <err> |
                    <ok>          {Response code or error codes as defined
                                   in section 5.3.2.2 "Response Codes".}
```

The response <ok> indicates that the Disconnect Service was accepted. If the C12.22 Node was in the session state then this is an indication of the successful abortion of the Session.

This is also an indication that the C12.22 Node aborted all Session-related processing, removed itself from the C12.22 Network Segment (by de-asserting the appropriate Physical Layer signals), entered the off-line state and reset its communication parameters to their default settings. The Association between the peer C12.22 Nodes was terminated and all other Associations that share the same Interface on the C12.22 Node network segment that processed this response were also terminated.

#### **5.3.2.4.9 Wait Service**

The Wait Service is used to maintain an established Session during idle periods, thus preventing automatic termination. This service temporarily extends the Session Time-out to the <time> specified in the request upon acknowledgement of the Wait Service request. The Session Time-out will be reset to the default value once the next valid service is received by this target.

Request:

```

<wait>          ::= 70H <time>
<time>          ::= <byte>          {Requested wait period in seconds. The
                                     value zero does not affect the Channel
                                     settings.}

```

**Response:**

The responses <sns>, <isss>, <bsy>, and <err> indicate a problem with the received service request and Session Time-out is not extended.

The response <ok> indicates the service request was accepted and the Session Time-out is extended to the value requested.

```

<wait-r>        ::= <sns> | <isss> | <bsy> | <err> |
                   <ok>

```

**5.3.2.4.10 Registration Service**

The Registration Service is used to add and keep routing-table entries of C12.22 Relays active. To be part of a C12.22 Network, a C12.22 Node shall send a Registration Service request to one of the C12.22 Master Relays. This service is carried in an ACSE Application Data Unit <acse-pdu> with the Calling ApTitle set to the ApTitle of the registering C12.22 Node and the Called ApTitle set to the ApTitle of the C12.22 Master Relay. This service carries the node type, the node class, serial number, registration lifetime parameters and an optional native address of the registering node. The native address is used only by the neighbor C12.22 Relay to send messages back to this node and it is ignored by all others nodes.

The C12.22 Master Relay shall send a copy of each first registration received ("first" here meaning an actual registration request as contrasted with the reuse of the register service as keep-alive) to all C12.22 Notification Hosts and all C12.22 Authentication Hosts that need to be notified. In this case, the Registration Service is sent with the Calling ApTitle set to the ApTitle of the C12.22 Master Relay and the Called ApTitle set, in turn, to the ApTitle of each C12.22 Host notified and all other parameters set in accordance with the registration response that was issued by the C12.22 Master Relay to the registered C12.22 Node. The C12.22 Master Relay shall send updated copies of each first registration to subscribed C12.22 Notification Hosts when the C12.22 Notification hosts subscribe to this service, including registration records that came into-effect before prior to the subscription for notifications.

**Request:**

```

<register>       ::= 27H <node-type> <connection-type> <device-class>
                   <ap-title> <electronic-serial-number>
                   <address-length> <native-address> <registration-period>
                   [<my-domain-pattern>]

<node-type>      ::= <byte>          {An identification of the C12.22 Node's
                                     Attributes. These values may be set to
                                     advertise the capabilities of this C12.22
                                     Node and to assist C12.22 Master Relay in
                                     the decision making for the successful
                                     completion of all communication with other
                                     C12.22 Nodes that participate in the
                                     registration process.

```

Bit 0 to 5: NODE\_TYPE as per  
INTERFACE\_STATUS\_TBL.NODE\_TYPE\_BFLD.

Bit 0: RELAY

When set to 1 it is an indication that this C12.22 Node is a C12.22 Relay. All C12.22 Relays shall set this bit to 1.

Bit 1: MASTER\_RELAY

When set to 1 it is an indication that this C12.22 Node is a C12.22 Master Relay. All C12.22 Master Relays shall set this bit to 1.

Bit 2: HOST

When set to 1 it is an indication that this C12.22 Node is a C12.22 Host.

Bit 3: NOTIFICATION\_HOST

When set to 1 it is an indication that this C12.22 Node is a C12.22 Notification Host. All C12.22 Notification Hosts shall set this bit to 1, if they wish the C12.22 Master Relay to add them to their list of Notification Hosts or issue notifications to this C12.22 Node when other C12.22 Nodes register with the servicing C12.22 Master Relay (See <called-AP-title>). Note: This does not preclude static registration; however notifications may not be received until the C12.22 Master Relay registers this C12.22 Node as a C12.22 Notification Host.

Bit 4: AUTHENTICATION\_HOST

When set to 1 it is an indication that this C12.22 Node is a C12.22 Authentication Host. All C12.22 Authentication Hosts shall set this bit to 1, if they wish the C12.22 Master Relay to add them to their list of Authentication Hosts or issue registration requests to this C12.22 Node when other C12.22 Nodes register with the servicing C12.22 Master Relay (See <called-AP-title>). Note: This does not preclude static registration; however notifications may not be received until the C12.22 Master Relay registers this C12.22 Node as a C12.22 Authentication Host.

Bit 5: END\_DEVICE

When set to 1 it is an indication that this C12.22 Node is a C12.19 Device. When set to 0 it is an indication the C12.22 Node is not a C12.19 Device.

Bit 6..7: Reserved and shall be set to 0.

<connection-type> ::= <byte>

{An indication of the type of connection requested and the core capability related to this C12.22 Node in regard to its connection to the C12.22 Network Segment. These bits are identical to those defined

by INTERFACE STATUS TBL.  
CONNECTION\_TYPE\_BFLD.

Bit 0: BROADCAST\_AND\_MULTICAST\_SUPPORTED  
Set to one (1) when the C12.22 Node has the capability to accept broadcast and multicast messages. Set to zero (0) otherwise.

Bit 1: MESSAGE\_ACCEPTANCE\_WINDOW\_SUPPORTED  
Set to one (1) when the C12.22 Node is capable of implementing time-based C12.22 Message acceptance windows. When set to 0 it is an indication that this C12.22 Node is not capable of implementing time-based C12.22 Message acceptance windows.

Bit 2: PLAYBACK\_REJECTION\_SUPPORTED  
This bit is set to one (1) when the C12.22 Node is capable of performing playback rejection algorithms on duplicate incoming C12.22 Messages. This bit is set to zero (0) when the C12.22 Node is not capable of performing playback rejection algorithms on duplicate incoming C12.22 Messages.

Bit 3: Reserved and shall be set to zero (0).

Bit 4: CONNECTIONLESS\_MODE\_SUPPORTED  
This bit is set to one (1) if the local network segment and the node support a connectionless-oriented protocol. This bit is set to zero (0) if the local network segment or the node supports do not support a connectionless protocol. At least one of CONNECTIONLESS\_MODE\_SUPPORTED or CONNECTION\_MODE\_SUPPORTED shall be set to one (1).

Bit 5: ACCEPT\_CONNECTIONLESS  
This bit is set to one (1) when the local network segment and the node support a connectionless-oriented protocol (Bit 4 set to 1) and the registering node accepts unsolicited incoming connectionless messages.

Bit 6: CONNECTION\_MODE\_SUPPORTED  
This bit is set to one (1) if the local network segment and the node supports a connection-oriented protocol. This bit is set to zero (0) if the local network segment or the node does not support a connectionless protocol. At least one of CONNECTIONLESS\_MODE\_SUPPORTED or CONNECTION\_MODE\_SUPPORTED shall be set to one (1).

Bit 7: ACCEPT\_CONNECTIONS

This bit is set to one (1) when the local network segment and the node support a connection-oriented protocol (Bit 6 set to 1) and the registering node accepts connections.}

<device-class> ::= <byte>+4

{A list of encoding of sub-identifiers (integers) expressed as the four bytes containing the MANUFACTURER\_ID as defined in Table 00 of ANSI C12.19-1997 or the DEVICE\_CLASS as defined by Version 2 of ANSI C12.19 registered class

This is subset of the Relative Object Identifier defined in ISO/IEC 8825-1: 1998/Amd.1: 1999 (E) and expressed by the <relative-uid-element>) follows:

The leading Relative Object Identifier introducer, 0D<sub>H</sub>, and length fields are not present in the Table 00 element. The length is assumed to be 4. Then each sub-identifier is represented as a series of (one or more) bytes. Bit 7 of each byte indicates shall be cleared to zero when it is the last member in the series. Bit 7 of each preceding octets in the series is set one. Bits 6-0 of the byte in the series collectively encode the sub-identifier concatenated to form an unsigned binary number whose most significant bit is bit 6 of the first octet and whose least significant bit is bit 0 of the last octet. Each sub-identifier value shall be encoded in the fewest possible bytes such that the leading octet of the sub-identifier will not have bit 7 set to one.

When the sub-identifiers that make up the device-class are encoded in this way, they will always result in four bytes of data.

For example if the value reported by MANUFACTURER\_ID is "TEMP", representing the relative object identifier 84.69.77.80 it is encoded as 54<sub>H</sub> 45<sub>H</sub> 4D<sub>H</sub> 50<sub>H</sub>. It shall be interpreted as a Relative UID encoding of 0D<sub>H</sub> 04<sub>H</sub> 54<sub>H</sub> 45<sub>H</sub> 4D<sub>H</sub> 50<sub>H</sub>.

Similarly when the value reported by DEVICE\_CLASS is 8571.3 or encoded as C2<sub>H</sub> 7B<sub>H</sub> 03<sub>H</sub> 00<sub>H</sub> it is interpreted as a Relative UID encoding of 0D<sub>H</sub> 04<sub>H</sub> C2<sub>H</sub> 7B<sub>H</sub> 03<sub>H</sub> 00<sub>H</sub>.}

<ap-title> ::= <universal-id-element> | <relative-uid-element>

{ApTitle of the C12.22 Node to be registered. The field is optional and when not provided, its content length is set to zero, implying that the ApTitle shall be automatically obtained from an authoritative proxy Relay. When the <ap-

<title> is expressed as <relative-uid-element> then it shall be encoded relative to <application-context-oid>.0. The relative portion (which follows the <application-context-oid>.0) of a registered ap-title shall not include assigned or reserved subbranches.}

<electronic-serial-number> ::= <universal-id-element> | <relative-uid-element>  
 {Unique ISO object identifier assigned to this C12.22 Device by its owner and provided for registration authentication. The field is optional and when not provided, its length is set to zero. This is the same number as appears as ELECTRONIC\_SERIAL\_NUMBER in Table 122 Interface Control Table.}

<address-length> ::= <byte> {Number of bytes in <native-address>.}

<native-address> ::= <byte>\* {Native address to use to forward messages to this node. This field is optional if lower layers of the protocol already provide it.

When defined for a specific protocol stack, this field can be subdivided in sub-fields to provide address type and address data to facilitate address resolution.}

<registration-period> ::= <word24> {Maximum period in seconds desired by the C12.22 Node to elapse between successive re-registration requests (keep-registration-alive). The value 0 implies that a re-registration life-timer is not supplied by the registering node.}

### Response:

The response <nok> indicates that the registration was rejected for some reason by the Master Relay. The Calling ApTitle shall be set to the ApTitle of the Master Relay that responded. The response <ok> indicates that this ApTitle's <universal-id-element> has been registered and routing tables have been updated.

<register-r> ::= <nok> |  
 <ok> <reg-ap-title> <reg-delay> <reg-period> <reg-info>

<reg-ap-title> ::= <universal-id-element> | <relative-uid-element>  
 {Registered ApTitle to be used by the C12.22 Node for future communication on this route. When the <reg-ap-title> is expressed as <relative-uid-element>, it shall be encoded relative to the <application-context-oid>.0. This ensures that even when relative encoding is used, the C12.22 Node can determine its absolute position in the C12.22 Network hierarchy. The relative portion (which follows the

		<application-context-oid>.0) of a registered ap-title shall not include assigned or reserved subbranches.)
<reg-delay>	::= <word16>	{Maximum delay in seconds that the device should wait before registering after a power up. A value of 3600 seconds shall be used as a default if this value cannot be retained. The actual delay used to re-register shall be a random value between 0 and this value.}
<reg-period>	::= <word24>	{Maximum period in seconds allowed to elapse between successive re-registration requests (keep-registration-alive). The device is automatically un-registered when this limit is reached. The value 0 implies that a re-registration is not required, the registration is static.}
<reg-info>	::= <byte>	<p>{Bit 0: DIRECT_MESSAGING_AVAILABLE Indicates whether direct messaging is available and provides performance benefit on this network segment. 0 = Use message forwarding only. 1 = Direct messaging is the preferred delivery method. When this mode is used then the C12.22 Node may optionally need to invoke the &lt;resolve&gt; service in order to discover the relative ApTitle of its local C12.22 Network Segment.</p> <p>Bit 1: MESSAGE_ACCEPTANCE_WINDOW_MODE Set to one (1) to indicate that this C12.22 Node may enable its incoming message acceptance window. Set to zero (0) to indicate that this C12.22 Node shall not enable its incoming message acceptance window.</p> <p>Bit 2: PLAYBACK_REJECTION_MODE Set to one (1) to indicate that this C12.22 Node may enable its playback rejection mechanism. Set to zero (0) to indicate that this C12.22 Node shall not enable its playback rejection mechanism.</p> <p>Bits 3: Reserved, shall be set to zero.</p> <p>Bit 4: CONNECTIONLESS_MODE This bit indicates whether this C12.22 Node shall enable its connectionless-mode communication capability. This bit is set to one (1) when the may enable its connectionless-oriented protocol on its local network segment. This bit is set to zero (0) when the local network segment shall not enable its connectionless-oriented protocol. When CONNECTION_MODE is set to zero (0) and CONNECTIONLESS_MODE is set to one (1) then</p>



this node shall use connectionless-mode communication exclusively on its local network segment.

Bit 5: ACCEPT\_CONNECTIONLESS

This bit is set to one (1) when the local network segment uses a connectionless-oriented protocol (Bit 4 set to 1) and the registering node shall accept unsolicited incoming connectionless messages. This bit is set to zero (0) when the registering node may not accept unsolicited incoming connectionless messages.

Bit 6: CONNECTION\_MODE

This bit indicates whether this C12.22 Node shall enable its connection-mode communication capability. This bit is set to one (1) when the local network segment may use a connection-oriented protocol. This bit is set to zero (0) when the local network segment shall not use a connection-oriented protocol. When CONNECTIONLESS\_MODE is set to zero (0) and CONNECTION\_MODE is set to one (1) then this node shall use connection-mode communication exclusively on its local network segment.

Bit 7: ACCEPT\_CONNECTIONS

This bit is set to one (1) when the local network segment uses a connection-oriented protocol (Bit 6 set to 1) and the registering node shall accept incoming connections. This bit is set to zero (0) when the registering node may not accept incoming connections.}

#### 5.3.2.4.11 Deregistration Service

The Deregistration Service is used to remove routing table entries of C12.22 Relays, Master Relays and provide service discontinuation to all of the C12.22 Master Relay authentication and notification hosts.

Request:

```
<deregister> ::= 24H <ap-title> {Request to deregister the <ap-title>
supplied.}
```

Response:

The response <nok> indicates that deregistration was rejected for identified <ap-title>. The response <ok> indicates that the requested <ap-title> was deregistered and routing tables have been updated. If the <ap-title> was not registered when the request was received, the response shall be <ok>.

```
<deregister-r> ::= <nok> |
<ok>
```

#### 5.3.2.4.12 Resolve Service

The Resolve Service is used to retrieve the native network address of a C12.22 Node. The native address is used to communicate directly with other C12.22 Nodes on the same native network. This mode of communication is referred throughout as “direct messaging”.

The Resolve request contains, as a parameter, the ApTitle of the C12.22 Node whose native address is requested. This request is carried in an ACSE Application Data Unit <acse-pdu> with the <aptitle> parameter set to the ApTitle of requested node, the <calling-AP-title> set to the ApTitle of requesting node and the <called-AP-title> set to the ApTitle of the requested C12.22 Relay. When the Calling ApTitle of requesting node is not known then this element is not present. The response shall be directed using the originating network address to the node that initiated the resolve service request.

On network segments capable of broadcast, this service can also be used to retrieve native addresses of C12.22 Relays. A node that wants to retrieve the list of local C12.22 Relays shall initiate a resolve request with the called ApTitle absent, and optionally the calling ApTitle absent (if not known). The requested ApTitle included in the request <ap-title> can have the following values:

- When the Master Relay ApTitle is pre-configured in the requesting node, the <ap-title> is set to this value. Every C12.22 Relay capable of forwarding information to this ApTitle shall return a Resolve response with its own ApTitle set as <calling-AP-title>, its Native Address set as <local-address> and its local C12.22 Network Segment <relative-uid-element> that designates the C12.22 Relay location relative to the top of the C12.22 Network hierarchy.
- When the Master Relay ApTitle is auto-assigned (not known), the requested <ap-title> length is shall be set to zero. Then every C12.22 Relay that has Master Relay ApTitle Auto-Assignment capability shall return a Resolve response with its own ApTitle set as <calling-AP-title>, its local address set as <local-address> and its local C12.22 Network Segment <relative-uid-element> that designates the C12.22 Relay location relative to the top of the C12.22 Network hierarchy

When responses arrive from more than one C12.22 Relay, the node may register through one or more of these C12.22 Relays. By registering with multiple C12.22 Relays, the C12.22 Node increases its chances to be located and be serviced. A different ApTitle or subbranch of the same ApTitle shall be used to register to each C12.22 Relay. It is the responsibility of the node that registers through multiple C12.22 Relays to maintain each route so it doesn't become obsolete.

This service can also be used to retrieve information about specific C12.22 Relays using a unicast message. A node that wants to retrieve ApTitle of a specific C12.22 Relay shall initiate a resolve request, optionally with the <called-AP-title> absent (if not known), and optionally the <calling-AP-title> absent (if not known) to the native address of the C12.22 Relay, which has to be known to the C12.22 Node either through prior configuration or from a prior network transaction). The responding C12.22 Relay shall return a Resolve response with its own ApTitle set as <calling-AP-title> and its Native Address set as <local-address>. The requested ApTitle included in the request <ap-title> shall have the following values:

- The ApTitle of the target C12.22 Relay or
- a <universal-uid-element> with the length field set to zero (0).

#### Request:

```
<resolve> ::= 25H <ap-title> {ApTitle of the requested C12.22 Node.}
```

#### Response:

The resolve <uat> is returned when the caller's ApTitle is unknown by the requested C12.22 Relay. The response <ok> indicates that the ApTitle has been found and its local address is returned.

```
<resolve-r> ::= <uat> |
```

```

        <ok> <local-address-length> <local-address>

<local-address-length> ::= <byte> {Length of <local-address> in bytes.}

<local-address> ::= <byte>* {Local address of the requested ApTitle}

```

#### 5.3.2.4.13 Trace Service

The Trace Service is used to retrieve the list of C12.22 Relays that have forwarded this C12.22 Message to a target C12.22 Node. This service is carried in an ACSE Application Data Unit <acse-pdu> with the Calling ApTitle set to the ApTitle of the node that have requested the trace and the Called ApTitle set to the ApTitle of the node traced.

Each time a C12.22 Relay receives this service request, it adds its own ApTitle to the list of C12.22 Relays stored in the User Information Element, <user-information-element> then it forwards it to the next C12.22 Relay. When the trace request reach the C12.22 Relay that have the target node as neighbor (the node whose ApTitle matches the Called ApTitle), this C12.22 Relay shall include its ApTitle in the list, replace the service code by a <ok> response code, set the Called ApTitle to the ApTitle of requesting node, set its own ApTitle as Calling ApTitle and return this information.

It is important to note that a Trace Service is only processed by the C12.22 Relays and is not processed by the target nodes and does not need to be implemented by these target nodes.

##### Request:

```

<trace> ::= 26H <ap-title>*

```

##### Response:

The response <nok> is returned when this request cannot be serviced. In this case, the response contains a list of <ap-title>s of all C12.22 Relays traversed up to the point of failure; i.e., the last <ap-title> is that of the C12.22 Relay rejecting the service request.

The response <ok> indicates that the Trace Service was successful and the response contains all C12.22 Relays used to forward this information.

```

<trace-r> ::= <nok> <ap-title>* |
               <ok> <ap-title>*
               {ApTitle of C12.22 Relays used to forward
               this service.}

```

#### 5.3.2.5 Service sequence state control

In a networking environment, the C12.22 Nodes may support one or multiple associations at the same time. Any association may be session oriented or sessionless. For each association supported, the C12.22 Nodes shall conform to the following state:

- |                 |  |
|-----------------|--|
| Off-line State: | Normal state upon C12.22 Node power-up. This is also the state upon completion of a Terminate or a Link Control interface-disable service request. An Off-line State implies that the C12.22 Node is not present on the C12.22 Network Segment that services it.                           |
| Idle State:     | Normal state upon an active or passive open of a network connection. A passive open implies that the C12.22 Node is ready to accept transactions from the network. An active open implies that the C12.22 Node is ready to initiate transactions to a peer C12.22 Node across the network. |

Sessionless State: State while processing transaction without entering the Session State.

Session State: State achieved after a Logon Service has been accepted.

The relationship between EPSEM services and service sequence states is:

Identification: This service request is accepted at the Idle State only. Upon completion, the Application returns to the Idle State.

Wait: This service request is accepted at the Session State only. Acceptance of this request does NOT result in any sequence state changes.

Logon: This service request is accepted at the Idle State only. Acceptance of this request results in a transition to the Session State. This service is optional and not implemented when NBR\_SESSION\_SUPPORTED returned by the Identification Service is set to 0.

Security: This service request is accepted at the Session State only. Acceptance of this request does NOT result in any sequence state changes.

Read and Write: These requests are accepted in Session State when NBR\_SESSION\_SUPPORTED, returned by the Identification Service, is greater than zero. In this case, acceptances of these requests do NOT result in any sequence state changes. They are also supported in Sessionless State when the SESSIONLESS\_SUPPORTED, returned by the Identification Service, is set to TRUE. In this case, the Application returns to the Idle State.

Logoff: This service request is accepted at the Session State only. Acceptance of this request results in a transition to the Idle State. This service is optional; however, it shall be supported if the Logon Service is supported. The Logoff Service is a request to initiate a normal Session termination. This may be used by C12.22 Nodes to complete Session-related data processing following the successful termination of the Session.

Terminate: This service request is accepted at the Session State only. Acceptance of this request results in a transition to the Idle State. This service is optional; however, it shall be supported if the Logon Service is also supported. The Terminate Service is an abnormal Logoff Service. Upon completion, the C12.22 Node returns the Idle State.

Disconnect: This optional service request is accepted in any state. When received in the Session State, an Application shall perform a Terminate Service then disconnect from the C12.22 Network. Upon completion, the C12.22 Node returns the Off-line State.

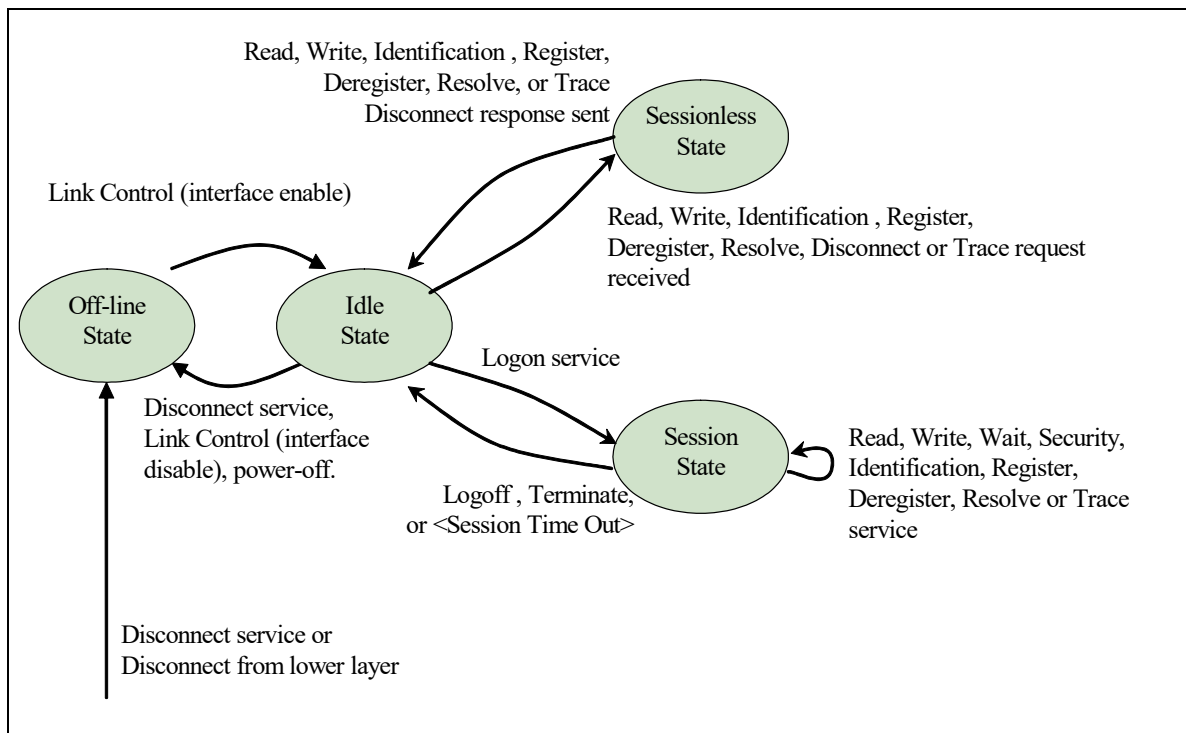


Figure 5.2: C12.22 Host Application Layer State Diagram

**5.3.2.6 Partial Table access using index/element-count Method**

1. An index sets up a start of selection into a Table object relative to the beginning of the Table as follows:
  - Each member of a PACKED RECORD is assigned a unique number.
  - The positional number of the first element of a PACKED RECORD is assigned the value zero.
  - The positional number of subsequent elements in the same PACKED RECORD is incremented by one for each subsequent element in the PACKED RECORD.
  - Each sub-element of a BIT FIELD is assigned a unique positional number.
  - The positional number of the first sub-element of a BIT FIELD is assigned the value zero.
  - The positional numbers of subsequent sub-elements in the same BIT FIELD are incremented by one for each subsequent sub-element in the BIT FIELD, independent of the bit range assigned to the sub-element.
  - Positional numbers are assigned independently of any IF or CASE statements that may be present inside PACKED RECORDS or BIT FIELDS, as if the elements or sub-elements were not enclosed within any IF or SWITCH statements.
  - For non-final elements one level of index is appended to the index of the parent's element index for use in selections.
  - Selection of Boolean members within a SET is referenced in the same manner as members of a single dimensional array.
  - For elements of an ARRAY one level of index is appended to the index of the array's element for each dimension (as per BNF.dim) for use in selections into entries of the ARRAY.

2. Selection based on an index method using <element-count> equal to one will deliver the whole selected element.
3. For the purpose of binary transmission, <index>+ cannot select into a sub-element or final elements that are not atomic, with the exception of SETs, where the first octet selected for transmission is that computed by integer division of the atomic index number requested by eight (8), and the number of elements is the number of bits requested.
4. For the purpose of transmission, an indexed selection into a non-existing element shall result in an "Inappropriate Action Requested" error. However, it is acceptable to append zeros at the end of an <index>+ to indicate the desired access level of an indexed selection within the Table element hierarchy.
5. When <element-count> is greater than one, the application shall return up to <element-count> elements at the same or lower hierarchical level of the <index>+ used to initiate the request traversing all element types serially.
6. When <element-count> is greater than the number of elements available for transmission, the number of elements transmitted will be limited to the maximum number elements available at the same or lower hierarchical level of the <index>+ used to initiated the request..
7. The number of numeric segments that make up the <index>+ defines the initial hierarchical level for element serialization and for <element-count> interpretation.
8. As a part of a Read Service request, <element-count> equal to zero shall be interpreted as **ALL** data starting from the <index>+ to the end of the Table requested.
9. As a part of a response, <count> equal to zero shall be interpreted as **NO** data was received.
10. As a part of a Write Service request, <count> shall correctly represent the actual number of bytes to be written, including the optional pending header, at the hierarchical level of the selection start <index>+.
11. As a part of a Read Service request, <element-count> represents the maximum number of elements requested.
12. Any element excluded from the data stream through the use of the IF or CASE conditional statements shall not be a candidate for transmission and shall not be counted.
13. Any element excluded from the data stream through the use of zero length ARRAYS, SETs, STRINGs, BINARYs or BCDs shall not be a candidate for transmission and shall not be counted.
14. Any element whose size is zero shall not be a candidate for transmission and shall not be counted.
15. The <element-count> counts elements and not octets.
16. If the respondent application does not support the transmission elements at the requested index level, or the respondent application cannot deliver the element requested from an ARRAY the respondent application shall assert the error condition "Inappropriate Action Requested". The requester may then attempt a retry of the Read or Write Service request on an <index>+ of an element that is higher or lower in hierarchy relative to the <index>+ of the failed attempt.

#### Index Count Access Method Examples

The following are examples for the use of the Index/Element-Count method to select data.

Example 1	Example 2	Example 3	Example 4
<index> = 1.0 <element-count> = 2	<index> = 1, <element-count> = 2 or <index> = 1.0, <element-count> = 4	<index> = 1.2.0, <element-count> = 4	<index> = 1.2, <element-count> = 4 or <index> = 1.2.0, <element-count> = 5
0	0	0	0
1.0 (Selected)	1.0 (Selected)	1.0	1.0
1.1 (Selected)	1.1 (Selected)	1.1	1.1
1.2	1.2 (Selected)	1.2 (Selected)	1.2 (Selected)
2	2 (Selected)	2 (Selected)	2 (Selected)
3.0	3.0	3.0 (Selected)	3.0 (Selected)
3.1.0	3.1.0	3.1.0 (Selected)	3.1.0 (Selected)
3.1.1	3.1.1	3.1.1	3.1.1 (Selected)
3.2	3.2	3.2	3.2
4	4	4	4

### 5.3.2.7 Partial Table access using offset/octet-count method

1. An <offset> sets up a start of selection into a Table object relative to the beginning of the Table.
2. <offset> zero (0) is the octet offset to the first octet of the first object in the Table as prescribed by the object data type and the value of DATA\_ORDER, found in the GEN\_CONFIG\_TBL (Table 00).
3. When <count> is greater than one, the application shall return no more than <count> octets from <offset> used to initiate the request traversing all element types serially, where each element will be transferred according to its type and the value of DATA\_ORDER, found in the GEN\_CONFIG\_TBL (Table 00).
4. When <count> is greater than the number of octets available for transmission, the number of octets transmitted will be limited to the maximum number octets available. The response <count> shall be adjusted to reflect the actual number of octets transferred in the response.
5. As a part of a request, <count> equal to zero shall be interpreted as **ALL** data starting from the <offset> to the end of the Table requested.
6. As a part of a response, <count> equal to zero shall be interpreted as **NO** data was received.
7. As a part of a Write Service request, <count> shall correctly represent the actual number of octets requested to be written starting at the Table offset requested including the optional pending header.
8. As a part of a Read Service request, <count> represents the maximum number of octets requested including the optional pending header.
9. Elements that are not present in the Table, by virtue of their being excluded from the data stream through the use of zero length ARRAYs, SETs, STRINGs, BINARYs or BCDs, shall not be candidates for transmission and not be counted.
10. Any element whose size is zero shall not be a candidate for transmission and shall not be counted.
11. The octet counter counts octets and not elements.

12. If the respondent application does not support the transmission octets at the requested offset if the respondent application shall assert the error condition "Inappropriate Action Requested".

### 5.3.3 EPSEM Envelope Structure

The EPSEM structure enables transportation of multiple requests and responses at the same time. It also provides response control and C12.19 Device Class. When <epsem> contains more than one <epsem-data> request, the corresponding <epsem-data> responses shall be returned in the same order such that there is a one-to-one matching between <epsem-data> requests and <epsem-data> responses contained in a single <epsem>.

```
<epsem> ::= <epsem-control> [<ed-class>] <epsem-data>+ | 00H
```

```
<epsem-control> ::= <byte> {Datagram control field.  
Bit 7: Reserved  
Shall be equal to 1
```

#### Bit 6: RECOVERY\_SESSION

Flag used to initiate a session for which the session establishment request is composed of a single Logon request and its response are not subject to the restrictions imposed by the message accepted window or playback rejection mechanism. It is important to note that the core of the session is intrinsically protected against playback by the initial exchange of IVs.

To initiate a node recovery session, a message is sent to a C12 Node with this flag set and with the <epsem-data> carrying a single <logon> service. On receipt, the C12.22 Node shall validate the authenticity and content of this message, but disregard the acceptance window and/or playback rejection requirements, subject to the C12.22 Node capabilities stipulated in Table 121.

#### Bit 5: PROXY\_SERVICE\_USED

0 = This message has not been send though a proxy C12.22 Relay.  
1 = This message was sent though a proxy C12.22 Relay and the <called-AP-title-element> shall not be included in the computation of the <mac>.

#### Bit 4: ED\_CLASS\_INCLUDED

When set to true, <ed-class> is included in this <acse-pdu>. <ed-class> shall be included in all unsolicited messages and one-way messages.

#### Bit 2 to 3: SECURITY\_MODE

0 = Cleartext  
1 = Cleartext with authentication  
2 = Ciphertext with authentication



Bit 0 to 1: RESPONSE\_CONTROL  
 Used by request messages to control the  
 return of corresponding responses.  
 0 = Always respond  
 1 = Respond on exception  
 2 = Never respond

RESPONSE\_CONTROL shall be set to 2 by all  
 one-way C12.22 Nodes. }

```

<ed-class>      ::= <byte>+4      {DEVICE_CLASS exactly as encoded in the
                                   General Configuration Table (Table 0) of
                                   ANSI C12.19 (Version 2.0) or the
                                   MANUFACTURER code as defined in the
                                   General Configuration Table (Table 0) of
                                   ANSI C12.19-1997 (Version 1.0).}

<epsem-data>    ::= <service-length> <req-res>

<service-length> ::= <byte>+      {Length of <req-res> field, encoded as
                                   defined by ISO/IEC 8825-1:2002 [BER]. When
                                   <service-length> is zero then it signifies
                                   the end of the <epsem-data> list.}

<req-res>       ::= <request> | <response>
                                   {EPSEM request or response as defined in
                                   section 5.3.2.2 "Response Codes".}
  
```

### 5.3.4 Association Control - Association Control Service Element (ACSE)

EPSEM relies on the use of Connectionless-mode ACSE (ISO/IEC 15955:1999) to convey association and security parameters. This includes the identification of the application context <aSO-context-element>, application process titles of called and calling process <called-AP-title-element> and <calling-AP-title>, authentication information if a secure transaction is required <mechanism-name-element> and <calling-authentication-value-element>. The encoding of the elements of Connectionless ACSE is based on ISO/IEC 10035-1:1995 and ISO/IEC 8825-1:2002, although represented here using the BNF notation whose production rules is identical to that of ASN.1 / BER.

The following syntax represents a conformant subset of those fields defined in the Connectionless ACSE standards for the services used. The Standard also extends the data type AP-title that is referenced by ISO/IEC 15955:1999 and imported from ISO/IEC 15954: 1999 by adding an ap-title-form4 to facilitate the inclusion of an ASN.1 RELATIVE\_OID in the definitions of <calling-AP-title-element> and <called-AP-title-element>.

The revised ASN.1 syntax that is referenced by ANSI C12.22 is as follows:

```

AP-title ::= CHOICE {
  ap-title-form1  AP-title-form1,      -- Not used by ANSI C12.22
  ap-title-form2  AP-title-form2,      -- Used by ANSI C12.22
  ...,
  ap-title-form3  AP-title-form3,      -- Not used by ANSI C12.22
  ap-title-form4  [0] IMPLICIT AP-title-form4 -- Used by ANSI C12.22
}
  
```

where

```

AP-title-form4 ::= RELATIVE-OID
  
```

When required or optional components of ACSE are present, they shall appear in the order presented below.

```

<acse-pdu> ::= 60H
               <elements-length>
               <elements>

<elements-length> ::= <byte>+ {Length of <elements>, encoded as defined
                               by ISO/IEC 8825-1:2002 [BER].}

<elements> ::= <unsegmented-acse-elements> |
               <segmented-acse-elements>
               {The <acse-pdu> that makes up a fully
                assembled application layer data unit or
                the <acse-pdu> that makes up a single
                application sub-layer data unit segment.
                For more details see section 5.3.5
                "Application Segmentation ".}

<unsegmented-acse-elements> ::= [ <aSO-context-element> ]
                                [ <called-AP-title-element> ]
                                [ <called-AP-invocation-id-element> ]
                                [ <calling-AP-title-element> ]
                                [ <calling-AE-qualifier-element> ]
                                <calling-AP-invocation-id-element>
                                [ <mechanism-name-element> ]
                                [ <calling-authentication-value-element> ]
                                <user-information-element>

<segmented-acse-elements> ::= [ <aSO-context-element> ]
                              [ <called-AP-title-element> ]
                              <called-AE-qualifier-element>
                              [ <calling-AP-title-element> ]
                              [ <calling-AE-qualifier-element> ]
                              <calling-AP-invocation-id-element>
                              <segment-user-information-element>

```

#### **5.3.4.1 Application Context Element (A1<sub>H</sub>)**

The Application Context Element <aSO-context-element> is used to uniquely identify the ANSI C12.22 Application from any other Application Layer that uses ACSE. This uniqueness is guaranteed by the use of a registered Universal Identifier. This specification allows messages that do not include the Universal Identifier. This can be done for efficiency reasons only on network segments dedicated to the ANSI C12.22 application (e.g. a homogeneous C12.22 Network), and this field must be reinserted when the message is relayed to a heterogeneous application network where ACSE frames of other contexts may be present.

```

<aSO-context-element> ::= A1H <aSO-context-length> <aSO-context>

<aSO-context-length> ::= <byte>+ {Length of <aSO-context>, encoded as
                                   defined by ISO/IEC 8825-1:2002 [BER].}

<aSO-context> ::= 06H <universal-id-length> <universal-id>
                 {Object identifier representing this
                  Application Layer (ANSI C12.22). This
                  value shall be set to the Standard
                  application context, <application-context-
                  oid>, defined in Annex D.}

```

#### **5.3.4.2 Called AP Title Element (A2<sub>H</sub>)**

The Called AP Title Element <called-AP-title> uniquely identifies the target of an ACSE message. This uniqueness is guaranteed by the use of an absolute or relative Universal Identifier. Relative Universal Identifiers are derived from the ANSI C12.22 ApTitle branch (<application-context-oid>.0) and are described in section 5.2.3 “Universal Identifiers Encoding”.

```
<called-AP-title-element> ::= A2H <called-AP-title-length> <called-AP-title>

<called-AP-title-length> ::= <byte>+
                               {Length of <called-AP-title> encoded as
                                defined by ISO/IEC 8825-1:2002 [BER].}

<called-AP-title> ::= <universal-aptitle-element> | <relative-aptitle-
                               element>
```

#### **5.3.4.3 Calling AP Title Element (A6<sub>H</sub>)**

The Calling AP Title Element <calling-AP-title> uniquely identifies the initiator of an ACSE message. This uniqueness is guaranteed by the use of an absolute or relative Universal Identifier. When relative then it is derived from the ANSI C12.22 ApTitle branch of the <application-context-oid>.0.

The <calling-AP-title-element> can be omitted when the target is on the same Network Segment or when Relay proxy services are available. This may be done for efficiency of communication or when the C12.22 Node's ApTitle is not assigned.

```
<calling-AP-title-element> ::= A6H <calling-AP-title-length> <calling-AP-
                               title>

<calling-AP-title -length> ::= <byte>
                               {Length of <calling-AP-title> encoded as
                                defined by ISO/IEC 8825-1:2002 [BER].}

<calling-AP-title> ::= <universal-aptitle-element> | <relative-aptitle-
                               element>
```

#### **5.3.4.4 Universal Identifier of Called and Calling AP Title Element (06<sub>H</sub>)**

A Universal Identifier that uniquely identifies a network address. See section 5.2.3 “Universal Identifiers Encoding” for more information.

```
<universal-aptitle-element> ::= 06H <universal-id-length> <universal-id>
```

#### **5.3.4.5 Relative Universal Identifier of Called and Calling AP Title Element (80<sub>H</sub>)**

For efficiency reasons, Relative Universal Identifiers <relative-aptitle-element> may be used to identify C12.22 Nodes, their interfaces, and Local Ports. Within an ANSI C12.22 Network, the Relative UID of a C12.22 Node shall be unique.

For internal communication, C12.22 Devices, C12.22 Communication Modules, and Local Ports may use Relative UIDs that begin with Assigned or Reserved subbranches within a C12.22 Node, as detailed in section 5.3.4.12. While this form of addressing must be unique within the C12.22 Node, it is likely that it will not to be unique within the C12.22 Network or across C12.22 Nodes; therefore Relative UIDs that begin with Assigned or Reserved subbranches shall not be present on the C12.22 Network.

```
<relative-aptitle-element> ::= 80H <relative-uid-length> <relative-uid>
```

#### **5.3.4.6 Calling Application Entity Qualifier Element (A7<sub>H</sub>)**

The <calling-AE-qualifier-element> is an optional element used to qualify the information transferred by

the application layer entity.

```

<calling-AE-qualifier-element> ::= A7H <calling-AE-qualifier-integer-length>
    <calling-AE-qualifier-integer>
    {The calling application entity qualifier.
    When the target device does not support
    this type of element qualification it will
    ignore it with no error. The initiator of
    the service can identify if the target
    supports this element by the presence of
    this element in the response.}

<calling-AE-qualifier-integer-length> ::= <byte>+
    {The size of the universal integer that
    encapsulates the <calling-AE-qualifier-
    integer>, encoded as defined by ISO/IEC
    8825-1:2002 [BER].}

<calling-AE-qualifier-integer> ::= 02H <calling-AE-qualifier-length>
    <calling-AE-qualifier>
    {The universal integer object that
    contains the <calling-AE-qualifier>..}

<calling-AE-qualifier-length> ::= <byte>+
    {Length of <calling-AE-qualifier>, encoded
    as defined by ISO/IEC 8825-1:2002 [BER].}

<calling-AE-qualifier> ::= <byte>+ {Application entity qualifier value:
    Bit 0 : TEST
    When set to 1, this message has to be
    interpreted as a test message. The called
    C12.22 Node shall ignore this message if
    it does not support the test feature.
    Otherwise it shall process the test
    message in such a manner that it does not
    affect the operation or the end state of
    the called node; then it shall include a
    <calling-AE-qualifier-element> in its
    response <acse-pdu> with bit 0 set to 1
    (response <acse-pdu> is a test response,
    thus acknowledging the fact that it
    supports tests).

    Bit 1 : URGENT
    Messages tagged with the calling
    application entity value bit 1 set to 1
    are expected to be forwarded by all C12.22
    Relays and acted upon by the called C12.22
    Node urgently (high priority).

    Bit 2: NOTIFICATION
    When this bit is set to true, write
    services contained within this <acse-pdu>
    shall be interpreted as notifications of
    information issued by the initiator of
    this message. No other services shall be
    affected by the state of this bit. All
    unsolicited notification shall also
    include the <epsem>s <ed-class>, i.e.
    <epsem-control>s ED_CLASS_INCLUDED bit
    shall be set to true. When NOTIFICATION is
  
```

set to true it is an indication that the information supplied in the <acse-pdu>s <user-information-element> is about the calling C12.22 Node using the operating model indicated by the provided <ed-class>. When NOTIFICATION is cleared to zero, this is information for the called C12.22 Node.

Bit 3 to 7: Reserved.}

#### **5.3.4.7 Mechanism Name Element (8B<sub>H</sub>)**

The Mechanism Name Element <mechanism-name> uniquely identifies the security mechanism used in the containing <acse-pdu>. This uniqueness is guaranteed by the use of a registered Universal Identifier. This element identifies the format of the Authentication Value Element <calling-authentication-value> and <user-information> and security algorithms involved. This element is optional and when not included, the default security mechanism defined by this document applies (<application-context-oid>.2.1).

<mechanism-name-element> ::= 8B<sub>H</sub> <mechanism-name-length> <mechanism-name>

<mechanism-name-length> ::= <byte>+  
{Length of <mechanism-name> encoded as defined by ISO/IEC 8825-1:2002 [BER].}

<mechanism-name> ::= <universal-id>  
{Object identifier representing the security mechanism used. Note: There is no ASN.1 tag and length added to <universal-id> since it is defined as IMPLICIT.}

#### **5.3.4.8 Calling Authentication Value Element (AC<sub>H</sub>)**

The structure and content of the <calling-authentication-value-element> is implied by the value of the authentication mechanism name identified in the message (either included or default) and further qualified by the <calling-authentication-value-encoding> described below.

The discussion in this section applies only to any <acse-pdu> identified by <mechanism-name> <application-context-oid>.2.0 and <application-context-oid>.2.1. These are Standard-registered authentication-mechanism names. Other mechanism names can be registered. When a registered mechanism can be expressed using the ASN.1 syntax then it shall be replace the element <calling-authentication-value-other-asn1>, otherwise it shall be encoded as <calling-authentication-value-other-octets>.

The optional Authentication Value Element <calling-authentication-value-element> is used to carry privacy and authentication parameters. When it contains an <calling-authentication-value-c1221> the <user-information> shall be transmitted unencrypted and the SECURITY\_MODE value of the <epsem-control> field shall be set to 1. When it is not included, the SECURITY\_MODE value of the <epsem-control> field shall be set to 0 (Note: In these cases the fields <mac> and <padding> in <user-information> are not included - see section 5.3.4.11 "User Information Element (BEH)").

When <calling-authentication-value-c1222> is included then the <user-information> is authenticated and private (when the SECURITY\_MODE value of the <epsem-control> field is set to 2), or just authenticated (when the SECURITY\_MODE value of the <epsem-control> field is set to 1). Note that the <epsem-control> field is transmitted as Cleartext (i.e. it may be authenticated, but never encrypted).

The C12.22 protocol for the exchange of the contents of <calling-authentication-value-element> is described in section 5.3.4.13 "C12.22 Security Mechanism". Other <mechanism-name> values may imply entirely different content for the <calling-authentication-value> and the protocols used.

```

<calling-authentication-value-element> ::= ACH <calling-authentication-value-length>
                                     <calling-authentication-value-external>

<calling-authentication-value-length> ::= <byte>+
                                     {Length of <calling-authentication-value-external> encoded as defined by ISO/IEC 8825-1:2002 [BER].}

<calling-authentication-value-external> ::= A2H <calling-authentication-value-external-length>
                                     [ <calling-authentication-value-indirect-reference> ]
                                     <calling-authentication-value-encoding>

<calling-authentication-value-external-length> ::= <byte>+
                                     {Length of the optional <calling-authentication-value-indirect-reference> and <calling-authentication-value-encoding> encoded as defined by ISO/IEC 8825-1:2002 [BER].}

<calling-authentication-value-indirect-reference> ::= 02H 01H 00H
                                     {Reserved.}

<calling-authentication-value-encoding> ::= <calling-authentication-value-single-asn1> | <calling-authentication-value-octet-aligned>

<calling-authentication-value-single-asn1> ::= A0H <calling-authentication-value-asn1-length> <calling-authentication-value-asn1>
                                     {The authentication value is represented by another ASN.1 module. This syntax shall be consistent with the value and algorithm assumed by the <mechanism-name>. Two modules are currently defined by this Standard: <calling-authentication-value-c1222> and <calling-authentication-value-c1221>..}

<calling-authentication-value-asn1-length> ::= <byte>+
                                     {Length of <calling-authentication-value-asn1> encoded as defined by ISO/IEC 8825-1:2002 [BER].}

<calling-authentication-value-asn1> ::=
                                     <calling-authentication-value-c1222> |
                                     <calling-authentication-value-c1221> |
                                     <calling-authentication-value-other-asn1>

<calling-authentication-value-octet-aligned> ::= 81H <calling-authentication-value-length> <calling-authentication-value-other>
                                     {The authentication value is represented by non-ASN.1 syntax. This syntax shall be consistent with the value and algorithm assumed by the <mechanism-name>. Currently none are defined by this Standard.}

```

**5.3.4.8.1 C12.22 Security Mechanism (<application-context-oid>.2.1)**

```
-- ASN.1 syntax for <calling-authentication-value-cl222>.

Module C12.22-Security-Mechanism {iso(1) member-body(2) us(840) c1219-
standard(10066) mechanism(12) cl222(1) version(1)}
DEFINITIONS ::=
BEGIN

Calling-authentication-value-cl222 ::= [1] IMPLICIT SEQUENCE {
    key-id-element      [0] IMPLICIT OCTET STRING (SIZE(1)) OPTIONAL,
    iv-element          [1] IMPLICIT OCTET STRING (SIZE(4)) OPTIONAL,
}

END
```

<calling-authentication-value-cl222> ::= A1<sub>H</sub> <calling-authentication-value-cl222-length>  
 [ <key-id-element> ]  
 <iv-element>  
 {Authentication or authentication plus  
 privacy encoding parameters that are  
 applicable when using the ANSI C12.22  
 native mechanism name <application-  
 context-oid>.2.1}.

<calling-authentication-value-cl222-length> ::= <byte><sup>+</sup>  
 {Length of the sequence <key-id-element>,  
 <iv-element> <user-element> and <message-  
 authentication-token>}

<key-id-element> ::= 80<sub>H</sub> <key-id-length> <key-id>  
 {The universal octet string that contains  
 <key-id>.}

<key-id-length> ::= <byte><sup>+</sup> {Length of <key-id> encoded as defined by  
 ISO/IEC 8825-1:2002 [BER].}

<key-id> ::= <byte> {Identifies the key used to encrypt and  
 decrypt the information. This value is  
 matched with an entry in the KEY\_ENTRIES  
 array located in the Security Table (Table  
 45, KEY\_TBL or Table 46, EXTENDED\_KEY\_TBL)  
 of the target C12.22 Node. This element is  
 optional and when not provided in a  
 sessionless message, <key-id> 0 shall be  
 used; otherwise, the previous key id of  
 the session shall be used.

The <key-id> is interpreted as an unsigned  
 integer.

The <key-id> may be different in each  
 C12.22 Message transmitted. The <key-id>  
 used in a request may be different from  
 the <key-id> used in the response message.}

<iv-element> ::= 81<sub>H</sub> <iv-length> <iv>  
 {The initialization vector element used by  
 the authentication and privacy algorithms.}

<code>&lt;iv-length&gt;</code>	<code>::= &lt;byte&gt;+</code>	{Length of <code>&lt;iv&gt;</code> encoded as defined by ISO/IEC 8825-1:2002 [BER].}
<code>&lt;iv&gt;</code>	<code>::= &lt;byte&gt;<sup>4</sup></code>	{ The <code>&lt;iv&gt;</code> element is a 32 bits unsigned integer counter that represents the UTC time in seconds since 00:00:00 January 1, 1970 UTC, with the most significant byte transmitted first. It is used as the timestamp for the C12.22 Message.

During sessionless communication the `<iv>` is used by the authentication nonce creation algorithm. It is also used to implement an acceptance window and playback rejection that may be optionally enabled. If supported, these optional functions may be enabled in the Registration service response.

When the acceptance window is enabled, the C12.22 Node shall reject any sessionless messages with an `<iv>` that is outside the message acceptance window. When comparing the timestamp that is encoded in the sender's `<iv>` with the message receipt time by the receiving C12.22 Node, an uncertainty of  $\pm 15$  minutes shall be allowed.

When the Interface Control Table (Table 122) is implemented, the message acceptance window is controlled by the **MESSAGE\_ACCEPTANCE\_WINDOW** Element.

When playback rejection is enabled, the C12.22 Node shall maintain a history of previous sessionless messages received. Messages received are filtered based on the uniqueness of the `<called-AP-title-element>`, `<calling-AP-title-element>`, `<calling-AP-invocation-id-element>` and `<iv>`. When both the acceptance window and playback rejection are enabled, the acceptance window may be temporary reduced when playback rejection can't be enforced because of the rate of messages received.

For session-oriented communication, only the Logon request and the Logon response carry the initial `<iv>` element. Any subsequent requests and responses within a session use the `<iv>`s that were received during the Logon service request and response. Playback rejection of messages exchanged after the initial login service is achieved by resetting then monotonically incrementing the `<calling-AP-invocation-id>`. This technique is used to provide additional session-mode



duplicate message detection as well as  
detection of sequencing errors.}

#### 5.3.4.8.2 C12.21 Security Mechanism (<application-context-oid>.2.0)

```
-- ASN.1 syntax for <calling-authentication-value-cl221>.

Module C1221-Security-Mechanism {iso(1) member-body(2) us(840) c1219-
standard(10066) mechanism(12) c1221(0) version(1)}
DEFINITIONS ::=
BEGIN
calling-authentication-value-cl221 ::= [0] IMPLICIT CHOICE {
    c1221-auth-identification [0] IMPLICIT OCTET STRING (SIZE(3 | 5..259)),
    c1221-auth-request        [1] IMPLICIT OCTET STRING (SIZE(1..255)),
    c1221-auth-response       [2] IMPLICIT OCTET STRING (SIZE(0 | 1..255)),
    ...
}
END
```

<calling-authentication-value-cl221> ::= A0<sub>H</sub> <c1221-authentication-length>  
<c1221-authentication>  
{Authentication encoding when using the  
ANSI C12.21 compatibility mode over ANSI  
C12.22 to communicate the equivalent of  
the PSEM Authentication Service (53<sub>H</sub>) of  
ANSI C12.21. This mechanism shall only be  
selected when the mechanism name is  
<application-context-oid>.2.0 or when the  
default mechanism is used together with  
the <calling-authentication-value-cl221>  
tag.}

<c1221-authentication-length> ::= <byte>+  
{Length of <c1221-authentication> encoded  
as defined by ISO/IEC 8825-1:2002 [BER].}

<c1221-authentication> ::= <c1221-auth-identification-octet-string> |  
<c1221-auth-request-octet-string> |  
<c1221-auth-response-octet-string>  
{The ANSI C12.21-compatible authentication  
mechanism element <c1221-authentication>  
shall only be issued in conjunction with a  
Logon Service request, a Logon Service  
response or an Identification Service  
response. This element is used to provide  
a two-way authentication with playback  
rejection during a session. The contents  
of <c1221-auth-identification-octet-  
string>, <c1221-auth-request-octet-  
string> and <auth-response-octet-string> fields  
depend on the authentication algorithm  
used in the context of the ANSI C12.21  
protocol. The content and implementation  
of this algorithm are identical to that of  
ANSI C12.21. For complete details see ANSI  
C12.21 Identification and Authentication  
Services.}

<c1221-auth-identification-octet-string> ::= 80<sub>H</sub> <c1221-auth-identification-  
length> <c1221-auth-identification>

{This element may only be present as part of an EPSEM Identification Service response. When it is present, the C12.21 authentication algorithm is requested by the responding C12.22 Node.}

<c1221-auth-identification-length> ::= <byte>+  
 {Length of <c1221-auth-identification> encoded as defined by ISO/IEC 8825-1:2002 [BER]. This length shall be either 3 octets or in the range of 5 to 259 octets in order to satisfy the constraints imposed by ANSI C12.21.}

<c1221-auth-identification> ::= <auth-ser> | <auth-ser-ticket>  
 {See ANSI C12.21 PSEM Identification Service response for encoding details of <auth-ser> and <auth-ser-ticket>.}

<c1221-auth-request-octet-string> ::= 81<sub>H</sub> <c1221-auth-request-length>  
 <c1221-auth-request>  
 {This element may only be present as part of an EPSEM Logon Service request. When it is present, the C12.21 authentication shall be initiated together with the Logon Service request. An EPSEM <ok> response indicates that both the Logon and Authentication Services were successful.}

<c1221-auth-request-length> ::= <byte>+  
 {Length of <c1221-auth-request> encoded as defined by ISO/IEC 8825-1:2002 [BER]. This length shall be in the range of 1 to 255 in order to satisfy the constraints imposed by ANSI C12.21.}

<c1221-auth-request> ::= <auth-request>  
 {This element contains the information used to authenticate the sender of this <acse-pdu> according to the authentication protocol of ANSI C12.21. This field is defined in the C12.21 PSEM Authentication Service and encoded as <auth-request> of ANSI C12.21.}

<c1221-auth-response-octet-string> ::= 82<sub>H</sub> <c1221-auth-response-length>  
 <c1221-auth-response>  
 {This element may only be present as part of an EPSEM Logon Service response. When it is present, the C12.21 authentication shall be supplied as together with the Logon Service response. The field is encoded as <auth-response> of ANSI C12.21.}

<c1221-auth-response-length> ::= <byte>+  
 {Length of <c1221-auth-response> encoded as defined by ISO/IEC 8825-1:2002 [BER]. The length of this field shall be set to zero if the Logon response returns an error. When the Logon response is <ok> it is also an indication that the

authentication was successful. This length shall then be in the range of 1 to 255 in order to satisfy the constraints imposed by ANSI C12.21.}

```
<c1221-auth-response> ::= <auth-response>
    {This element contains the information
    used to authenticate the sender of this
    <acse-pdu> according to the authentication
    protocol of ANSI C12.21. This field is
    defined in the C12.21 PSEM Authentication
    Service and encoded as <auth-response> of
    ANSI C12.21.}
```

#### 5.3.4.8.3 C12.22 Other Security Mechanisms

```
<calling-authentication-value-other-asn1> ::= <byte>*
    {Authentication encoding when using an
    ANSI C12.22 registered mechanism name
    identifier, such as <application-context-
    oid>.2.n, (where n > 1). Actual encoding
    depends on the encoding rules set forth by
    the authentication mechanism registered
    ASN.1 syntax.}
```

```
<calling-authentication-value-other-octets> ::= <byte>*
    {Authentication encoding when using an
    ANSI C12.22 registered mechanism name
    identifier, such as <application-context-
    oid>.2.n, (where n > 1). Actual encoding
    is arbitrary and depends on the encoding
    rules set forth by the authentication
    mechanism.}
```

#### 5.3.4.9 Called AP Invocation ID Element (A4<sub>H</sub>)

The Called Invocation ID element <called-AP-invocation-id-element> provides the called entity Application Layer with information about another (past) APDU that is related to this APDU. A likely user of this capability may be an entity that was requested to respond on exception, or a C12.22 Relay which responds with a <nok> code as a result of a request that could not be processed to completion. It is also provided in an <ok> response to identify the APDU that was the trigger of this APDU response.

The <called-AP-invocation-id-element> shall be placed before the <user-information-element>.

```
<called-AP-invocation-id-element> ::= A4H <called-AP-invocation-id-integer-
length>
    <called-AP-invocation-id-integer>
    {The <called-AP-invocation-id-element> is
    associated with the recipient of this APDU.
    It uniquely binds this APDU to another
    APDU that was sent originally by the
    Application Layer of the recipient of this
    ACSE message. The uniqueness of the
    binding is established by matching the
    called and calling ApTitles of this APDU
    with the members of the <called-AP-
    invocation-id>. The actual encoding rules
    of <called-AP-invocation-id> are identical
```

```

to those of the <calling-AP-invocation-id>.)

<called-AP-invocation-id-integer-length> ::= <byte>+
    {The size of the universal integer that
    encapsulates <called-AP-invocation-id>
    encoded as defined by ISO/IEC 8825-1:2002
    [BER].}

<called-AP-invocation-id-integer> ::= 02H <called-AP-invocation-id-length>
    <called-AP-invocation-id>
    {The universal integer that contains
    <called-AP-invocation-id>.)

<called-AP-invocation-id-length> ::= <byte>+
    {Length of <called-AP-invocation-id>
    encoded as defined by ISO/IEC 8825-1:2002
    [BER].}

<called-AP-invocation-id> ::= <byte>*
    {It conveys association information about
    an APDU that was originally issued by the
    called entity Application Layer. This
    value shall range between 0 to 4294967295.}

```

#### **5.3.4.10 Calling AP Invocation ID Element (A8<sub>H</sub>)**

The Calling AP Invocation ID element <calling-AP-invocation-id-element> provides for duplicate APDU rejection. This identifier is also the value returned by the <called-AP-invocation-id-element> to match corresponding requests and responses.

The <calling-AP-invocation-id-element> shall be placed before the <user-information-element>.

```

<calling-AP-invocation-id-element> ::= A8H <calling-AP-invocation-id-integer-length> <calling-AP-invocation-id-integer>
    {The <calling-AP-invocation-id-element> is
    associated with the originator of an ACSE
    message that was sent by the Application
    Layer of the originator of this ACSE
    message. The binding is temporary and it
    shall last only for the duration of the
    duplicate Datagram recognition period. In
    session-mode communication this period is
    the session duration. In sessionless
    communication the period is not less than
    one minute for any given association
    between a <called-AP-title-element>,
    <calling-AP-title-element> and <calling-AP-invocation-id-element> and optionally
    the supplied <iv>, subject to security
    parameters used. }

<calling-AP-invocation-id-integer-length> ::= <byte>+
    {The size of the universal integer that is
    the <calling-AP-invocation-id> encoded as
    defined by ISO/IEC 8825-1:2002 [BER].}

<calling-AP-invocation-id-integer> ::= 02H <calling-AP-invocation-id-length>
    <calling-AP-invocation-id>

```

{The universal integer that contains  
<calling-AP-invocation-id>..}

<calling-AP-invocation-id-length> ::= <byte>+  
{Length of <calling-AP-invocation-id>  
encoded as defined by ISO/IEC 8825-1:2002  
[BER].}

<calling-AP-invocation-id> ::= <byte>\*  
{In a sessionless transmission this is a  
number that is initially created by the  
Application Layer. The initial value is  
not defined by this Standard and therefore  
it is arbitrary. This value shall be an  
unsigned integer in the range of 0 to  
4294967295.

Upon session establishment (following the  
C12.22 Server's receipt of the Login  
service request and before the issuance of  
the <ok> response) both the C12.22 Server  
and the C12.22 Client shall reset the  
<calling-AP-invocation-id> to zero (0).  
Subsequently, within a session, the C12.22  
Server and the C12.22 Client shall  
increment monotonically the <calling-AP-  
invocation-id> up to the limit of  
4294967295, before the transmission of the  
next <acse-pdu>. i.e. the <calling-AP-  
invocation-id> shall remain unique for the  
entire duration of the session and when  
the maximum value is reached the session  
shall implicitly be terminated (as if the  
<terminate> service request was received).

The Application Layer shall supply this  
<calling-AP-invocation-id> upon  
construction of an initial <acse-pdu> and  
it shall use this element to reject  
Application Layer duplicate or out-of-  
sequence APDUs. It gets changed each time  
a new unsegmented APDU is created by the  
Application Layer. In sessionless  
exchanges This value shall not repeat  
within one minute and it may roll back to  
zero when it reaches its exceeds the  
maximum allowed value.}

#### **5.3.4.11 User Information Element (BE<sub>H</sub>)**

The User Information Element <user-information-element> is used to carry one or multiple requests or responses.

<user-information-element> ::= BE<sub>H</sub> <user-information-external-length>  
<user-information-external>

<user-information-external-length> ::= <byte>+  
{The size of the <user-information-  
external> encoded as defined by ISO/IEC  
8825-1:2002 [BER].}

```

<user-information-external> ::= 28H <user-information-length>
                                [ <user-information-indirect-reference> ]
                                <user-information-octet-string>

<user-information-length> ::= <byte>+
                                {The size of the <user-information-
                                indirect-reference> and <user-information-
                                octet-string> encoded as defined by
                                ISO/IEC 8825-1:2002 [BER].}

<user-information-indirect-reference> ::= 02H <user-information-encoding-
                                length> <user-information-encoding>
                                {Identifies the data encoding used to
                                decipher the <user-information> element.
                                This element is optional when
                                communicating over strictly homogeneous
                                C12.22 Networks. When this element is
                                absent, then <user-information-encoding>
                                shall assume the only currently defined
                                value of 00H.}

<user-information-encoding-length> ::= <byte>+
                                {The size of the <user-information-
                                encoding> encoded as defined by ISO/IEC
                                8825-1:2002 [BER].}

<user-information-encoding> ::= <byte>
                                {Default value is 00H. The value 00H
                                indicates that <user-information> is
                                transmitted exactly as prescribed in this
                                Standard. All other values are reserved.}

<user-information-octet-string> ::= 81H <user-information-octet-string-
                                length> <user-information>
                                {A collection of the bytes of <user-
                                information>. These are not encoded in
                                ASN.1 rules; therefore they are
                                transmitted as an implicit octet string.}

<user-information-octet-string-length> ::= <byte>+
                                {Length of <user-information> encoded as
                                defined by ISO/IEC 8825-1:2002 [BER].}

<user-information> ::= <epsem> | <c1222sm-user-information>

<c1222sm-user-information> ::= <epsem> [<padding>] <mac>
                                {Content of the User Information Element
                                when the Mechanism Name is <application-
                                context-oid>.2.1 and
                                the the SECURITY_MODE within the <epsem-
                                control> is set to a value different than
                                zero.}

<epsem>
                                {As defined in section 5.3.3 EPSEM
                                Envelope Structure.}

<padding> ::= <byte>*
                                {Zero or more bytes added to the end of
                                the <epsem> message to facilitate message
                                segmentation and size as required by the
                                encryption algorithm. The first byte of

```

padding must be 0 to identify the end of the requests and responses list <epsem-data>+. The other padding bytes can be set to any random pattern.}

<mac> ::= <byte><sup>4</sup> {Message authentication code as defined by annex I – “The EAX’ Cryptographic Mode”.}

#### 5.3.4.12 Use of Subbranches of a Registered ApTitle

Any subbranch of the registered root ApTitle can be used to communicate with the C12.22 Node that registered that root under controlled conditions. All subbranches are assumed to be registered and managed by the root ApTitle holder as long as the root ApTitle is registered. There are three subbranch categories supported by the Standard

1. Assigned Subbranch: Those subbranches whose function and numeric values are assigned by the standard. Subbranches 0 to 15 are reserved for this purpose.
2. Reserved Subbranch: Those subbranches whose function and numeric values are assigned by the vendor application of the C12.22 Node. Subbranches 16 to 31 are reserved for this purpose.
3. Dynamic Subbranch: Those subbranches that are assigned dynamically by the C12.22 Node. Subbranches starting at 32 are reserved for this purpose

When a C12.22 Device receives a message targeted to an unsupported subbranch, the node shall return an <uat> Unknown Aptitle error response.

The relative portion of a registered (which follows the <application-context-oid>.0) ap-title shall not include assigned or reserved subbranches. This is requirement is to ensure unambiguous conversion between relative and universal ApTitles.

Subbranches are used for the efficient facilitation of the following:

#### Access to Interfaces and Local Ports

Reserved Subbranches were assigned by the Standard to access the C12.22 Devices’ Local Ports and C12.22 Communication Modules’ interfaces. These values are pre-assigned as follows:

Branch	Description
relay-ap-title.0 or relay-ap-title.0.group	Defined for broadcast and multicast addressing as defined by “Annex D - Universal Identifier”.
registered-ap-title.1 or registered-ap-title.1.interface	These subbranches provide access to C12.22 Communication Module tables of a C12.22 Node. The default Communication Module assigned subbranch is .1; other Communication Modules may be accessed using subbranches .1.x, where x =1,2... n, is the Communication Module interface number; and x = 0 is the default Communication Module interface. For example, if registered-ap-title is set to “<application-context-oid>.0.69.987”, the default interface can be accessed using the ApTitle “<application-context-oid>.0.69.987.1” and the second interface can be accessed using the ApTitle “<application-context-oid>.0.69.987.1.2”.
registered-ap-title.2 or registered-ap-title.2.local-port	These subbranches provide access to the application entity of a device attached to a Local Port. The default Local Port assigned subbranch is .2; other local ports may be accessed using subbranches .2.x, where x =1,2... n, is the Local Port number; and x = 0 is the default Local Port

Branch	Description
	<p>(e.g. ANSI Type II connector). For example, if registered-ap-title is set to “&lt;application-context-oid&gt;.0.69.987”, the default local port can be accessed using either the ApTitle “&lt;application-context-oid&gt;.0.69.987.2” or “&lt;application-context-oid&gt;.0.69.987.2.0”.</p> <p>The following Local Port numbers are defined in this Standard. These are identical to those defined in 6.9.2 “Packet Definition”.</p> <p>0 = C12.22 Device 1 = Local Port 0 (Default) 2 = Local Port 1 (Alternate) 3 = Interface 0 (Default WAN) 4 = Interface 1 (Alternate WAN) 5 = Interface 2(Default POT modem) 6 = Interface 3 (Alternate POT modem) 7 to 12 = Reserved 13 to 28 = Manufacturer defined</p> <p>All other values are reserved.</p> <p>For example “&lt;application-context-oid&gt;.0.69.987.1.1” is equivalent to “&lt;application-context-oid&gt;.0.69.987.2.4”; however “&lt;application-context-oid&gt;.0.69.987.1.100” may be bound by the manufacturer to “&lt;application-context-oid&gt;.0.69.987.2.1”.</p>

## Mailbox

Reserved Subbranches were assigned by the Standard to identify common mailbox. These values are pre-assigned as follows:

Branch	Description
registered-ap-title or registered-ap-title.3.0	Default mailbox
registered-ap-title.3.1	Notification mailbox Provides a mechanism for the transmission of messages tagged as information C12.22 Node.
registered-ap-title.3.2	Alarm mailbox Provides a mechanism for the transmission of alarm messages about the values or states of a C12.22 Node.

## Use of Assigned Subbranches in Relative ApTitle

Assigned Subbranches and Reserved Subbranches may be used in the first arc of a <called-AP-title> or <calling-AP-title> when encoded as <relative-aptitle-element>. i.e. when exchanging messages between a C12.22 Device and a C12.22 Communication Module or any of the C12.22 Node local ports or interfaces. E.g. When C12.22 Device sends a C12.22 Message through a local interface to its local C12.22 Communication Module, it may encode the <called-AP-title> as 80<sub>H</sub> 01<sub>H</sub> 01<sub>H</sub> (indicating that it is a relative called ApTitle that refers to the assigned subbranch 1, that is the C12.22 Communication Module).

When C12.22 Authentication mechanism is used in conjunction with Assigned Subbranches in Relative ApTitle to form a <called-AP-title> or <calling-AP-title>, the canonical form of the corresponding <called-AP-title-element> or <calling-AP-title-element> shall be derived by appending <called-AP-title> or <calling-AP-title> to the canonical form of C12.22 Device's ApTitle (obtained the TLS Get Configuring Service).



## Multiple Associations

The use of subbranches also facilitates the management of concurrent associations from the same targets and initiators. Both the initiator and the target C12.22 Nodes sharing an association may assign an unreserved subbranch for their ApTitle. In this case, the initiator assigns a subbranch of its calling ApTitle then it uses the target's root ApTitle as the called ApTitle. The target may respond with its called ApTitle or assign a new subbranch for its called ApTitle for subsequent exchanges.

The <calling-AP-invocation-id-element> shall reset to zero, by both the sender and receiver, upon session establishment then increment monotonically upon the issuance of a new un-segmented C12.22 Message inside the session. The sessionless-mode initial value of the <calling-AP-invocation-id-element> is not defined by this Standard therefore it is arbitrary.

Example of multiple sessions that use subbranches:

Process ID	Action	Calling ApTitle	Called ApTitle	Calling AP Invocation ID	Called AP Invocation ID	Session
1	Logon →	208.20.50.69	208.20.30	8362 (Next)	N.A.	1
1	← Ok	208.20.30.81	208.20.50.69	0	8362	1 (Started)
1	Read →	208.20.50.69	208.20.30.81	0	N.A.	1
2	Logon →	208.20.50.47	208.20.30	9273 (Next)	N.A.	2
1	← Data	208.20.30.81	208.20.50.69	1	0	1
2	← Ok	208.20.30.75	208.20.50.47	0	9273	2 (Started)
1	Read →	208.20.50.69	208.20.30.81	1	N.A.	1
1	← Data	208.20.30.81	208.20.50.69	2	1	1
2	Read →	208.20.50.47	208.20.30.75	0	N.A.	2
2	← Data	208.20.30.75	208.20.50.47	1	0	2
2	Logoff →	208.20.50.47	208.20.30.75	1	N.A.	2
1	Logoff →	208.20.50.69	208.20.30.81	2	N.A.	1
2	← Ok	208.20.30.75	208.20.50.47	2	1	2 (End)
1	← Ok	208.20.30.81	208.20.50.69	3	2	1 (End)
1	Read →	208.20.50.102	208.20.30.0 (comm. Module)	7514 (Next)	N.A.	N.A.
1	← Data	208.20.30.0.2	208.20.50.102	261 (Next)	7514	N.A.

## Proxy Server

A C12.22 Relay can also act as a proxy server for C12.22 Nodes that are clustered on the same local network segment of the C12.22 Relay. This service is initiated by any C12.22 Node when it sends a <acse-pdu> to the Proxy Relay without including its own calling ApTitle element in the <acse-pdu>. The Relay can either reject this message by returning <onp> or it can dynamically assign one of its subbranches as the calling ApTitle, then forward the resulting <acse-pdu> to its destination. Messages that are directed to that dynamically assigned subbranch are forwarded by the C12.22 Relay to the corresponding clustered node.

Dynamically assigned subbranches shall remain assigned to the same C12.22 Node for the duration of the connection (in a connected environment) or a limited time (in a connectionless environment) before it is returned to the available subbranches. The choice of this timing value is left to the C12.22 Relay developer.

### 5.3.4.13C12.22 Security Mechanism

This section defines the security mechanisms used when the Mechanism Name Element <mechanism-name> is set to a value that is well-known according to this Standard. This is the case when an ACSE message explicitly includes a Mechanism Name Element <mechanism-name-element> as follows:

- <application-context-oid>.2.0 for ANSI C12.22 compatibility mode with ANSI C12.21 Authentication algorithm 00<sub>H</sub> according to ANSI C12.21-2006 ANNEX G - Data Encryption Standard; or
- <application-context-oid>.2.1 for ANSI C12.22 native mechanism as defined in this document.

The Standard native mechanism names supported are <application-context-oid>.2.1 or <application-context-oid>.2.0. In the absence of a <mechanism-name-element>, the default mechanism is <application-context-oid>.2.1.

#### **5.3.4.13.1 C12.22 Security Mechanism (<application-context-oid>.2.1)**

##### **Security Modes**

The C12.22 Security mechanism supports the transportation of messages in three forms:

1. Cleartext, where all the elements of the C12.22 Message are transmitted as unauthenticated readable fields.
2. Authenticated Cleartext, where all the elements of the C12.22 Message are transmitted as authenticated readable fields.
3. Authenticated Ciphertext, where all the elements of the C12.22 Message are transmitted as authenticated fields, with portions of the <user-information> element are encrypted to provide privacy of the data.

The standard C12.22 security mechanism relies on a mode of encryption derived from EAX, as defined by M. Bellare, P. Rogaway, and D. Wagner. As of this writing, the EAX mode is not a recognized NIST standard, but it has been submitted to NIST, and can be obtained from the NIST web site:

<http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/eax/eax-spec.pdf>

ANSI C12.22 uses a mode called EAX', defined by annex I "The EAX' Cryptographic Mode", in conjunction with the Advanced Encryption Standard (AES) block Cipher with 128-bit keys. The selected mode provides the ability to both protect the privacy of portions of a message, as well as authenticate the entire message. Some portions of the C12.22 Message, such as headers, are sent as Cleartext. ANSI C12.22 uses this feature to provide authentication, as well as (optionally) allowing for confidentiality. When confidentiality is specified for a C12.22 Message select contents of the <user-information-element> are replaced with the Ciphertext generated by the AES-EAX' algorithm.

To successfully exchange authenticated or authenticated and private messages, each side of the communication link must share the same cipher-algorithm and key. To help the target C12.22 Node to select the appropriate key, a key id may be included in each sessionless message or at the beginning of each new session.

The SECURITY\_MODE field present in the <epsem-control> byte is used to indicate whether the message is transmitted authenticated or authenticated and private, where the optional <padding> and required <mac> fields are appended to the <epsem>. The <mac> is the cryptographically-derived message authentication code, which may be used to establish the authenticity of the C12.22 Message.

When the SECURITY\_MODE field is set to zero (0), all of the elements of the C12.22 Message are sent as Cleartext without any authentication and with neither <mac> nor <padding> appended at the end of the message.

It should be noted that the Security Mechanism, as described in this section provides authentication and, optionally, privacy for messages. It does not provide a mechanism for establishing or changing the level of authorization for a given message. Authorization is achieved by using the security service as described in section 5.3.2.4.5 "Security Service".

##### **Rules For Responses**

As a rule, the target of a sessionless request always responds using the mode and algorithm (Cleartext, authenticated Cleartext, or authenticated Cleartext plus Ciphertext) as requested unless it has been configured to reject it.

Within a session, the target always returns responses using the mode and Cipher algorithm used by the initial Logon request unless it has been configured to reject it. The security mode can not be modified during a session.

### **Key ID**

The key used to encrypt the information is identified by the <key-id-element>. This element is optional and when not provided in a sessionless message, <key-id> 0 shall be used; otherwise, the previous key id of the session shall be used.

### **Access Privileges**

During both session-based and sessionless transactions, it is possible to establish or modify access privileges by providing a password using the <security> service. For a sessionless transaction, the <security> request also includes the <user-id>. The <security> service is transported in the <epsem-data> portion of the <user-information-element>.

### **Initialization Vector**

An Initialization Vector is provided by the <iv-element>. A new value should be generated each time a new sessionless C12.22 Message is generated. Within a session (following a Logon Service Response up to the next Logoff Service Response inclusive), no <iv> is transmitted. Each side shall use the Initialization Vector, <iv>, received in the C12.22 Message that carried the Logon Service by the other end. This <iv> is then used for all Datagrams transmitted within that session. Please note that <calling-AP-invocation-id> element is used in the nonce calculations for the cryptographic processing, ensuring that, while the <iv> does not change within a session, the nonces that are used do, so that the C12.22 Message sequence remains cryptographically secure. Generation of the <calling-AP-invocation-id> is discussed in detail below.

The <iv> portion of the <iv-element> is used by the authentication and privacy algorithms. It provides a 32 bit unsigned integer that represents the UTC time counter in seconds since 00:00:00 January 1, 1970 UTC. The accuracy of the <iv> is assumed to be better than  $\pm 15$  minutes, therefore enabling the receiver of the C12.22 Message to reject incoming messages based on their time-stamp (an optional capability).

### **Calling AP Invocation ID**

In a sessionless transmission the <calling-AP-invocation-id> is a number that is initially created by the Application Layer. The initial value is not defined by this Standard and therefore it is arbitrary. Upon session establishment (following the C12.22 Server's receipt of the Login service request and before the issuance of the <ok> response) both the C12.22 Server and the C12.22 Client reset the <calling-AP-invocation-id> to zero (0). Subsequently, within a session, the C12.22 Server and the C12.22 Client increment the <calling-AP-invocation-id> monotonically, without limits, before the transmission of the next <acse-pdu>; therefore the <calling-AP-invocation-id> is unique for the entire duration of the session (this is possible due to the fact that the <calling-AP-invocation-id> is encoded as an ASN.1 Universal Integer).

The Application Layer generates a new <calling-AP-invocation-id> upon construction of a new (non-fragment) <acse-pdu> and it may use this element to detect Application Layer duplicate or out-of-sequence APDUs. In sessionless exchanges this value shall not repeat within one minute.

### **Syntax Mapping**

As detailed in Annex I, EAX'.Encrypt takes two arguments: *N*, the Cleartext portion of the message and, *P*, the Plaintext portion of the message. The result of EAX'.Encrypt operation are *C*, the Ciphertext and *T*, the message authentication code. Similarly, EAX'.Decrypt takes three arguments: *N*, *C*, and *T*; the Cleartext, the Ciphertext, and message authentication code, respectively. Consistency across all C12.22 Nodes in the creation and processing of *N* is required, otherwise messages cannot be correctly authenticated. The purpose of this sub-section is to detail exactly how the contents of a C12.22 APDU are mapped into *N* and *P* when sending messages, and into *N*, *C*, and *T*, when receiving messages.

This standard's use of the EAX' mode of operation is designed to facilitate processing in which the portions of the message that are required for cryptographic operations can be processed as they are validated without requiring any backtracking. Conceptually, this means that while validating a message that has been received, or while generating a message for transmission, data could be placed in a buffer for cryptographic operations. To facilitate explanation of the use of EAX', this sub-section is written as if such a buffer exists. Specific implementations may or may not be implemented in such a fashion.

Cryptographic operations always occur on the unsegmented form of the C12.22 Message as defined in Section 5.3.4. In secured messages, the optional <called-AP-title-element> is always required and the optional <calling-authentication-value-element> is also required only in sessionless messages (for more details see definition of <iv>). However the <iv> and the <key> are always processed as part of the canonical form of the secured message.

When securing a message within the EAX' mode of operation, portions of the message may be Cleartext, and portions may be Plaintext. When messages are secured as Cleartext with Authentication, the canonified form of the entire message is used to calculate the nonce. The nonce calculation is the basis for message authentication code. When messages are secured as Ciphertext with Authentication, the ACSE headers and the EPSEM headers are the Cleartext used to calculate the nonce, and EPSEM data is treated as Plaintext. In both cases a canonified version of the message headers is used to form the basis of the nonce calculation.

### ***Cleartext with Authentication***

When producing the canonified Cleartext *N*, for a message being sent as Cleartext with Authentication, the following steps are taken, in order:

1. The leading ACSE APDU tag (60<sub>H</sub>) and the <elements-length> are omitted.
2. The <aSO-context-element>, if present, is included in the canonified Cleartext.
3. The <called-AP-title-element> is converted to absolute form, and included in the canonified Cleartext.
4. The <called-AP-invocation-id-element>, if present, is included in the canonified Cleartext.
5. The <calling-AP-title-element> is temporarily skipped, until the <epsem-control> can be evaluated to determine whether the <calling-AP-title> should be included in security calculations.
6. The <calling-AE-qualifier-element>, if present, is included in the canonified Cleartext.
7. The <calling-AP-invocation-id-element> is included in the canonified Cleartext.
8. The <mechanism-name-element>, if present, is included in the canonified Cleartext.
9. The <calling-authentication-value-element>, if present, is included in the canonified Cleartext.
10. The <user-information-element>, including any optional sub-elements, up to and including the <epsem-control> is included in the canonified Cleartext.
11. If, and only if, the PROXY\_SERVICE\_USED flag is not set, the absolute form of the <calling-AP-title-element> is included in the canonified Cleartext.
12. The <key-id> is appended to the end of the canonified Cleartext.
13. The <iv> is appended to the end of the canonified Cleartext.
14. The <ed-class>, <epsem-data>, 00<sub>H</sub>, and <padding> are included in the canonified Cleartext.

If the <calling-authentication-value-element> already includes the <key-id> and <iv>, they will be present twice in the canonified Cleartext.

C12.22 APDU	Canonified Cleartext
60 <sub>H</sub> <elements-length> [ <aSO-context-element> ] A2 <sub>H</sub> <called-AP-title-length> <universal-aptitle-element>   <relative-aptitle-element> [ <called-AP-invocation-id-element> ] [ A6 <sub>H</sub> <calling-AP-title-length> <universal-aptitle-element>   <relative-aptitle-element> ] [ <calling-AE-qualifier-element> ] <calling-AP-invocation-id-element> [ 8B <sub>H</sub> <mechanism-name-length> <application-context-oid>.2.1 ] [ <calling-authentication-value-element> ] BE <sub>H</sub> <user-information-external-length> 28 <sub>H</sub> <user-information-length> [ <user-information-indirect-reference> ] 81 <sub>H</sub> <user-information-octet-string-length> <epsem-control>  [<ed-class>] <epsem-data>+ [ <padding> ]	[ <aSO-context-element> ] A2 <sub>H</sub> <called-AP-title-length> <universal-aptitle-element> [ <called-AP-invocation-id-element> ]  [ <calling-AE-qualifier-element> ] <calling-AP-invocation-id-element> [ 8B <sub>H</sub> <mechanism-name-length> <application-context-oid>.2.1 ] [ <calling-authentication-value-element> ] BE <sub>H</sub> <user-information-external-length> 28 <sub>H</sub> <user-information-length> [ <user-information-indirect-reference> ] 81 <sub>H</sub> <user-information-octet-string-length> <epsem-control>  [ A6 <sub>H</sub> <calling-AP-title-length> <universal-aptitle-element> ] <key-id> <iv> [<ed-class>] <epsem-data>+ [ <padding> ]

For Cleartext messages with Authentication,  $N$  is the concatenated form of the canonified Cleartext as described above, in the order described. Elements that are not provided in the message are simply not included in  $N$ . As described in Annex I EAX'Encrypt, this byte string is sent through the algorithm CMAK, producing the nonce,  $\underline{N}$ . The least significant 4 bytes are used for the "tag" or message authentication code. The message authentication code is delivered in the <mac> element. Thus, an authenticated message would be sent as:

```

60H <elements-length>
[ <aSO-context-element> ]
A2H <called-AP-title-length>
    <universal-aptitle-element> |
    <relative-aptitle-element>
[ <called-AP-invocation-id-element> ]
[ A6H <calling-AP-title-length>
    <universal-aptitle-element> |
    <relative-aptitle-element> ]
[ <calling-AE-qualifier-element> ]
<calling-AP-invocation-id-element>
[ 8BH <mechanism-name-length>
    <application-context-oid>.2.1 ]
<calling-authentication-value-element>
BEH <user-information-external-length>
    28H <user-information-length>
        [ <user-information-indirect-reference> ]
        81H <user-information-octet-string-length>
        <epsem-control>
[<ed-class>]

```

```

    <epsem-data>+
    [ <padding> ]
    <mac>

```

### ***Ciphertext with Authentication***

When producing the canonified Cleartext *N* for a message being sent as Ciphertext with Authentication the following steps are taken, in order:

1. The leading ACSE APDU tag (60<sub>H</sub>) and the <elements-length> are omitted.
2. The <aSO-context-element>, if present, is included in the canonified Cleartext.
3. The <called-AP-title-element> is converted to absolute form, and included in the canonified Cleartext.
4. The <called-AP-invocation-id-element>, if present, is included in the canonified Cleartext.
5. The <calling-AP-title-element> is temporarily skipped, until the <epsem-control> can be evaluated to determine whether the <calling-AP-title> should be included in security calculations.
6. The <calling-AE-qualifier-element>, if present, is included in the canonified Cleartext.
7. The <calling-AP-invocation-id-element> is included in the canonified Cleartext.
8. The <mechanism-name-element>, if present, is included in the canonified Cleartext.
9. The <calling-authentication-value-element>, if present, is included in the canonified Cleartext.
10. The <user-information-element>, including any optional sub-elements, up to and including the <epsem-control> is included in the canonified Cleartext. The <ed-class>, <epsem-data>, 00<sub>H</sub>, and <padding> are not included in the canonified Cleartext.
11. If, and only if, the PROXY\_SERVICE\_USED flag is not set, the absolute form of the <calling-AP-title-element> is included in the canonified Cleartext.
12. The <key-id> is appended to the end of the canonified Cleartext.
13. The <iv> is appended to the end of the canonified Cleartext.
14. The <ed-class>, <epsem-data>, 00<sub>H</sub>, and <padding>, when present, are the Plaintext, *P*, to be enciphered.

If the <calling-authentication-value-element> already includes the <key-id> and <iv>, they will be present twice in the canonified Cleartext.

C12.22 APDU (as transmitted)	Canonified Cleartext (as processed)
60 <sub>H</sub> <elements-length> [ <aSO-context-element> ] A2 <sub>H</sub> <called-AP-title-length> <universal-aptitle-element>   <relative-aptitle-element> [ <called-AP-invocation-id-element> ] [ A6 <sub>H</sub> <calling-AP-title-length> <universal-aptitle-element>   <relative-aptitle-element> ] [ <calling-AE-qualifier-element> ] <calling-AP-invocation-id-element> [ 8B <sub>H</sub> <mechanism-name-length> <application-context-oid>.2.1 ] [ <calling-authentication-value-element> ] BE <sub>H</sub> <user-information-external-length> 28 <sub>H</sub> <user-information-length> [ <user-information-indirect-reference> ] 81 <sub>H</sub> <user-information-octet-string-length> <epsem-control>	[ <aSO-context-element> ] A2 <sub>H</sub> <called-AP-title-length> <universal-aptitle-element> [ <called-AP-invocation-id-element> ] [ <calling-AE-qualifier-element> ] <calling-AP-invocation-id-element> [ 8B <sub>H</sub> <mechanism-name-length> <application-context-oid>.2.1 ] [ <calling-authentication-value-element> ] BE <sub>H</sub> <user-information-external-length> 28 <sub>H</sub> <user-information-length> [ <user-information-indirect-reference> ] 81 <sub>H</sub> <user-information-octet-string-length> <epsem-control> [ A6 <sub>H</sub> <calling-AP-title-length> <universal-aptitle-element> ] <key-id> <iv>
[<ed-class>] <epsem-data>+ [ <padding> ]	Plaintext (as processed)
	[<ed-class>] <epsem-data>+ [ <padding> ]

For Ciphertext messages with Authentication,  $N$  is the concatenated form of the canonified Cleartext as described above, in the order described.  $P$  is the concatenated form of the Plaintext as described above, in the order described. Elements that are not provided in the message are simply not included in  $N$  or  $P$ . As described in Annex I EAX'.Encrypt, the byte string  $N$  is sent through the algorithm  $\text{CMAC}_K$ , producing the nonce,  $\underline{N}$ . In the case of messages that are secured using Ciphertext with Authentication, the nonce is used to support CTR mode encryption, as described in Annex I EAX'.Encrypt, of the Plaintext  $P$ . The message authentication code is the XOR of the nonce  $\underline{N}$ , with the  $\text{CMAC}_K$  of the Ciphertext.

The output of a message secured using Ciphertext with Authentication would be:

```

60H <elements-length>
[ <aSO-context-element> ]
[ A2H <called-AP-title-length>
  <universal-aptitle-element> |
  <relative-aptitle-element> ]
[ <called-AP-invocation-id-element> ]
[ A6H <calling-AP-title-length>
  <universal-aptitle-element> |
  <relative-aptitle-element> ]
[ <calling-AE-qualifier-element> ]
<calling-AP-invocation-id-element>
[ 8BH <mechanism-name-length>
  <application-context-oid>.2.1 ]
<calling-authentication-value-element>
BEH <user-information-external-length>

```

```

28H <user-information-length>
    [ <user-information-indirect-reference> ]
81H <user-information-octet-string-length>
    <epsem-control>
    encrypted form of <ed-class>, <epsem-data>, and <padding>
    <mac>

```

### **Validating Messages**

Receiving and validating messages requires a similar process. The receiving device must first canonify the received message, exactly as described above for Cleartext with Authentication or Ciphertext with Authentication, as appropriate.

For Cleartext messages with Authentication, the <mac> as received in the C12.22 Message is provided as *T* to the EAX'.decrypt algorithm; the canonified Cleartext is provided as *N*; and *C*, the Ciphertext, is null. Within the EAX'.decrypt algorithm, the canonified Cleartext is used to generate the nonce, and the message authentication code, as previously described. Then the generated message authentication code is compared with the received <mac>. If the value does not match, the message is rejected.

For Ciphertext messages with Authentication, the <mac> as received in the C12.22 Message is provided as *T* to the EAX'.decrypt algorithm; the canonified Cleartext is provided as *N*; and bytes following the <epsem-control> up to, but exclusive of, the <mac> are provided as *C*, the Ciphertext. Within the EAX'.decrypt algorithm, the canonified Cleartext is used to generate the nonce. The least four significant bytes are used as the "tag" for the message. The CMAC<sub>K</sub> of the Ciphertext is XOR'd with the tag to generate the message authentication code. The generated message authentication code is compared with the received <mac>. If the value does not match, the message is rejected. If the value matches, the Ciphertext is returned to Plaintext by performing CTR mode encryption on it using the nonce.

### **5.3.5 Application Segmentation Sub-layer**

APDU Segmentation is required when the underlying Network Layer does not provide a delivery mechanism with sufficient capability to handle large APDUs (see , [HCCS 1: 1987] p18-19).

This Application Segmentation Sub-layer is responsible for the segmentation and reassembly of C12.22 Messages that exceed the Transport Layer maximum size limit. Segmentation is the process of breaking the C12.22 Message into smaller pieces, and reassembly is the process of putting the smaller pieces back together to reconstruct the original C12.22 Message. Because each APDU Segment must be sent by the Transport Layer independently, each must carry sufficient information to be able to be routed to the target C12.22 Node. The detailed implementation algorithm for APDU Segmentation is described in the following subsections.

This Standard requires that the delivery of C12.22 Messages cannot fail because of their size. C12.22 Relays and C12.22 Communication Modules shall, therefore, segment or re-segment C12.22 Messages received, adjusting their sizes to facilitate a reliable transportation of C12.22 Message.

A C12.22 Node Application Layer may reject a request because of its size or its related response size (for example, because the assembled C12.22 Message would exceed the memory capacity of the device). In these cases, the C12.22 Node returns the error message <rqt!> or <rst!>.

At the Transport Layer, each C12.22 Node has separate responsibilities for transmit and receive. For transmit, each C12.22 Node must assure that APDU Segments it sends do not exceed the capacity of the underlying Transport Layer. In the case of a C12.22 Device attached to a C12.22 Communications Module, this maximum segment size may be negotiated between the two via the Negotiate service. For receive, the C12.22 Node must accept any size APDU Segment that the Transport Layer is capable of carrying, since no end-to-end negotiation is possible.

C12.22 Relays or C12.22 Communication Modules may reassemble APDU Segments to optimize the



overall performance of the C12.22 Network.

### 5.3.5.1 APDU Segmentation

APDUs as defined by <acse-pdu> are used to request services and transfer payloads. APDUs are delivered to the lower layers in the OSI stack for encapsulation, address resolution and subsequent transmission to their destinations through the Transport Layer. In general, the Application Layer does not have knowledge (nor should it have any such knowledge) of network segment size and transmission constraints and similarly the Application Segmentation Sub-layer does not have any knowledge of the limits that may be imposed by the remote Application Layer or remote Application Segmentation Sub-layer.

### 5.3.5.2 APDU Segment

Each APDU Segment is constructed as an ACSE PDU with the following constraints:

- the <aSO-context-element>, <called-AP-title-element>, <calling-AP-title-element>, <calling-AE-qualifier-element> and <calling-AP-invocation-id-element>, when present in the unsegmented APDU, shall be identically present in each of the APDU segments.
- the <mechanism-name-element>, <calling-authentication-value-element> and <called-AP-invocation-id-element> shall not be present in an APDU segment.
- the <called-AE-qualifier-element> shall be present and <segment-user-information-element> shall be constructed as per "Segment User Information Element (BE<sub>H</sub>)" section below.
- the <called-AE-qualifier-element> and <calling-AP-invocation-id-element> shall be placed before the <segment-user-information-element>.
- the <segment-user-information-octet-string> shall immediately follow <segment-user-information-indirect-reference> as per <segment-user-information-associated>.

#### 5.3.5.2.1 Called AE Qualifier Element (A3<sub>H</sub>)

The <called-AE-qualifier-element> is used to carry the size of the fully assembled <acse-pdu>. This element shall be present in all <segmented-acse-elements> and absent in all <unsegmented-acse-elements>. The presence of the <called-AE-qualifier-element> is an indication that the <acse-pdu> contains <segmented-acse-elements>.

```
<called-AE-qualifier-element> ::= A3H <called-AE-qualifier-element-length>
                                <called-AE-qualifier-integer>
```

```
<called-AE-qualifier-element-length> ::= <byte>+
                                         {The size of the <called-AE-qualifier-integer>
                                          encoded as defined by ISO/IEC 8825-1:2002 [BER].}
```

```
<called-AE-qualifier-integer> ::= 02H <called-AE-qualifier-integer-length>
                                <called-AE-qualifier>
```

```
<called-AE-qualifier-integer-length> ::= <byte>+
                                         {The size of the <called-AE-qualifier>
                                          encoded as defined by ISO/IEC 8825-1:2002 [BER].}
```

```
<called-AE-qualifier> ::= <byte> | <word16> | <word24>
                          {This shall be the size of the fully assembled APDU, <acse-pdu>, when measured
                           in bytes. It shall be identical in value to all related segments, and set to the
                           size of the fully assembled <acse-pdu> (1 + size of <elements-length> + <elements-length>
                           in bytes).}
```

### 5.3.5.2.2 Segment User Information Element (BE<sub>H</sub>)

The <segment-user-information-element> is used to carry one Application Sub-layer segment and its related data offset in the fully assembled <acse-pdu>.

```

<segment-user-information-element> ::= BEH <segment-user-information-
    external-length>
    <segment-user-information-external>

<segment-user-information-external-length> ::= <byte>+
    {The size of the implicit sequence that
    encapsulates <segment-user-information-
    indirect-reference> and <segment-user-
    information-octet-string> encoded as
    defined by ISO/IEC 8825-1:2002 [BER].}

<segment-user-information-external> ::= 28H <segment-user-information-
    associated-length>
    <segment-user-information-associated>

<segment-user-information-associated-length> ::= <byte>+
    {The size of the <segment-user-
    information-indirect-reference> and
    <segment-user-information-octet-string>
    encoded as defined by ISO/IEC 8825-1:2002
    [BER].}

<segment-user-information-associated> ::= <segment-user-information-
    indirect-reference>
    <segment-user-information-octet-string>
    {See definitions in "Segment Association
    Information Element" below.}

```

#### 5.3.5.2.2.1 Segment Association Information Element

```

<segment-user-information-indirect-reference> ::= 02H <segment-byte-offset-
    length> <segment-byte-offset>

<segment-byte-offset-length> ::= <byte>+ {The size of the <segment-byte-
    offset> encoded as defined by ISO/IEC
    8825-1:2002 [BER].}

<segment-byte-offset> ::= <byte> | <word16> | <word24>
    {It is the segment <segment-user-
    information> offset in bytes relative to
    the beginning of the fully assembled APDU.
    Segment offset zero points to the value
    60H of the APDU header and it cannot
    exceed the size of the fully assembled
    APDU - 1, when measured in bytes. This
    segment reference may not be preserved
    when going through a relay application.}

```

#### 5.3.5.2.2.2 Segment Data Elements

The <segment-user-information-octet-string> is used to carry one Application Sub-layer segment that is encoded as an ordered sequence of octets in <segment-user-information>.

```

<segment-user-information-octet-string> ::= 81H <segment-user-information-
length> <segment-user-information>

<segment-user-information-length> ::= <byte>+
{Length of <segment-user-information>
encoded as defined by ISO/IEC 8825-1:2002
[BER].}

<segment-user-information> ::= <byte>*
{Segment data bytes.}

```

### 5.3.5.3 The Segmentation and Reassembly

The following subsections detail the APDU Segmentation Algorithm, and error exception reporting algorithm.

#### 5.3.5.3.1 The Segmentation Algorithm

When the C12.22 Message exceeds the maximum transmit size of the Transport Layer, the Application Segmentation Sub-layer shall either return an error (<snp> or <snerr>) to the Application Layer or fragment the APDU. The algorithm for fragmentation is described below; conforming implementations may optimize or alter this algorithm as long as they adhere to the production rules expressed by this Standard.

If the <acse-pdu> being fragmented is not the product of an earlier segmentation (by virtue of the absence of the <called-AE-qualifier-element>) then perform the following steps in this order:

1. Copy the entire unsegmented <acse-pdu> into a Datagram capture buffer. This includes all the <acse-pdu> components: the byte 60<sub>H</sub>, <elements-length> and <elements> unaltered.
2. Split the captured data buffer into smaller non-overlapping fragments that are suitable (size-wise) for inclusion into the newly generated <acse-pdu>.
3. For each fragment of the Datagram capture buffer build a new <acse-pdu> by placing a Datagram capture buffer fragment into the <segment-user-information> field of the <segment-user-information-element> then setting the <segment-user-information-length> to the size of the Datagram capture buffer corresponding fragment in bytes.
4. Create the following fields ahead of the <segment-user-information-element>:
  - <called-AP-title-element> from the unsegmented <acse-pdu>.
  - <called-AE-qualifier-element> and set the <called-AE-qualifier> to the size of the fully assembled <acse-pdu> as stored in the Datagram capture buffer, including all the <acse-pdu> components such as the byte 60<sub>H</sub>, <elements-length> and all unaltered <elements>.
  - <calling-AP-title-element> from the unsegmented <acse-pdu>.
  - <calling-AE-qualifier-element> from the unsegmented <acse-pdu>.
  - <calling-AP-invocation-id-element> from the unsegmented <acse-pdu>, as all related segments shall have the same value.
  - Set the <segment-byte-offset> of the <segment-user-information-element> to the fragment's byte offset relative to the beginning of the Datagram capture buffer.
5. Append the newly constructed <segment-user-information-element>, to the end of the <acse-pdu>.
6. Update the <elements-length> of the <acse-pdu>..
7. Finish sending the fragments to its Transport Layer.

If the <acse-pdu> that is being fragmented is a product of a previous segmentation (by virtue the presence of the <called-AE-qualifier-element>) then:

1. Copy all the <acse-pdu>s <elements> excluding the element <segment-user-information-element> to the target <acse-pdu> construction area.

2. Copy the segment's <segment-user-information> portion from the <acse-pdu>s <segment-user-information-element> into a Datagram capture buffer, to be place as a new user information element <user-information-element>.
3. Split the captured Datagram buffer into smaller non-overlapping fragments that are suitable (size-wise) for inclusion into the newly generated <acse-pdu> segments.
4. Adjust all length and offset fields in each of the resulting <acse-pdu> segments. Specifically adjust each
  - <segment-user-information-length> to the length of the newly reduced data fragment in bytes.
  - <segment-byte-offset> being the fragment's revised <segment-user-information> offset in the fully-assembled <acse-pdu> as originally delivered by its Application Layer.
5. Update the <elements-length> of the <acse-pdu>.
6. Finish the sending the fragments to the Transport Layer.

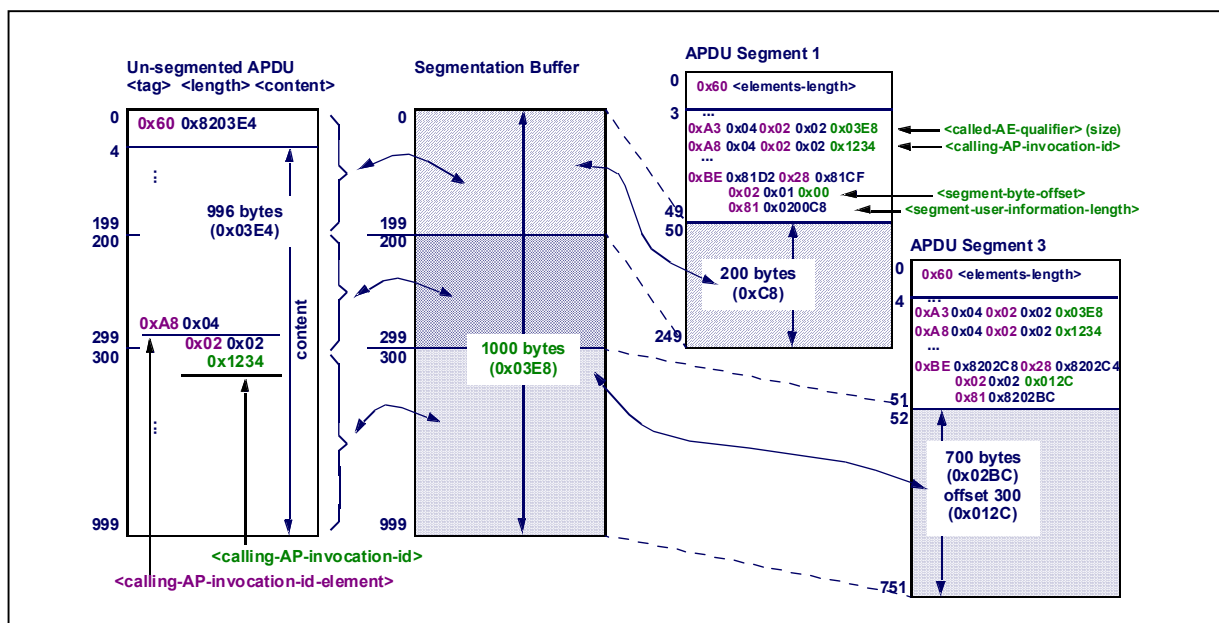


Figure 5.3: Un-segmented APDU segmentation and re-assembly algorithm

### 5.3.5.3.2 The Reassembly Algorithm

This section describes how the Application Segmentation Sub-layer layer reassembles APDU fragments received from its Transport Layer into larger fragments or the original <acse-pdu> for delivery to the Application Layer.

Reassembly is not required for strictly forwarding functions, such as C12.22 Relays. However, any C12.22 Relay may choose, at its option, to perform full or partial APDU reassembly based on its system's requirements and knowledge.

When the Datagram received from the Transport Layer is a segment of a fragmented <acse-pdu> the Application Segmentation Sub-layer shall either return an error (<sgnp> or <sgerr>) to the <calling-AP-title-element> (calling entity) or reassemble the APDU before delivering it to the C12.22 Application Layer. The presence of <called-AE-qualifier-element> indicates that the Datagram is a segment of a fragmented <acse-pdu> whose <called-AE-qualifier> is the amount of buffer space needed to assemble the <acse-pdu>.

The reassembly algorithm is outlined below:

1. Allocate an <acse-pdu> re-assembly buffer that is capable of holding the <called-AE-qualifier> in bytes. This field is found in the <called-AE-qualifier-element> element.
2. For each segment received copy the <segment-user-information> portion of the <segment-user-information-element> into the buffer at the offset specified by the <segment-byte-offset> of the <segment-user-information-element> up to length specified in the <segment-user-information-length> in bytes.
3. Repeat step 2 above until all segments have been received. All segments are received when the <acse-pdu> re-assembly buffer contains no gaps or the duplicate packet recognition period expires.
4. If all segments are received with no gaps in the user data then forward the re-assembled buffer to its Application Layer for processing.
5. If a full re-assembly cannot be completed due to missing segments, then (the assembler) will send a <sgerr> response indicating the lowest segment that is missing (by including its starting segment offset in the error response) then reset the time-out timer. This may trigger the remote segmentation sub-layer to re-transmit the missing segment starting at that offset or including that offset. Only if that segment is not received the assembler may discard all related segments silently. If subsequently another segment is not received within the time-out interval then this process (item 5) repeats.

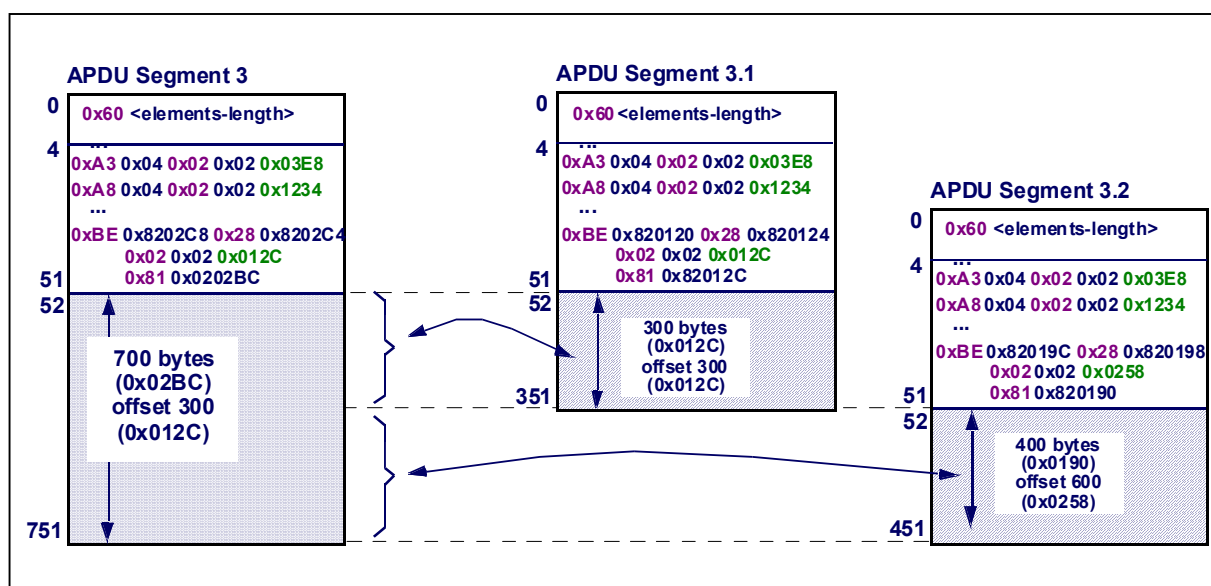


Figure 5.4: Segmented APDU re-segmentation and partial assembly algorithm

## 5.4 Layer 6 - Presentation Layer

Not defined by this Standard, open to any network protocol.

## 5.5 Layer 5 - Session Layer

Not defined by this Standard, open to any network protocol.

## 5.6 Layer 4 - Transport Layer

Not defined by this Standard, open to any network protocol.

## 5.7 Layer 3 - Network Layer

Not defined by this Standard, open to any network protocol.

### **5.8 Layer 2 - Data link Layer**

Not defined by this Standard, open to any network protocol.

### **5.9 Layer 1 - Physical Layer**

Not defined by this Standard, open to any network protocol.

## 6 Protocol Details: C12.22 Device to C12.22 Communication Module Interface

### 6.1 Interface Architecture

This section describes the interface (Physical, Data link, Transport, and Application layers) between a C12.22 Device and a C12.22 Communication Module. The intent of this architecture is to allow the creation of C12.22 Devices that can reside on any type of network. This architecture also allows development of C12.22 Communication Modules that can interface any C12.22 Device to specific networks. A C12.22 Device plus a C12.22 Communication Module arranged in this fashion constitutes a C12.22 Node.

In this model, all of the services that are normally provided by a single application entity are split between two physically and logically distinct modules, the C12.22 Communication Module and the C12.22 Device. The roles of each are divided as follows:

#### C12.22 Communication Module

- Implementation of Layers 1 through 6 of the target network
- Implementation of the Register Service (Start-up and keep-alive service sequences) for C12.22 Devices with one or more interfaces to C12.22 Communication Modules
- Implementation of the Resolve Service (direct messaging vs. message forwarding services)
- Forwarding <acse-pdu> from the network to a C12.22 Device (or other C12.22 Communication Modules)

#### C12.22 Device

- Implementation of the EPSEM Logon, Logoff, Security, Read and Write services
- Encryption and Authentication
- Application Layer <acse-pdu> segmentation and reassembly
- Application Layer addressing
- Application Layer <acse-pdu> processing or forwarding
- C12.22 Communication Module control and management services

### 6.2 Interface Diagram

The block diagram in Figure 6.1, C12.22 Communication Module Implementation Model, shows the minimal and optional layers and services supported by a C12.22 Device that is connected to a C12.22 Communication Module via a point-to-point interface.

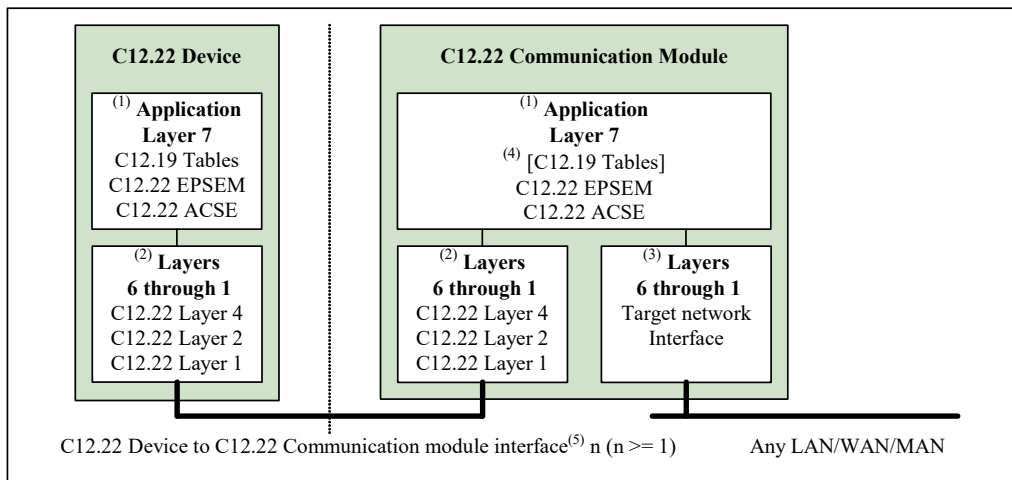


Figure 6.1: C12.22 Communication Module Implementation Model

## Annotations:

1. Application Layer is managed using ANSI C12.19 Tables, ANSI C12.22 EPSEM language and ACSE encapsulation.
2. ANSI C12.22 Layers 6 through 1 used for communication and control messages exchanged between the C12.22 Device and the C12.22 Communication Module, as described in this section.
3. Corresponding Layers 6 through 1 of the target network.
4. Optional content and services.
5. The number of ANSI C12.22 local Interfaces (n may be zero, one or greater than one).

## 6.3 Implementation Guidelines

### 6.3.1 C12.22 Communication Module

The C12.22 Communication Module shall minimally be able to:

- Implement Layers 1-6 services as defined in this Standard.
- Initiate a Negotiate Service to maximize packet sizes at startup.
- Optionally initiate a Get Configuration Service at startup and upon receipt of Link Control Service requests when the RELOAD\_CONFIG\_FLAG is set to true.
- Process and honor all configuration control directives received by the "Get Configuration Service".
- Emit empty packets as prescribed for line supervision in order to transition the attached C12.22 Device into the "Comm Module Present State".
- Implement Layers 1, 2 and 4 on the C12.22 Device interface side.
- Implement Layers 1 to 6 on the C12.22 Network Segment side.
- By default, forward all Datagrams from the network side to the local Interface side.
- By default, forward Datagrams from the local Interface side to the C12.22 Network Segment side that are not addressed to it.
- Intercept and process (may reject) the Send Service <acse-pdu> ApTitles that are addressed to it through its local Interface.
- Recognize and perform Layer 2 local routing in support of data-link packet forwarding to the other attached C12.22 Communication Modules.
- Recognize and correctly register the C12.22 Device using single or multiple interfaces ApTitles as needed.



- Recognize any ApTitle having its root ApTitle.1.<interface-number> as its own ApTitle, where <interface-number> is the interface number assigned to this C12.22 Communication Module.

The C12.22 Communication Module can optionally implement its own Table set (Table 0 and any other Tables as needed by its application). This Table set can be accessed from the network side by using the ANSI C12.22 protocol. In this case, the C12.22 Communication Module does not need to be registered. It is accessed using the C12.22 Device "<ap-title>.1 [<interface-number>]", which is a subbranch of any of the registered ApTitles for the attached C12.22 Device.

This Table set can also be accessed through the C12.22 Device Local Port. In this case, the called ApTitle shall be set to ".2[.local-port]", which in turn will be forwarded by the C12.22 Device to the C12.22 Communication Module on the requested interface.

### 6.3.2 C12.22 Device

ANSI C12.22 Interfaces are numbered sequentially from one to the number of Interfaces supported. For example, the second Interface can be accessed using the called ApTitle "<ap-title>.1.2". The special C12.22 ApTitle "<ap-title>.1" and "<ap-title>.1.0" shall be interpreted when received by the C12.22 Device as the "default C12.22 Communication Module interface", and when received by the C12.22 Communication Module as "this C12.22 Communication Module".

The C12.22 Device shall minimally be able to:

- Implement the Negotiate, Get Configuration, Link control and Send services as defined by this Standard.
- Initiate a Link Control Service as needed (e.g., after updates to network tables) after the Negotiate Service.
- Provide routing of APDUs between the C12.22 Device Local Port (if one is available) and any C12.22 Communication Module that is attached to it.

In addition the C12.22 Device may be able to:

- Provide a communication path from the local access port to any node on the LAN/WAN side of its C12.22 Communication Modules (optional capability subject to the C12.22 Application <acse-pdu> forwarding restrictions).
- Provide a communication path from the LAN/WAN of any of its C12.22 Communication Modules to any C12.22 Communication Module that is attached to it (optional capability subject to the C12.22 Application <acse-pdu> forwarding restrictions).
- Manage and control its network interfaces and C12.22 Communication Modules.
- Provide network Tables in support of the management and operation of all of its network interfaces, attached C12.22 Communication Modules, Local Ports and services.
- Provide network and interface state information using its network Tables.
- Accept and optionally process or reject APDUs arriving from any C12.22 Communication Module using the Send Service.
- Provide routing of APDUs from and to the C12.22 Device's local Interfaces, if any are available, to and from any C12.22 Communication Module attached to them.

## 6.4 Layer 7 - Application Layer

Same as section 5.3 "Layer 7 - Application Layer".

## 6.5 Layer 6 - Presentation Layer

Null layer.

## 6.6 Layer 5 - Session Layer

Null layer.

## 6.7 Layer 4 - Transport Layer

Transport Layer services are defined to facilitate setup, management and communication with one or more C12.22 Communication Modules.

For the purposes of enhanced clarity, the Negotiate Service defined in Layer 7 of ANSI C12.18 is presented within this layer of this Standard.

```

<tls-requests> ::= <tls-negotiate>      |      {Negotiate Request}
                   <tls-get-config>     |      {Get Configuration Request}
                   <tls-link-control>    |      {Link Control Request}
                   <tls-send-message>    |      {Send Message Request}
                   <tls-get-status>      |      {Get Status Request}
                   <tls-get-reg-status>   |      {Get Registration Status
                                           Request}

<tls-responses> ::= <tls-negotiate-r>    |      {Negotiate Response}
                   <tls-get-config-r>    |      {Get Configuration Response}
                   <tls-link-control-r>   |      {Link Control Response}
                   <tls-send-message-r>   |      {Send Message Response}
                   <tls-get-status-r>     |      {Get Status Response}
                   <tls-get-reg-status-r> |      {Get Registration Status
                                           Response}

```

### 6.7.1 Negotiate Service

The Negotiate Service is initiated by the C12.22 Communication Module after detection of the presence of an attached C12.22 Device.

#### Request

```

<tls-negotiate> ::= <baud-rate-selector> <rec-packet-size> <rec-nbr-packet>
                   <baud-rates> <rec-nbr-of-channels>

<baud-rate-selector> ::= 60H |      {No <baud rate> included in request. Stay
                                     at default baud rate}
                        61H |      {1 <baud rate> included in request}
                        62H |      {2 <baud rate>s included in request}
                        63H |      {3 <baud rate>s included in request}
                        64H |      {4 <baud rate>s included in request}
                        65H |      {5 <baud rate>s included in request}
                        66H |      {6 <baud rate>s included in request}
                        67H |      {7 <baud rate>s included in request}
                        68H |      {8 <baud rate>s included in request}
                        69H |      {9 <baud rate>s included in request}
                        6AH |      {10 <baud rate>s included in request}
                        6BH |      {11 <baud rate>s included in request}
                        6CH |      {12 <baud rate>s included in request}
                        6DH |      {13 <baud rate>s included in request}
                        6EH |      {14 <baud rate>s included in request}
                        6FH <byte> {Reserved for protocol extension}

```

```

<rec-packet-size> ::= <word16> {Maximum packet size in bytes that can be
                                received by the C12.22 Communication
                                Module. This value shall not be greater
                                than 8192 bytes.}

<rec-nbr-packet> ::= <byte> {Maximum number of packets the C12.22
                             Communication Module is able to reassemble
                             into an upper layer data structure at one
                             time.}

<baud-rates> ::= <baud-rate>* {List of baud rates supported by the
                                C12.22 Communication Module. If the C12.22
                                Device cannot select one of these baud
                                rates, the original baud rate is echoed in
                                the response.}

<baud-rate> ::= 00H | {Reserved}
                 01H | {300 baud}
                 02H | {600 baud}
                 03H | {1200 baud}
                 04H | {2400 baud}
                 05H | {4800 baud}
                 06H | {9600 baud}
                 07H | {14400 baud}
                 08H | {19200 baud}
                 09H | {28800 baud}
                 0AH | {57600 baud}
                 0BH | {38400 baud}
                 0CH | {115200 baud}
                 0DH | {128000 baud}
                 0EH | {256000 baud}
                 0FH | {230400 baud}
                 10H | {460800 baud}
                 11H | {500000 baud}
                 12H | {576000 baud}
                 13H | {921600 baud}
                 14H | {1000000 baud}
                 15H | {1152000 baud}
                 16H | {1500000 baud}
                 17H | {2000000 baud}
                 18H | {2500000 baud}
                 19H | {3000000 baud}
                 1AH | {3500000 baud}
                 1BH | {4000000 baud}
                    {1CH - FFH reserved}

<rec-nbr-of-channels> ::= <byte> {The maximum number of concurrent channels
                                supported by the C12.22 Communication
                                Module.}

```

### Response

The responses <sns>, <isss>, <bsy>, and <err> indicate a problem with the received service request and the C12.22 Communication channel retains its current settings.

The response <ok> indicates the service request was accepted and the new settings now apply. The new channel settings shall be the values communicated in the request (for the initiator) and the response (for the respondent) in regards to packet size, number of packets and number of channels.

```

<tls-negotiate-r> :=<sns> | <isss> | <bsy> | <err> |

```

```

<ok><trs-packet-size><trs-nbr-packet>
<negotiated-baud-rate><trs-nbr-of-channels>
<interface-number>

<trs-packet-size> ::= <word16> {Maximum packet size in bytes that can be
                                received by the attached C12.22 Device.
                                This value shall not be greater than 8192
                                bytes.}

<trs-nbr-packet> ::= <byte> {Maximum number of packets the attached
                              C12.22 Device is able to reassemble into
                              an upper layer data structure at one time.}

<negotiated-baud-rate> ::= <baud-rate> {Baud rate to use for future communication
                                         on this interface.}

<trs-nbr-of-channels> ::= <byte> {The maximum number of concurrent channels
                                   supported in reception by the C12.22
                                   Device.}

<interface-number> ::= <byte> {The interface number assigned to this
                                C12.22 Communication Module.}

```

### 6.7.2 Get Configuration Service

The Get Configuration Service is used solely by a C12.22 Communication Module to request its configuration information from the C12.22 Device to which it is attached. This service is initiated by the C12.22 Communication Module as needed, any time after it detects the presence of the C12.22 Device. The information carried by this service is the same as that found in the Interface Control Table (Table 122).

#### Request:

```
<tls-get-config> ::= 72H
```

#### Response:

The response <err> indicates a problem with the received service request.

The response <ok> indicates that the request has been accepted and the C12.22 Communication Module configuration follows in the response.

```

<tls-get-config-r> ::= <err> |
    <ok> <control> <interface-status> <node-
    type> <device-class> <esn> <native-
    address-len> <native-address> <broadcast-
    address-len> <broadcast-address> <relay-
    native-address-len> <relay-native-address>
    <node-ap-title> <master-relay-aptitle>
    <relay-aptitle> [<nbr-of-retry><response-
    timeout>]

<control> ::= <byte> {This is a bit field that provides
                      information and directives to the C12.22
                      Communication Module about its expected
                      operation. The control bits are defined as
                      follows:

```

Bit 0: INTERCEPT MODE

This is a directive to the C12.22 Communication Module regarding the forwarding to the C12.22 Device of messages that originate from the network side and that are addressed to the C12.22 Communication Module.

The value 0 (false) indicates that the C12.22 Communication Module is required to forward to the C12.22 Device all messages even if they are addressed directly to the C12.22 Communication Module and regardless of whether it has the capability to process them. This is the default operating mode of any C12.22 Communication Module at start-up.

The value 1 (true) indicates that the C12.22 Communication Module is permitted to intercept and process messages that are addressed directly to it if it has the capability to recognize them and process them. If it does not have such a capability, the C12.22 Communication Module shall forward the messages to the C12.22 Devices across the local interface, according to the default behavior at start-up.

Bits 1..7: Reserved.}

<interface-status>	::= <byte>	{As defined by the Link Control Service. This field is used by the C12.22 Device to configure its interface without issuing a Link Control Service after each power-up.}
<node-type>	::= <byte>	{Node type as defined by the Registration Service.}
<device-class>	::= <byte>+	{C12.19 Device Class of the C12.22 Device. Four bytes containing the MANUFACTURER ID as defined in Table 0 of ANSI C12.19-1997 or the registered DEVICE_CLASS as defined by Version 2 of ANSI C12.19.}
<esn>	::= <universal-id-element>   <relative-uid-element>	{A unique electronic serial number (ESN) that is assigned to a C12.22 Device by its owner (or operator). The use of an ESN is optional and, when not provided, the length of this element is set to zero.}
<native-address-len>	::= <byte>	{Number of bytes in <native-address>. This length is set to zero when this value is not configurable.}
<native-address>	::= <byte>*	{Native address assigned to the C12.22 Communication Module.}

```

<broadcast-address-len> ::= <byte> {Number of bytes in <broadcast-address>.
                                     This length is set to zero when this value
                                     is not configurable.}

<broadcast-address> ::= <byte>*    {Broadcast address assigned to the C12.22
                                     Communication Module.}

<relay-native-address-len> ::= <byte>
                                {Number of bytes in <relay-native-address>.
                                When the length of this element is zero
                                then the nearest C12.22 Relay address is
                                resolved and assigned by the C12.22
                                Communication Module. When the length of
                                this element is not zero then the C12.22
                                Communication Module shall use this value
                                as its new nearest C12.22 Relay address
                                until such time that it is reassigned by
                                the C12.22 Device.}

<relay-native-address> ::= <byte>* {Native address of the nearest C12.22
                                     Relay used to maintain registration of
                                     this node.}

<node-ap-title> ::= <universal-id-element> | <relative-uid-element>
                   {The ApTitle of the C12.22 Node to be
                   registered with a C12.22 Master Relay.
                   When the length of this element is zero
                   then the C12.22 Node's ApTitle will be
                   dynamically assigned. When the C12.22
                   Device handles registration, this field
                   can also be used to provide the registered
                   ApTitle of the C12.22 Device to the C12.22
                   Communication Module.}

<master-relay-aptitle> ::= <universal-id-element> | <relative-uid-element>
                          {The ApTitle of the C12.22 Master Relay
                          used to register this C12.22 Device. When
                          the length of this element is zero then
                          the C12.22 Master Relay's ApTitle will be
                          dynamically assigned.}

<relay-aptitle> ::= <universal-id-element> | <relative-uid-element>
                   {The ApTitle of the nearest C12.22 Relay
                   used to maintain registration of this node.
                   The length field shall be set to zero (0)
                   if not known at the time of this response.}

<nbr-of-retries> ::= <byte>      {Defines the number of times a request is
                                   sent if the response is not received
                                   within a <response-timeout>. The default
                                   value is 3.}

<response-timeout> ::=          <word16>    {Controls the number of
                                               seconds this C12.22 Communication
                                               Module has to wait for a response
                                               to a request before failing or
                                               sending retries. The default
                                               value is 300 seconds.}

```

### 6.7.3 Link Control Service

This service is used by a C12.22 Device to control a C12.22 Communication Module. A C12.22 Communication Module shall never initiate this service.

The C12.22 Device may initiate a Link Control Service request upon responding to both C12.22 Communication Module Negotiate Service request and Get Configuration Service Request. Alternatively it may initiate a Link Control Service request upon receipt of TLS service request that is other than the Negotiate Service request or the Get Configuration Service Request. Otherwise it shall wait a minimum of 60 seconds following the detection of the data-link Communication Module Present State, before initiating Link Control Service (or any other TLS service request).

When a C12.22 Node sends a Registration (or re-registration) Service Request and the response is <ok><reg-ap-title> where the received <reg-ap-title> is different from the ApTitle previously registered by the C12.22 Node, then the C12.22 Node shall accept the <reg-ap-title> as its new ApTitle or it may re-register for a new ApTitle.

#### Request:

```
<tls-link-control> ::= 73H <interface-ctrl>
```

```
<interface-ctrl> ::= <byte>          {Bit 0 to 1: INTERFACE_CTRL. Controls the
                                       presence of the C12.22 Communication
                                       Module on the Native Network.
                                       0 = Maintain current state.
                                       1 = Enable this interface. The C12.22
                                       Communication Module shall present itself
                                       to the Native Network and take all the
                                       necessary action enable it to communicate
                                       over the Native Network (e.g. so that one
                                       can initiate an application layer
                                       registration service request)
                                       2 = Disable this interface. Disable this
                                       interface. The C12.22 Communication Module
                                       shall disconnect from the Native Network
                                       in a manner that it has no presence,
                                       logical or physical on network
                                       3 = Reset this interface. Equivalent to
                                       disabling this Native Network interface
                                       (per code 2), followed by the enabling of
                                       this Native Network interface (per code 1).
                                       Note: This is not a C12.22 Communication
                                       Module software reset function; this is a
                                       network side interface reset.
```

```
Bit 2 to 3: AUTO_REGISTRATION_CTRL
0 = Maintain current state
1 = Enable auto-registration
2 = Disable auto-registration
3 = Force one-time registration now, then
   return to the previous state of auto-
   registration. Once registered, the C12.22
   Module shall keep the C12.22 Device
   registration state alive until instructed
   otherwise or the C12.22 Device becomes de-
   registered due to external reasons.
```

```
Bit 4 to 5: DIRECT_MESSAGING_CTRL
0 = Maintain current state
1 = Enable direct communication with
   target nodes on the same network segment
```

2 = Disable direct communication with target nodes on the same network segment

Bit 6: RESET\_STATISTIC\_FLAG  
false = Maintain current state  
true = Reset all statistics and counters

Bit 7: RELOAD\_CONFIG\_FLAG  
false = Maintain current state  
true = Initiate a Get configuration service to retrieve new configuration.}

#### Response:

The response <sns> indicates a problem with the received service request.

The response <ok> indicates that the request has been processed successfully.

```
<tls-link-control-r> ::= <sns> |
                        <ok><interface-status><ap-title>
```

```
<interface-status> ::= <byte> {Bit 0: INTERFACE_FLAG
This bit is set to true when the physical
interface is enabled. The C12.22
Communication Module is up and it is
configured to communicate on the Native
Network (e.g. ready to accept or deliver
Registration Service Requests)
```

```
Bit 1: AUTO_REGISTRATION_FLAG
This bit is set to true when C12.22
Communication Module automatically
registers the C12.22 Device on this
interface.
```

```
Bit 2: DIRECT_MESSAGING_FLAG
This bit is set to true when direct
messaging is in use on this interface.
```

```
Bits 3..7: Reserved.}
```

```
<ap-title> ::= <universal-id-element> | <relative-uid-element>
{ApTitle registered on this interface. If
the ApTitle has a length of zero, it means
that the C12.22 Node is not yet registered;
however, a registration may be in
progress.}
```

### 6.7.4 Send Message Service

The Send Message Service is used by a C12.22 Device to send messages through an attached C12.22 Communication Module or by the C12.22 Communication Module to transfer each message received for the C12.22 Device. This service carries the <acse-pdu> generated by two-way-devices or <short-pdu>s generated by one-way devices to be sent and the native address of the target or the initiator.

#### Request:

```
<tls-send-message> ::= <tls-send-acse-message> | <tls-send-short-message>
```



{The service request, address and payload to be transmitted to and from a C12.22 Communication Module across the Data Link as described in section 6.9 "Layer 2 - Data Link Layer".}

<tls-send-acse-message> ::= 74<sub>H</sub> <address-lgn> <address> <acse-pdu>  
 {The message format of a send service request that carries <acse-pdu>s as its payload data. It is the message format used by all two-way devices. This message format is the only one supported on all C12.22 Nodes on any C12.22 Network Segment.}

<tls-send-short-message> ::= 77<sub>H</sub> <address-lgn> <address> <short-pdu>  
 {The message format of a send service request that carries <short-pdu>s as its payload data. It is the message format used by some one-way devices to reduce the message size. This message format is recognized only by one-way designated native network segments and by one-way message enabled C12.22 Communication Modules. It is the responsibility of C12.22 Communication Module to map the <short-pdu> to an <acse-pdu> upon delivery to its C12.22 Network Segment.}

<address-lgn> ::= <byte> {Number of bytes in <address>.}

<address> ::= <byte>\* {When issued by the C12.22 Communication Module, the native address represents the source of the <acse-pdu> or the <short-pdu> on the local network segment. The C12.22 Communication Module shall always provide the native address.

When issued by the C12.22 Device Transport Layer, the native address represents the target node on the local network segment for this <acse-pdu> or <short-pdu>. This field is optional and when not provided, <address-lgn> is set to zero (00<sub>H</sub>).

Three methods are available to send information.

#### Directed message:

When the <native-address> is provided, the C12.22 Communication Module sends the <acse-pdu> or <short-pdu> to the specified address.

#### Message forwarding:

When the <native-address> is not provided and DIRECT\_MESSAGING\_FLAG is set to false, the C12.22 Communication Module sends the <acse-pdu> to one of its nearest C12.22 Relay based on a default or internal routing table (C12.22 Communication

Modules shall not transmit <short-pdu>s onto the C12.22 Network).

Direct messaging:  
When the <native-address> is not provided and DIRECT\_MESSAGING\_FLAG is set to true, the C12.22 Communication Module shall use the <called-AP-title> of the <acse-pdu> or <short-pdu> to resolve the native address to which to send this message.)

#### Response:

The responses <nett> and <netr> indicate a problem during the transmission of this message and can be returned only when this service is requested by the C12.22 Device.

The response <ok> indicates that the message has been transmitted successfully. Responses may be expected only following a <tls-send-acse-message> service request. Responses shall not be expected following a <tls-send-short-message> service request

```
<tls-send-message-r> ::=          <tls-send-acse-message-r> | <tls-send-
                                   short-message-r>

<tls-send-acse-message-r> ::= <nett> | <netr> | <ok>

<tls-send-short-message-r> ::=    {Short messages do not expect responses.}
```

### 6.7.5 Get Status Service

The Get Status Service is used by a C12.22 Device to retrieve information from a C12.22 Communication Module. The C12.22 Communication Module shall not initiate this service request. The information carried by this service is the same as that found in the Interface Status Table (Table 125).

#### Request:

```
<tls-get-status> ::= 75H <status-ctrl>

<status-ctrl>      ::= <byte>          {Controls the information returned by this
                                         service.
                                         0 = Interface information and statistics
                                         1 = Interface information only
                                         2 = Statistics only
                                         3 = Statistics changed or not delivered
                                         since last issued request}
```

#### Response:

The response <err> indicates a problem with the received service request.

The response <ok> is returned by the C12.22 Communication Module with the available information.

```
<tls-get-status-r> ::=          <err> |
                                   <ok>[<interface-info>]<statistics>*

<interface-info> ::= <interface-name-len>
                     <interface-name>
                     <interface-status>
```

```

<interface-name-len> ::= <byte>          {Number of bytes in
<interface-name>      {Textual description of the interface in
                        8-bit ASCII format. }

<interface-status>    ::= <byte>          {As described by the Link Control Service.}

<statistics>          ::= <statistic code> <statistic-value>
                        {The contents of the response are governed
                        solely by the C12.22 Communication Module.

                        It is conceivable that the number of
                        <statistics> returned in the response can
                        cause the response to exceed the
                        negotiated data link transmission
                        parameters of the C12.22 Device. For this
                        reason, when the C12.22 Communication
                        Module is constrained by the C12.22
                        Device's data link in a manner that
                        prevents it from sending all of the
                        statistics requested, the C12.22
                        Communication Module shall deliver the
                        maximum number of <statistics> it can
                        transmit according to the data link
                        configuration parameters.}

<statistic-code> ::= <word16>             {This code identifies the associated
<statistic-value>    {The list of standard
                        codes available is defined in the Network
                        Statistics Table (Table 128). <service-
                        code> zero (00h) followed by <statistic-
                        value> of zero (00h) represents the end-
                        of-list.}

<statistic-value>    ::= <int48>          {Statistic returned.}

```

### 6.7.6 Get Registration Status Service

The Get Registration Status Service is used by a C12.22 Device to get registration information from a C12.22 Communication Module. The information carried by this service is the same as that found in the Registration Table (Table 126).

Note: The current version of the Standard does not provide a capability for a C12.22 Communication Module to perform an authenticated or an encrypted Registration Service on behalf of the C12.22 Device. Therefore, it shall be the responsibility of the C12.22 Device to manage C12.22 Node Registration when the service calls for the use of authenticated or encrypted Registration Service Requests.

#### Request:

```
<tls-get-reg-status> ::= 76H
```

#### Response:

The response <err> indicates a problem with the received service request.

The response <ok> is returned by the C12.22 Communication Module with the available information.

```
<tls-get-reg-status-r> ::= <err> |
```

```

    <ok>
    <relay-native-address-len> <relay-native-address>
    <master-relay-aptitle> <relay-ap-title>
    <node-ap-title> <reg-delay> <reg-period>
    <reg-count-down>

    <relay-native-address-len> ::= <byte>      {Number of bytes in <relay-native-
                                              address>..}

    <relay-native-address> ::= <byte>* {Native address of the nearest C12.22
                                      Relay used to maintain registration of
                                      this C12.22 Node..}

    <master-relay-aptitle> ::=                <universal-id-element> | <relative-uid-
                                              element>
                                              {ApTitle of the C12.22 Master Relay used
                                              to register this C12.22 Device..}

    <relay-ap-title> ::= <universal-id-element> | <relative-uid-element> {
                                      The Calling ApTitle of the local C12.22
                                      Relay that was used to register this
                                      C12.22 Node. If the C12.22 Communication
                                      Module used direct messaging (it is co-
                                      located on the same C12.22 Network Segment
                                      of the C12.22 Master Relay) then the
                                      <relay-ap-title> shall be encoded as a
                                      <universal-id-element> with the length
                                      field set to zero (0)..}

    <node-ap-title> ::= <universal-id-element> | <relative-uid-element>
                      {ApTitle used to register this C12.22
                      Device through this interface..}

    <reg-delay>      ::= <word16>      {Maximum delay in seconds that the C12.22
                                      Device should wait before registering
                                      after a power-up..}

    <reg-period>     ::= <word24>      {Maximum period in seconds allowed by the
                                      local C12.22 Relay between two messages
                                      received from the C12.22 Node. The C12.22
                                      Device will be automatically deregistered
                                      when this limit is reached..}



    <reg-count-down> ::= <word16>      {The amount of time in minutes left before
                                      the registration period expires..}

```

### 6.7.7 Service Time Sequence Diagrams

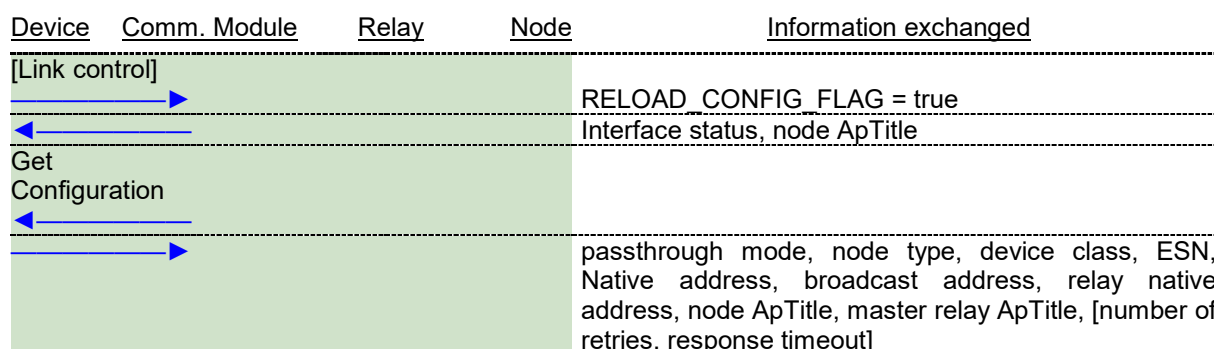
#### Negotiate

The following diagram shows information exchanged between a C12.22 Device and a C12.22 Communication Module at start-up.

Device	Comm. Module	Relay	Node	Information exchanged
Negotiate				
				packet size, number of packets, baud rate(s), number of channels
				packet size, number of packets, baud rate number of channels, interface number

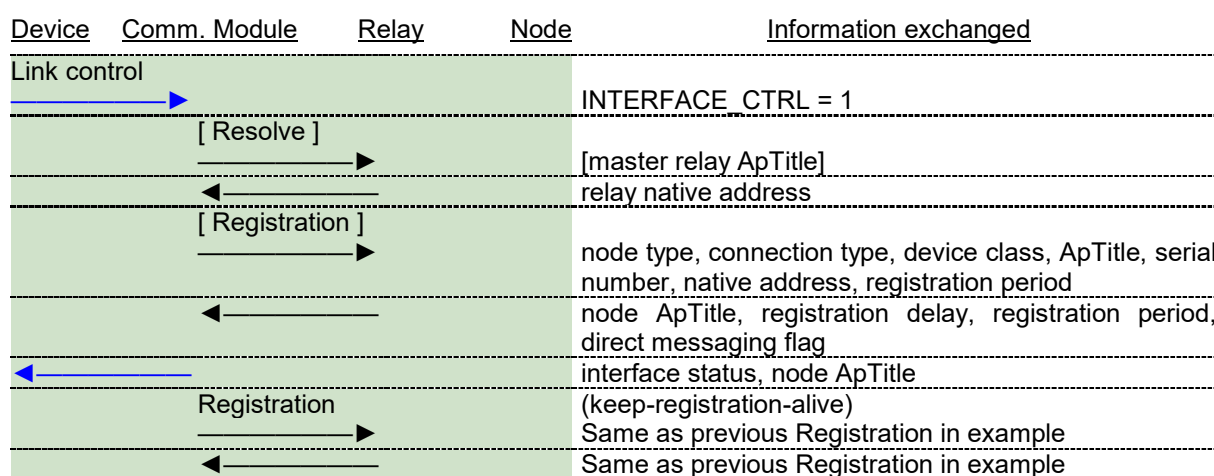
## Get Configuration

The following diagram shows information exchanged when the C12.22 Communication module retrieves its configuration from the C12.22 Device.



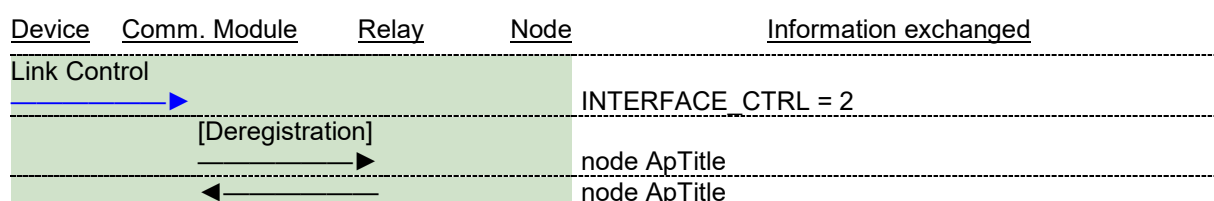
## Link Control (Enable Network-side Interface)

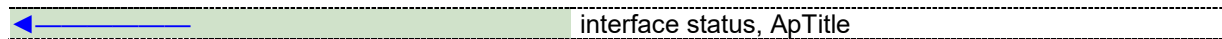
The following diagram shows the information exchanged for enabling a C12.22 Communication Module interface on the network side. It is important to note that the use of the Resolve Service is optional and not needed if the C12.22 Communication Module already knows the native address of its nearest C12.22 Relay. When the C12.22 Communication Module supports multiple routes, the Resolve and Register services are used multiple times to register each route. A different ApTitle shall be used for each route.



## Link Control (Disable Network-side Interface)

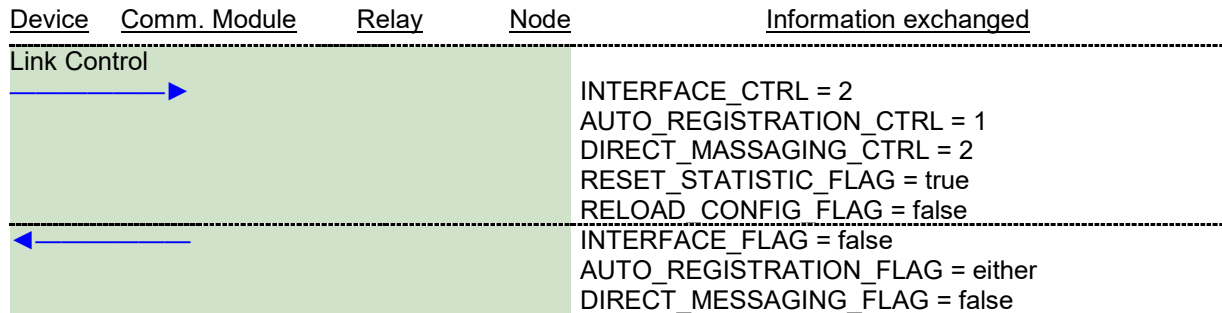
The following diagram shows the information exchanged for disabling a C12.22 interface.





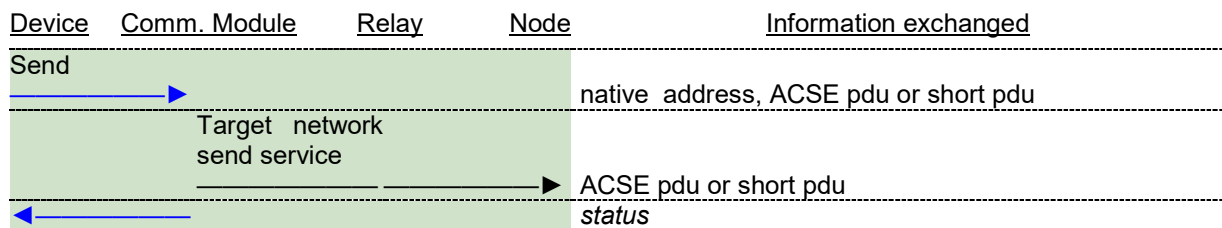
### Link Control (Auto Registration, Direct Messaging, Reset Statistic Control or Reload Configuration)

The following diagram shows the information exchanged for disabling a C12.22 interface.



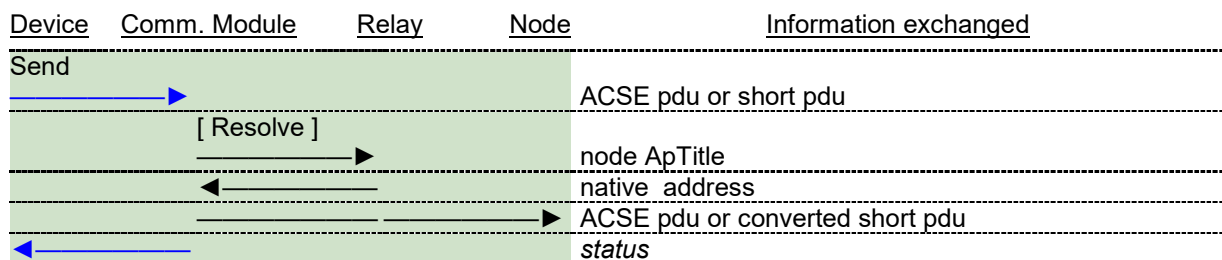
### Send (Directed Message)

The following diagram shows the information exchanged when the Send Service is generated by a C12.22 Device and the "Directed Message" option has been invoked.



### Send (Direct Messaging)

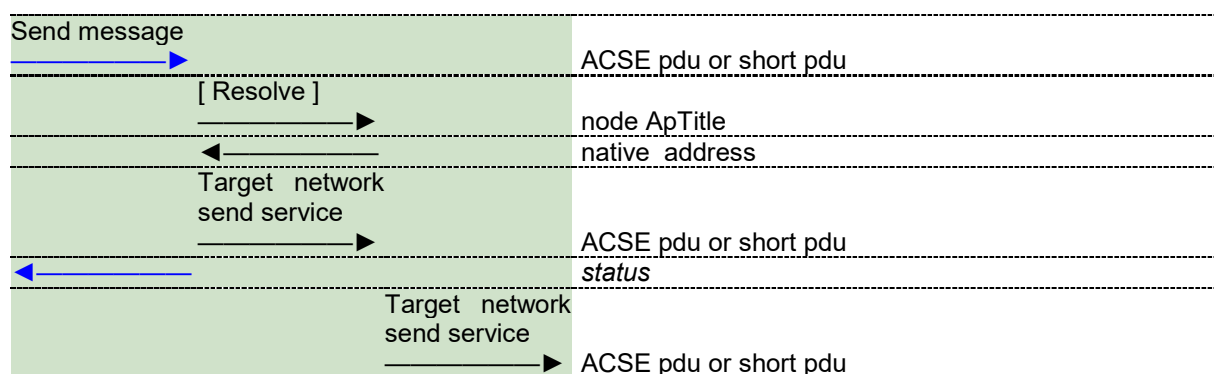
The following diagram shows the information exchanged when the Send Service is generated by a C12.22 Device and the "Direct Messaging" option has been invoked.



### Send (Message Forwarding)

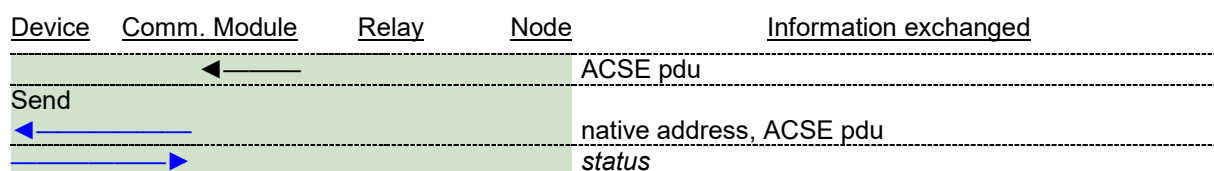
The following diagram shows the information exchanged when the Send Service is generated by a C12.22 Device and the "Message Forwarding" option has been invoked.





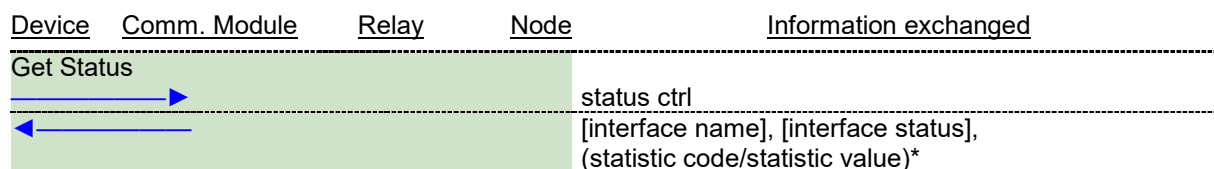
### Send (Message Received)

The following diagram shows the information exchanged when the Send Service is received by a C12.22 Device.



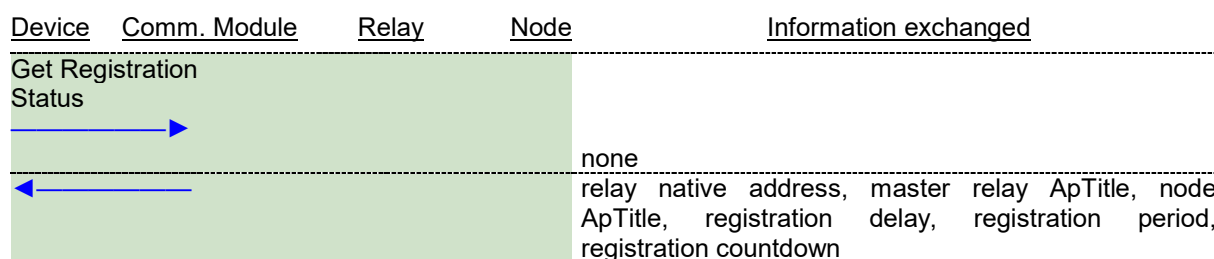
### Get Status

The following diagram shows the information exchanged when the Get Status Service is initiated by a C12.22 Device.



### Get Registration Status

The following diagram shows the information exchanged when the Get Registration Status Service is initiated by a C12.22 Device.



## 6.7.8 Service Sequence States

The Transport Layer is defined as a series of Service Sequence States. For the C12.22 Communication Module, valid states include:

- Disconnected State:** Initial state entered after power-up. The module initializes itself (possibly using the Get Configuration service) and then enters the Enabled state.
- Enabled State:** It is only in this state that the C12.22 Communication Module can actually send a message over the network.
- Disabled State:** State entered after the C12.22 Communication Module has been requested to disable its network interface via the Link Control service INTERFACE\_CTRL field.

The relationship between services and service sequence states for a C12.22 Communication Module is:

- Negotiate:** This service may be issued while the C12.22 Communication Module in any state.
- Get Configuration:** This service may be issued while the C12.22 Communication Module is in any state.
- Link Control:** This service is accepted in the Enabled or Disabled state, but not in the Disconnected state. The Link Control service causes the transitions from Enabled or Disabled state to any other state.
- Send Message:** This service can be accepted or issued in the Enabled or Disabled state. If the C12.22 Communication Module is requested to send a message to the C12.22 Network while it is in the Disabled state, it shall respond with a <netr> response.
- Get Status:** This service is accepted both in the Enabled and Disabled states.
- Get Registration Status:** This service is accepted in the Enabled and in the Disabled states.

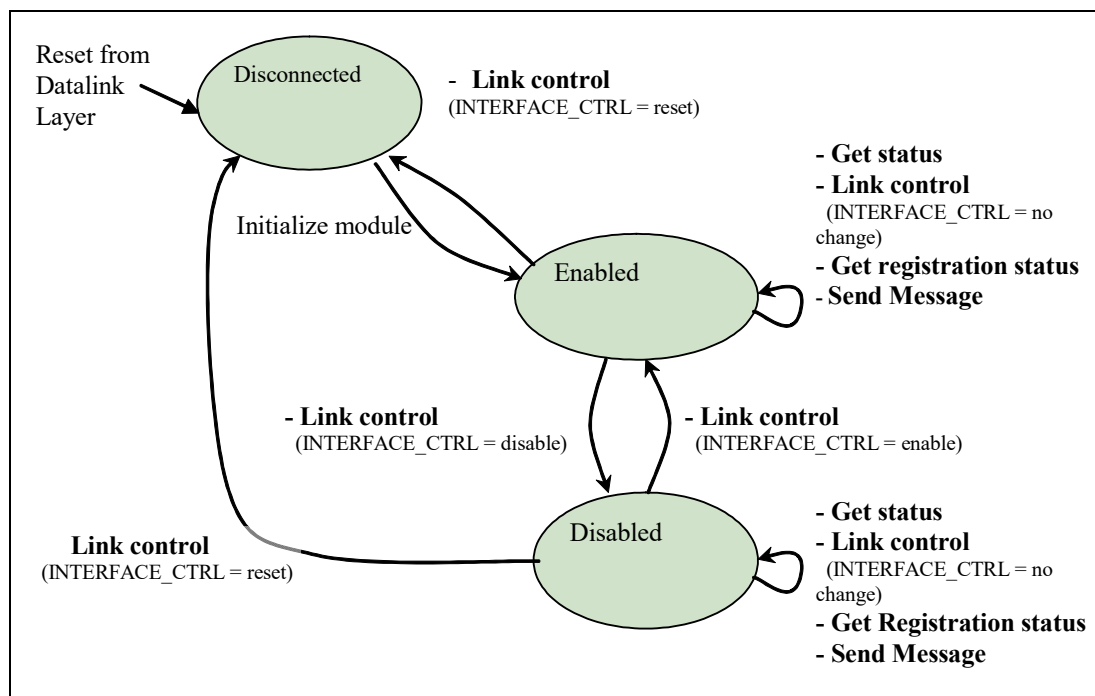


Figure 6.2: Transport Layer State Diagram for C12.22 Communication Module

For the C12.22 Device, the states and transitions are different. For a C12.22 Device, valid states include:



Disconnected State:	Initial state entered after power-up. The module, if present, is initializing itself and may not be immediately available.
Comm Module Present State:	The C12.22 Device may receive Get Configuration requests from the C12.22 Communication Module in this state. The C12.22 Device may send any service requests to the Comm Module during this state.
Comm Module Disabled State:	State entered after the C12.22 Communication Module has been instructed to disable itself.

The relationship between services and service sequence states is for a C12.22 Device:

Negotiate:	This service may be issued during the Comm Module Present state only.
Get Configuration:	This service may be received from the Comm Module in any state.
Link Control:	This service may be issued in either the Comm Module Present or Comm Module Disabled state.
Send Message:	This service can be sent in either the Comm Module Present or the Comm Module Disabled states.
Get Status:	This service may be sent in either the Comm Module Present or the Comm Module Disabled states.
Get Registration Status:	This service should be sent in the Comm Module Present state only.

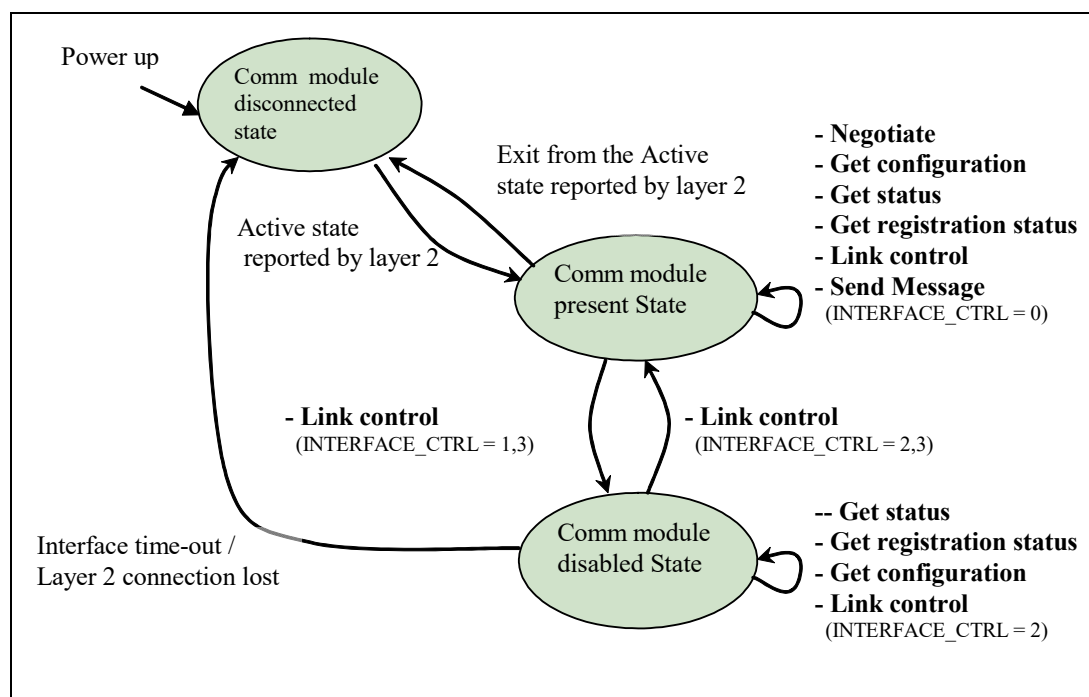


Figure 6.3: Transport Layer State Diagram for C12.22 Device

## 6.8 Layer 3 - Network Layer

Null layer.

## 6.9 Layer 2 - Data Link Layer

The Data Link Layer is used only for communication between the C12.22 Device and the C12.22 Communication Module.

Datagrams of upper layers are transported in one or more packets. Each packet is variable in length but

cannot exceed a maximum packet size. The maximum packet size has a default value when the communication link is opened. The maximum packet size and number of packet attributes are changed through the use of the Negotiate Service.

The C12.22 Device may support transfer of multiple Datagrams at the same time via interleaving packets. To accomplish this, a different channel number is associated for each simultaneous Datagram transmitted. The C12.22 Communication Module shall be responsible for the generation of channel numbers for all packets it sends to the C12.22 Device. Similarly the C12.22 Device shall be responsible for the generation of channel numbers for all packets it sends to the C12.22 Communication Module. All packets belonging to one Datagram shall have the same channel number. Channel numbers generated by the C12.22 Communication Module are independent of the channel numbers that are generated by the C12.22 Device.

For each packet received, the target sends a positive or negative acknowledgment. This acknowledgment consists of a single byte transmitted outside of the packet structure or inside a packet structure if requested by the TRANSPARENCY flag. If the requester does not receive an acknowledgment before the Response Timeout, or a negative acknowledgment is received, the same packet is re-transmitted up to the maximum number of three (3) retry attempts. After the final retry attempt, the requester should assume that the communication link is down and try to reestablish it.

The Data Link Layer also supports line supervision by the transmission of empty packets to avoid Traffic Time-out. An empty packet is defined as a <packet> with the <length> field set to zero.

### 6.9.1 Basic Data Information

Communication link settings are specified below.

#### 6.9.1.1 Fixed Settings

<b>Data Format</b>	8 data bits, 1 start bit, 1 stop bit, no parity
<b>Data Type</b>	Asynchronous, serial bit (start - stop), full duplex
<b>Data Polarity</b>	Start bit, space, logical 0 Stop bit, mark, logical 1, quiescent state
<b>Bit Order</b>	Bits within each byte are transmitted least significant bit first with the least significant bit identified as bit 0 and most significant bit as bit 7.
<b>Traffic Timeout</b>	6 seconds
<b>Inter-character Timeout</b>	500 milliseconds
<b>Response Timeout</b>	2 seconds
<b>Retry Attempts</b>	3

#### 6.9.1.2 Variable Settings

Default settings apply at the initiation of communication. These settings may be changed through the Negotiate Service. Communication link settings will return to defaults as a result of a Traffic Time-out.

Setting	Default Value	Changed by
<b>Data Rate</b>	9600 baud	Negotiate Service
<b>Number of Packets</b>	1	Negotiate Service
<b>Packet Size</b>	64 bytes	Negotiate Service

### 6.9.2 Packet Definition

<packet> ::= <stp> <identity> <ctrl> <seq-nbr> <length> <data> <crc>

<stp> ::= EE<sub>H</sub> {Start of packet character.}

<code>&lt;identity&gt;</code>	<code>::= &lt;byte&gt;</code>	<p>{Identity.  Bits 3 to 7: LOCAL_ADDRESS_NUMBER  This field is used to facilitate routing of information between a Local Port and the local C12.22 Communication Modules. This field can be set to the following values:  0 = C12.22 Device  1 = Local Port 0 (Default)  2 = Local Port 1 (Alternate)  3 = Interface 0 (Default WAN)  4 = Interface 1 (Alternate WAN)  5 = Interface 2 (Default POT modem)  6 = Interface 3 (Alternate POT modem)  7 to 12 = Reserved  13 to 28 = Manufacturer-defined  29 = Invalid (to avoid &lt;stp&gt;)  30 to 31 = Reserved for future extension of this field</p> <p>Bits 0 to 2: CHANNEL NUMBER  This field allows simultaneous transfer of multiple segmented &lt;acse-pdu&gt;s. A different number is assigned dynamically and independently by the C12.22 Device or by the C12.22 Communication module for each &lt;acse-pdu&gt; message transferred simultaneously in any one direction. It is permissible for the channel numbers assigned by the C12.22 Device or by the C12.22 Communication module to overlap.}</p>
<code>&lt;ctrl&gt;</code>	<code>::= &lt;byte&gt;</code>	<p>{Control field.  Bit 7: MULTI_PACKET_TRANSMISSION  If true, then this packet is part of a multiple packet transmission.</p> <p>Bit 6: FIRST_PACKET  If true, then this packet is the first packet of a multiple packet transmission (MULTI_PACKET_TRANSMISSION, bit-7, is also set to 1), otherwise this bit shall be set to 0.</p> <p>Bit 5: TOGGLE_BIT  Represents a toggle bit to reject duplicate packets. This bit is toggled for each new packet sent. Retransmitted packets keep the same state as the original packet sent. After a Traffic Time-out, the state of the toggle bit may be re-initialized; thus, it is assumed to be indeterminate.</p> <p>Bit 4: TRANSPARENCY  If true, this packet uses the transparency mechanism as described in section 6.9.10 "Transparency". Also the target shall respond with an ACK or optional NAK inside</p>

the packet structure as described in section 6.9.4 "Acknowledgment".

#### Bits 2 to 3: ACKNOWLEDGMENT

Used to acknowledge the reception of a valid or invalid packet. Acknowledgment and response data can be sent together in the same packet or as two consecutive packets.

0 = No acknowledgment included of the previous packet.

1 = Positive acknowledgment of the previous packet.

2 = Negative acknowledgment of the previous packet.

3 = Unacknowledged packet sent.

A possible use of option 3 is when a one-way device sends blurts which do not require acknowledgments. In this context, the C12.22 Device shall be the originator of all messages. This is an indication to the C12.22 Communication Module to disable the ACK/NAK packet handshake for one-way devices.

#### Bit 0 to 1: DATA FORMAT

0 = C12.18 or C12.21 Device

1 = C12.22 Device

2 = Invalid (to avoid <stp>)

3 = Reserved}

<seq-nbr>	::= <byte>	{Number that is decremented by one for each new packet sent. The first packet in a multiple packet transmission shall have a <seq-nbr> equal to the total number of packets minus one. A value of zero in this field indicates that this packet is the last packet of a multiple packet transmission. If only one packet is in a transmission, this field shall be set to zero.}
<length>	::= <word16>	{Number of bytes of <data> in packet.}
<data>	::= <byte>*	{<length> number of bytes of actual packet data. The length of the data is limited by the maximum packet size minus the packet overhead. This value can never exceed 8183.}
<crc>	::= <word16>	{HDLC implementation of the polynomial $X^{16} + X^{12} + X^5 + 1$ }

### 6.9.3 CRC Selection

The CRC selected for implementation is the polynomial  $X^{16} + X^{12} + X^5 + 1$ . The method for calculation and insertion is the HDLC standard per ISO/IEC 13239 (2002) Annex A.

In the EPSEM packet, there is no opening flag as referenced in ISO/IEC 13239 (2002) Annex A. The

EPSEM start of packet character EE is included in the CRC calculation. The result of the CRC calculation shall be transmitted least significant byte first per the ISO/IEC 13239 (2002) Annex. See “Annex H - CRC Examples” for examples of computation and coding.

#### 6.9.4 Acknowledgment

A positive or negative acknowledgment is used by the communicating devices to indicate either acceptance or rejection of a packet.

An <ack> shall be issued by the receiving device to the transmitting device for each valid packet received. When an <ack> or <nak> is encoded as a <packet> then the ACKNOWLEDGMENT field of <ctrl> shall be set to 1 (one) or 2 (two) and the CHANNEL NUMBER of the <identity> field shall be identical to the CHANNEL NUMBER of the <packet> that is positively or negatively acknowledged.

<ack> ::= 06<sub>H</sub> | <packet>

A <nak> shall be issued by the receiving device to the transmitting device for each packet received that

1. begins with <stp>, and
2. must be followed by five additional bytes followed by <length> bytes followed by two additional bytes, and
3. is completely received without any intervening inter-character time-outs, and
4. contains some other error.

<nak> ::= 15<sub>H</sub> | <packet>

#### 6.9.5 Retry Attempts

The same packet shall be retransmitted if a <nak> is received or the Response Time-out occurs.

After the failure of the final retry attempt, the communication link is considered down, and the C12.22 Communication Module shall attempt to reestablish the link by repeatedly either sending a Negotiate Service request or Get Configuration Service request until the receipt of a valid response from the C12.22 Device for the initiated service.

#### 6.9.6 Timeouts

##### 6.9.6.1 Traffic Time-out

The C12.22 Device will consider the communications link down after waiting some period of time for a valid packet or acknowledgment. This period of time is defined as the Traffic Time-out.

##### 6.9.6.2 Inter-character Time-out

The recipient of the packet must handle the case when the end of a packet is lost. Inter-character Time-out is defined as the minimum amount of time the recipient shall wait between characters within a packet when receiving data over the communication link. The recipient shall wait at least this amount of time before it ceases to wait for the remainder of the packet and sends a <nak>.

##### 6.9.6.3 Response Time-out

The sender of the packet must handle the case when the acknowledgment or negative acknowledgment is lost. Response Time-out is defined as the minimum amount of time the sender shall wait after a packet is sent to receive an acknowledgment over the communication link. If this amount of time elapses, the

sender will send the packet again.

### 6.9.7 Turn Around Delay

The Turn Around Delay is the minimum delay the recipient must wait after receiving a packet and before sending a positive or negative acknowledgement.

### 6.9.8 Collision

A C12.22 Device (or a C12.22 Communication Module) shall be able to process an incoming <packet> across its interface while it transmits an outgoing <packet> across that same interface. The C12.22 Device (or a C12.22 Communication Module) shall send acknowledge (<ack> or <nak>) immediately, if appropriate, for any incoming <packet> received across that interface before sending a next <packet>. The C12.22 Device shall cease transmission and wait for the transmission from the C12.22 Client. When a C12.22 Device (or a C12.22 Communication Module) expects an <ack> or <nak>, but receives an incoming <packet> across the interface, it shall first process the incoming <packet>; then it shall reset its response time-out timer and finally it shall resume waiting for an <ack> or <nak>.

### 6.9.9 Duplicate Packets

A duplicate packet is defined as a packet whose identity, toggle bit and valid CRC are identical to those of the immediate previous packet. A second check may be performed on all bytes in the packet to prevent false detection of duplicate packets, as CRC16 is not foolproof in a multi-channel setting. If a duplicate packet is received by the target device, the target should discard the duplicate packet and return an <ack>, unless ACKNOWLEDGMENT = 3 in the packet's <ctrl> field. The toggle bit in the first packet may assume either state following link establishment or following link re-establishment.

### 6.9.10 Transparency

The "Basic Transparency" method is requested by the TRANSPARENCY bit in the <ctrl> byte. This method replaces any occurrence of the <stp> by a two-octet escape sequence. It shall also dictate that <ack> and <nak> responses be encapsulated within packets.

After CRC computation, the transmitter shall replace any occurrence of the <stp> or 1B<sub>H</sub> (escape flag as defined in ISO/IEC 13239) found inside the packet by a two-octet sequence consisting of the 1B<sub>H</sub> and the original octet with bit 5 complemented. Prior to CRC computation, the receiver removes every occurrence of 1B<sub>H</sub> and inverts bit 5 of the following octet.

Therefore, the following replacement shall result:

<u>Transmitted</u>	<u>Received</u>
1B <sub>H</sub> -> 1B <sub>H</sub> 3B <sub>H</sub>	1B <sub>H</sub> 3B <sub>H</sub> -> 1B <sub>H</sub>
EE <sub>H</sub> -> 1B <sub>H</sub> CE <sub>H</sub>	1B <sub>H</sub> CE <sub>H</sub> -> EE <sub>H</sub>

### 6.9.11 Supervision of the Communications Link

The C12.22 Communications Module is assumed to continually poll the C12.22 Device using an empty packet (packet with the <length> field set to 0). Any message transported in either direction that is confirmed by an acknowledgement can be accepted to reset the traffic timer. If the timer expires, the link can enter the Disconnected State. The C12.22 Device, at its option, can activate a reset of the C12.22 Communications Module (and perhaps itself) to re-establish the link and re-enter the Connected State.

The C12.22 Device has the responsibility to supervise the C12.22 Communication Module for proper performance on all communications levels. The method for concluding a C12.22 Communication Module malfunction is at the discretion of the manufacturer of the C12.22 Device. Upon determination that the C12.22 Communication Module is in an "insane" state or needs resetting for any reason, the C12.22

Device may perform a reset of the C12.22 Communication Module by dropping the RESET line for greater than 50 msec.

### 6.9.12 Local Routing

The LOCAL\_ADDRESS\_NUMBER field defined in the <identity> byte of the packet is used to facilitate routing of <acse-pdu> among Local Ports and C12.22 Communication Modules. Layer 2 local routing shall be implemented by all C12.22 Devices to facilitate C12.22 Communication Module setup and diagnostics. It is highly recommended that this service be used by C12.22 Communication Modules to route <acse-pdu>s received from one interface and targeting a different interface or Local Port as defined by section 5.3.4.12 "Use of Subbranches of a Registered ApTitle", subsection "Access to Interface and Local Ports".

#### General Considerations

1. Each packet sent by a device attached to a Local Port or a C12.22 Communication Module has its LOCAL\_ADDRESS\_NUMBER set to the desired destination.
2. Each packet received by a C12.22 Device with the LOCAL\_ADDRESS\_NUMBER field set to non-zero is forward to the requested destination. Because routed packets exist outside the context of a negotiated session on the device doing the routing, packet size for routed packets must not exceed the Layer 2 default size set by this Standard (64 bytes) and the number of packets that make up an <acse-pdu> shall not be greater than 1, in order to avoid re-segmentation by the C12.22 Device.
3. Each packet received by a C12.22 Device with the LOCAL\_ADDRESS\_NUMBER field set to zero is intended for that device. Such packets may therefore be larger than 64 bytes if a larger size has been successfully negotiated via the protocol.

#### Local Routing Rules for C12.22 Devices

The routing rules are applied only after the packet has been verified to be valid and non-duplicate. The LOCAL\_ADDRESS\_NUMBER refers to a destination port.

When the C12.22 Device receives a packet that has the LOCAL\_ADDRESS\_NUMBER equal to zero, it means that the packet is for this C12.22 Device. Such packets should be acknowledged (unless ACKNOWLEDGMENT = 3 in the packet's <ctrl> field) with an <ack> and passed up to the upper layers of this C12.22 Device for processing.

If the C12.22 Device receives a packet with a LOCAL\_ADDRESS\_NUMBER equal to some non-zero port number, the C12.22 Device shall apply the following rules, in this order:

1. If the destination port is busy (i.e. a packet has been sent to that port, but has not yet been acknowledged by an <ack>), silently discard the packet.
2. Otherwise if the <packet> is invalid (as per section 6.9.4 "Acknowledgment") send a <nak> (unless ACKNOWLEDGMENT = 3 in the packet's <ctrl> field).
3. Otherwise if the destination port does not exist or it is already known to the C12.22 Device that no C12.22 Communications Module is attached to that port, acknowledge the packet (unless ACKNOWLEDGMENT = 3 in the packet's <ctrl> field) with an <ack> and then either discard it or optionally pass it up to higher layers so that an appropriate Application Layer error response may be returned.
4. Otherwise if the <packet> is valid send an <ack> (unless ACKNOWLEDGMENT = 3 in the packet's <ctrl> field) to the source port and then modify the packet, replacing the LOCAL\_ADDRESS\_NUMBER in the packet with the source port number. Recalculate the CRC and forward this packet to the destination port.

#### Local Routing Rules for C12.22 Communication Modules

For the purposes of this section, the word "port" is intended to mean either a Local Port of a C12.22

Device or a C12.22 Communications Module interface to a C12.22 Device. The routing rules are applied only after the packet has been verified to be valid and non-duplicate.

When the C12.22 Communication Module receives a packet, from the C12.22 Device, which has the LOCAL\_ADDRESS\_NUMBER equal to zero, it means that the packet is for this C12.22 Communication Module, so such packets should be acknowledged by an <ack> and passed up to the upper layers of this C12.22 Communication Module for processing.

If the C12.22 Communication Module receives a packet from the C12.22 Device, the LOCAL\_ADDRESS\_NUMBER signifies where the response (if any) should be sent. The C12.22 Communication Module should acknowledge the packet with an <ack> or <nak> (unless ACKNOWLEDGMENT = 3 in the packet's <ctrl> field) then pass the packet's contents, including the LOCAL\_ADDRESS\_NUMBER, up to the upper layers of this C12.22 Communication Module for processing. Subsequently the C12.22 Communication Module shall generate applicable service-related packets to this LOCAL\_ADDRESS\_NUMBER.

### **6.9.13 Service Sequence States**

Data Link Layer communications is defined as a series of "Service Sequence States".

Valid states include:

- |                     |  |
|---------------------|--|
| Disconnected State: | State entered after power-up or upon communication link drop detection. All parameters defined in section 6.9.1.2 "Variable Settings" return to their defaults.            |
| Connected State:    | State established upon detection of the other device (i.e., the C12.22 Device for a C12.22 Communications Module or the C12.22 Communications Module for a C12.22 Device). |
| Active State:       | State entered when any valid local traffic is sent or received.  |



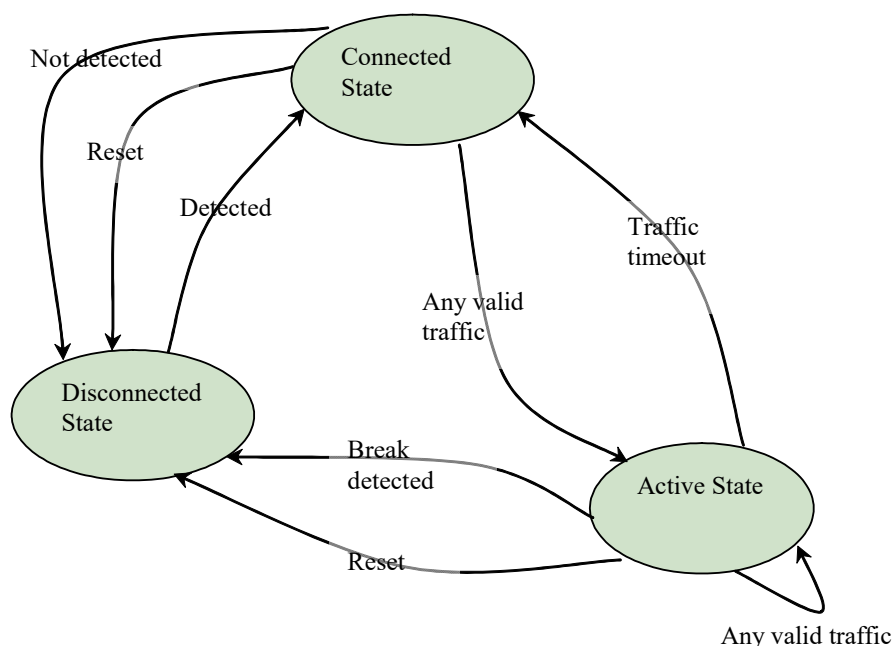


Figure 6.4: Data Link Layer State Diagram

## 6.10 Layer 1 - Physical Layer

The Physical Layer is defined for the Interface between the C12.22 Device and the C12.22 Communication Modules.

### 6.10.1 Signal Definition

Signal #	Signal	Description	C12.22 Device DTE	C12.22 Comm. Module DCE
1	RxD	Receive Data	Input	Output
2	TxD	Transmit Data	Output	Input
3	RESET	Module Reset	Output	Input
4	HSCD	High-speed cable detect	Input	Input
5	VPLUS	C12.22 Comm. Module power supply	Output	Input
6	GND	Ground/Common		

### 6.10.2 Electrical Properties of Connection

The following properties are required for the interface lines:

- Each side of the interface shall provide a pull-down resistor to Ground/Common on its input (Signal #1 and #4 for the C12.22 Device, Signals #2 and #4 for the C12.22 Communication Module). This pull-down enables supervision of the signal line and enables detection of a mated connection on that line. There shall be a pull-up resistor on Signal line #3 on both sides of the link. All transmitters output an idle logical 1, positive voltage, so that connection can be detected. Also this sets the default communication speed to low speeds (less than or equal to 256000 bits / sec).
- VPLUS may provide power to the C12.22 Communication Module. If VPLUS does provide power, a C12.22 Device is expected to provide up to 1.5 W with a voltage range of 5-12V.

3. If a C12.22 Device does not provide power to VPLUS, it shall be left disconnected allowing a compliant external power supply to drive VPLUS for the attached C12.22 Communication Module.
4. The RESET is the means for the C12.22 Device to reset the C12.22 Communication Module. The RESET signal should be active low with a pulse of 50 ms or greater.
5. The HSCD is the means for the C12.22 Device and the C12.22 Communication Module to detect the presence of a high-speed cable connection. The HSCD signal shall be active high (high speed is enabled).
6. Voltage input high  $>2.5V$ , logical 1.
7. Voltage input low level  $\leq 0.8V$ , logical 0
8. Idle voltage of link is logical 1.
9. Maximum output voltage shall be less than or equal to the supplied VPLUS voltage.
10. Source impedance of output lines shall be  $\leq 1K$  Ohms.
11. Input impedance of input lines shall be  $\geq 100K$  Ohms.
12. Data Rate
  - Minimum of 300 bits / sec.
  - Nominal of 9600 bits / sec (default).
  - Maximum of 256000 bits / sec (using standard cable with a maximum length of 1m).
  - Maximum of 4000000 bits / sec (using high-speed rated cable).
13. All signals shall meet IEEE C37.90.1-2002 requirements.
14. The C12.22 Device or C12.22 Communication Module shall provide line isolation on any and all physical wire and plug pins exiting from under the cover. The insulation must be rated to ANSI C12.1-2001 insulation 2.5kV test requirements between all voltage and current carrying parts and all other metallic parts.
15. All C12.22 Device or C12.22 Communication Module wires, plugs, contacts and metallic parts exiting the device cover shall meet electrostatic discharge (ESD) testing to 10kV.
16. All C12.22 Device or C12.22 Communication Module wires, plugs, contacts and metallic parts exiting the C12.22 Device cover shall meet IEEE C62.41-2002 for voltage and current carrying parts.
17. Maximum connection length is 1 meter at up to 256000 bits / sec, using standard cable.

### 6.10.3 Mechanical and Environmental Properties

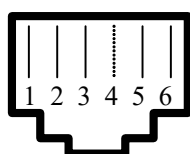
This section defines connectors for the Physical Layer of a C12.22 interface between a C12.22 Device and a C12.22 Communication Module when this connection is exposed externally on the respective parts.

This Standard assumes that the environmental protection and strain relief is integrated into the enclosure containing the connector. Modular plug cabling is used to connect from a C12.22 Device to a C12.22 Communications Module that will each contain a modular jack for termination. Below are typical parts for the C12.22 Device and C12.22 Communications Module. Equivalent parts are acceptable.

Connector for C12.22 Device: RJ11 Jack (typical part AMP520250-2)

Connector for C12.22 Communications Module: RJ11 Jack (typical part AMP520250-2)

RJ11 Jack Front View



Cable Plug View

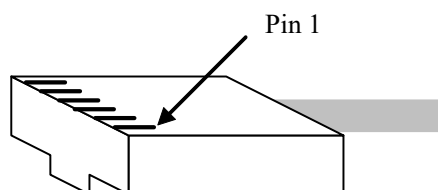


Figure 6.5: Jack and Cable Plug Wiring Diagram

Figure 6.5 shows the mapping of the signals defined in 7.10.1 to the pins in the 6-pin RJ11 jack and plug.

#### **6.10.4 Supervision of the Communications Link**

Either the C12.22 Communication Module or the C12.22 Device can detect the other by the presence of a marking signal (logical 1) on the corresponding transmit or receive data line. By detecting the “break” condition (continuous logical 0) disconnection can be determined.

## 7 Local Port Communication Protocol Details

This section defines the protocol stack used by C12.22 Local Ports.

### 7.1 Protocol Definition

#### 7.1.1 Layer 7 - Application Layer

As defined in section 5.3 “Layer 7 - Application Layer”.

To access a C12.22 Node or one of its interfaces through a Local Port, a C12.22 Node that is attached through the Local Port does not have to include the <calling-AP-title-element> and <called-AP-title-element> in <acse-pdu>s transmitted. The data link LOCAL\_ADDRESS\_NUMBER is used as described in section 6.9.12 “Local Routing”.

Optionally, an interface may be used as a proxy to send messages over the C12.22 Network as defined in section 5.3.4.12 “Use of Subbranches of a Registered ApTitle”. In this case, both the <called-AP-title-element> of the targeted C12.22 Node and the LOCAL\_ADDRESS\_NUMBER of the interface used to send this message are provided.

#### 7.1.2 Layer 6 - Presentation Layer

Null layer.

#### 7.1.3 Layer 5 - Session Layer

Null layer.

#### 7.1.4 Layer 4 - Transport Layer

The Local Port shall operate as an ANSI C12.22 Device. As such, Layer 4 shall operate as per section 6’ “Protocol Details: C12.22 Device to C12.22 Communication Module Interface”. Also, the only services supported in Layer 4 shall be the <tls-negotiate> and the <tls-send-message>.

The state diagram of this layer used in this context differs from the one defined in section 6.7.8, “Service Sequence States”. For point-to-point communication, the following states are defined:

Base State:	This is the state at which communication begins. At this point the default data transmission parameters apply.
-------------	--

Negotiated State:	At this state, the parameters negotiated by the Negotiate Service apply.
-------------------	--

The relationship between services and service sequence states is:

Negotiate:	This service is accepted at the Base State only. The successful completion of this service (including the transmission of the response) transitions communications to the Negotiated state. This service cannot be used to negotiate the <rec-packet-size> and <rec-nbr-packet> when communicating with an interface.
------------	---

Send message:	This service can be accepted at Base State and Negotiated state to send and receive Layer 7 messages.
---------------	---

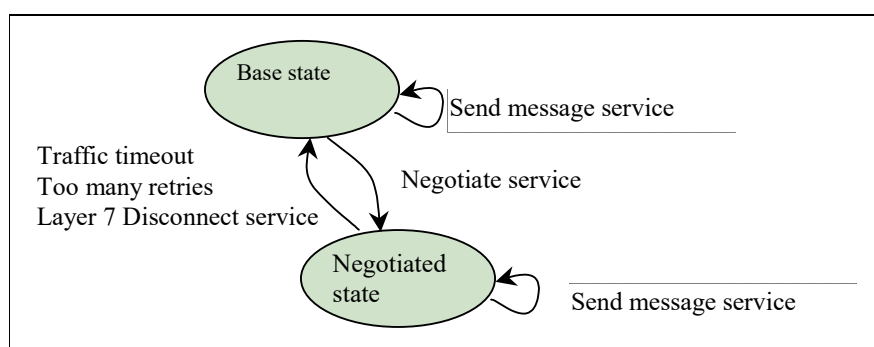


Figure 7.1: Transport Layer State Diagram

### 7.1.5 Layer 3 - Network Layer

Null layer.

### 7.1.6 Layer 2 - Data Link Layer

As defined in section 6.9 “Layer 2 - Data Link Layer”.

Time-outs are fixed for each type of communication link and are defined as follows:

Local Port timeouts	Interface	Optical port	Modem
TRAFFIC TIMEOUT	6 seconds	6 seconds	30 seconds
INTER CHARACTER TIMEOUT	500 milliseconds	500 milliseconds	1 second
RESPONSE TIMEOUT	2 seconds	2 seconds	4 seconds
TURN AROUND DELAY	Unspecified	175 microseconds	Unspecified

### 7.1.7 Layer 1 - Physical Layer

The Physical Layer is not defined in this Standard and is left to the discretion of implementers.

## 7.2 C12.22 Local Port Communication Using a C12.18 Optical Port

Some C12.22 Nodes may implement a Local Port (see definition of Local Port) of the form and type defined in ANSI C12.18. Under this condition two possibilities are considered:

- The Local Port (such as ANSI Type 2 attached to a C12.22 Device) supports both the ANSI C12.22 Standard and ANSI C12.18 Standard. As such, this section describes a methodology of a “smart” C12.18 Device (e.g. handheld reader), to determine that the Local Port supports the C12.22 communication protocol, and then choose to communicate using ANSI C12.18 or ANSI C12.22 at its option.
- The Local Port supports only ANSI C12.22 communication. As such, the C12.22 reader can only communicate (no option) using the C12.22 standard communication protocol.

Another possible implementation of a C12.22 Node is one that has an optical port but does not support C12.22 over the optical port. This possibility is not considered in this Standard.

When communicating in ANSI C12.18 compatibility mode, the optical port shall operate as an ANSI C12.18 device. As such, all Physical, Data Link and Application Layer services shall comply with ANSI C12.18.

When communicating in ANSI C12.22 compatibility mode, the optical Local Port shall operate as defined

in section 7.1 "Protocol Definition" and Layer 1 (Physical Layer of the optical port) shall operate according to ANSI C12.18 Physical Layer requirements.

### **7.2.1 Establishment of ANSI C12.18 Protocol Compatibility Mode**

ANSI C12.22 provides optional optical port compatibility with ANSI C12.18.

Any C12.18-compatible reading device desiring to successfully connect to a Local Port shall set bit 0 in DATA\_FORMAT of the <ctrl> byte to 0 in each transmitted <packet>. This control bit is found in the <packet> defined in section 6.9.2 "Packet Definition".

Consequently, the C12.22 Node having a Local Port shall also set bit 0 in DATA\_FORMAT of the <ctrl> byte in each transmitted <packet> to 0 then enter C12.18 compatibility mode, if supported. If the C12.18 compatibility mode is not supported by the Local Port, the C12.22 Node having a Local Port shall either "not acknowledge" any incoming <packet> having DATA\_FORMAT bit 0 set to 0 or respond with a <nak>. If the Local Port of the C12.22 Node does support C12.18 communication, it shall communicate according to ANSI C12.18 protocol definition thereafter. Note that C12.18 specifies half-duplex operation, so communicating devices should accommodate this fact.

The actual protocol implementation type may be confirmed by the reader by inspecting the <std> field returned being zero in the <ident-r> service response from the optical port.

### **7.2.2 Establishment of ANSI C12.22 Protocol Compatibility Mode**

ANSI C12.22 provides optional optical port compatibility with ANSI C12.22.

Any ANSI C12.22 compatible reading device, desiring to successfully connect to a Local Port, shall set bit 0 in DATA\_FORMAT of the <ctrl> byte to 1 in each transmitted <packet>. This control bit is found in the <packet> defined in C12.22 Communication Module section 6.9.2 "Packet Definition".

Consequently, the C12.22 Node having a Local Port shall also set bit 0 in DATA\_FORMAT of the <ctrl> byte in each transmitted <packet> to 1 and then enter C12.22 compatibility mode. The C12.22 Nodes shall communicate according to ANSI C12.22 protocol definition thereafter, as described in this Standard. Note that such devices may be half-duplex, so communicating devices should accommodate this possibility.

The actual protocol implementation type may be confirmed by the reader by inspecting the <std> field returned being three in the <ident-r> service response from the optical port.

## 8 Backward Compatibility

Backward compatibility of ANSI C12.22 with ANSI C12.18 and ANSI C12.21 is in relation to Layer 7 services, service parameters, Datagram data format, and transmission order.

ANSI C12.22 is backward compatible with ANSI C12.18 and ANSI C12.21 at the Application Layer. The ANSI C12.22 Application Layer Datagram is formatted in a way that enables ANSI C12.18 or ANSI C12.21 compliant devices and gateways to recognize the ANSI C12.22 Application Layer Datagrams (and services) so that they can take one of the following actions:

- reject the datagram; or
- strip the ANSI C12.22 additional information to yield a ANSI C12.18 or ANSI C12.21 compliant datagram in its original form; or
- strip the ANSI C12.22 additional information to yield an ANSI C12.18 or ANSI C12.21 equivalent Datagram for the purpose of interpretation or translation into other protocol.

## 9 Compliance

A C12.22 Network is considered to be in compliance with this Standard if the following conditions are met:

- 1.) It contains one or more C12.22 Network Segments.
- 2.) It contains at least one C12.22 Master Relay.
- 3.) It contains one or more C12.22 Nodes.

A C12.22 Node is considered to be in compliance with this Standard if the following conditions are met:

- 1.) It contains at least one C12.22 Application.
- 2.) The C12.22 Application supports at least one C12.22 Service (see Section 5.3.2.4, "Services").  
When a C12.22 Application does not support a C12.22 Service and it is a two-way device, it shall respond with an appropriate error code.
- 3.) The C12.22 Node has a unique ApTitle.

It is important to note that it is not necessary that every feature in the Standard be implemented at every level in a network (see Figure 5.1, C12.22 Reference Network Model).



## **Annex A - Relays**

**(normative)**

### **Description**

C12.22 Relays provide two basic services: address resolution and message forwarding.

C12.22 Message forwarding is an optional service and implemented only if the lower layers do not already provide routing (heterogeneous network). C12.22 Message forwarding is considered less desirable than using lower-layer routing since it allows only transmission of C12.22 Messages. This service is accessed by simply sending an <acse-pdu> to a C12.22 Relay with a called ApTitle different from the C12.22 Relay ApTitle. The C12.22 Relay will automatically forward this C12.22 Message to the next C12.22 Relay on this route or its final destination.

Address resolution is used to map C12.22 ApTitles to the corresponding native address. This method consists of using a Resolve Service Request with the target ApTitle as a parameter. The Resolve Service Response contains the native address of the requested C12.22 Node if this C12.22 Node resides on the same C12.22 Network Segment. Future communication with this C12.22 Node can be accomplished by addressing this C12.22 Node with its native address.

Note: The term "Routing table" is used loosely in this section to represent a general concept. In Annex C - "Modifications And Extensions to C12.19-1997", this concept is implemented by two tables, the Registration List Table (Table 132) and the Static Routing Table (Table 133).

### **A.1 Hierarchical Topology**

C12.22 Relay services are used between heterogeneous C12.22 Network Segments for forwarding C12.22 traffic. Source and destination addresses of each C12.22 Message are provided respectively by the Calling ApTitle and Called ApTitle as defined in the ACSE Application Data Unit <acse-pdu>. For each C12.22 Message received, the C12.22 Relay extracts the Called ApTitle and compares it to ApTitles contained in its routing tables. If a match is found, the C12.22 Relay forwards this message to the next C12.22 Relay or the final C12.22 Node associated with the Called ApTitle.

Routing tables update dynamically upon a successful completion of the Registration Service. The Registration Service is initiated by C12.22 Nodes at installation time and is to be repeated periodically within a time interval that is shorter than that published during the registration process using <reg-time-out>. Routing tables can also contain static routes that are configured manually. C12.22 Relays are organized in a hierarchical structure where a C12.22 Master Relay contains routing information for all accessible devices in the hierarchy and a list of C12.22 Notification Hosts. Other C12.22 Relays maintain only routing information of C12.22 Nodes under them.

Is it important to note that C12.22 Relays are also C12.22 Nodes on the C12.22 Network; as such, they must be registered.

### **A.2 C12.22 Master Relays**

A C12.22 Master Relay contains in its routing table all C12.22 Nodes registered to a C12.22 Network service provider. C12.22 Master Relays have some specific responsibilities compared to normal C12.22 Relays:

- C12.22 Master Relays are responsible for sending registration notification to C12.22 Notification and Authentication Hosts that are listed in their host tables (see "Annex A.3 Registration Notification").

- C12.22 Master Relays are the target of all Registration Service requests.
- C12.22 Master Relays shall not use default route entries in their routing tables.

### **A.3 Registration Notification**

C12.22 Master Relay shall maintain a list of C12.22 Host ApTitles that need to receive registration notifications. C12.22 Master Relays shall forward a copy of each first registration service ("first" here meaning an actual registration request as contrasted with the reuse of the register service as keep-alive) received by the C12.22 Hosts listed in its hosts notification tables. A notification is a Registration Service Request <register> with the Calling ApTitle set to the ApTitle of the C12.22 Master Relay, the Called ApTitle set to the ApTitle of the host and the ACSE Application Data Unit <acse-pdu> containing the node associated with this notification.

### **A.4 Registration Algorithm Details**

The following steps summarize the algorithm used for routing a Registration Service request.

1. C12.22 Node with UID Px.Ny attaches to a C12.22 Network Segment.
2. C12.22 Node Px.Ny sends a Registration Service Request <register> to one or more C12.22 Relays on the C12.22 Network Segment.
3. All C12.22 Relays receiving the Registration Service request forward it to higher C12.22 Relay(s) in the hierarchy until a C12.22 Master Relay is reached or return an error response as appropriate.
4. The C12.22 Master Relay deletes any duplicate Registration Service requests it received.
5. The C12.22 Master Relay forwards a copy of the Registration Service request to each C12.22 Host that needs to be notified, in accordance with its Host registration notification table. A C12.22 Master Relay can also act as a C12.22 Authentication Host, so its ApTitle may appear in the hosts' authentication and registration notification table.
6. The C12.22 Master Relay monitors the responses from all of the C12.22 Hosts that were notified for a minimum of one <register-r> with an <ok> response code from a C12.22 Authentication Host.
7. If one or more C12.22 Authentication Hosts respond with an <ok>, the C12.22 Master Relay sends an <ok> Registration Service response to the registering C12.22 Node by one of the possible (preferably the shortest) routes. Otherwise it replies with a <nok> and does not add the registering C12.22 Node to the C12.22 Relays' routing tables.
8. If a C12.22 Notification Host that is known to the C12.22 Master Relay does not respond, the C12.22 Master Relay shall cache the notification request for the non-responding C12.22 Host and it shall retry to notify that C12.22 Host periodically until the C12.22 Notification Host acknowledges (with an <ok> or a <nok>) the registration notification request, or until the registered C12.22 Node gets de-registered, whichever occurs first.

### **A.5 C12.22 Node ApTitle Auto-assignment**

A C12.22 Node that needs to get an ApTitle assigned to it dynamically shall initiate a Registration Service request with the calling ApTitle absent. This Registration Service request includes all the required parameters, except that the registering ApTitle length is set to zero. Also, the native address and device serial number, normally optional, shall be provided for validation.

The C12.22 Relay that receives this C12.22 Message has two options:

- In the first scheme, the C12.22 Relay directly assigns one of its subbranches to this C12.22 Node. The C12.22 Relay modifies the Registration Service request to add the assigned ApTitle as calling ApTitle and registered ApTitle and forwards the resulting C12.22 Message to the C12.22 Master Relay. The C12.22 Master Relay processes this C12.22 Message like any other Registration Service request it receives.

- The second scheme consists of forwarding the Registration Service request based on the rules described in section 5.3.4.12 "Use of Subbranches of a Registered ApTitle", Proxy server subsection. The C12.22 Master Relay then multi-casts this information to all C12.22 Authentication Hosts in its domain. If authenticated by a C12.22 Authentication Host, the C12.22 Master Relay assigns an ApTitle that is included in the Registration Service response. This response is then sent back to the registering C12.22 Node.

In either case, the registering C12.22 Node retrieves its new ApTitle from the Registration Service response.

## **A.6 C12.22 Master Relay ApTitle Auto-assignment**

A C12.22 Node that wants to get an ApTitle assigned to it dynamically shall initiate a Registration Service request with the called ApTitle absent. A C12.22 Relay that receives this request has two options:

- It can reject it by sending back an <sns> error message.
- The C12.22 Relay modifies the Registration Service request to add the called ApTitle. The ApTitle used can be a static ApTitle in this C12.22 Relay or based on the C12.22 Node serial number or native address received.

The auto-discovery of nearest C12.22 Relays (presented in section 5.3.2.4.12 "Resolve Service") together with the C12.22 Node ApTitle Auto-assignment and the C12.22 Master Relay ApTitle Auto-assignment can be used to automate the installation of a new C12.22 Node.

## **A.7 Obsolete Routes**

A C12.22 Node may request removal from the C12.22 Network by sending a Deregistration Service request. This service request is sent to the C12.22 Master Relay that originally received the Registration Service. The C12.22 Relay shall remove all routing information for the requested C12.22 Node only when an <ok> response is return by the C12.22 Master Relay.

C12.22 Relays shall also remove routing information of C12.22 Nodes that did not communicate or did not register within the Registration Time-out period (<registration-period>), following notification of the C12.22 Master Relay by the C12.22 Relay.

Following the removal of a C12.22 Node, a C12.22 Master Relay shall notify all C12.22 Hosts associated with this C12.22 Node about this removal.

## **A.8 Multiple Routes**

In some network topologies, messages can propagate through many paths (multiple C12.22 Relays). A C12.22 Node has the option to register to one or more of the available paths. For each path, the C12.22 Node shall register with a different ApTitle or a different subbranch of the same ApTitle. It is the responsibility of the C12.22 Node to maintain the registration of each path that it doesn't become obsolete. A C12.22 Client that wants to communicate with a C12.22 Device registered through multiple routes can select the route used for each transmission by using the corresponding ApTitle.

## **A.9 Application Layer Supervision**

It is assumed that C12.22 ACSE APDUs are transferred through reliable communication links. Any entity that provides Application Layer supervision shall generate a <nok> response to the Calling ApTitle upon discovery of network delivery failure. This assures that the C12.22 Application always gets a response.

## **A.10 Routing**

All routes to C12.22 Nodes are stored in the C12.22 Relay's routing tables. Each of the routing tables contains ApTitleMask / FwdAddress pairs. An ApTitleMask is represented as dot-delimited numeric or '\*' strings as shown below.

ApTitleMask	Description
<application-context-oid>.0.100.234	Represents the ApTitleMask for node <application-context-oid>.0.100.234
<application-context-oid>.0.100.*	Represents the mask for any node whose ApTitle starts with <application-context-oid>.0.100
*	Represents any node.

When searching for a forwarding address, the routing tables are parsed sequentially until a match is found or the end of the table is reached. When a match is found and the destination address is another C12.22 Relay, then the ACSE APDU is forwarded to that C12.22 Relay. If the destination is a C12.22 Node, then the ACSE APDU is delivered to that C12.22 Node directly on the local C12.22 Network Segment.

The following steps summarize the algorithm used for routing any C12.22 Messages:

1. Arbitrary C12.22 Node on the network wants to send a C12.22 Message to C12.22 Node Px.Ny
2. A C12.22 Message is sent to one of the local C12.22 Relays with Px.Ny as Called ApTitle, with interface Ax to C12.22 Node Px.Ny.
3. Local C12.22 Relay searches for Px.Ny in its routing table and, if found, it forwards the C12.22 Message to it via interface Ay.
4. If no route is available to forward or deliver this message, a <nok> is returned.

## Annex B - Routing Examples

(informative)

### B.1 C12.22 Relays With a Single Service Provider

The diagram below shows an example of C12.22 Relays and the routing tables associated to each of them. In this diagram, each routing table contains the following information:

- the node type
- the ApTitle of this node represented as “Pn.Nn” where Pn is the Service Provider code and Nn is the Subscriber code assigned to this node. For example, ApTitle <application-context-oid>.0.55.234 represent Service Provider <application-context-oid>.0.55, and Subscriber code .234
- the native address of the next node where to forward messages for this ApTitle

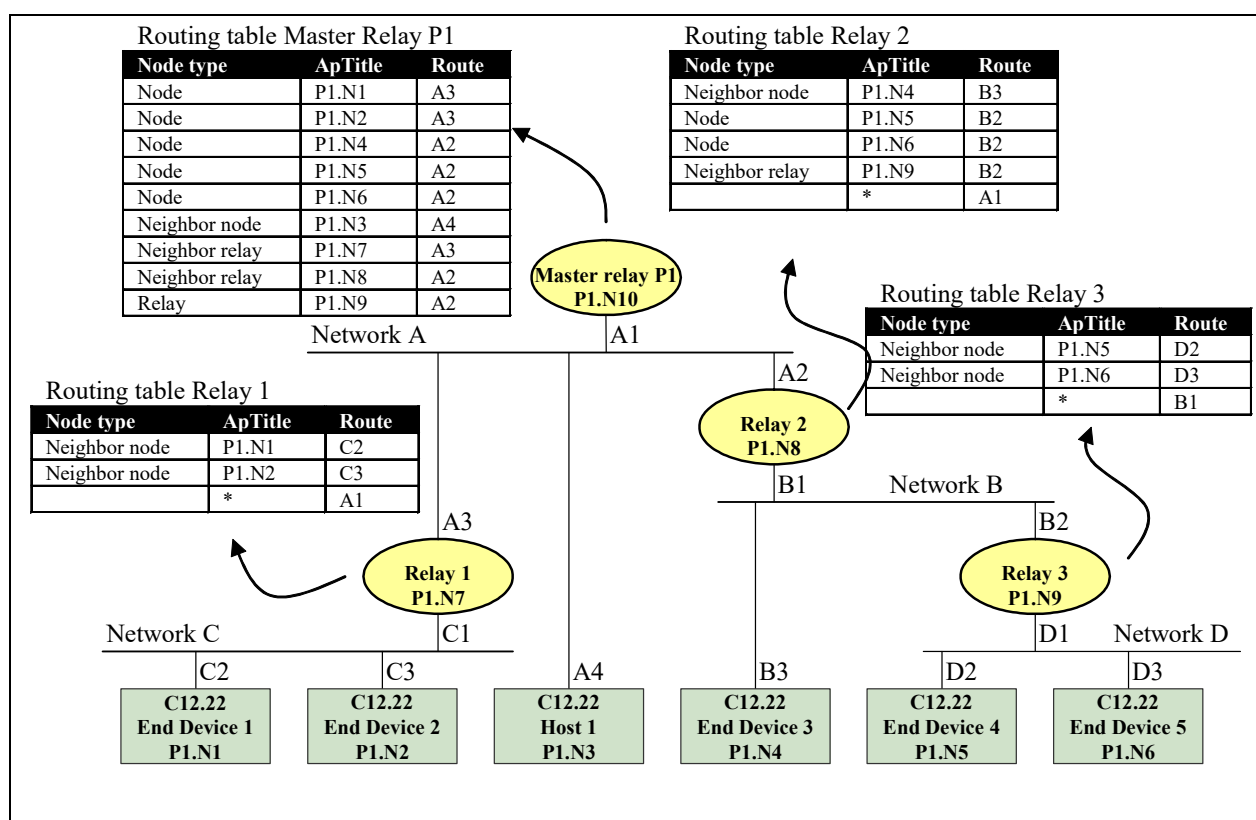


Figure B.1: C12.22 Relays and Routing Tables

### B.2 C12.22 Relays Shared by Multiple Service Providers

Each provider of C12.22 services maintains its own hierarchy of C12.22 Relays. To allow sharing of C12.22 Relays between service providers, ApTitles are broken into two parts. The first part identifies the service provider responsible for this C12.22 Device and the second part identifies a subscriber code that makes the entire ApTitle unique. As shown in Figure B.2, C12.22 Relays can take advantage of the

ApTitle structure to route C12.22 Messages.

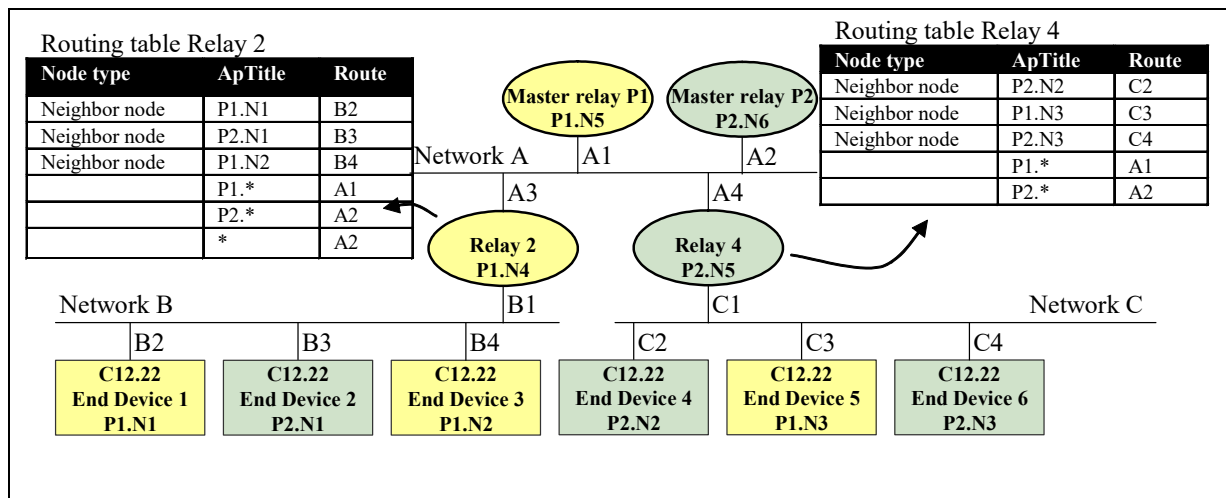


Figure B.2: C12.22 Relays and Routing Tables for Multiple Service Providers

## **Annex C - Modifications And Extensions to C12.19-1997**

**(normative)**

This Annex describes:

- C.1- New Decade 12, Network Control Tables, which are not currently in ANSI C12.19-1997.
- C.2- New Decade 13, Relay Control Tables, which are not currently in ANSI C12.19-1997.
- C.3- Universal ID pattern description of ApTitles
- C.4- Modified Decade 0, General Configuration Tables - addition of network registration and interface control procedures to Table 07, Procedure Initiate Table.
- C.5- Modified Decade 4, Security Tables - addition of Table 46, Key Table, which is not currently in ANSI C12.19-1997.

## C.1 Decade 12: Node Network Control Tables

This decade contains tables associated with C12.22 Node access to C12.22 Networks.

**TABLE 120 Dimension Network Table**

### Table 120 Data Description

**DIM\_NETWORK\_TBL** (Table 120) specifies the maximum dimensional values for this decade.

**Global Default Table Property Overrides:** Role="LIMITING", Accessibility="READONLY"

**TYPE DIM\_NETWORK\_BFLD = BIT FIELD OF UINT8**

```

TIME_STAMP_ENABLE_FLAG      : BOOL(0);
PROG_NATIVE_ADDRESS         : BOOL(1);
PROG_BROADCAST_ADDRESS      : BOOL(2);
STATIC_RELAY                 : BOOL(3);
STATIC_APTITLE               : BOOL(4);
STATIC_MASTER_RELAY          : BOOL(5);
CLIENT_RESPONSE_CTRL         : BOOL(6);
COMM_MODULE_SUPP_FLAG        : BOOL(7)
END;
```

**TYPE DIM\_MESSAGE\_BFLD = BIT FIELD OF UINT8**

```

MESSAGE_ACCEPTANCE_WINDOW_FLAG : BOOL(0);
ACCEPTANCE_RECOVERY_SESSION_FLAG : BOOL(1);
END;
```

**TYPE DIM\_NETWORK\_RCD = PACKED RECORD**

```

CONTROL                      : DIM_NETWORK_BFLD;

NBR_INTERFACES                : UINT8;
NBR_REGISTRATIONS             : UINT8;
NBR_FILTERING_RULES           : UINT16;
NBR_EXCEPTION_HOSTS           : UINT16;
NBR_EXCEPTION_EVENTS          : UINT16;
NBR_STATISTICS                : UINT16;
NBR_MULTICAST_ADDRESSES       : UINT8;

NATIVE_ADDRESS_LEN            : UINT8;
FILTERING_EXP_LEN              : UINT8;
MESSAGE_CONTROL                : DIM_MESSAGE_BFLD;
NBR_PLAYBACK_REJECT_MSGS      : UINT32;
END;
```

**TABLE 120 DIM\_NETWORK\_TBL = DIM\_NETWORK\_RCD;**

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>DIM_NETWORK_BFLD</b>		
<b>TIME_STAMP_ENABLE_FLAG</b>	FALSE	Network statistics table (Table 125) does not contain time-related information.



	TRUE	Network statistics table (Table 125) contains time-related information.
<b>PROG_NATIVE_ADDRESS</b>	FALSE	Interface native addresses cannot be set in the Interface control table (Table 122).
	TRUE	Interface native addresses can be set in the Interface control table (Table 122).
<b>PROG_BROADCAST_ADDRESS</b>	FALSE	Interface broadcast addresses cannot be set in the Interface control table (Table 122).
	TRUE	Interface broadcast addresses can be set in the Interface control table (Table 122).
<b>STATIC_RELAY</b>	FALSE	Local relay native static address cannot be set in the Interface control table (Table 122).
	TRUE	Local relay native address can be set in the Interface control table (Table 122).
<b>STATIC_APTITLE</b>	FALSE	This C12.22 Node cannot be programmed with a static ApTitle.
	TRUE	This C12.22 Node can be programmed with a static ApTitle.
<b>STATIC_MASTER_RELAY</b>	FALSE	The association with a C12.22 Master Relay cannot be programmed with a static ApTitle in the Interface control table (Table 122).
	TRUE	The association with a C12.22 Master Relay can be programmed with a static ApTitle in the Interface control table (Table 122).
<b>CLIENT_RESPONSE_CTRL</b>	FALSE	The Interface control table (Table 122) is not capable of providing client response control parameters.
	TRUE	The Interface control table (Table 122) is capable of providing client response control parameters.
<b>COMM_MODULE_SUPP_FLAG</b>		States the capability of this C12.22 Node to act as a C12.22 Device and be attached to a C12.22 Communication Module.
	FALSE	This C12.22 Node does not have the capability to assume the role of a C12.22 Device and it cannot be attached to a C12.22 Communication Module.
	TRUE	This C12.22 Node has the capability to assume the role of a C12.22 Device and it can be attached to a C12.22 Communication Module.
<b>DIM_MESSAGE_BFLD</b>		
<b>MESSAGE_ACCEPTANCE_WINDOW_FLAG</b>		This flag indicates whether the C12.22 Node has the capability to reject incoming C12.22 Messages strictly based on the timestamp of the message.
	FALSE	The C12.22 Node's time-based C12.22 Message acceptance window is not implemented.

	TRUE	The C12.22 Node's time-based C12.22 Message acceptance window is implemented and may be enabled.
<b>ACCEPTANCE_RECOVERY_SESSION_FLAG</b>		This flag indicates whether the C12.22 Node has the capability to ignore the incoming C12.22 Message acceptance window and playback rejection when processing incoming EPSEM <logon> service request.
	FALSE	The C12.22 Node's does not have the capability to override and ignore that acceptance window and playback rejection when processing an EPSEM <logon> service request.
	TRUE	The C12.22 Node's has the capability to override and ignore that acceptance window and playback rejection when processing an EPSEM <logon> service request.
<b>DIM_NETWORK_RCD</b>		
<b>NBR_INTERFACES</b>	0..255	Maximum number of network interfaces supported by this implementation.
<b>NBR_REGISTRATIONS</b>	0..255	Maximum number of concurrent registrations supported by each node. Each registration provides an independent route to a C12.22 Network.
<b>NBR_FILTERING_RULES</b>	0..65535	Maximum number of filtering rules supported in the Filtering rules table (Table 124).
<b>NBR_EXCEPTION_HOSTS</b>	0..65535	Maximum number of exception hosts supported in the Exception Report Configuration Table (Table 123).
<b>NBR_EXCEPTION_EVENTS</b>	0..65535	Maximum number of events supported for each entry in the Exception Report Configuration Table (Table 123).
<b>NBR_STATISTICS</b>	0..65535	Maximum number of statistics supported by the Network statistics table (Table 128).
<b>NBR_MULTICAST_ADDRESSES</b>	0..255	Maximum number multicast addresses supported in the Interface Control Table (Table 122).
<b>NATIVE_ADDRESS_LEN</b>	0..255	Maximum length of a native address supported by this implementation.
<b>FILTERING_EXP_LEN</b>	0..255	Maximum number of characters used as parameter for each filtering condition in the Filtering rules table (Table 124).

**MESSAGE\_CONTROL**

This Element contains attribute for C12.22 Message generation and acceptance. See **DIM\_MESSAGE\_BFLD**.

**NBR\_PLAYBACK\_REJECT\_MSGS**

0..4294967295 Maximum number of the messages maintained to implement playback rejection. Value greater than zero indicates that the C12.22 Node supports playback rejection. Playback rejection is based on the uniqueness of the <called-AP-title-element>, <calling-AP-title-element>, <calling-AP-invocation-id-element> and <iv> of previous messages.

**TABLE 121 Actual Network Table****Table 121 Data Description**

**ACT\_NETWORK\_TBL** (Table 121) specifies the actual dimensional values for this decade.

**Global Default Table Property Overrides:** Role="Actual"

**TABLE 121 ACT\_NETWORK\_TBL = DIM\_NETWORK\_RCD;**

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>DIM_NETWORK_BFLD</b>		<b>Redefines:</b> <b>DIM_NETWORK_TBL.DIM_NETWORK_BFLD.</b>
<b>TIME_STAMP_ENABLE_FLAG</b>	FALSE	Network statistics table (Table 125) does not contain time-related information.
	TRUE	Network statistics table (Table 125) contains time-related information.
<b>PROG_NATIVE_ADDRESS</b>	FALSE	Interface native addresses are not available in the Interface control table (Table 122).
	TRUE	Interface native addresses are settable in the Interface control table (Table 122).
<b>PROG_BROADCAST_ADDRESS</b>	FALSE	Interface broadcast addresses are not available in the Interface control table (Table 122).
	TRUE	Interface broadcast addresses are settable in the Interface control table (Table 122).
<b>STATIC_RELAY</b>	FALSE	Local relay native static address is not available in the Interface control table (Table 122).
	TRUE	Local relay static native address is settable in the Interface control table (Table 122).
<b>STATIC_APTITLE</b>	FALSE	This C12.22 Node is not programmed with a static ApTitle.
	TRUE	This C12.22 Node is programmed with a static ApTitle.
<b>STATIC_MASTER_RELAY</b>	FALSE	The association with a C12.22 Master Relay is not programmed with a static ApTitle in the Interface control table (Table 122).
	TRUE	The associated master relay is programmed with a static ApTitle in the Interface control table (Table 122).
<b>CLIENT_RESPONSE_CTRL</b>	FALSE	The Interface control table (Table 122) does not provide client response control parameters.
	TRUE	The Interface control table (Table 122) provides client response control parameters.

<b>COMM_MODULE_SUPP_FLAG</b>		States whether of this C12.22 Node is a C12.22 Device and may be attached to a C12.22 Communication Module.
	FALSE	This C12.22 Node is not enabled to assume the role of a C12.22 Device.
	TRUE	This C12.22 Node is enabled to assume the role of a C12.22 Device.
<b>DIM_MESSAGE_BFLD</b>		<b>Redefines:</b> <b>DIM_NETWORK_TBL.</b>
<b>MESSAGE_ACCEPTANCE_WINDOW_FLAG</b>		<b>DIM_MESSAGE_BFLD.</b>
		This flag indicates whether the C12.22 Node rejects incoming C12.22 Messages strictly based on the timestamp of the message.
	FALSE	The C12.22 Node's time-based C12.22 Message acceptance window not allowed.
	TRUE	The C12.22 Node's time-based C12.22 Message acceptance window is allowed and can be enabled.
<b>ACCEPTANCE_RECOVERY_SESSION_FLAG</b>		This flag indicates whether the C12.22 Node is programmed to ignore the incoming C12.22 Message acceptance window and playback rejection when processing incoming EPSEM <logon> service request.
	FALSE	The C12.22 Node's shall not override and shall not ignore the acceptance window and playback rejection settings when processing an EPSEM <logon> service request.
	TRUE	The C12.22 Node's shall override and ignore that acceptance window when processing an EPSEM <logon> service request.
<b>DIM_NETWORK_RCD</b>		
<b>NBR_INTERFACES</b>	0..255	Actual number of network interfaces supported by this implementation.
<b>NBR_REGISTRATIONS</b>	0..65535	Actual number of concurrent routes supported.
<b>NBR_FILTERING_RULES</b>	0..65535	Actual number of filtering rules available in the Filtering rules table (Table 124).
<b>NBR_EXCEPTION_HOSTS</b>	0..65535	Actual number of exception hosts available in the Exception Report Configuration Table (Table 123).
<b>NBR_EXCEPTION_EVENTS</b>	0..65535	Actual number of events supported for each entry in the Exception Report Configuration Table (Table 123).
<b>NBR_STATISTICS</b>	0..65535	Actual number of statistics supported by the Network statistics table (Table 128).
<b>NBR_MULTICAST_ADDRESSES</b>	0..255	Actual number of multicast addresses supported in the Interface control table (Table 122).

<b>NATIVE_ADDRESS_LEN</b>	0..255	Actual length of a native address supported.
<b>FILTERING_EXP_LEN</b>	0..255	Actual number of characters in a filtering expression used in the Filtering rules table (Table 124).
<b>MESSAGE_CONTROL</b>		This Element contains the programmable attribute for C12.22 Message generation and acceptance.
<b>NBR_PLAYBACK_REJECT_MSGS</b>	0..4294967295	Actual number of messages that can be monitored to implement playback rejection.

**TABLE 122 Interface Control Table****Table 122 Data Description**

**INTERFACE\_CTRL\_TBL** (Table 122) contains the configuration of each interface supported by this implementation.

**Global Default Table Property Overrides:** Role="CONTROL"

```

TYPE INTERFACE_CTRL_ENTRY_RCD = PACKED RECORD
  IF ACT_NETWORK_TBL.PROG_NATIVE_ADDRESS THEN
    NATIVE_ADDRESS          : BINARY(ACT_NETWORK_TBL.NATIVE_ADDRESS_LEN);
  END;
  IF ACT_NETWORK_TBL.PROG_BROADCAST_ADDRESS THEN
    BROADCAST_ADDRESS       : BINARY(ACT_NETWORK_TBL.NATIVE_ADDRESS_LEN);
  END;
  IF ACT_NETWORK_TBL.STATIC_RELAY THEN
    RELAY_NATIVE_ADDRESS     : BINARY(ACT_NETWORK_TBL.NATIVE_ADDRESS_LEN);
  END;
  IF ACT_NETWORK_TBL.STATIC_APTITLE THEN
    NODE_APTITLE            : BINARY(20);
  END;
  IF ACT_NETWORK_TBL.STATIC_MASTER_RELAY THEN
    MASTER_RELAY_APTITLE     : BINARY(20);
  END;
  IF ACT_NETWORK_TBL.CLIENT_RESPONSE_CTRL THEN
    NBR_OF_RETRIES          : UINT8;
    RESPONSE_TIMEOUT        : UINT16;
  END;
  IF ACT_NETWORK_TBL.COMM_MODULE_SUPP_FLAG THEN
    COMM_MODULE_FEATURE_CTRL: INTERFACE_STATUS_TBL.INTERFACE_STATE_BFLD;
  END;
  IF ACT_NETWORK_TBL.MESSAGE_ACCEPTANCE_WINDOW_FLAG THEN
    ACCEPTANCE_WINDOW       : UINT32;
  END;
  INTERFACE_STATE           : INTERFACE_STATUS_TBL.INTERFACE_STATE_BFLD;
  INTERFACE_CONNECTION      : INTERFACE_STATUS_TBL.CONNECTION_TYPE_BFLD;
END;

TYPE INTERFACE_CTRL_RCD = PACKED RECORD
  ELECTRONIC_SERIAL_NUMBER: BINARY(20);
  NODE_TYPE                : INTERFACE_STATUS_TBL.NODE_TYPE_BFLD;
  INTERFACE_ENTRIES        : ARRAY[ACT_NETWORK_TBL.NBR_INTERFACES]
    OF INTERFACE_CTRL_ENTRY_RCD;
  MCAST_ADDRESSES         : ARRAY[ACT_NETWORK_TBL.NBR_MULTICAST_ADDRESSES]
    OF BINARY(20);
END;

TABLE 122 INTERFACE_CTRL_TBL = INTERFACE_CTRL_RCD;

```

IdentifierValueDefinition

**INTERFACE\_CTRL\_ENTRY\_RCD  
NATIVE\_ADDRESS**

Native address assigned to this interface. This address can be pre-configured or dynamically received from the C12.22 Communication Module. See section 6.7.2 "Get Configuration Service".

**BROADCAST\_ADDRESS**

Native address used to broadcast messages on the local C12.22 Network Segment. This address can be pre-configured or dynamically received from the C12.22 Communication Module. See section 6.7.2 "Get Configuration Service".

**RELAY\_NATIVE\_ADDRESS**

Native address used to access the relay on this C12.22 Network Segment. When not provided, this field can be pre-configured or assigned automatically by the EPSEM <resolve> service.

**NODE\_APTITLE**

Relative or absolute object identifier assigned to this node. This value can be pre-configured or dynamically assigned to this node. This field is encoded as <universal-id-element> or <relative-uid-element>.

**MASTER\_RELAY\_APTITLE**

Relative or absolute object identifier assigned to the C12.22 Master Relay responsible for this node. This value can be pre-configured or dynamically assigned to this node. This field is encoded as <universal-id-element> or <relative-uid-element>.

**NBR\_OF\_RETRIES**

The number of retries defines the number of times an EPSEM request is repeated if the EPSEM response is not received within a RESPONSE\_TIMEOUT interval.

**RESPONSE\_TIMEOUT**

This parameter controls the number of seconds a C12.22 Client has to wait for an EPSEM response to an EPSEM request before failing or sending retries.

**COMM\_MODULE\_FEATURE\_CTRL**

This is a C12.22 Network management control parameter that defines the features that shall be enabled and delegated to the attached C12.22 Communication Module on this interface. When this Element is present then it controls the information passed by the C12.22 Device to the C12.22 Communication Module. The actions that are performed by the C12.22 Communication Module shall be reported in **INTERFACE\_STATUS\_TBL.INTERFACE\_STATE**.

**MESSAGE\_ACCEPTANCE\_WINDOW**

This Element defines the incoming C12.22 Message acceptance window in minutes.



- 0 The incoming C12.22 Message acceptance window is automatic property of the C12.22 Node.
- 1..4294967295 The C12.22 Message incoming acceptance window in minutes. Any sessionless C12.22 Message that is received whose <iv> element is not older than (MESSAGE\_ACCEPTANCE\_WINDOW  $\pm 15$ ) minutes relative to the C12.22 Node's internal clock shall be considered to have arrived within the C12.22 Node's Message acceptance window. All other C12.22 Messages may be rejected as too old.

**INTERFACE\_STATE**

The C12.22 Node interface capabilities attributes used to present this node through this interface. See **INTERFACE\_STATUS\_TBL**. **INTERFACE\_STATE\_BFLD**.

**INTERFACE\_CONNECTION**

The C12.22 Node network connection attributes used to register this node through this interface. See **INTERFACE\_STATUS\_TBL**. **CONNECTION\_TYPE\_BFLD**.

**INTERFACE\_CTRL\_RCD**  
**ELECTRONIC\_SERIAL\_NUMBER**

Universal identifier used by the C12.12 Node and the C12.22 Master Relay in algorithms for the auto-assignment of the C12.22 Node's Aptitle.

**NODE\_TYPE**

The C12.22 Node type attributes used to register this node through this interface. See **INTERFACE\_STATUS\_TBL**. **NODE\_TYPE\_BFLD**.

**INTERFACE\_ENTRIES**

Array containing the configuration information for each interface supported by this implementation.

**MCAST\_ADDRESSES**

Array of relative object identifiers. This field is encoded as a <relative-uid-element> supported by this node. The value of 0 is reserved for unused.

### TABLE 123 Exception Report Configuration Table

### Table 123 Data Description

**EXCEPTION\_REPORT\_CONFIG\_TBL** (Table 123) is used to configure exception messages that can be sent by this C12.22 Node to C12.22 Hosts as a result of a triggering event or through the invocation of Procedure 26 Exception Report.

### Global Default Table Property Overrides: role="CONTROL"

### Table 123 Type Definitions

```

TYPE EXCEPTION_REPORT_ENTRY_RCD = PACKED RECORD
    APTITLE_NOTIFY           : BINARY(20);
    MAX_NUMBER_OF_RETRIES   : UINT8;
    RETRY_DELAY              : UINT16;
    EXCLUSION_PERIOD        : UINT16;
    HOST_SECURITY            : UINT16;
    EVENT_REPORTED           : ARRAY[ACT_NETWORK_TBL.NBR_EXCEPTION_EVENTS]
                                OF TABLE IDB BFLD;

```

**END;**

```
TYPE EXCEPTION_REPORT_RCD = PACKED RECORD
    EXCEPTION_REPORT      : ARRAY[ACT_NETWORK_TBL.NBR_EXCEPTION_HOSTS]
                           OF EXCEPTION_REPORT_ENTRY_RCD;
```

**END;**

**TABLE 123 EXCEPTION REPORT CONFIG TBL = EXCEPTION REPORT RCD;**

### Table 123 Element Descriptions

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>TABLE_IDB_BFLD</b>		<b>Redefines: STD. TABLE_IDB_BFLD.</b>
<b>TBL_PROC_NBR</b>	0..2047	Event code that triggers this exception report. For standard event codes, refer to “History & Event Log Codes” as defined in ANSI C12.19.
<b>STD_VS_MFG_FLAG</b>	FALSE TRUE	Event number is defined in standard. Event number is defined by the manufacturer.
<b>SELECTOR</b>		Reserved.
<b>EXCEPTION_REPORT_ENTRY_RCD</b>		
<b>APTITLE_NOTIFY</b>		The C12.22 Host-called ApTitle for reporting exceptions that are generated for the associated events. This is also the C12.22 Host-calling root ApTitle for corresponding response messages. As part of a response, a <b>KEY</b> shall be considered a match if in addition the calling ApTitle matches <b>APTITLE_NOTIFY</b> or is a branch of <b>APTITLE_NOTIFY</b> . An

**APTITLE\_NOTIFY** that is all binary zeros implies that this exception report entry is not used.

<b>MAX_NUMBER_OF_RETRIES</b>		Maximum numbers of times the C12.22 Node shall attempt to report an exception. This process stops after the reception by the node of an <ok> response or a non fatal <nok> response from the C12.22 Host.
	0	Exception report is unconfirmed. The C12.22 Message shall be formulated in a manner that does not solicit a response from the C12.22 Host.
	1..255	Exception report is confirmed. The C12.22 Message shall be formulated in a manner that solicits a response from the C12.22 Host.
<b>RETRY_DELAY</b>	0..65535	Minimum interval in minutes between two consecutive retry attempts for the same event.
<b>EXCLUSION_PERIOD</b>	0..65535	Minimum interval in minutes before initiating a consecutive report for the same event.
<b>HOST_SECURITY</b>	0..65535	A index into the Element <b>HOST_SECURITY</b> of <b>HOST_ACCESS_SECURITY_TBL</b> (Table 47). This selection provides information about the C12.22 Host password, keys and authentication values that shall be encoded in the originating C12.22 Message.
<b>EVENT_REPORTED</b>		List of events reported to the associated ApTitle.
<b>EXCEPTION_REPORT_RCD</b> <b>EXCEPTION_REPORT</b>		Array containing a list of exception report control records to manage where to send exception reports for the associated events.

**TABLE 124 Filtering Rules Table****Table 124 Data Description**

FILTERING\_RULES\_TBL (Table 124) contains a collection of rules used to filter traffic across interfaces. A filtering rule may define actions taken based on information matched such as interface identifier, ApTitle, native address, electronic serial number and data flow. The filtering rules allow use-pattern matching as described in "Annex C.3 Universal ID Pattern Description of ApTitles".

**Global Default Table Property Overrides:** Role="CONTROL"

**TYPE FILTERING\_CONDITION\_BFLD = BIT FIELD OF UINT16**

```

ACTION_RULE      : UINT(0..3);
LABEL            : UINT(4..7);
DIRECTION        : UINT(8..11);
CONDITION        : UINT(12..15);
END;
```

**TYPE FILTERING\_RULES\_ENTRY\_RCD = PACKED RECORD**

```

CONDITION        : FILTERING_CONDITION_BFLD;
PATTERN          : STRING(ACT_NETWORK_TBL.FILTERING_EXP_LEN);
END;
```

**TYPE RULES\_PER\_INTERFACE\_RCD = PACKED RECORD**

```

FILTERING_RULES  : ARRAY[ACT_NETWORK_TBL.NBR_FILTERING_RULES]
                  OF FILTERING_RULES_ENTRY_RCD;
END;
```

**TYPE FILTERING\_RULES\_RCD = PACKED RECORD**

```

RULES_PER_INTERFACE : ARRAY[ACT_NETWORK_TBL.NBR_INTERFACES]
                  OF RULES_PER_INTERFACE_RCD;
END;
```

**TABLE 124 FILTERING\_RULES\_TBL = FILTERING\_RULES\_RCD;**

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>FILTERING_CONDITION_BFLD</b>		
<b>ACTION_RULE</b>		
	0	Defines the action taken after evaluating this rule.
	1	IGNORE, Take no action, continue with next rule.
	2	REJECT, Reject immediately if the associated condition matches.
	3	DENY, Mark as reject if the associated condition matches. This condition may be allowed by a following rule.
	4	ALLOW, Accept message immediately if the associated condition matches.
	5	GOTO, Continue processing starting at LABEL.
		COND, Conditionally continue processing at LABEL immediately if the associated condition matches.

<b>LABEL</b>		Label used by the options 4 and 5 of the ACTION_RULE field.
	0	No label
	1..15	LABEL 1 to LABEL 15
<b>DIRECTION</b>		Only messages corresponding to this selection are tested for the associated condition.
	0	Not applicable, don't care.
	1	Any <acse-pdu> received by this interface.
	2	Any <acse-pdu> transmitted from this interface.
	3	<acse-pdu> transmitted from this interface directly to a destination on the same C12.22 Network Segment (Not through a C12.22 Relay).
	4	<acse-pdu> transmitted from this interface to a destination using a C12.22 Relay.
	5..15	Reserved.
<b>CONDITION</b>	0	The condition is TRUE.
	1	Any <acse-pdu> with matching calling or called ApTitle pattern.
	2	Any <acse-pdu> with matching calling ApTitle pattern.
	3	Any <acse-pdu> with matching called ApTitle pattern.
	4	Any <acse-pdu> with matching source or destination native address pattern.
	5	Any <acse-pdu> with matching source native address pattern.
	6	Any <acse-pdu> with matching destination native address pattern.
	7	<registration> request with matching <electronic-serial-number> pattern.
	8..15	Reserved. Shall be set to 0 (zero).
<b>FILTERING_RULES_ENTRY_RCD CONDITION</b>		See FILTERING_CONDITION_BFLD above.
<b>PATTERN</b>		Matching pattern that is applied subject to the CONDITION field.
<b>RULES_PER_INTERFACE_RCD FILTERING_RULES</b>		See FILTERING_RULES_ENTRY_RCD above.
<b>FILTERING_RULES_RCD RULES_PER_INTERFACE</b>		See RULES_PER_INTERFACE_RCD above.

**TABLE 125 Interface Status Table****Table 125 Data Description**

**INTERFACE\_STATUS\_TBL** (Table 125) contains status and information about each interface supported by the implementation.

**Global Default Table Property Overrides:** Role="DATA", Accessibility="READONLY"

**TYPE INTERFACE\_STATE\_BFLD = BIT FIELD OF UINT8**

```

INTERFACE_ENABLE_FLAG      : BOOL(0);
AUTO_REGISTRATION_FLAG     : BOOL(1);
DIRECT_MESSAGING_FLAG      : BOOL(2);
FILLER                     : FILL(3..7);
END;
```

**TYPE NODE\_TYPE\_BFLD = BIT FIELD OF UINT8**

```

RELAY_FLAG                 : BOOL(0);
MASTER_RELAY_FLAG          : BOOL(1);
HOST_FLAG                  : BOOL(2);
NOTIFICATION_HOST_FLAG     : BOOL(3);
AUTHENTICATION_HOST_FLAG   : BOOL(4);
END_DEVICE_FLAG            : BOOL(5);
FILLER                     : FILL(6..7);
END;
```

**TYPE CONNECTION\_TYPE\_BFLD = BIT FIELD OF UINT8**

```

BROADCAST_AND_MULTICAST_FLAG : BOOL(0);
IF ACT_NETWORK_TBL.MESSAGE_ACCEPTANCE_WINDOW_FLAG THEN
  MESSAGE_ACCEPTANCE_WINDOW_FLAG : BOOL(1);
END;
PLAYBACK_REJECTION_FLAG      : BOOL(2);
FILLER_2                     : FILL(3);
CONNECTIONLESS_MODE_FLAG     : BOOL(4);
ACCEPT_CONNECTIONLESS_FLAG    : BOOL(5);
CONNECTION_MODE_FLAG         : BOOL(6);
ACCEPT_CONNECTIONS_FLAG      : BOOL(7);
END;
```

**TYPE INTERFACE\_STATUS\_ENTRY\_RCD = PACKED RECORD**

```

INTERFACE_NAME              : STRING(20);
INTERFACE_STATE              : INTERFACE_STATE_BFLD;
INTERFACE_CONNECTION        : CONNECTION_TYPE_BFLD;
IF ACT_NETWORK_TBL.TIME_STAMP_ENABLE_FLAG THEN
  INTERFACE_ON_TIME          : LTIME_DATE;
  INTERFACE_OFF_TIME         : LTIME_DATE;
  LAST_STAT_RESET_TIME       : LTIME_DATE;
  LAST_ACCESS_TIME           : LTIME_DATE;
END;
END;
```

**TYPE INTERFACE\_STATUS\_RCD = PACKED RECORD**

```

NODE_TYPE                   : NODE_TYPE_BFLD;
```

**INTERFACE\_STATUS** : ARRAY[ACT\_NETWORK\_TBL.NBR\_INTERFACES]  
OF INTERFACE\_STATUS\_ENTRY\_RCD;  
**END;**

**TABLE 125 INTERFACE\_STATUS\_TBL = INTERFACE\_STATUS\_RCD;**

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>INTERFACE_STATE_BFLD</b>		
<b>INTERFACE_ENABLE_FLAG</b>	FALSE	This interface is deactivated and does not manifest or present itself on the C12.22 Network Segment.
	TRUE	This interface is functional and it manifests and presents itself on the C12.22 Network Segment.
<b>AUTO_REGISTRATION_FLAG</b>	FALSE	The C12.22 Communication Module attached to this interface does not register this C12.22 Node on the C12.22 Network. For example, if the interface of a C12.22 Relay on the LAN side does not need to be registered then this field should be set to FALSE.
	TRUE	The C12.22 Communication Module attached to this interface registers this C12.22 Node on the C12.22 Network.
<b>DIRECT_MESSAGING_FLAG</b>	FALSE	All messages sent through this interface are sent via C12.22 Relays.
	TRUE	For all messages sent through this interface, the node always tries to resolve the native addresses. If the native address is available, all subsequence exchanges are done directly with the target C12.22 Node.
<b>NODE_TYPE_BFLD</b>		
<b>RELAY_FLAG</b>		Identification of the type of the C12.22 Node exposed through this interface.
	FALSE	An indication of whether or not this C12.22 Node is a C12.22 Relay.
	TRUE	The C12.22 Node is not a C12.22 Relay. If the MASTER_RELAY_FLAG is set to TRUE, the C12.22 Master Relay will not route C12.22 Messages to other C12.22 Nodes.
	TRUE	The C12.22 Node is a C12.22 Relay. When MASTER_RELAY_FLAG is also set to TRUE, the C12.22 Relay is also a C12.22 Master Relay
<b>MASTER_RELAY_FLAG</b>		
	FALSE	An indication of whether or not this C12.22 Node is a C12.22 Master Relay.
	TRUE	The C12.22 Node is not a C12.22 Master Relay.
	TRUE	The C12.22 Node is a C12.22 Master Relay.

<b>HOST_FLAG</b>		An indication of whether or not this C12.22 Node is a C12.22 Host. A Host is an un-embedded application process that runs on a computer. A non-host is a C12.22 embedded application.
	FALSE	The C12.22 Node is not a C12.22 Host.
	TRUE	The C12.22 Node is a C12.22 Host.
<b>NOTIFICATION_HOST_FLAG</b>		An indication of whether or not this C12.22 Node is a C12.22 Notification Host.
	FALSE	The C12.22 Node is not a C12.22 Notification Host.
	TRUE	The C12.22 Node is a C12.22 Notification Host.
<b>AUTHENTICATION_HOST_FLAG FLAG</b>		An indication of whether or not this C12.22 Node is a C12.22 Authentication Host.
	FALSE	The C12.22 Node is not a C12.22 Authentication Host.
	TRUE	The C12.22 Node is a C12.22 Authentication Host.
<b>END_DEVICE_FLAG</b>		An indication whether this C12.22 Node is a C12.22 Device, i.e. it has metrological sensors and C12.19 Tables.
	FALSE	The C12.22 Node does not have metrological sensors and C12.19 Tables.
	TRUE	The C12.22 Node has metrological sensors and C12.19 Tables.
<b>CONNECTION_TYPE_BFLD BROADCAST_AND_MULTICAST_FLAG</b>		The status whether this C12.22 Node accepts broadcast and multicast messages.
	FALSE	The C12.22 Node does not accept broadcast and multicast messages.
	TRUE	The C12.22 Node accepts broadcast and multicast messages.
<b>ACCEPTANCE_WINDOW_FLAG</b>		Status for whether or not this C12.22 Node enabled the time-based C12.22 Message acceptance window.
	FALSE	This C12.22 Node does not reject incoming C12.22 Messages based on the timestamp of the message.
	TRUE	This C12.22 Node rejects incoming C12.22 Messages based on the timestamp of the message.



<b>PLAYBACK_REJECTION_FLAG</b>	Status indicator for whether or not this C12.22 Node enabled the C12.22 Message playback rejection mechanism.
FALSE	This C12.22 Node does not reject played back messages.
TRUE	This C12.22 Node rejects played back messages.
<b>CONNECTIONLESS_MODE_FLAG</b>	Status indicator for whether or not this C12.22 Node enabled the connectionless-oriented protocol. Note that this flag and <b>CONNECTION_MODE_FLAG</b> may both be set.
FALSE	The node does not use a connection-less oriented protocol on its local network segment.
TRUE	The local network segment uses a connectionless oriented protocol on its local network segment.
<b>ACCEPT_CONNECTIONLESS_FLAG</b>	Status indicator for whether or not this C12.22 Node accepts unsolicited incoming connectionless messages. This Sub-element shall be set to FALSE when <b>CONNECTIONLESS_MODE_FLAG</b> is FALSE.
FALSE	The C12.22 Node does not accept unsolicited incoming connectionless messages.
TRUE	The C12.22 Node listens for and accepts unsolicited incoming connectionless messages.
<b>CONNECTION_MODE_FLAG</b>	Status indicator for whether or not this C12.22 Node enabled the connection-oriented protocol. Note that this flag and <b>CONNECTIONLESS_MODE_FLAG</b> may both be set
FALSE	The node does not use a connection oriented protocol on its local network segment.
TRUE	The local network segment uses a connection oriented protocol on its local network segment.
<b>ACCEPT_CONNECTIONS_FLAG</b>	Status indicator for whether or not this C12.22 Node accepts incoming connections. This Sub-element shall be set to FALSE when <b>CONNECTION_MODE_FLAG</b> is FALSE.
FALSE	The C12.22 Node does not accept incoming connection.
TRUE	The C12.22 Node listens for and accepts connection incoming connections.
<b>INTERFACE_STATUS_ENTRY_RCD INTERFACE_NAME</b>	This field contains a textual description of technology used by this interface. This

description can be pre-configured or dynamically received from the C12.22 Communication Module. See section 6.7.2 “Get Configuration Service”.

<b>INTERFACE_STATE</b>	The active state of this interface, see <b>INTERFACE_STATE_BFLD</b> defined above.
<b>INTERFACE_CONNECTION</b>	The active protocol connection modes and capabilities used by this interface. See <b>CONNECTION_TYPE_BFLD</b> .
<b>NODE_TYPE</b>	See <b>NODE_TYPE_BFLD</b> defined above.
<b>INTERFACE_ON_TIME</b>	The last time the interface was enabled.
<b>INTERFACE_OFF_TIME</b>	The last time the interface was disabled.
<b>LAST_STAT_RESET_TIME</b>	The last time the statistics values were reset.
<b>LAST_ACCESS_TIME</b>	The last time the interface received or transmitted any C12.22 Message.
<b>INTERFACE_STATUS_RCD</b> <b>NODE_TYPE</b>	See <b>NODE_TYPE_BFLD</b> defined above.
<b>INTERFACE_STATUS</b>	Array containing information for each interface supported by this implementation.

**TABLE 126 Registration Status Table****Table 126 Data Description**

**REGISTRATION\_STATUS\_TBL** (Table 126) contains the information and configuration parameters required to register and maintain registrations for one or more routes.

**Global Default Table Property Overrides:** Role="CONTROL"

**TYPE REGISTRATION\_ENTRY\_RCD = PACKED RECORD**

```

INTERFACE_ID          : UINT8;
RELAY_NATIVE_ADDRESS  : BINARY(ACT_NETWORK_TBL.NATIVE_ADDRESS_LEN);
MASTER_RELAY_APTITLE  : BINARY(20);
NODE_APTITLE          : BINARY(20);
REGISTRATION_DELAY    : UINT16;
REGISTRATION_PERIOD    : UINT24;
REGISTRATION_COUNT_DOWN : UINT24;
END;
```

**TYPE REGISTRATION\_STATUS\_RCD = PACKED RECORD**

```

REGISTRATION          : ARRAY[ACT_NETWORK_TBL.NBR_REGISTRATIONS]
                      OF REGISTRATION_ENTRY_RCD;
END;
```

**TABLE 126 REGISTRATION\_STATUS\_TBL = REGISTRATION\_STATUS\_RCD;**

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>REGISTRATION_ENTRY_RCD</b>		
<b>INTERFACE_ID</b>	0..255	The physical interface number used to attach this C12.22 Node through this Interface to a C12.22 Network Segment. This ID is an index in the INTERFACE_ENTRIES array of the Interface control table (Table 122).
<b>RELAY_NATIVE_ADDRESS</b>		Native address used to access the C12.22 Relay on this route for the C12.22 Node's local C12.22 Network Segment. This field can be pre-configured or assigned automatically by the EPSEM <resolve> service.
<b>MASTER_RELAY_APTITLE</b>		Relative or absolute object identifier assigned to the C12.22 Master Relay responsible for this C12.22 Node.
<b>NODE_APTITLE</b>		Relative or absolute object identifier assigned to this C12.22 Node encoded as <relative-uid-element> or <universal-id-element>, respectively.
<b>REGISTRATION_DELAY</b>	0..65535	Maximum random delay, in seconds, between each power up and the automatic issuance of the first Registration Service request by the C12.22 Node. This function is disabled when this field is set to 0 (zero).

<b>REGISTRATION_PERIOD</b>	0..16777215	The maximum duration, in seconds, before the C12.22 Node's registration expires. The C12.22 Node should reregister itself before this period lapses in order to remain registered. This value may be lowered through Registration Service parameters.
<b>REGISTRATION_COUNT_DOWN</b>	0..16777215	The amount of time in seconds left before the registration period expires.
<b>REGISTRATION_RCD REGISTRATION</b>		Array that contains the registration information for this C12.22 Node. This array contains more than one entry only if this C12.22 Node supports multiple routes. See "Annex A.8 Multiple Routes".

**TABLE 127 Network Statistics Selections Table****Table 127 Data Description**

**NETWORK\_STATISTICS\_CTRL\_TBL** (Table 127) contains selections of the statistics recorded for each interface supported by the implementation. The entries made in this Table select the statistics that will be reported in **NETWORK\_STATISTICS\_TBL** (Table 127).

**Global Default Table Property Overrides:** role="CONTROL"

**Table 127 Type Definitions**

```

TYPE NETWORK_STATISTICS_CTRL_ENTRY_RCD = PACKED RECORD
    INTERFACE_ID      : UINT8;
    STATISTIC_ID      : NETWORK_STATISTICS_TBL.TABLE_IDB_BFLD;
END;

TYPE NETWORK_STATISTICS_CTRL_RCD = PACKED RECORD
    STATISTICS        : ARRAY[ACT_NETWORK_TBL.NBR_STATISTICS]
                      OF NETWORK_STATISTICS_CTRL_ENTRY_RCD;
END;

TABLE 127 NETWORK_STATISTICS_CTRL_TBL = NETWORK_STATISTICS_CTRL_RCD;

```

**Table 127 Element Descriptions**

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>NETWORK_STATISTICS_CTRL_ENTRY_RCD</b>		This packed record contain a single control Element that identifies one statistic to be reported in <b>NETWORK_STATISTICS_TBL</b> (Table 128).
<b>INTERFACE_ID</b>	0..255	The physical interface number used to register this C12.22 Node. This ID is an index in the <b>INTERFACE_ENTRIES</b> array of the Interface control table (Table 122).
<b>STATISTIC_ID</b>		See <b>NETWORK_STATISTICS_TBL .TABLE_IDB_BFLD</b> .
<b>VALUE</b>		Statistic reported.
<b>NETWORK_STATISTICS_CTRL_RCD</b>		This packed record contains a collection of all node statistics to be reported. This collection spans all interfaces, devices and communication modules. The statistics reported in <b>NETWORK_STATISTICS_TBL</b> (Table 128) shall be ordered according to the entries defined in this Table (Table 127).
<b>STATISTICS</b>		Array containing a series of selectors to be reported in <b>NETWORK_STATISTICS_TBL</b>

(Table 128).

**TABLE 128 Network Statistics Table****Table 128 Data Description**

**NETWORK\_STATISTICS\_TBL** (Table 128) contains statistics for each interface supported by the implementation.

**Global Default Table Property Overrides:** role="DATA", accessibility="READONLY"

**Table 128 Type Definitions**

**TYPE NETWORK\_STATISTICS\_ENTRY\_RCD = PACKED RECORD**

INTERFACE\_ID : UINT8;  
 STATISTIC\_ID : TABLE\_IDB\_BFLD;  
 VALUE : INT48;

**END;**

**TYPE NETWORK\_STATISTICS\_RCD = PACKED RECORD**

STATISTICS : ARRAY[ACT\_NETWORK\_TBL.NBR\_STATISTICS]  
 OF NETWORK\_STATISTICS\_ENTRY\_RCD;

**END;**

**TABLE 128 NETWORK\_STATISTICS\_TBL = NETWORK\_STATISTICS\_RCD;**

**Table 128 Element Descriptions**

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>TABLE_IDB_BFLD</b> <b>TBL_PROC_NBR</b>		<b>Redefines: STD.TABLE_IDB_BFLD.</b> Identifies the statistic reported. When the associated STD_VS_MFG_FLAG is set to FALSE, this field is defined as follows:
	0	No statistic reported
	1	Number of octets sent
	2	Number of octets received
	3	Number of <acse-pdu> sent
	4	Number of <acse-pdu> received
	5	Number of <acse-pdu> forwarded
	6	Number of <acse-pdu> dropped
	7	Number of transmission errors
	8	Number of reception errors
	9	Number of collisions
	10	Number of message overruns
	11	Number of framing errors
	12	Number of checksum errors
	13	Number of active Associations
	14	Number of Associations timeouts
	15	Number of signal carriers lost
	16	Signal strength (0-100%)
	17	Signal strength (dBm)
	18	Number of registrations sent
	19	Number of registrations received

	20	Number of registration denials (received but rejected)
	21	Number of registrations failed (issued but rejected or lost)
	22	Number of de-registrations requested
	23..2047	Reserved
<b>STD_VS_MFG_FLAG</b>	FALSE	The statistic is defined by this standard.
	TRUE	The statistic is defined by the manufacturer.
<b>SELECTOR</b>	0	Identifies the statistics belonging (or derived by) the C12.22 Device. If the C12.22 Node is an integrated device (i.e. it is not a C12.22 Device that connects to a C12.22 Communication Module) then this value shall be used for the C12.22 Node.
	1	Identifies the statistics belonging (or delivered from) the C12.22 Communication Module that is attached to the C12.22 Device.
	2..15	Reserved.
<b>NETWORK_STATISTICS_ENTRY_RCD</b>		
<b>INTERFACE_ID</b>	0..255	The physical interface number used to register this C12.22 Node. This ID is an index in the INTERFACE_ENTRIES array of the Interface control table (Table 122).
<b>STATISTIC_ID</b>		See TABLE_IDB_BFLD defined above.
<b>VALUE</b>		Statistic reported.
<b>NETWORK_STATISTICS_RCD</b>		
<b>STATISTICS</b>		Array containing a series of statistics. Each entry contains an identifier and an associated value per <b>NETWORK_STATISTICS_CTRL_TBL</b> (Table 127).

## C.2 Decade 130 - Relay Control Tables

This decade contains tables associated with the management of a C12.22 Relay.

**TABLE 130 Dimension Relay Table**

### Table 130 Data Description

**DIM\_RELAY\_TBL** (Table 130) specifies the maximum dimensional values for this decade.

**Global Default Table Property Overrides:** Role="LIMITING", Accessibility="READONLY"

**TYPE RELAY\_CONFIGURATION\_BFLD = BIT FIELD OF UINT8**

```

    ASSIGN_APTITLE_LOCALLY      : BOOL(0);
    FILLER                      : FILL(1..7);
END;
```

**TYPE DIM\_RELAY\_RCD = PACKED RECORD**

```

    RELAY_CONFIGURATION          : RELAY_CONFIGURATION_BFLD;
    NBR_REGISTRATION_ENTRIES     : UINT32;
    NBR_STATIC_ROUTING_ENTRIES   : UINT16;
    NBR_DYNAMIC_ROUTING_ENTRIES  : UINT16;
    NBR_ASSIGNED_MASTER_RELAY    : UINT16;
    NBR_HOSTS                    : UINT16;
END;
```

**TABLE 130 DIM\_RELAY\_TBL = DIM\_RELAY\_RCD;**

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>RELAY_CONFIGURATION_BFLD</b> <b>ASSIGN_APTITLE_LOCALLY</b>		Any C12.22 Relay may act to provide ApTitles that are derived from its registered C12.22 Node ApTitle to C12.22 Nodes in its domain that do not register themselves with a C12.22 Master Relay.
	FALSE	This relay does not support assignment of ApTitles locally.
	TRUE	This relay does support assignment of ApTitles locally.
<b>DIM_RELAY_RCD</b> <b>NBR_REGISTRATION_ENTRIES</b>	0..4294967295	Maximum number of C12.22 Node registration entries that can be contained in the Registration List Table (Table 132) of this C12.22 Relay.
<b>NBR_STATIC_ROUTING_ENTRIES</b>	0.. 65535	Maximum number of static routing rules supported by this C12.22 Relay.
<b>NBR_DYNAMIC_ROUTING_ENTRIES</b>		



	0..65535	Maximum number of dynamic routing rules supported by this C12.22 Relay.
<b>NBR_ASSIGNED_MASTER_RELAY</b>	0..65535	Maximum number of different C12.22 Master Relays that can be automatically assigned by this C12.22 Relay.
<b>NBR_HOSTS</b>	0..65535	Maximum number of C12.22 Authentication Hosts and C12.22 Notification Hosts that can be serviced by this C12.22 Master Relay.

**TABLE 131 Actual Relay Table****Table 131 Data Description**

**ACT\_RELAY\_TBL** (Table 131) specifies the actual dimensional values for this decade.

**Global Default Table Property Overrides:** Role="ACTUAL"

**TABLE 131 ACT\_RELAY\_TBL = DIM\_RELAY\_RCD;**

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>RELAY_CONFIGURATION_BFLD ASSIGN_APTITLE_LOCALLY</b>		Any C12.22 Relay may act to provide ApTitles that are derived from its registered C12.22 Node ApTitle to C12.22 Nodes in its domain that do not register themselves with a C12.22 Master Relay.
	FALSE	This relay does not assign ApTitles locally.
	TRUE	This relay assigns ApTitles locally.
<b>DIM_RELAY_RCD</b>		
<b>NBR_REGISTRATION_ENTRIES</b>	0..4294967295	Actual number of C12.22 Node registration entries contained in the Registration List Table (Table 132) of this C12.22 Relay
<b>NBR_STATIC_ROUTING_ENTRIES</b>	0.. 65535	Actual number of static routing rules supported by this C12.22 Relay.
<b>NBR_DYNAMIC_ROUTING_ENTRIES</b>	0.. 65535	Actual number of dynamic routing rules supported by this C12.22 Relay.
<b>NBR_ASSIGNED_MASTER_RELAY</b>	0..65535	Actual number of different C12.22 Master Relays that can be automatically assigned by this C12.22 Relay.
<b>NBR_HOSTS</b>	0..65535	Actual number of C12.22 Authentication Hosts and C12.22 Notification Hosts that are serviced by this C12.22 Master Relay.

**TABLE 132 Registration List Table****Table 132 Data Description**

**REGISTRATION\_LIST\_TBL** (Table 132) is used to access the registration information of a C12.22 Relay. Entries in this table are added upon successful new registration of C12.22 Nodes. Entries are removed for each de-registration service received for a C12.22 Node or for a C12.22 Node for which the REGISTRATION\_TIME\_OUT timer has expired. To forward a C12.22 Message, a C12.22 Relay shall first attempt to find an entry in this table. If no match is found, the C12.22 Relay shall attempt to forward the C12.22 Message as per static routing information located in the Static Routing Table (Table 133).

**Global Default Table Property Overrides:** Role="DATA", Accessibility="READONLY"

**TYPE NODE\_QUALIFIER\_BFLD = BIT FIELD OF UINT8**

```

RELAY_FLAG           : BOOL(0);
MASTER_RELAY_FLAG    : BOOL(1);
HOST_FLAG            : BOOL(2);
NOTIFICATION_HOST_FLAG : BOOL(3);
AUTHENTICATION_HOST_FLAG : BOOL(4);
END_DEVICE_FLAG      : BOOL(5);
NEIGHBOR_FLAG        : BOOL(6);
APTITLE_ASSIGNED_FLAG : BOOL(7);
END;
```

**TYPE REGISTRATION\_LIST\_ENTRY\_RCD = PACKED RECORD**

```

NODE_APTITLE          : BINARY(20);
NODE_NATIVE_ADDRESS   : BINARY(ACT_NETWORK_TBL.NATIVE_ADDRESS_LEN);
INTERFACE             : UINT8;
NODE_QUALIFIER        : NODE_QUALIFIER_BFLD;
NODE_CLASS            : BINARY(4);
ELECTRONIC_SERIAL_NUMBER : BINARY(20);
ASSIGNED_MASTER_RELAY : BINARY(20);
REGISTRATION_TIME_OUT : UINT16;
REGISTRATION_COUNT_DOWN : UINT16;
MESSAGES_SENT         : UINT48;
MESSAGES_RECEIVED     : UINT48;
END;
```

**TYPE REGISTRATION\_LIST\_RCD = PACKED RECORD**

```

REGISTRATIONS : ARRAY[ACT_RELAY_TBL.NBR_REGISTRATION_ENTRIES]
                OF REGISTRATION_LIST_ENTRY_RCD;
END;
```

**TABLE 132 REGISTRATION\_LIST\_TBL = REGISTRATION\_LIST\_RCD;**

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>NODE_QUALIFIER_BFLD</b> <b>RELAY_FLAG</b>	FALSE	An indication of whether or not this C12.22 Node is a C12.22 Relay. The C12.22 Node is not a C12.22 Relay. If the MASTER_RELAY_FLAG is set to TRUE, the

	TRUE	C12.22 Master Relay will not route C12.22 Messages to other C12.22 Nodes. The C12.22 Node is a C12.22 Relay. When MASTER_RELAY_FLAG is also set to TRUE, the C12.22 Relay is also a C12.22 Master Relay
<b>MASTER_RELAY_FLAG</b>	FALSE TRUE	An indication of whether or not this C12.22 Node is a C12.22 Master Relay. The C12.22 Node is not a C12.22 Master Relay. The C12.22 Node is a C12.22 Master Relay.
<b>HOST_FLAG</b>	FALSE TRUE	An indication of whether or not this C12.22 Node is a C12.22 Host. The C12.22 Node is not a C12.22 Host. The C12.22 Node is a C12.22 Host.
<b>NOTIFICATION_HOST_FLAG</b>	FALSE TRUE	An indication of whether or not this C12.22 Node is a C12.22 Notification Host. The C12.22 Node is not a C12.22 Notification Host. The C12.22 Node is a C12.22 Notification Host.
<b>AUTHENTICATION_HOST_FLAG</b>	FALSE TRUE	An indication of whether or not this C12.22 Node is a C12.22 Authentication Host. The C12.22 Node is not a C12.22 Authentication Host. The C12.22 Node is a C12.22 Authentication Host.
<b>END_DEVICE_FLAG</b>	FALSE TRUE	An indication of whether or not this C12.22 Node has metrological sensors and C12.19 Tables. The C12.22 Node has metrological sensors and C12.19 Tables. The C12.22 Node does not have metrological sensors and C12.19 Tables.
<b>NEIGHBOR_FLAG</b>	FALSE TRUE	The C12.22 Node does not reside on the same C12.22 Network Segment of the C12.22 Relay. The C12.22 Node resides on the same C12.22 Network Segment of the C12.22 Relay.
<b>APTITLE_ASSIGNED_FLAG</b>	FALSE TRUE	The C12.22 Node uses a statically assigned ApTitle. The C12.22 Node ApTitle is dynamically assigned by this C12.22 Relay.
<b>REGISTRATION_LIST_ENTRY_RCD NODE_APTITLE</b>		ApTitle of a C12.22 Node encoded as <universal-id-element> or <relative-uid-element>.
<b>NODE_NATIVE_ADDRESS</b>		Native address used to access this node.
<b>INTERFACE</b>		Interface ID used to access this node. This ID can be used as an index in the INTERFACE_ENTRIES array in the Network control table (Table 122)

<b>NODE_QUALIFIER</b>		See NODE_QUALIFIER_BFLD defined above.
<b>NODE_CLASS</b>		Four bytes containing the MANUFACTURER field as defined in Table 0 of ANSI C12.19-1997 or the DEVICE_CLASS as defined by the revised ANSI C12.19.
<b>ELECTRONIC_SERIAL_NUMBER</b>		Universal identifier used by the auto ApTitle assignment algorithm.
<b>ASSIGNED_MASTER_RELAY</b>		ApTitle of the master relay responsible of this node encoded as <universal-id-element> or <relative-uid-element>.
<b>REGISTRATION_TIME_OUT</b>	0..65535	Time-out value in minutes received in the last registration request.
<b>REGISTRATION_COUNT_DOWN</b>	0..65535	Countdown in minutes before removing this entry. This value is reset to the REGISTRATION_COUNT_DOWN at each new C12.22 Message forwarded from this C12.22 Node.
<b>MESSAGES_SENT</b>		Number of messages forwarded from this node since its installation (initial registration).
<b>MESSAGES_RECEIVED</b>		Number of messages forwarded to this node since its installation (initial registration).
<b>REGISTRATION_LIST_RCD REGISTRATIONS</b>		Array containing information about each C12.22 Node registered to this C12.22 Relay.

**TABLE 133 Static Routing Table****Table 133 Data Description**

**STATIC\_ROUTING\_TBL** (Table 133) is used by C12.22 Relays that support static routing to multiple C12.22 Master Relays. When a C12.22 Message is received by a C12.22 Relay, the relay first tries to locate the destination C12.22 Node (using the message's called ApTitle field) in its Registration list table (Table 132). If this ApTitle is not found and a single C12.22 Master Relay is supported, the relay forwards this message to one of the routes defined in the Registration table (Table 132). If multiple C12.22 Master Relay forwarding is supported, the static routing table (Table 133) is used to identify which route to use to forward this message.

**Global Default Table Property Overrides:** Role="CONTROL"

**TYPE ROUTING\_TABLE\_ENTRY\_RCD = PACKED RECORD**

**APTITLE\_PATTERN** : STRING(30);  
**NATIVE\_ADDRESS** : BINARY(ACT\_NETWORK\_TBL.NATIVE\_ADDRESS\_LEN);  
**INTERFACE\_ID** : UINT8;  
**END;**

**TYPE STATIC\_ROUTING\_TABLE\_RCD = PACKED RECORD**

**ROUTING\_TABLE** : ARRAY[ACT\_RELAY\_TBL.NBR\_STATIC\_ROUTING\_ENTRIES]  
OF ROUTING\_TABLE\_ENTRY\_RCD;  
**END;**

**TABLE 133 STATIC\_ROUTING\_TBL = STATIC\_ROUTING\_TABLE\_RCD;**

IdentifierValueDefinition

**ROUTING\_TABLE\_ENTRY\_RCD**  
**APTITLE\_PATTERN**

Each entry contains ApTitle patterns that are associated with forwarding interfaces. An ApTitle pattern is represented as dot delimited numeric or '\*' strings as shown below (for more details see "Annex C.3 Universal ID Pattern Description of ApTitles").

ApTitleMask	Description
2.16.840.1.234	Represents the ApTitle pattern for node 2.16.840.1.234.
2.16.840.1.*	Represents the pattern for any C12.22 Node whose ApTitle begins with 2.16.840.1 and is followed by any sub branch derived from of this ApTitle.
*	Represents any C12.22 Node (matches anything).

When searching for a forwarding C12.22 Node's address, the routing tables are parsed

sequentially from low array indices toward higher indices until a match is found or until the end of a table is reached.

When a match is found and the C12.22 Node's destination address is reachable through another enabled interface (of this relay) to another remote C12.22 Relay, then the C12.22 Message may be forwarded to that relay. If the destination is a C12.22 Node (e.g. a C12.19 Device) then the C12.22 Message may be delivered to that node directly on the local network segment, thus completing the delivery of the message.

#### **INTERFACE\_ID**

Interface number used to forward C12.22 Messages to C12.22 Nodes having an ApTitle that matches the associated pattern (APTITLE\_PATTERN).

#### **NATIVE\_ADDRESS**

Native address of the C12.22 forwarding Node on a C12.22 Network Segment that is attached to this C12.22 Relay.

#### **ROUTING\_TABLE\_RCD ROUTING\_TABLE**

An array containing a collection of static ApTitle patterns and corresponding destinations used to forward messages. All entries are evaluated sequentially, from index 0 to index NBR\_STATIC\_ROUTING\_ENTRY-1. The search stops at the first pattern that matches, thus resulting in the C12.22 Message being forwarded to the corresponding destination.

**TABLE 134 Host Notification Table****Table 134 Data Description**

**HOST\_NOTIFICATION\_TBL** (Table 134) contains the list of authentication and notification hosts that shall be contacted when a registration or de-registration service is successfully processed by a master relay. The following conditions shall be true for any notification or authentication host to be contacted by this master relay:

1. The C12.22 Notification Host has been registered with this C12.22 Master Relay as a notification host.
2. The C12.22 Authentication Host has been registered with this C12.22 Master Relay as an authentication host.
3. The registered ApTitle matches the **NOTIFICATION\_PATTERN** Element that is bound to the **HOST\_APTITLE** (whether manually entered by the administrator or obtained as a Registration Service parameter).

**Global Default Table Property Overrides:** role="CONTROL", volatile="true"

**Table 134 Type Definitions**

**TYPE HOST\_NOTIFICATION\_ENTRY\_RCD = PACKED RECORD**

```

HOST_APTITLE           : BINARY(20);
HOST_TYPE              : INTERFACE_STATUS_TBL.NODE_TYPE_BFLD;
NBR_PENDING_NOTIFICATION : UINT16 ;
NOTIFICATION_RETRY_INTERVAL : UINT16;
NOTIFICATION_RETRY_COUNT  : UINT16;
NOTIFICATION_TIME_OUT     : UINT16;
NOTIFICATION_EXCLUSION   : UINT16;
HOST_SECURITY           : UINT16;
NOTIFICATION_PATTERN     : STRING(30);
END;
```

**TYPE HOST\_NOTIFICATION\_RCD = PACKED RECORD**

```

NOTIFICATION_HOSTS      : ARRAY[ACT_RELAY_TBL.NBR_HOSTS]
                        OF HOST_NOTIFICATION_ENTRY_RCD;
END;
```

**TABLE 134 HOST\_NOTIFICATION\_TBL = HOST\_NOTIFICATION\_RCD;**

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>HOST_NOTIFICATION_ENTRY_RCD</b> <b>HOST_APTITLE</b>		ApTitle of an authentication or notification host. A <b>HOST_APTITLE</b> that is all binary zeros implies that this host notification entry is not used.
<b>HOST_TYPE</b>		Indicates if this host is used to authenticate registration and deregistration or needs only to be notified for each registration received (when <b>AUTHENTICATION_HOST_FLAG</b> or <b>NOTIFICATION_HOST_FLAG</b> flag is set to



		TRUE). See <b>INTERFACE_STATUS_TBL</b> . <b>NODE_TYPE_BFLD</b> for more details).
<b>NBR_PENDING_NOTIFICATION</b>	0..65535	Number of notification pending because this host is not available.
<b>NOTIFICATION_RETRY_INTERVAL</b>	0..65535	Minimum period in minutes following a failed attempt before a next attempt is made to notify a host. Zero means that the C12.22 Master Relay will not try again to send notifications and instead will wait for the corresponding hosts to register (or re-register) before sending the pending notifications.
<b>NOTIFICATION_RETRY_COUNT</b>	0..65535	Maximum number of notification retry attempts following the first unsuccessful notification attempt. Zero means that the C12.22 Master Relay will not attempt to resend notifications to a non-responsive host until it registers (or re-registers). Non-zero values will cause the C12.22 Master Relay to retry up to the stated value before classifying the C12.22 Host as a non-responsive host.
<b>NOTIFICATION_TIME_OUT</b>	0..65535	Number of minutes after the last unsuccessful retry that a pending notification can be deleted and not sent to the corresponding host. Zero means never delete.
<b>NOTIFICATION_EXCLUSION</b>	0..65535	Minimum period in minutes between two consecutive notifications to the same C12.22 Host.
<b>HOST_SECURITY</b>	0..65535	A index into the Element <b>HOST_SECURITY</b> of <b>HOST_ACCESS_SECURITY_TBL</b> (Table 47). This index is used to select an appropriate password, keys and authentication data that shall be encoded in the C12.22 Message registration message that is sent to the C12.22 Authentication Host or C12.22 Notification Host.
<b>NOTIFICATION_PATTERN</b>		An ApTitle pattern that is associated with the C12.22 Nodes population that is of interest this authentication or notification host. An ApTitle pattern is represented as dot delimited numeric or '*' strings as described in "Annex C.3 Universal ID Pattern Description of ApTitles"). This value may be statically entered programmatically or acquired its value through the Registration Service.
<b>HOST_NOTIFICATION_RCD</b> <b>NOTIFICATION_HOSTS</b>		Array containing a list of C12.22 Authentication Hosts and C12.22 Notification Hosts.

**TABLE 135 Master Relay Assignment Table****Table 135 Data Description**

**MASTER\_RELAY\_ASSIGNMENT\_TBL** (Table 135) is used to define the ApTitles of the C12.22 Master Relays that shall be assigned to C12.22 Nodes based on their registration-time-provided serial number patterns. See "Annex A.6 C12.22 Master Relay ApTitle Auto-assignment" for more detail.

**Global Default Table Property Overrides:** Role="CONTROL"

**TYPE ASSIGNMENT\_ENTRY\_RCD = PACKED RECORD**

**SERIAL\_NUMBER\_PATTERN** : STRING(30);

**MASTER\_RELAY\_ASSIGNED** : BINARY(20);

**END;**

**TYPE MASTER\_RELAY\_ASSIGNMENT\_RCD = PACKED RECORD**

**MASTER\_RELAYS** : ARRAY[ACT\_RELAY\_TBL.NBR\_ASSIGNED\_MASTER\_RELAY]  
OF ASSIGNMENT\_ENTRY\_RCD;

**END;**

**TABLE 135 MASTER\_RELAY\_ASSIGNMENT\_TBL = MASTER\_RELAY\_ASSIGNMENT\_RCD;**

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>ASSIGNMENT_ENTRY_RCD</b>		
<b>SERIAL_NUMBER_PATTERN</b>		The serial number received is compared to each of these patterns. The search stops at the first match and the corresponding master relay ApTitle is assigned. This pattern is constructed following the rules defined in annex "Annex C.3 Universal ID Pattern Description of ApTitles".
<b>MASTER_RELAY_ASSIGNED</b>		ApTitle of the C12.22 Master Relay assigned.
<b>MASTER_RELAY_ASSIGNMENT_RCD</b>		
<b>MASTER_RELAYS</b>		List of serial number patterns and corresponding C12.22 Master Relay ApTitles to assign. All entries are evaluated sequentially, from index 0 to index NBR_ASSIGNED_MASTER_RELAY-1, the search stop at the first pattern that match and the corresponding master relay ApTitle is assigned.

**TABLE 136 Dynamic Routing Report Table****Table 136 Data Description**

**DYNAMIC\_ROUTING\_REPORT\_TBL** (Table 136) is used by C12.22 Relays to report their routing patterns that are dynamically derived from actual traffic patterns and registration. It is intended to be used for diagnostic reports.

**Global Default Table Property Overrides:** Role="DATA"

```

TYPE DYNAMIC_ROUTING_TABLE_RCD = PACKED RECORD
  ROUTING_TABLE      : ARRAY[ACT_RELAY_TBL.NBR_DYNAMIC_ROUTING_ENTRIES]
                      OF STATIC_ROUTING_TBL.ROUTING_TABLE_ENTRY_RCD;
END;

TABLE 136 DYNAMIC_ROUTING_REPORT_TBL = DYNAMIC_ROUTING_TABLE_RCD;

```

### C.3 Universal ID Pattern Description of ApTitles

Patterns are used in these decades to select ApTitles and serial numbers. This section describes how these patterns are constructed and evaluated.

<i>n</i>	Universal ID segment value expressed as a decimal number (e.g. "123").
.	The (dot) is the Universal ID segment delimiter. Example: "2.20.30" is an Universal ID with three segments 2, 20, and 30.
*	Matches zero or more branches of the Universal ID. Example: "2.20.*" match "2.20.2" and "2.20.3.45".
[ ]	Delimits and groups a range of Universal ID patterns, using the "-" (minus) as a range indicator and the "," (comma) as a range alternatives separator. Example: "2.20.[1-10,40-50,100].*" will match the universal ID "2.20.1.*" through "2.20.10.*", "2.20.40.*" through "2.40.50.*" and "2.20.100.*".
!	Reverses the sense result of the mask (interpreted as "not") on the group or Universal ID element immediately on its right. The reversal operator applies only to the next segment. This option can be used with the range delimiter introduced above. Example: "2.20.!30.100" will reverse the sense of the "30" and match any universal ID which starts with "2.20" and ends with 100 and does not have 30 in its third segment. Similarly "2.20.!(30.100)" will reverse the sense of the "30.100" and match any universal ID which starts with "2.20" and does not end with "30.100".
( )	Groups Universal ID patterns. Groups can be separated by " " (vertical bar) to introduce alternatives. The alternatives are processed from left to right until a match is found. Groups may be nested. Example: "(2.20.100.*) (2.10.5.*)" will match either the UID pattern "2.20.100.*" or "2.10.5.*".
?	Matches this branch of the Universal ID. Example: "2.20.?" matches "2.20.2" but not "2.20.3.45".
<i>pattern{n}</i>	Matches exactly <i>n</i> repetitions of a pattern, where <i>n</i> is a decimal number. A pattern is any of the Universal ID constructors described above. Example: "2.20.*.10" will match anything starting with "2.20" and ending with ".10", whereas "2.20(?:){3}.10" will match anything that starts with "2.20" followed by any three segments then ending with ".10".
<i>pattern{n,m}</i>	Matches <i>n</i> to <i>m</i> repetitions of a pattern, where <i>n</i> and <i>m</i> are decimal numbers. A pattern is any of the Universal ID constructors described above. Example: "2.20.*.10" will match anything starting with "2.20" and ending with ".10", whereas "2.20(?:){1,3}.10" will match anything that starts with "2.20" followed by any one to three segments then ending with ".10".

## C.4 Additions to TABLE 07 - Procedure Initiate Table

This Annex describes procedures associated with the Network Control Tables (Decade 12).

### PROCEDURE 23 Register

The Registration service is used to add and maintain ("keep-alive") routing information of C12.22 Relays. To be part of C12.22 Network, a node shall send a registration service to one of the C12.22 Relays. This procedure is typically initiated through the local port.

**TYPE REGISTER\_RESPONSE\_RCD = PACKED RECORD**

**RESPONSE\_CODE : UINT8;**

**END;**

**PROCEDURE 23 REGISTER\_PROC**

**RESPONSE = REGISTER\_RESPONSE\_RCD;**

Identifier	Value	Definition
<b>RESPONSE_CODE_RCD</b> <b>    RESPONSE_CODE</b>	0..31	PSEM response code received when executing the C12.22 Registration Service.

### PROCEDURE 24 Deregister

The Deregistration Service is used to remove routing information in C12.22 Relays. This procedure, typically used on the local port (ANSI C12.18), is used to initiate this process.

**PROCEDURE 24 DEREGISTER\_PROC**

**RESPONSE = REGISTER\_PROC.REGISTER\_RESPONSE\_RCD;**

### PROCEDURE 25 Network Interface Control

This procedure is used to invoke an operation on a specific network interface.

**TYPE INTERFACE\_CONTROL\_REQUEST\_BFLD = BIT FIELD OF UINT8**

**INTERFACE\_CTRL : UINT(0..1);**

**AUTO\_REGISTRATION\_CTRL : UINT (2..3);**

**DIRECT\_MESSAGING\_CTRL : UINT (4..5);**

**RESET\_STATISTICS\_FLAG : BOOL(6);**

**ALL\_INTERFACES\_FLAG : BOOL(7);**

**END;**

**TYPE INTERFACE\_CONTROL\_REQUEST\_RCD = PACKED RECORD**

**INTERFACE\_ID : UINT8;**

**INTERFACE\_CONTROL : INTERFACE\_CONTROL\_REQUEST\_BFLD;**

**END;**

**PROCEDURE 25 NETWORK\_INTERFACE\_CONTROL\_PROC**

**REQUEST = INTERFACE\_CONTROL\_REQUEST\_RCD;**

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>INTERFACE_CONTROL_REQUEST_BFLD</b>		
<b>INTERFACE_CTRL</b>		
	0	Activates or deactivates the selected network interface communication hardware.
	1	Maintain current state.
	2	Enable this interface.
	3	Disable this interface.
	3	Reset interface.
<b>AUTO_REGISTRATION_CTRL</b>		
	0	Controls if the C12.22 Communication Module registers this interface on the C12.22 Network.
	1	Maintain current state.
	2	Enable auto-registration.
	3	Disable auto-registration.
	3	Force one-time registration now; then return to the previous state of auto-registration. Once the interface is registered the C12.22 Communication Module shall keep the C12.22 Device registration state alive until instructed otherwise or the C12.22 Device becomes de-registered due to external reasons.
<b>DIRECT_MESSAGING_CTRL</b>		
	0	Enables or disables direct addressing of nodes on same network segment.
	1	Maintain current state.
	2	Enable direct communication with target nodes on the same network segment.
	3	Disable direct communication with target nodes on the same network segment.
	3	Reserved.
<b>RESET_STATISTICS_FLAG</b>		
	FALSE	Do not reset statistics available from the Interface Status table (Table 125).
	TRUE	Reset statistics available from the Interface Status table (Table 125).
<b>ALL_INTERFACES_FLAG</b>		
	FALSE	Only the interface specified by the INTERFACE_ID field is affected.
	TRUE	All interfaces supported by the node are affected.
<b>INTERFACE_CONTROL_REQUEST_RCD</b>		
<b>INTERFACE_ID</b>	0..255	Interface number affected when the ALL_INTERFACES_FLAG is set to FALSE.
<b>INTERFACE_CONTROL</b>		
		See INTERFACE_CONTROL_REQUEST_BFLD defined above.

## PROCEDURE 26 Exception Report

This procedure is used by an ANSI C12.22 Node to report an exception. In this context the C12.22 Node acts as a one-way device, as per "Annex E - One-way Devices".

One-way C12.22 Nodes shall encode their data for transmission as a PSEM Write Service request,

<write>, while setting bit 2, NOTIFICATION, of the <ae-qualifier-value> to 1. This is an indication to the called C12.22 Node that the information provided is about the calling C12.22 Node.

Using RESPONSE\_CONTROL, the C12.22 Node may control whether the target responds. If retries are used, then the C12.22 Node should cease retrying when it receives a EPSEM <ok> response or the retry count is reached, whichever is first.

One-way nodes shall set RESPONSE\_CONTROL to two (2, never respond) and otherwise the node may set RESPONSE\_CONTROL to any allowable value.

```
TYPE EXCEPTION_REPORT_REQUEST_RCD = PACKED RECORD
    EXCEPTION          : EXCEPTION_REPORT_CONFIG_TBL.TABLE_IDB_BFLD;
END;
```

```
PROCEDURE 26 EXCEPTION_REPORT_PROC
    REQUEST = EXCEPTION_REPORT_REQUEST_RCD;
```

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
EXCEPTION_REPORT_REQUEST_RCD EXCEPTION		Exception code reported, see EXCEPTION_REPORT_CONFIG_TBL.TABLE_I DB_BFLD.

## C.5 Table 46: Extended Key Table

### Table 46 Data Description

**EXTENDED\_KEY\_TBL** (Table 46) is used to store authentication and encryption keys for use by this End Device. Table entries shall be applied by the End Device upon receipt of any originating C12.22 Message from a remote C12.22 Node and when encoding corresponding responses. Entries from this Table shall not be used to encode initiating-messages as part of registration services, de-registration services, exception reports or messages emitted by C12.22 Master Relays or C12.22 Relays to C12.22 Authentication Hosts and C12.22 Notification Hosts as part of the registration process of other C12.22 Nodes. See also **HOST\_ACCESS\_SECURITY\_TBL** (Table 47).

#### Notes:

1. Although it may be possible to read the **EXTENDED\_KEY\_TBL** (Table 46), the values reported (read back) are not defined. The motivation is to prevent accidental or intentional exposure of the Cipher keys.
2. The entries in this table provide only authentication and message encryption parameters. Group-access attributes to Tables and Procedures shall be determined from the passwords and user identifiers that accompany C12.22 Messages. In fact there is no assumed or implied positional relationship among entries in **SECURITY\_TBL** (Table 42) and entries in **EXTENDED\_KEY\_TBL** (Table 46).

**Global Default Table Property Overrides:** Role="CONTROL", Accessibility="WRITEONLY"

### Table 46 Type Definitions

**TYPE EXTENDED\_KEY\_ENTRY\_RCD = PACKED RECORD**

```

    KEY_ID           : UINT8;
    CIPHER_CODE      : UINT8;
    KEY_LENGTH       : UINT8;
    KEY              : BINARY(ACT_SECURITY_LIMITING_TBL.KEY_LEN);

```

**END;**

**TYPE EXTENDED\_KEY\_RCD = PACKED RECORD**

```

    KEY_ENTRIES      : ARRAY[ACT_SECURITY_LIMITING_TBL.NBR_KEYS]
                      OF EXTENDED_KEY_ENTRY_RCD;

```

**END;**

**TABLE 46 EXTENDED\_KEY\_TBL = EXTENDED\_KEY\_RCD;**

### Table 46 Element Descriptions

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
<b>EXTENDED_KEY_ENTRY_RCD</b>		
<b>KEY_ID</b>	0..255	Key selector used to match a selection for a key that is provided by the communication protocol against a <b>KEY</b> entry in this Table.
<b>CIPHER_CODE</b>		A cipher code used to encrypt, decrypt or authenticate C12.22 Messages.



	0	AES-128/EAX', When AES-128/EAX' is selected, <b>KEY_LENGTH</b> shall be set to 16 octets. This indicates a key length of 128 bits.
	1..255	Reserved.
<b>KEY_LENGTH</b>	0..255	The number of octets used in this KEY entry. The value 0 (zero) implies that this entry is not used.
<b>KEY</b>		Key to be applied in the authentication and/or encryption processes.
<b>EXTENDED_KEY_RCD KEY_ENTRIES</b>		Array of keys used to establish authentication and/or encryption.

## C.6 Table 47 Host Access Security Table

### Table 47 Data Description

**HOST\_ACCESS\_SECURITY\_TBL** (Table 47) is used to store authentication keys, encryption keys and access grants used by remote C12.22 Nodes, such as C12.22 Master Relays, C12.22 Relays, C12.22 Notification Hosts and C12.22 Authentication Hosts. Table entries shall only be used by this the End Device to encode initiating-messages whose purpose is to register a node, de-register a node, report exceptions, or by C12.22 Master Relays that need to send registration service messages to C12.22 Notifications Hosts and C12.22 Authentication Hosts. Entries from this Table shall also be used to decode and authenticated corresponding responses.

The C12.22 Nodes shall use **EXTENDED\_KEY\_TBL** (Table 46) and **SECURITY\_TBL** (Table 42) for all other application related communication (e.g. meter reading, un-solicited data delivery by an End Device).

**Note:** Although it may be possible to read the **HOST\_ACCESS\_SECURITY\_TBL** (Table 47), the values reported (read back) are not defined, but the actual key values shall not be returned.

**Global Default Table Property Overrides:** Role="CONTROL", Accessibility="WRITEONLY"

### Table 47 Type Definitions

**TYPE SECURITY\_QUALIFIER\_BFLD = BIT FIELD OF UINT8**

```

PASSWORD_FLAG          : BOOL(0);
REGISTRATION_FLAG       : BOOL(1);
NOTIFICATION_FLAG       : BOOL(2);
AUTHENTICATION_FLAG     : BOOL(3);
EXCEPTION_FLAG          : BOOL(4);
END;
```

**TYPE HOST\_SECURITY\_ENTRY\_RCD = PACKED RECORD**

```

SECURITY_QUALIFIER      : SECURITY_QUALIFIER_BFLD;
SECURITY_ACCESS          : LOGIN_PROC.PARM_DATA_RCD;
CIPHER_MODE              : UINT8;
KEY_ID                   : UINT8;
CIPHER_CODE              : UINT8;
KEY_LENGTH               : UINT8;
KEY                      : BINARY(ACT_SECURITY_LIMITING_TBL.KEY_LEN);
END;
```

**TYPE HOST\_ACCESS\_SECURITY\_RCD = PACKED RECORD**

```

HOST_SECURITY            : ARRAY [
                                ACT_NETWORK_TBL.NBR_REGISTRATIONS +
                                ACT_NETWORK_TBL.NBR_EXCEPTION_HOSTS +
                                ACT_RELAY_TBL.NBR_HOSTS
                              ] OF HOST_SECURITY_ENTRY_RCD;
END;
```

**TABLE 47 HOST\_ACCESS\_SECURITY\_TBL = HOST\_ACCESS\_SECURITY\_RCD;**

### Table 47 Element Descriptions

<u>Identifier</u>	<u>Value</u>	<u>Definition</u>
-------------------	--------------	-------------------

**SECURITY\_QUALIFIER\_BFLD  
PASSWORD\_FLAG**

A flag that indicates whether the C12.22 Node is required to provide a password and user identification in order to establish the necessary role and access permissions for operation it initiated.

FALSE

**PASSWORD\_ACCESS** is not required. The **PASSWORD\_ACCESS** Element shall not be used.

TRUE

**PASSWORD\_ACCESS** is required. The **PASSWORD\_ACCESS** field shall be communicated as part of the C12.22 Message to establish necessary access permissions per **SECURITY\_TBL** (Table 42) of the C12.22 Host.

**REGISTRATION\_FLAG**

A flag that indicates whether this **HOST\_SECURITY** entry may be for used in messages destined to C12.22 Master in the encoding of registration or de-registration services requests.

FALSE

The associated **HOST\_SECURITY** entry cannot be used to register or to de-register a node.

TRUE

The associated **HOST\_SECURITY** entry can be used to register or to de-register a node.

**NOTIFICATION\_FLAG**

A flag that indicates whether this **HOST\_SECURITY** entry may be uses in messages destined to C12.22 Notification Hosts.

FALSE

**HOST\_SECURITY** associated entry may not be used in messages for C12.19 Notification Hosts.

TRUE

**HOST\_SECURITY** associated entry may be used in messages for C12.19 Notification Hosts.

**AUTHENTICATION\_FLAG**

A flag that indicates whether this **HOST\_SECURITY** entry may be used in messages destined to C12.22 Authentication Hosts.

FALSE

**HOST\_SECURITY** associated entry may not be used in messages for C12.19 Authentication Hosts.

TRUE

**HOST\_SECURITY** associated entry may be used in messages for C12.19 Authentication Hosts.

**EXCEPTION\_FLAG**

A flag that indicates whether this **HOST\_SECURITY** entry may be used in exception report messages destined to C12.22 Node.

FALSE

**HOST\_SECURITY** associated entry may not be used in exception report messages.

		TRUE	<b>HOST_SECURITY</b> associated entry may be used in messages.
<b>HOST_SECURITY_ENTRY_RCD</b>			The necessary security attributes of a remote C12.22 Node such as a C12.22 Master Relay, C12.22 Notification Host or C12.22 Authentication Host that may be accessed by this node.
<b>SECURITY_QUALIFIER</b>			See <b>SECURITY_QUALIFIER_BFLD</b> .
<b>SECURITY_ACCESS</b>			The password and user identification that may be required by the remote C12.22 Node in order to grant the necessary access privileges to this C12.22 Node. This Element shall be communicated as part of the C12.22 Message when the entry's Element <b>SECURITY_QUALIFIER.PASSWORD_FLAG</b> is TRUE. For more details see <b>LOGIN_PROC.PARM_DATA_RCD</b> .
<b>CIPHER_MODE</b>	0		The C12.22 Messages shall be transmitted in plain text without encryption and without authentication fields.
	1		The C12.22 Messages shall be only authenticated using the provided entry's Elements <b>KEY_ID</b> , <b>CIPHER_CODE</b> and <b>KEY</b> . Response messages (if any) shall be authenticated using the first entry in this Table which matches <b>KEY_ID</b> , <b>CIPHER_CODE</b> and <b>KEY</b> from this Table and matches the role one the <b>SECURITY_QUALIFIER</b> flags.
	2		Originating C12.22 Messages shall be encrypted using entry's Elements <b>KEY_ID</b> , <b>CIPHER_CODE</b> and <b>KEY</b> . Response messages (if any) shall be decrypted using the first entry which matches <b>KEY_ID</b> , <b>CIPHER_CODE</b> and <b>KEY</b> from this Table and matches the role one the <b>SECURITY_QUALIFIER</b> flags.
	3..255		Reserved.
<b>KEY_ID</b>	0..255		Key selector used to select a <b>CIPHER_CODE</b> .
<b>CIPHER_CODE</b>			A cipher code used to encrypt or authenticate C12.22 Messages that are initiated by this C12.22 Node and to decrypt or authenticate corresponding response C12.22 Messages received from a remote C12.22 Node.
	0		AES-128/EAX', When AES-128/EAX' is selected, <b>KEY_LENGTH</b> shall be set to 16 octets. This indicates a key length of 128 bits.
	1....255		Reserved.

<b>KEY_LENGTH</b>	0..255	The number of octets used in this <b>KEY</b> entry. This value shall be less or equal to <b>ACT_SECURITY_LIMITING_TBL.KEY_LEN</b> . The value zero (0) implies that this <b>KEY</b> is not used and has the same effect as setting the <b>CIPHER_MODE</b> to 0.
<b>KEY</b>		The key to be applied in the authentication, or encryption or decryption processes.
<b>HOST_ACCESS_SECURITY_RCD</b> <b>HOST_SECURITY</b>		An array of passwords, keys and cipher codes that may be used to encoded C12.22 Message, as part of a request, or decode C12.22 Messages, as part of a corresponding response.

## Annex D - Universal Identifier

(normative)

ANSI C12.19 and C12.22 make use of the ISO Universal Identifier to uniquely identify objects. These Universal Identifiers are registered under two branches. The first one is used for ANSI C12.19 and related standards to uniquely identify components of the End Device Class, EDL and TDL.

```
<device-class-root-oid> ::= 2.16.124.113620.1.19      {ISO registered absolute
                                                    object identifier root for End Device
                                                    Classes. This value shall be encoded for
                                                    the purpose of transmission using ISO/IEC
                                                    8825-1:2002 [BER] as:
                                                    06 07 60 7C 86 F7 54 01 13H }
```

The second one is used for uniquely identifying Standard communication parameters, such as the Standard-defined communication application context, Standard-defined security mechanisms, and Standard's root ApTitle for all C12.22 Nodes.

```
<application-context-oid> ::= 2.16.124.113620.1.22    {ISO registered absolute
                                                    object identifier for the context of this
                                                    Standard. This value shall be encoded for
                                                    the purpose of transmission using ISO/IEC
                                                    8825-1:2002 [BER] as:
                                                    06 07 60 7C 86 F7 54 01 16H }
```

The following table summarizes the list of objects actually defined:

Use	Universal identifier
ANSI C12.19 Device Class	<device-class-root-oid>.<device class id>
ANSI C12.22 Application context	<application-context-oid>
(Reserve for future sub context)	<application-context-oid>.1
ANSI C12.22 Mechanism name for compatibility with ANSI C12.21 Authentication Service algorithm 00 <sub>H</sub> .	<application-context-oid>.2.0
ANSI C12.22 native Mechanism Name	<application-context-oid>.2.1
ANSI C12.22 ApTitles	<application-context-oid>.0.<service provider id>.<node id> or any other registered Universal Identifier.
ANSI C12.22 ApTitles Broadcast	<application-context-oid>.0.[<service provider id>].<node id>].0
ANSI C12.22 ApTitles Multicast	<application-context-oid>.0.[<service provider id>].<node id>].0.xxx

### ANSI C12.19 Device Class

Absolute C12.19 Device Class identifiers shall be globally unique. To assure this, organizations implementing this Standard can register a Device Class Universal Identifier. This identifier can be used for one or multiple C12.22 Node types that share the same data structure (C12.19 EDL and TDL). This identifier is used by upstream devices to understand incoming data structures.

Device Classes will be assigned on a first come first serve basis. The first 128 Device Class IDs are reserved for registration of one-way devices. Preferred Device Class IDs may also be requested and assigned if available.

Also submitted with the registration request is a simple XML-text TDL file (as defined in Version 2 of ANSI C12.19) and an optional EDL if desired. For one-way devices, EDL and TDL shall include enough information to completely describe any unsolicited messages that the C12.22 Node might generate. For two-way devices, no specific information is required to be included in the EDL and TDL.

### **ANSI C12.22 Node ApTitles**

C12.22 Node ApTitles shall be globally unique. To assure this, organizations implementing this Standard can register an ApTitle Universal Identifier. Under this registered branch, each organization can assign a unique ApTitle for each node installed on the network. There is no limit on the number of C12.22 Nodes assigned under the same branch.

For use in ACSE messages, this ApTitle forms the local root object identifier for either a client or server in a C12.22 Network. This ApTitle is used by C12.22 Networks to propagate C12.22 Messages from source to destination over any network architecture.

### **Multicast addressing**

Multicast addressing is similar to broadcast except that it can be assumed that some communications process has a distribution list (found in Table 122 Interface Control Table) and only routes the message to certain recipients. These recipients, knowledgeable about their membership in the multicast group can respond to such a message as if directly addressed to them.

Broadcast and multicast messages shall be targeted to one C12.22 Network Segment. A C12.22 Relay may forward broadcast and multicast C12.22 Messages to other network segments according to its internal configuration.

A broadcast is addressed to <application-context-oid>.0.[<service provider id>[.<node id>].]0. A multicast is addressed to <application-context-oid>.0.[<service provider id>[.<node id>].]0.xxx; where xxx is a specific multicast address. The specific multicast address can be any relative branch of a Universal Identifier. Note that routers and C12.22 Relays may want to optimize the distribution of such messaging.

### **Registration**

Information on registration can be found at <http://www.naedra.org>. This site is under the management of the ANSI/IEEE/MC OID Oversight group.

## Annex E - One-way Devices

(normative)

One-way C12.22 Nodes fall into two categories:

1. Issuers of ACSE unsolicited messages to the C12.22 Network; or
2. Issuers of blurts (short messages) to a C12.22 Communication Module or a blurt only capable native network.

A C12.22 Network shall only process ACSE encapsulated messages. One way devices shall not send <short-pdu> on a C12.22 Network. When a C12.22 Node is implemented as a C12.22 Device and a C12.22 Communication Module then the C12.22 Communication Module may be configured to process ACSE messages or short messages. It is then the responsibility of the C12.22 Communication Module to map the short messages into proper ACSE encapsulated messages before presenting the messages to the C12.22 Network.

One way C12.22 Nodes shall encode their data for transmission as a PSEM Write Service request, <write>, while setting bit 2, NOTIFICATION, of the <calling-AE-qualifier> to 1. This is an indication to the called C12.22 Node that the information provided is about the calling C12.22 Node. Also they shall set the <epsem-control> bits in the RESPONSE\_CONTROL to 2, requesting the target not to respond.

A C12.22 Communication Module that attaches to a one-way short-message-emitting C12.22 Device shall set bit 2, NOTIFICATION, of the <calling-AE-qualifier> to 1 on behalf of the one-way C12.22 Device to assure the correct interpretation of the information provided by the C12.22 Device.

One-way short-message-emitting devices shall provide exactly two-level relative calling ApTitle (.subbranch-1.subbranch-2) or none (00<sub>H</sub>). It is the responsibility of the first C12.22 Relay (receiving the ACSE message from the C12.22 Host for transmission on the C12.22 Network) or the C12.22 Communication Module (connecting a C12.22 Device to a C12.22 Network) to extend relative calling ApTitle for proper location of the initiator of the message on the C12.22 Network.

A typical example transmitted by a one-way device using ANSI C12.22 has the following ACSE Datagram format:

60 2E	<acse-pdu>
A2 05	<called-AP-title-element>
80 03 7B A1 21	<called-AP-title> = .123.4257
A6 05	<calling-AP-title-element>
80 03 7B A3 54	<calling-AP-title> = .123.4567
A7 03	<calling-AE-qualifier-element>
02 01 04	NOTIFICATION = "true"
A8 03	<calling-AP-invocation-id-element>
02 01 03	<calling-AP-invocation-id> = 3
BE 14	<user-information-element>
28 12	<user-information-external>
81 10	<user-information-octet-string>
90	<epsem-control>
14 00 00 00	<ed-class>
0A	<service-length>
40	<full-write>
00 03	<tableid>
00 04	<count>
00 08 00 00	<data>
F8	<cksum>

The ACSE Datagram format above may be considered to have too much overhead for some technologies used by one-way devices. For this reason a stripped down shorter (blurt) message format is provided as an alternative. This format has been designed to be mapped to an <acse-pdu>.



```

<short-pdu> ::= <short-calling-ApTitle> <short-device-class> <psem-
write-request>

<short-calling-ApTitle> ::= <byte>+ | 00H
    {The calling ApTitle is a two-level
    relative universal to the locally register
    root Aptitle representing the ApTitle and
    encoded without tag and length. The value
    00H is the null ApTitle, which is an
    indication to receiver of this message to
    fully resolve the calling ApTitle.}

<short-device-class> ::= <byte>+
    {The device class is a single level
    universal ID representing the C12.22 Host
    model and encoded without tag and length.
    This universal ID shall be registered as a
    one-way device class derived from the root
    class: <device-class-root-oid>.<short-
    device-class>}

<psem-write-request> ::= <write>
    {As defined in by the PSEM Write Service
    request in this Standard. It is the
    responsibility of the C12.22 Communication
    Module or C12.22 Node, which translates
    the <short-pdu> to an <acse-pdu>, to set
    bit 2, NOTIFICATION, of the <calling-ae-
    qualifier > to 1 as an indication to the
    called C12.22 Node that the information
    provided by the write service request is
    about the calling C12.22 Node.}

```

The previous example can be encoded as follow using this format:

7B A3 54	<relative-uid>
14	<ed-class>
40	<full-write>
00 03	<tableid>
00 04	<count>
01 02 03 04	<data>
F6	<cksum>

## Annex F - APDU Response Timeout Algorithm

(informative)

The Application Response Time-out is used by a C12.22 Node that issues EPSEM requests to another C12.22 Node and needs to know how long to wait for responses. This annex illustrates one possible behavior that an implementer may use in a client application to deal with dynamically changing network latency.

This algorithm is applicable both to session and session-less transactions.

In this example we set the initial value of the APDU response timeout to 30 seconds and the number of retries to three. Ideally the number of retry attempts when multiplied by the default time out interval length should be set to at least one fifth of the C12.22 session timeout value. The time-out value should be reset to its initial value each time a new session or session-less association is created. Upon timeout, the C12.22 Node may attempt one or more transmissions of the failing request where the number of retries is set to an application provided value and can be programmed in the INTERFACE\_CTRL\_TBL when implemented.

Each time an Application Layer <acse-pdu> response is received, the response time-out duration may be re-calculated and revised to a value that is two times the average response time of previous three responses that were received (use the time-out value for responses that were not received in this calculation).

Each time an Application Layer APDU response is not received within the anticipated time interval, the latest timeout interval shall be increased by a factor of two, and then a duplicate Application Layer APDU shall be transmitted as shown in the table below.

Steps	Response interval (seconds)	Response timeout (seconds)
1		30 (Example of a initial value)
2	7	$(30 + 30 + 2*7) / 3 = 25$
3	9	$(30 + 2*7 + 2*9) / 3 = 21$
4	6	$(2*7 + 2*9 + 2*6) / 3 = 15$
5	fail	$2*15 = 30$
7	5	$(2*6 + 30 + 2*5) / 3 = 17$

The retransmission process may be performed up to three times. If after three re-transmission attempts the Application Layer does not receive a valid APDU response and the calculated time-out is less than the default time-out value, the time-out value shall be set to the default time-out and up to three more transmission attempts shall be performed.

If after three consecutive re-transmission attempts, where the time-out value is greater than the default time-out value and the Application Layer does not receive a valid APDU response an application timeout shall occur and the Application Layer shall return <nett> to its Application Process.

The <nett> error response shall be placed in the <epsem>s <response> member of the <acse-pdu>s <user-data-element>. The <acse-pdu>s <called-AP-title-element> in the response shall be set to the value found in the <calling-AP-title-element> of the original request and the <acse-pdu>s <calling-AP-title-element> in the response shall be set to the <called-AP-title-element> value found in the original request. All other fields, such as <calling-AE-qualifier-element>, shall be replicated in the <nett> response.

If the C12.22 Node is the originator of the request, then the request shall be aborted upon receipt of a <nett> error by the Application Process and terminate its session (if any). If the C12.22 Node is not the originator of the service request (i.e., it is a C12.22 Relay) then it shall not participate in this algorithm.

## Annex G - Communication Example

(informative)

### Example #1: Unsecured session

The following example represents a session initiated by node “.123.4” to read Table 05 of node “.123.8437”. Messages exchanges are sent in cleartext.

Logon request

```
60 29
  A2 05
    80 03 7B C1 75
  A6 04
    80 02 7B 04
  A8 03
    02 01 07
  BE 15
    28 13
      81 11
        80
        0F
        50
        00 02
        55 53 45 52 20 4E 41 4D 45 20
        00 3C
```

```
<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.8437
<calling-AP-title-element>
<calling-AP-title> .123.4
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 7
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control>
<service-length>
<logon>
<user-id> = 2
<user> = "USER NAME"
<req-session-idle-timeout> = 60 sec
```

Logon response

```
60 22
  A2 04
    80 02 7B 04
  A4 03
    02 01 07
  A6 05
    80 03 7B C1 75
  A8 03
    02 01 00
  BE 09
    28 07
      81 05
        80
        03
        00
        00 3C
```

```
<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.4
<called-AP-invocation-id-element>
<called-AP-invocation-id> = 7
<calling-AP-title-element>
<calling-AP-title> = .123.8437
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 0
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control>
<service-length>
<ok>
<resp-session-idle-timeout> = 60 sec
```

Read request

```
60 1D
  A2 05
    80 03 7B C1 75
  A6 04
    80 02 7B 04
  A8 03
    02 01 00
  BE 09
    28 07
      81 05
        80
        03
        30
        00 05
```

```
<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.8437
<calling-AP-title-element>
<calling-AP-title> .123.4
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 0
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control>
<service-length>
<full-read>
<table-id> = 5
```

Read response

```
60 37
  A2 04
    80 02 7B 04
  A4 03
    02 01 00
  A6 05
    80 03 7B C1 75
  A8 03
```

```
<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.4
<called-AP-invocation-id-element>
<called-AP-invocation-id> = 0
<calling-AP-title-element>
<calling-AP-title> = .123.8437
<calling-AP-invocation-id-element>
```

02 01 01	<calling-AP-invocation-id> = 1
BE 1E	<user-information-element>
28 1C	<user-information-external>
81 1A	<user-information-octet-string>
80	<epsem-control>
18	<service-length>
00	<ok>
00 14	<count>
44 45 56 49 43 45 20 49 44 20 20 20 20	20 20 20 20 20 20 20 "DEVICE ID"
43	<cksum>
Logoff request	
60 1B	<acse-pdu>
A2 05	<called-AP-title-element>
80 03 7B C1 75	<called-AP-title> = .123.8437
A6 04	<calling-AP-title-element>
80 02 7B 04	<calling-AP-title> .123.4
A8 03	<calling-AP-invocation-id-element>
02 01 01	<calling-AP-invocation-id> = 1
BE 07	<user-information-element>
28 05	<user-information-external>
81 03	<user-information-octet-string>
80	<epsem-control>
01	<service-length>
52	<logoff>
Logoff response	
60 20	<acse-pdu>
A2 04	<called-AP-title-element>
80 02 7B 04	<called-AP-title> = .123.4
A4 03	<called-AP-invocation-id-element>
02 01 01	<called-AP-invocation-id> = 1
A6 05	<calling-AP-title-element>
80 03 7B C1 75	<calling-AP-title> = .123.8437
A8 03	<calling-AP-invocation-id-element>
02 01 02	<calling-AP-invocation-id> = 2
BE 07	<user-information-element>
28 05	<user-information-external>
81 03	<user-information-octet-string>
80	<epsem-control>
01	<service-length>
00	<ok>

## Example #2: Unsecured sessionless

The following example represents a sessionless transaction initiated by ".123.4" to read field MFG\_SERIAL\_NUMBER of node ".123.8437". Messages exchanges are sent in cleartext.

Offset partial read request	
60 22	<acse-pdu>
A2 05	<called-AP-title-element>
80 03 7B C1 75	<called-AP-title> = .123.8437
A6 04	<calling-AP-title-element>
80 02 7B 04	<calling-AP-title> .123.4
A8 03	<calling-AP-invocation-id-element>
02 01 14	<calling-AP-invocation-id> = 20
BE 0E	<user-information-element>
28 0C	<user-information-external>
81 0A	<user-information-octet-string>
80	<epsem-control>
08	<service-length>
3F	<pread-offset>
00 01	<tableid> = 1
00 00 10	<offset> = 16
00 10	<octet-count> = 16
Offset partial read response	
60 33	<acse-pdu>
A2 04	<called-AP-title-element>
80 02 7B 04	<called-AP-title> = .123.4
A4 03	<called-AP-invocation-id-element>
02 01 14	<called-AP-invocation-id> = 20
A6 05	<calling-AP-title-element>

```

      80 03 7B C1 75
A8 03
      02 01 14
BE 1A
      28 18
          81 16
              80
              14
              00
              00 10
          4D 41 4E 55 46 41 43 54 55 52 45 52 20 53 4E 20 "MANUFACTURER SN"
          92

```

```

<calling-AP-title> = .123.8437
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 20
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control>
<service-length>
<ok>
<count>
4D 41 4E 55 46 41 43 54 55 52 45 52 20 53 4E 20 "MANUFACTURER SN"
<cksum>

```

### Example #3: Unsecured notification

The following example represents an unacknowledged notification of Procedure 26 initiated by node .123.273 and targeted to node .123.2. This notification is sent in cleartext.

```

Write request
60 2E
  A2 04
    80 02 7B 02
  A6 05
    80 03 7B 82 11
  A7 03
    02 01 06
  A8 03
    02 01 18
  BE 15
    28 13
        81 11
        92
        54 45 4D 50
        0B
        40
        00 07
        00 05
        1A 00
        00
        01 00
        E5

```

```

<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.2
<calling-AP-title-element>
<calling-AP-title> = .123.273
<calling-AE-qualifier-element>
URGENT = TRUE, NOTIFICATION = TRUE
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 24
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> ED_CLASS_INCLUDED = true,
                  RESPONSE_CONTROL = 2
<ed-class>
<service-length>
<full-write>
<tableid> = 7 (In little endian)
<count> = 5
PROC = 26
SEQ_NBR = 0
PARM = 1 (Primary power down event code)
<cksum>

```

### Example #4: Authenticated session

The following example represents a session initiated by node ".123.4" to read Table 05 of node ".123.8437". Messages exchanges include a Message Authentication Code.

```

Logon request
60 3E
  A2 05
    80 03 7B C1 75
  A6 04
    80 02 7B 04
  A8 03
    02 01 04
  AC 0F
    A2 0D
      A0 0B
        A1 09
          80 01 02
          81 04 48 F3 C2 05
  BE 19
    28 17
      81 15
          84
          0F
          50
          00 02
          55 53 45 52 20 4E 41 4D 45 20
          00 3C

```

```

<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.8437
<calling-AP-title-element>
<calling-AP-title> .123.4
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 4
<calling-authentication-value-element>
<calling-authentication-value-external>
<calling-authentication-value-single-asn1>
<calling-authentication-value-cl222>
<key-id-element> = 2
<iv-element> = 2008-10-13 21:47:49 UTC
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 1
<service-length>
<logon>
<user-id> = 2
<user> = "USER NAME"
<req-session-idle-timeout> = 60 sec

```

# ANSI C12.22-2008

AD DC 46 60

Logon response

```
60 37
  A2 04
    80 02 7B 04
  A4 03
    02 01 04
  A6 05
    80 03 7B C1 75
  A8 03
    02 01 00
  AC 0F
    A2 0D
      A0 0B
        A1 09
          80 01 02
          81 04 48 F3 C2 04
  BE 0D
    28 0B
      81 09
        84
        03
        00
        00 3C
        C0 7D B1 65
```

Read request

```
60 21
  A2 05
    80 03 7B C1 75
  A6 04
    80 02 7B 04
  A8 03
    02 01 00
  BE 0D
    28 0B
      81 09
        84
        03
        30
        00 05
        75 E7 5A 51
```

Read response

```
60 3B
  A2 04
    80 02 7B 04
  A4 03
    02 01 00
  A6 05
```

```
<mac>
Network context = 2.16.124.113620.1.22.0
K = 08070605040302010807060504030201
L = B86D06FD5929DFAAF6BE44CED8939EC3
D = 70DA0DFAB253BF55ED7C899DB1273D01
Q = E1B41BF564A77EABDAF9133B624E7A02
N = A20D060B607C86F7540116007BC175A8
    03020104AC0FA20DA00BA10980010281
    0448F3C205BE192817811584A60C060A
    607C86F7540116007B040248F3C2050F
    5000025534552204E414D4520003C80
Tag = ADDC4660BD37FFC17E38A52A9E9F9902
T = ADDC4660
```

```
<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.4
<called-AP-invocation-id-element>
<called-AP-invocation-id> = 4
<calling-AP-title-element>
<calling-AP-title> = .123.8437
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 0
<calling-authentication-value-element>
<calling-authentication-value-external>
<calling-authentication-value-single-asn1>
<calling-authentication-value-cl222>
<key-id-element> = 2
<iv-element> = 2008-10-13 21:47:48 UTC
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 1
<service-length>
<ok>
<resp-session-idle-timeout> = 60 sec
<mac>
N = A20C060A607C86F7540116007B04A403
    020104A803020100AC0FA20DA00BA109
    800102810448F3C204BE0D280B810984
    A60D060B607C86F7540116007BC17502
    48F3C2040300003C8000000000000000
Tag = C07DB165308C0AF031240676F624C8B5
T = C07DB165
```

```
<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.8437
<calling-AP-title-element>
<calling-AP-title> .123.4
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 0
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 1
<service-length>
<full-read>
<table-id> = 5
<mac>
N = A20D060B607C86F7540116007BC175A8
    03020100BE0D280B810984A60C060A60
    7C86F7540116007B040248F3C2040330
    00058000000000000000000000000000
Tag = 75E75A517893E5F2F83D725ACD3BD8F9
T = 75E75A51
```

```

      80 03 7B C1 75
A8 03
      02 01 01
BE 22
      28 20
          81 1E
              84
              18
              00
              00 14
              44 45 56 49 43 45 20 49 44 20
              20 20 20 20 20 20 20 20 20 20
              43
              93 96 5A F1

```

Logoff request

```

60 1F
      A2 05
          80 03 7B C1 75
      A6 04
          80 02 7B 04
      A8 03
          02 01 01
      BE 0B
          28 09
              81 07
                  84
                  01
                  52
                  B6 47 27 4F

```

Logoff response

```

60 24
      A2 04
          80 02 7B 04
      A4 03
          02 01 01
      A6 05
          80 03 7B C1 75
      A8 03
          02 01 02
      BE 0B
          28 09
              81 07
                  84
                  01
                  00
                  AD 30 B6 E7

```

```

<calling-AP-title> = .123.8437
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 1
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 1
<service-length>
<ok>
<count>
"DEVICE ID"

<cksum>
<mac>
N =  A20C060A607C86F7540116007B04A403
    020100A803020101BE222820811E84A6
    0D060B607C86F7540116007BC1750248
    F3C20518000014444556494345204944
    20202020202020202020204380000000
Tag = 93965AF1592D4316C4422B93F9C4769A
T = 93965AF1

```

```

<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.8437
<calling-AP-title-element>
<calling-AP-title> .123.4
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 1
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 1
<service-length>
<logoff>
<mac>
N =  A20D060B607C86F7540116007BC175A8
    03020101BE0B2809810784A60C060A60
    7C86F7540116007B040248F3C2040152
Tag = B647274FA20F920E367AF64DEDA56119
T = B647274F

```

```

<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.4
<called-AP-invocation-id-element>
<called-AP-invocation-id> = 1
<calling-AP-title-element>
<calling-AP-title> = .123.8437
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 2
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 1
<service-length>
<ok>
<mac>
N =  A20C060A607C86F7540116007B04A403
    020101A803020102BE0B2809810784A6
    0D060B607C86F7540116007BC1750248
    F3C20501008000000000000000000000
Tag = AD30B6E76134AED93E3F212FF2674040
T = AD30B6E7

```

### Example #5: Authenticated sessionless

The following example represents a sessionless transaction initiated by ".123.4" to read field MFG\_SERIAL\_NUMBER of node ".123.8437". Messages exchanged include a Message Authentication Code.

## ANSI C12.22-2008

Offset partial read request

```
60 37
  A2 05
    80 03 7B C1 75
  A6 04
    80 02 7B 04
  A8 03
    02 01 09
  AC 0F
    A2 0D
      A0 0B
        A1 09
          80 01 02
          81 04 48 F3 C6 07
  BE 12
    28 10
      81 0E
        84
        08
        3F
        00 01
        00 00 10
        00 10
        F0 F6 4B C4
```

```
<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.8437
<calling-AP-title-element>
<calling-AP-title> .123.4
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 9
<calling-authentication-value-element>
<calling-authentication-value-external>
<calling-authentication-value-single-asn1>
<calling-authentication-value-cl222>
<key-id-element> = 2
<iv-element> = 2008-10-13 22:04:55 UTC
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 1
<service-length>
<pread-offset>
<tableid> = 1
<offset> = 16
<octet-count> = 16
<mac>
Network context = 2.16.124.113620.1.22.0
K = 08070605040302010807060504030201
L = B86D06FD5929DFAAF6BE44CED8939EC3
D = 70DA0DFAB253BF55ED7C899DB1273D01
Q = E1B41BF564A77EABDAF9133B624E7A02
N = A20D060B607C86F7540116007BC175A8
    03020109AC0FA20DA00BA10980010281
    0448F3C607BE122810810E84A60C060A
    607C86F7540116007B040248F3C60708
    3F000100001000108000000000000000
Tag = F0F64BC4DB64D0212F986D44207904F8
T = F0F64BC4
```

Offset partial read response

```
60 48
  A2 04
    80 02 7B 04
  A4 03
    02 01 09
  A6 05
    80 03 7B C1 75
  A8 03
    02 01 09
  AC 0F
    A2 0D
      A0 0B
        A1 09
          80 01 02
          81 04 48 F3 C6 06
  BE 1E
    28 1C
      81 1A
        84
        14
        00
        00 10
        4D 41 4E 55 46 41 43 54 55 52 45 52 20
        92
        45 92 25 55
```

```
<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.4
<called-AP-invocation-id-element>
<called-AP-invocation-id> = 9
<calling-AP-title-element>
<calling-AP-title> = .123.8437
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 9
<calling-authentication-value-element>
<calling-authentication-value-external>
<calling-authentication-value-single-asn1>
<calling-authentication-value-cl222>
<key-id-element> = 2
<iv-element> = 2008-10-13 22:04:54 UTC
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 1
<service-length>
<ok>
<count>
53 4E 20 "MANUFACTURER SN"
<cksum>
<mac>
N = A20C060A607C86F7540116007B04A403
    020109A803020109AC0FA20DA00BA109
    800102810448F3C606BE1E281C811A84
    A60D060B607C86F7540116007BC17502
    48F3C606140000104D414E5546414354
    5552455220534E209280000000000000
Tag = 4592255508295586868105DA6EBEA0F5
T = 45922555
```

## Example #6: Authenticated notification



The following example represents an unacknowledged notification of Procedure 26 initiated by node .123.273 and targeted to node .123.2. This notification includes a Message Authentication Code.

```

Write request
60 43
  A2 04
    80 02 7B 02
  A6 05
    80 03 7B 82 11
  A7 03
    02 01 04
  A8 03
    02 01 0C
  AC 0F
    A2 0D
      A0 0B
        A1 09
          80 01 02
          81 04 48 F3 C9 E5
    BE 19
      28 17
        81 15
          96

      54 45 4D 50
      0B
      40
      00 07
      00 05
      1A 00
      00
      02 00
      E4
      D7 A4 84 41

<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.2
<calling-AP-title-element>
<calling-AP-title> = .123.273
<calling-AE-qualifier-element>
NOTIFICATION = TRUE
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 12
<calling-authentication-value-element>
<calling-authentication-value-external>
<calling-authentication-value-single-asn1>
<calling-authentication-value-cl222>
<key-id-element> = 2
<iv-element> = 2008-10-13 22:21:25 UTC
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> ED_CLASS_INCLUDED = true,
SECURITY_MODE = 1,
RESPONSE_CONTROL = 2

<ed-class>
<service-length>
<full-write>
<tableid> = 7 (In little endian)
<count> = 5
PROC = 26
SEQ_NBR = 0
PARM = 2 (Primary power up event code)
<cksum>
<mac>
Network context = 2.16.124.113620.1.22.0
K = 08070605040302010807060504030201
L = B86D06FD5929DFAAF6BE44CED8939EC3
D = 70DA0DFAB253BF55ED7C899DB1273D01
Q = E1B41BF564A77EABDAF9133B624E7A02
N = A20C060A607C86F7540116007B02A703
020104A80302010CAC0FA20DA00BA109
800102810448F3C9E5BE192817811596
A60D060B607C86F7540116007B821102
48F3C9E554454D500B40000700051A00
000200E4800000000000000000000000
Tag = D7A484411EB10A789868F4B2FFF1C3E1
T = D7A48441

```

### Example #7: Encrypted session

The following example represents a session initiated by node “.123.4” to read Table 05 of node “.123.8437”. Messages exchanges include a Message Authentication Code and the application payload is encrypted for privacy.

```

Logon request
60 3E
  A2 05
    80 03 7B C1 75
  A6 04
    80 02 7B 04
  A8 03
    02 01 05
  AC 0F
    A2 0D
      A0 0B
        A1 09
          80 01 02
          81 04 48 F3 CA BD
    BE 19
      28 17
        81 15

<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.8437
<calling-AP-title-element>
<calling-AP-title> .123.4
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 5
<calling-authentication-value-element>
<calling-authentication-value-external>
<calling-authentication-value-single-asn1>
<calling-authentication-value-cl222>
<key-id-element> = 2
<iv-element> = 2008-10-13 22:25:01 UTC
<user-information-element>
<user-information-external>
<user-information-octet-string>

```

# ANSI C12.22-2008

```

88
D0 60 62 64 AE B9 18 1D 92 2C EE 71
56 01 85 63
68 77 A1 57

```

Logon response

```

60 37
  A2 04
    80 02 7B 04
  A4 03
    02 01 05
  A6 05
    80 03 7B C1 75
  A8 03
    02 01 00
  AC 0F
    A2 0D
      A0 0B
        A1 09
          80 01 02
            81 04 48 F3 CA BC
  BE 0D
    28 0B
      81 09
        88
        9E DF 3C F4
        C0 CE B3 04

```

Security request

```

60 33
  A2 05
    80 03 7B C1 75
  A6 04
    80 02 7B 04
  A8 03
    02 01 00
  BE 1F
    28 1D
      81 1B
        88
        EC B8 17 16 5A 0D BF C3 CE 0C 42 83
        E3 14 25 55 2D D2 7D 98 08 4D

```

```

<epsem-control> SECURITY_MODE = 2
Encrypted payload
Encrypted payload
<mac>
Network context = 2.16.124.113620.1.22.0
K = 08070605040302010807060504030201
L = B86D06FD5929DFAAF6BE44CED8939EC3
D = 70DA0DFAB253BF55ED7C899DB1273D01
Q = E1B41BF564A77EABDAF9133B624E7A02
N = A20D060B607C86F7540116007BC175A8
    03020105AC0FA20DA00BA10980010281
    0448F3CABDBE192817811588A60C060A
    607C86F7540116007B040248F3CABD80
Tag = 36E55BDFD84A6FD88B41B9214849BBFC
C = C4A76212A94BA5379C6AB57386FC055B
Tag = 5E92FA8811F8AB1F30DC1B3BC17874C5
T = 6877A157

```

when decrypted:

```

0F <service-length>
50 <logon>
0002 <user-id>
55534552204E414D4520 <user>
003C <req-session-idle-timeout>

```

```

<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.4
<called-AP-invocation-id-element>
<called-AP-invocation-id> = 5
<calling-AP-title-element>
<calling-AP-title> = .123.8437
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 0
<calling-authentication-value-element>
<calling-authentication-value-external>
<calling-authentication-value-single-asn1>
<calling-authentication-value-cl222>
<key-id-element> = 2
<iv-element> = 2008-10-13 22:25:00 UTC
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 2
Encrypted payload
<mac>
N = A20C060A607C86F7540116007B04A403
    020105A803020100AC0FA20DA00BA109
    800102810448F3CABCE0D280B810988
    A60D060B607C86F7540116007BC17502
    48F3CABC800000000000000000000000
Tag = DA739FA27808328FCF372DB1AFB302B4
C = 9EDF3CF4800000000000000000000000
Tag = 1ABD2CA62B64080B8CDAB4E8AB3932C
T = C0CEB304

```

when decrypted:

```

03 <service-length>
00 <ok>
003C <resp-session-idle-timeout>

```

6F C2 04 83

# Security response

```
60 24
  A2 04
    80 02 7B 04
  A4 03
    02 01 00
  A6 05
    80 03 7B C1 75
  A8 03
    02 01 01
  BE 0B
    28 09
      81 07
        88
        CB 22
        2B 67 76 D8
```

# Read request

```
60 21
  A2 05
    80 03 7B C1 75
  A6 04
    80 02 7B 04
  A8 03
    02 01 01
  BE 0D
    28 0B
      81 09
        88
        81 5A 8C 8B
        0E 51 DC 15
```

# Read response

```
60 3B
  A2 04
    80 02 7B 04
  A4 03
```

```
<mac>
N = A20D060B607C86F7540116007BC175A8
    03020100BE1F281D811B88A60C060A60
    7C86F7540116007B040248F3CABC8000
Tag = 51B5921B580ABBFDA91F34EC11B7F12C
C = ECB817165A0DBFC3CE0C4283E3142555
    2DD27D98084D80000000000000000000
Tag = 3E7796982B5EADA775FE1E017F79F74C
T = 6FC20483
```

```
when decrypted:
15 <service-length>
51 <security>
50415353574F52442020202020202020202020
```

```
<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.8437
<called-AP-invocation-id-element>
<called-AP-invocation-id> = 0
<calling-AP-title-element>
<calling-AP-title> = .123.8437
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 1
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 2
Encrypted payload
<mac>
N = A20C060A607C86F7540116007B04A403
    020100A803020101BE0B2809810788A6
    0D060B607C86F7540116007BC1750248
    F3CABD80000000000000000000000000
Tag = 90579F871DB72E6BBC9EB60A3532BBAA
C = CB22800000000000000000000000000000
Tag = BB30E95F51346D9790AFC5DEFEB6A44
T = 2B6776D8
```

```
when decrypted:
01 <service-length>
00 <OK>
```

```
<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.8437
<calling-AP-title-element>
<calling-AP-title> .123.4
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 1
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 2
Encrypted payload
<mac>
N = A20D060B607C86F7540116007BC175A8
    03020101BE0D280B810988A60C060A60
    7C86F7540116007B040248F3CABC8000
Tag = CC45A38E02132F56E7627F7473DFFA54
C = 815A8C8B80000000000000000000000000
Tag = C2147F9B3EA9C327694BB691BA8D5361
T = 0E51DC15
```

```
<epsem>, when decrypted:
03 <service-length>
30 <full-read>
0005 <table-id>
```

```
<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.4
<called-AP-invocation-id-element>
```

# ANSI C12.22-2008

```

02 01 01
A6 05
80 03 7B C1 75
A8 03
02 01 02
BE 22
28 20
81 1E
88
8A E2 91 7E 92 A9 77 E4 A2 B6 3E 55
23 0A B7 B4 4B 0A 70 21 07 63 6B E4 5F
DB 4C 2D B4

<called-AP-invocation-id> = 1
<calling-AP-title-element>
<calling-AP-title> = .123.8437
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 2
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 2
Encrypted payload
Encrypted payload
<mac>
N = A20C060A607C86F7540116007B04A403
020101A803020102BE222820811E88A6
0D060B607C86F7540116007BC1750248
F3CABD80000000000000000000000000
Tag = 6F5AD2095A8AE1B5861F53A39855678E
C = 8AE2917E92A977E4A2B63E55230AB7B4
4B0A702107636BE45F80000000000000
Tag = B416FFBD91B8148A57C38075E4979FCA
T = DB4C2DB4

<epsem>, when decrypted:
18 <service-length>
00 <ok>
0014 <count>
444556494345204944202020202020202020
43 <cksum>

Logoff request
60 1F
A2 05
80 03 7B C1 75
A6 04
80 02 7B 04
A8 03
02 01 02
BE 0B
28 09
81 07
88
C9 3B
B9 E9 74 72

<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.8437
<calling-AP-title-element>
<calling-AP-title> .123.4
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 2
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 2
Encrypted payload
<mac>
N = A20D060B607C86F7540116007BC175A8
03020102BE0B2809810788A60C060A60
7C86F7540116007B040248F3CABC8000
Tag = 0ECDB72AF9DD68EAD130E155E458D200
C = C93B8000000000000000000000000000
Tag = B724C3581DE04C68EFA2C394D09965A1
T = B9E97472

<epsem>, when decrypted:
01 <service-length>
52 <logoff>

Logoff response
60 24
A2 04
80 02 7B 04
A4 03
02 01 02
A6 05
80 03 7B C1 75
A8 03
02 01 03
BE 0B
28 09
81 07
88
14 54
1C 78 B2 F0

<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.4
<called-AP-invocation-id-element>
<called-AP-invocation-id> = 2
<calling-AP-title-element>
<calling-AP-title> = .123.8437
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 3
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 2
Encrypted payload
<mac>
N = A20C060A607C86F7540116007B04A403
020102A803020103BE0B2809810788A6
0D060B607C86F7540116007BC1750248
F3CABD80000000000000000000000000
Tag = 3EDD871457BA6B65E498C6D1B48FC4F7

```

```
C =      14548000000000000000000000000000  
Tag =    22A535E44C89F56DE35CA7E5F975BC97  
T =     1C78B2F0
```

```
<epsem>, when decrypted:
01      <service-length>
00      <ok>
```

### Example #8: Encrypted sessionless

The following example represents a sessionless transaction initiated by “.123.4” to read field MFG\_SERIAL\_NUMBER of node “.123.8437”. Messages exchanged include a Message Authentication Code and the application payload is encrypted for privacy.

Offset partial read request

```

60 4F
    A2 05
        80 03 7B C1 75
    A6 04
        80 02 7B 04
    A8 03
        02 01 03
    AC 0F
        A2 0D
            A0 0B
                A1 09
                    80 01 02
                    81 04 48 F3 D0 61
    BE 2A
        28 28
            81 26
                88
                D0 71 80 54 4A FA BF 05 0B 09 32 AA
                40 A8 38 2E C5 72 C8 C2 F1 BE 5A 6E
                1A 1E 6A 06 27 5F C4 BA C3
                20 D9 92 C2

```

```
<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.8437
<calling-AP-title-element>
<calling-AP-title> .123.4
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 3
<calling-authentication-value-element>
<calling-authentication-value-external>
<calling-authentication-value-single-asn1>
<calling-authentication-value-c1222>
<key-id-element> = 2
<iv-element> = 2008-10-13 22:49:05 UTC
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 2
Encrypted payload
Encrypted payload
Encrypted payload
<mac>
Network context = 2.16.124.113620.1.22.0
K = 08070605040302010807060504030201
L = 8B6D06FD5929DFAAF6BE44CED8939EC3
D = 70DA0DFAB253BF55ED7C899DB1273D01
Q = E1B41BF564A77EABDAF9133B624E7A02
N = A20D060B607C86F7540116007BC175A8
03020103AC0FA20DA00BA10980010281
0448F3D061BE2A282812688A60C060A
607C86F7540116007B040248F3D06180
Tag = 1C1459EA80D87BFCA7ED47A1C9DE60BB
C = D07180544AFABF050B0932AA40A8382E
C572C8C2F1BE5A6E1A1E6A06275FC4BA
C3800000000000000000000000000000
Tag = 3CCDCB2844D4C8801395C010956ECFFE
T = 20D992C2
```

```
when decrypted:
17      <service-length>
51      <security>
50415353574F5244202020202020202020202020202020
      <password> = "PASSWORD"
0002    <user-id>
08      <service-length>
3F      <pread-offset>
0001    <tableid>
000010  <offset>
0010    <octet-count>
```

Offset partial read response

```

60 48
    A2 04
        80 02 7B 04
    A4 03
        02 01 03
    A6 05
        80 03 7B C1 75
    A8 03

```

```
<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.4
<called-AP-invocation-id-element>
<called-AP-invocation-id> = 3
<calling-AP-title-element>
<calling-AP-title> = .123.8437
<calling-AP-invocation-id-element>
```

```

02 01 03
AC 0F
  A2 0D
    A0 0B
      A1 09
        80 01 02
        81 04 48 F3 D0 60
BE 1E
  28 1C
    81 1A
      88
      18 43 71 B3 7C BD 85 44 B1 EA 91 05
      30 20 EC 99 AB 5E 63 72 27
      F0 F1 02 C4

<calling-AP-invocation-id> = 3
<calling-authentication-value-element>
<calling-authentication-value-external>
<calling-authentication-value-single-asn1>
<calling-authentication-value-cl222>
<key-id-element> = 2
<iv-element> = 2008-10-13 22:49:04 UTC
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> SECURITY_MODE = 2
Encrypted payload
Encrypted payload
<mac>
N = A20C060A607C86F7540116007B04A403
    020103A803020103AC0FA20DA00BA109
    800102810448F3D060BE1E281C811A88
    A60D060B607C86F7540116007BC17502
    48F3D060800000000000000000000000
Tag = 39531F4567DD7B9384DA8459E1233334
C = 184371B37CBD8544B1EA91053020EC99
    AB5E6372278000000000000000000000
Tag = C9A21D8169E59A96BFACC0A3529E0A00
T = F0F102C4

when decrypted:
14 <service-length>
00 <ok>
0010 <count>
4D414E55464143545552455220534E20 <data>
92 <cksum>

```

### Example #9: Encrypted notification

The following example represents an unacknowledged notification of Procedure 26 initiated by node .123.273 and targeted to node .123.2. This notification includes a Message Authentication Code and the application payload is encrypted for privacy.

```

60 43
  A2 04
    80 02 7B 02
  A6 05
    80 03 7B 82 11
  A7 03
    02 01 04
  A8 03
    02 01 02
AC 0F
  A2 0D
    A0 0B
      A1 09
        80 01 02
        81 04 48 F3 D2 F8
BE 19
  28 17
    81 15
      9A
      34 B7 27 6F 54 06 D2 5D 4E 3A 51 73
      1D 88 A5 D9
      1B D7 8F 32

<acse-pdu>
<called-AP-title-element>
<called-AP-title> = .123.2
<calling-AP-title-element>
<calling-AP-title> = .123.273
<calling-AE-qualifier-element>
NOTIFICATION = TRUE
<calling-AP-invocation-id-element>
<calling-AP-invocation-id> = 2
<calling-authentication-value-element>
<calling-authentication-value-external>
<calling-authentication-value-single-asn1>
<calling-authentication-value-cl222>
<key-id-element> = 2
<iv-element> = 2008-10-13 23:00:08 UTC
<user-information-element>
<user-information-external>
<user-information-octet-string>
<epsem-control> ED_CLASS_INCLUDED = true,
    SECURITY_MODE = 2,
    RESPONSE_CONTROL = 2
Encrypted payload
Encrypted payload
<mac>
Network context = 2.16.124.113620.1.22.0
K = 08070605040302010807060504030201
L = B86D06FD5929DFAAF6BE44CED8939EC3
D = 70DA0DFAB253BF55ED7C899DB1273D01
Q = E1B41BF564A77EABDAF9133B624E7A02
N = A20C060A607C86F7540116007B02A703
    020104A803020102AC0FA20DA00BA109
    800102810448F3D2F8BE19281781159A
    A60D060B607C86F7540116007B821102
    48F3D2F8800000000000000000000000
Tag = E61DF9AF96D333E8962A0D4E394BBF0F

```

```
C = 34B7276F5406D25D4E3A51731D88A5D9
Tag = FDCA769DFAEC95AAEA3F8C396E0936E9
T = 1BD78F32
```

```
when decrypted:
54454D50      <ed-class>
0B            <service-length>
40            <full-write>
0007          <tableid>
0005          <count>
1A00          PROC
00            SEQ_NBR
0200          PARM =
E4            <cksum>
```

## Annex H - CRC Examples

(informative)

### H.1 Trace

This example shows the actual bit manipulations performed in calculation of the frame check sequence (CRC) for an example PSEM packet in compliance with ANSI C12.18 and this document. In order to be in compliance with the CRC calculations, the bit ordering shall be consistent with this example.

```

packet without crc      = ee 00 00 00 00 01 20
                          = 11101110 00000000 00000000 00000000 00000000 00000001 00100000 00000000 00000000
LSBit First            01110111 00000000 00000000 00000000 00000000 10000000 00000100 00000000 00000000
XOR FF                 11111111 11111111

Start Tx               = 10001000 11111111 00000000 00000000 00000000 10000000 00000100 00000000 00000000
Apply P(x)             10001000 00010000 1
                          00000000 11101111 10000000 00000000 00000000 10000000 00000100 00000000 00000000
Apply P(x)             10001000 00010000 1
                          00000000 01100111 10010000 10000000 00000000 10000000 00000100 00000000 00000000
Apply P(x)             1000100 00001000 01
                          00000000 00100011 10011000 11000000 00000000 10000000 00000100 00000000 00000000
Apply P(x)             100010 00000100 001
                          00000000 00000001 10011100 11100000 00000000 10000000 00000100 00000000 00000000
Apply P(x)             1 00010000 00100001
                          00000000 00000000 10001100 11000001 00000000 10000000 00000100 00000000 00000000
Apply P(x)             10001000 00010000 1
                          00000000 00000000 00000100 11010001 10000000 10000000 00000100 00000000 00000000
Apply P(x)             100 01000000 100001
                          00000000 00000000 10010001 00000100 10000000 00000100 00000000 00000000 00000000
Apply P(x)             10001000 00010000 1
                          00000000 00000000 00011001 00010100 10000000 00000100 00000000 00000000 00000000
Apply P(x)             10001 00000010 0001
                          00000000 00000000 00001000 00010110 10000000 00000100 00000000 00000000 00000000
Apply P(x)             1000 10000001 00001
                          00000000 00000000 00000000 10010111 10000000 00000100 00000000 00000000 00000000
Apply P(x)             100010111 10000000 00000100 00000000 00000000 10001
                          00000000 00000000 00000000 00001110 00001010 10010100 00000000 00000000 00000000
Apply P(x)             1000 10000001 00001
                          00000000 00000000 00000000 00000110 10001011 10011100 00000000 00000000 00000000
Apply P(x)             100 01000000 100001
                          00000000 00000000 00000000 00000010 11001011 00011000 00000000 00000000 00000000
Apply P(x)             10 00100000 0100001
                          00000000 00000000 00000000 00000000 11101011 01011010 00000000 00000000 00000000
Apply P(x)             10001000 00010000 1
                          00000000 00000000 00000000 00000000 01100011 01001010 10000000 00000000 00000000
Apply P(x)             1000100 00001000 01
                          00000000 00000000 00000000 00000000 00100111 01000010 11000000 00000000 00000000
Apply P(x)             100010 00000100 001
                          00000000 00000000 00000000 00000000 00000101 01000110 11100000 00000000 00000000
Apply P(x)             100 01000000 100001
                          00000000 00000000 00000000 00000000 00000001 00000110 01100100 00000000 00000000
Apply P(x)             1 00010000 00100001
                          00000000 00000000 00000000 00000000 00000000 00010110 01000101 00000000 00000000
Apply P(x)             10001 00000010 0001
                          00000000 00000000 00000000 00000000 00000111 01000111 00010000 00000000 00000000
Apply P(x)             100 01000000 100001
                          00000000 00000000 00000000 00000000 00000000 00000011 00000111 10010100 00000000
Apply P(x)             10 00100000 0100001
                          00000000 00000000 00000000 00000000 00000000 00000001 00100111 11010110 00000000
Apply P(x)             1 00010000 00100001
                          00000000 00000000 00000000 00000000 00000000 00110111 11110111 11001000 00001000
XOR FF                 11001000 00001000
MSBit First            00010011 00010000
crc                    = 1310

```



## H.2 CRC Code Example

The following is an example of C code which calculates the <crc> field in a manner compliant with C12.22 Layer 2 and Layer 7. This code is provided as an example only.

```
#include <stdio.h>

unsigned short crc16(unsigned char octet, unsigned short crc);
unsigned short crc(int size, unsigned char *packet);

unsigned short crc16(unsigned char octet, unsigned short crc)
{
    int i;

    for (i = 8; i; i--)
    {
        if (crc & 0x0001)
        {
            crc >>= 1;
            if (octet & 0x01)
                crc |= 0x8000;
            crc = crc ^ 0x8408; /* 0x1021 inverted = 1000 0100 0000 1000 */
            octet >>= 1;
        }
        else
        {
            crc >>= 1;
            if (octet & 0x01)
                crc |= 0x8000;
            octet >>= 1;
        }
    }

    return crc;
}

unsigned short crc(int size, unsigned char *packet)
{
    int i;
    unsigned short crc;

    crc = (~packet[1] << 8) | (~packet[0] & 0xFF);

    for (i=2; i<size; i++)
        crc = crc16(packet[i], crc);

    crc = crc16(0x00, crc);
    crc = crc16(0x00, crc);
    crc = ~crc;

    crc = crc >> 8 | crc << 8; /* SWAP the bytes for Little Endian presentation per HDLC */

    return crc;
}

int main()
{
    unsigned char packet[] = { 0xEE, 0x00, 0x00, 0x00, 0x00, 0x01, 0x20 };

    printf("Crc    = %04x [expected 0x1310]\n", crc(sizeof(packet), packet));
    return 0;
}
```

## Annex I - The EAX' Cryptographic Mode

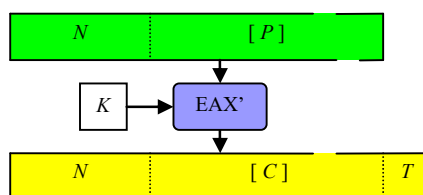
(normative)

Acknowledgment: This EAX' description is derived from [EAX MO 2004]. Its presentation is based on the algorithms specified in Figures 1 - 3 of that paper, recast into a syntax appropriate to this Standard. The authors of the referenced EAX papers [EAX 2003, EAX MO 2004], generously granted permission to use those figures. While the material below is not a direct reproduction of those figures, they are clearly the motivation for this presentation.

### I.1 EAX' description

Acknowledgment: This EAX' description is derived from [EAX MO 2004]. Its presentation is based on the algorithms specified in Figures 1 - 3 of that paper, recast into a syntax appropriate to this Standard. The authors of the referenced EAX papers [EAX 2003, EAX MO 2004], generously granted permission to use those figures. While the material below is not a direct reproduction of those figures, they are clearly the motivation for this presentation.

EAX' takes as input a Cleartext ( $N$ ) and an optional Plaintext ( $P$ ) and produce as output a Message Authentication Code ( $T$ ) and a corresponding optional Ciphertext ( $C$ ).



Each component of the message is defined as follows:

- $N$  : Cleartext, part of the ANSI C12.22 Message that is authenticated but not encrypted for secrecy.
- $P$  : Plaintext, part of the ANSI C12.22 Message which is encrypted for secrecy. When used for message secrecy, the Plaintext is composed of the <epsem-data> and the optional <padding>.
- $C$  : Ciphertext, part of the ANSI C12.22 Message after encryption for secrecy, corresponding directly to the plain text  $P$ .
- $T$  : Four bytes MAC (Message Authentication Code) inserted at the end of the ANSI C12.22 Message.

NOTE To provide interoperable security it is absolutely critical that  $N$ ,  $P$ ,  $C$ , and  $T$  are generated and processed in a consistent fashion. The C12.22 standard allows substantial flexibility in message transmission. Section 错误!未找到引用源。 of this standard provides specific details regarding how a C12.22 Message is canonified into a consistent standard form for cryptographic processing.  $N$  and  $P$  must be generated according to these rules.  $C$  and  $T$  are generated based on the algorithms below.

The cryptographic key is defined as follows:

- $K$  : Cryptographic secret key. Keys are managed through Tables 46 & 47. The key length is sized appropriate to the block cipher. The specific length in use is defined by the **CIPHER\_CODE** in use.

Within C12.22, the AES block cipher is used. The EAX' mode supports using alternate block ciphers. AES has a block size of 16 bytes (128 bits). EAX' is described in a series of algorithms, where these

algorithms make use of the following operators:

$X \leftarrow Y$	Y assigned to X
$X \text{ XOR } Y$	Bitwise exclusive-or of X and Y
$X \text{ XOR}_{\text{end}} Y$	Bitwise exclusive-or (XOR) of Y after alignment with the end of string X
$\text{size}(X)$	Length in bits of X
$\text{AES}_K(X)$	Encryption of X using AES with key K
$X \parallel Y$	The concatenation of strings X and Y
$X[\text{first } m \text{ bits}]$	Most significant m bits of X
$X \ll 1$	Left shift of X by one position (i.e., with a carry-in of 0)
$\text{ceiling}(X)$	Ceiling function, which returns the smallest integer not less than X
$n$	The block size in bits of the underlying block cipher (for AES, $n = 128$ )
$0^m$	m bits of 0. (i.e. $0^4 10^3$ would be 00001000)
$1^m$	m bits of 1. (i.e. $1^4 01^2 0$ would be 11110110)

**Algorithm**  $\text{dbl}(X)$  -- modulo the first minimal weight irreducible polynomial of degree n

10 **define**  $r_b$  **as**  $0^{120}10000111$  -- for  $n = 128$ , the Cipher block size (in bits)

11 **if**  $\text{msb}(X) = 0$  -- most significant bit

12 **then return**  $X \ll 1$

13 **else return**  $(X \ll 1) \text{ XOR } r_b$

**Algorithm**  $\text{deriveKeyDependentConstants}(K)$

20  $L \leftarrow \text{AES}_K(0^n)$  -- For the AES block cipher this is  $0^{128}$

21  $D \leftarrow \text{dbl}(L)$  --  $D$  is double the value of  $L$  in an appropriate field, as defined by  $r_b$

22  $Q \leftarrow \text{dbl}(D)$  --  $Q$  is quadruple the value of  $L$  --  $\text{dbl}(\text{dbl}(L))$  -- in the same field

23 **return**  $D$  and  $Q$

**Algorithm**  $\text{pad}(S, D, Q)$

30 **if**  $(\text{size}(S) > 0)$  and  $(\text{size}(S) \bmod n = 0)$  --  $n$  equals Cipher block size (in bits)

31 **then return**  $S \text{ XOR}_{\text{end}} D$  -- XORs into end of string

32 **else return**  $(S \parallel 10^{n-1-(\text{size}(S) \bmod n)}) \text{ XOR}_{\text{end}} Q$  -- XORs into end of string; applies to null string

**Algorithm**  $\text{CBC}'_K(t, S)$

40 Let  $S_1 \dots S_m \leftarrow S$  where  $\text{size}(S_i) = n$  --  $n$  equals Cipher block size (in bits)

41  $C_0 \leftarrow t$

42 **for**  $i \leftarrow 1$  **to**  $m$

43 **do**  $C_i \leftarrow \text{AES}_K(S_i \text{ XOR } C_{i-1})$

44 **return**  $C_m$

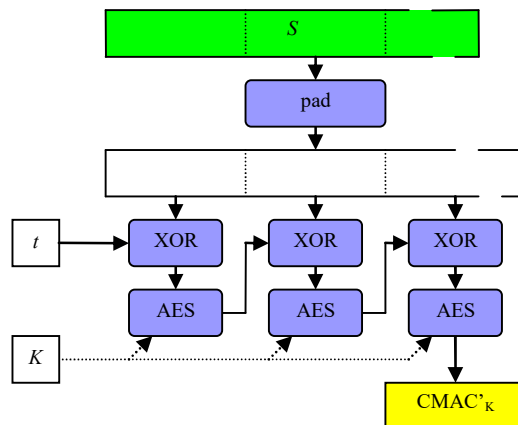
The following definition of CMAC' is derived from NIST SP 800-38B,  
[http://csrc.nist.gov/publications/nistpubs/800-38B/SP\\_800-38B.pdf](http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf), titled

**Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication**

It has been recast into nomenclature similar to that used in the EAX papers and the syntax used in this standard.

**Algorithm** CMAC' $_K(t, S, D, Q)$

50 **return** CBC' $_K(t, \text{pad}(S, D, Q))$



The following definition of CTR chaining mode is derived from NIST SP 800-38A,  
<http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf>, titled

**Recommendation for Block Cipher Modes of Operation: Methods and Techniques**

It has been recast into nomenclature similar to that used in the EAX papers and the syntax used in this standard.

**Algorithm** CTR' $_K(N, S)$

60  $m \leftarrow \text{ceiling}(\text{size}(S)/n)$

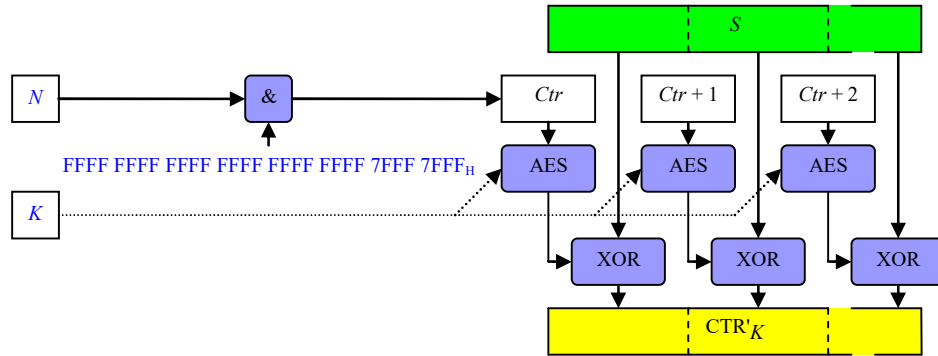
61  $\text{Ctr} \leftarrow N \ \& \ 1^{n-32} 01^{15} 01^{15}$  -- Optimization, to avoid inter-word carries.

62 -- For AES,  $n=128$  and  $1^{n-32} 01^{15} 01^{15}$  resolves to

63 -- FFFF FFFF FFFF FFFF FFFF FFFF 7FFF 7FFF<sub>H</sub>

64  $\text{Pad} \leftarrow \text{AES}_K(\text{Ctr}) \parallel \text{AES}_K(\text{Ctr}+1) \parallel \dots \parallel \text{AES}_K(\text{Ctr}+m-1)$

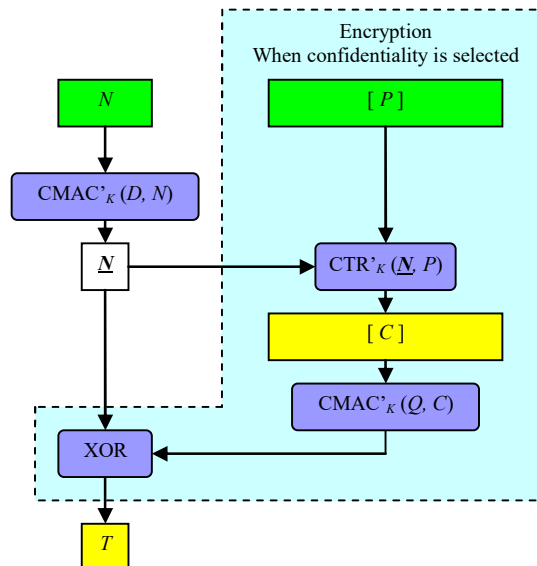
65 **return**  $S \text{ XOR } \text{Pad}$  [first  $\text{size}(P)$  bits]


**Algorithm EAX'.Encrypt $_K(N, P)$** 

```

70 deriveKeyDependentConstants( $K$ ) -- need be done only once per new key value
71  $Tag \leftarrow \underline{N} \leftarrow \text{CMAC}_K(D, N, D, Q)$  -- EAX' optimization, first  $D$  used instead of  $E_K(0^n)$ 
72 if  $\text{size}(P) > 0$  -- EAX' optimization, no contribution of a key-dependent constant for an empty  $C$ 
73 then  $C \leftarrow \text{nullString}$ 
74 else
75    $C \leftarrow \text{CTR}'_K(\underline{N}, P)$ 
76    $Tag \leftarrow Tag \text{ XOR } \text{CMAC}_K(Q, C, D, Q)$  -- EAX' optimization, first  $Q$  used instead of  $E_K(0^{n-210})$ 
77  $T \leftarrow Tag$  [first 32 bits]
78 return  $C$  and  $T$ 

```

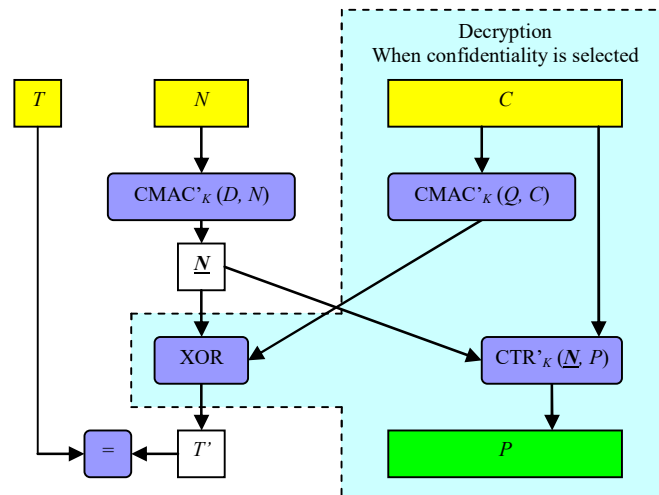


**Algorithm EAX'.Decrypt<sub>K</sub>(N, C, T)**

```

80 deriveKeyDependentConstants(K) -- need be done only once per new key value
81 Tag ←  $\underline{N}$  ← CMACK(D, N, D, Q) -- EAX' optimization, first D used instead of EK(0n)
82 if size(C) = 0 then -- EAX' optimization, no contribution of a key-dependent constant for an empty C
83   P ← nullString
84   if T ≠ Tag [first 32 bits] then return INVALID else return P -- message is VALID
85 else
86   Tag ← Tag XOR CMACK(Q, C, D, Q) -- EAX' optimization, first Q used instead of EK(0n-210)
87   if T ≠ Tag [first 32 bits] then return INVALID
88   P ← CTR'K( $\underline{N}$ , C) -- decrypt Ciphertext only when tags match
89 return P -- message is VALID

```

**I.2 Justifications for selection of EAX rather than CCM**

CCM is a NIST-standardized Authenticated Encryption with Associated Data (AEAD) cryptographic mode developed for IEEE 802.11, now also used by IEEE 802.15.4. CCM has five functional restrictions (which EAX lacks) and one common implementation restriction that make it less appropriate for use in ANSI C12.22:

1. **Nonce length:** CCM has a fixed maximum nonce input size of 13 bytes. The ANSI C12.22 Message format requires use of ApTitles and other information that may exceed 50 bytes, all of which must be included in the nonce in a manner unpredictable to an attacker. EAX directly processes nonces of arbitrary and variable size.
2. **Nonce unpredictability:** CCM generates nonces that are predictable to an attacker, making it trivial for an attacker to determine that two distinct messages under the same key are using the same nonce value. This makes it imperative that the nonce construction never duplicate a nonce value, even after the compression step that would be required to reduce the large C12.22 nonce

inputs to the required CCM 13-byte nonce. EAX computes a nonce value that is unpredictable to an attacker, in a range of  $2^{128}$  nonce values, so no special protection need be taken against duplicate nonce values (provided that the inputs to the nonce computation do not duplicate).

3. **"Online" capability:** CCM requires that the entire message be available before authentication and plaintext encryption can begin, requiring that an entire message be buffered before CCM can be applied. ANSI C12.22 can have message sizes of many thousand bytes., with maximum sizes exceeding 60,000 bytes. Together these features make it difficult to use add-on hardware or software for existing ANSI C12.22 implementations, to minimize time to initial deployment of the new standard. EAX has "online" capability, which makes it suitable for use in add-on hardware or software serial link encryptors.
4. **Authenticate-before-decrypt:** CCM requires that the entire received message be decrypted before it can be authenticated. Thus the extra processing required to decrypt Ciphertext must be applied whether the received message authenticates or not. EAX authenticates the Ciphertext rather than the plaintext, so this extra processing is required only for messages that have been authenticated. This reduces the ability of an attacker to mount a DoS (denial of service) attack on the receiving device, causing it to consume significant resources before determining that the received message is invalid.
5. **Block cipher independence:** CCM alone, of all recognized block cipher modes, is not a procedure applied to a generic underlying block cipher; CCM is defined only for AES-128. A standard with a potentially long life and international usage should remain applicable even when events such as a successful mode of attack on the selected block cipher, or national regulations, require use of a different underlying block cipher. EAX has this capability.
6. **Hardware non-support of canonicalized messages:** Some modem chips such as many of those for IEEE 802.15.4 provide hardware support for CCM-mode transmission and reception when the message to be CCM-authenticated consists exactly of the message Cleartext and plaintext, in that order, that is to be sent or that was received. However, ANSI C12.22 requires that the authenticated message use canonical forms of ApTitles, and exclude the <sending-ApTitle> when it was added by a Relay. The resultant software-based sequencing of invocations of the underlying block cipher (AES-128) is equivalent to that required for EAX.

### I.3 Justifications for the EAX' Optimizations

This section provides justifications for the simplifications of EAX' over the basic EAX mode specified by Bellare, Rogaway and Wagner: These simplifications are motivated by a desire to reduce the number of AES block encryptions required by EAX, and to reduce the amount of per-key-related storage needed by a time-optimized implementation, without weakening the cryptographic strength and resistance to attack that EAX offers. The EAX' optimizations reduce the required per-key storage for a time-optimized implementation from those of unmodified EAX by  $K+4B+2T$  bytes per key (88 bytes when AES128 is the block cipher) to  $K+2B$  bytes per key (48 bytes when AES128 is the block cipher), where  $K$  is the key size of the underlying block cipher,  $B$  is the block size of that block cipher, and  $T$  is the desired authentication tag size, all in bytes. For space-optimized implementations, where only  $K$  bytes of storage per key are required, these optimizations eliminate either three extra invocations of AES per message when <epsem-data> secrecy is used, or five extra invocations when it is not used.

**BACKGROUND:** Block ciphers such as AES-128 are keyed pseudo-random permutations (PRPs), which map a block-size input (e.g., 16 bytes = 128 bits) to a block-size output. The strength and ability to resist cryptanalysis of composite cryptographic modes which use this block cipher are directly related to the strength and ability to resist cryptanalysis of the underlying PRP. Analysis of a new mode must examine its impact on increasing or decreasing this strength, as well as any avenues of cryptanalytic attack that such a new mode may expose.

Consideration of the security proof of EAX, which is published in [EAX MO 2004], indicates that the proof can be modified to account for the following optimizations. Thus the security properties of EAX' are equivalent to those of EAX. Here is the list of ways that EAX' differs from EAX and an overview of the basic rationale for each.

1. **ISSUE: Reduction of EAX' to two input strings:** Non-inclusion of  $\text{CMAC}_k(0^{n-11} \parallel \text{pad}(\text{nullString}))$  when the EAX input H is null.

JUSTIFICATION:

$\text{CMAC}_k(0^{n-11} \parallel \text{pad}(\text{nullString}))$  is precisely  $\text{CBC}_k(0^{1271} \parallel (10^{127} \text{ XOR } \text{dbl}(\text{dbl}(\text{ECB}_k(0))))$ , which is a key-dependent constant. A predictable-to-an-attacker XOR of a key-dependent constant that is the output of a keyed PRP, into a key-dependent variable that is itself a combination of keyed PRP outputs, does not strengthen the cryptanalytic resistance of the computation, because the dimensionality of the resultant composite PRP is identical to that without the inclusion. Therefore its exclusion does not weaken the cryptanalytic resistance of the computation.

2. **ISSUE: Exclusion of the ciphertext tag input from a null plaintext string:** Conditional non-inclusion of  $\text{CMAC}_k(0^{n-210} \parallel \text{CTR}_k(N, \text{nullString}))$  when the EAX input M is null.

JUSTIFICATION:

$\text{CTR}_k(N, \text{nullString})$  is the null string. Thus  $\text{CMAC}_k(0^{n-210} \parallel \text{pad}(\text{CTR}_k(N, \text{nullString})))$  is precisely  $\text{CBC}_k(0^{n-210} \parallel (10^{n-1} \text{ XOR } \text{dbl}(\text{dbl}(\text{ECB}_k(0))))$ , which is a key-dependent constant. A predictable-to-an-attacker conditional XOR of a key-dependent constant that is the output of a keyed PRP, into a key-dependent variable that is itself a combination of keyed PRP outputs, does not strengthen the cryptanalytic resistance of the computation, because the dimensionality of the resultant composite PRP is identical to that without the inclusion. Therefore the conditional exclusion of this XOR does not weaken the cryptanalytic resistance of the computation.

3. **ISSUE: Simplified nonce incrementation in CTR-mode processing:** Forcing two bits of the initial nonce input to a CTR-mode keyed encryption to zero, to eliminate inter-word carries, reduces the average number of messages that can be encrypted before nonce reuse without reducing the maximum message length that can be protected.

JUSTIFICATION: Rogaway introduced a similar optimization in his SIV combined authentication and encryption mode, which was developed after EAX and which has also been submitted to NIST, (see <http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/siv/siv.pdf>), titled **The SIV Mode of Operation for Deterministic Authenticated-Encryption (Key Wrap) and Misuse-Resistant Nonce-Based Authenticated-Encryption**. The resulting reduction in the average number of messages is at most a factor of four. However, since the other nonce bits are unpredictable to an attacker, the attacker cannot determine when nonce reuse might occur, and thus cannot exploit this potential but undetectable-to-the-attacker nonce reuse. Thus there is no practical reduction in the cryptanalytic resistance of the result.

4. **4) ISSUE: Alternate initial CMAC blocks:** Use of constants other than  $0^n$ ,  $0^{n-11}$ , and  $0^{n-210}$  as the initial constants for the three CMAC computations of EAX.

JUSTIFICATION: The real requirement in EAX is that the three CMAC sequences each start with a value in the first block that is disjoint from the values of the first blocks of the other sequences, so that even if the remainder of their input sequences are identical, their outputs will not be. This will cause the initial values of the CBC sequences for each block to differ from that of the other blocks in a key-dependent way that is unpredictable to an attacker. The original EAX choices of  $0^n$ ,  $0^{n-11}$ , and  $0^{n-210}$  were arbitrary.

5. **ISSUE: Pre-encryption of each initial CMAC block:** Inclusion of a key-encrypted starting value in the CMAC' CBC computation, in lieu of prepending a block containing a known constant to the



sequence of blocks to be processed by CBC.

JUSTIFICATION:  $\text{CMAC}_K(J \parallel S, D, Q)$  is identical to  $\text{CMAC}'(K, E_K(J), S, D, Q)$ . Equivalently,  $\text{CMAC}'(K, J', S, D, Q)$  is identical to  $\text{CMAC}_K(D_K(J') \parallel S, D, Q)$ , where  $D_K$  is the keyed decryption operator – the inverse of the keyed encryption operator  $E_K$ . Thus this issue reduces to issue 4.

6. **ISSUE: Use of  $D$  and  $Q$  for the initial CMAC blocks:** The issue is use of a derived initial value for the first block of the conditional CMAC' computation that includes a CTR-mode output, where that value is a constant multiple (in an appropriate predefined canonical GF(2) extension field) of the initial value for the first block of the other CMAC' computation, where that second initial value is the output of a keyed PRP, in contrast to requiring that the initial value for that second CMAC computation (which is conditional) be a keyed PRP output of an independent initial value.

JUSTIFICATION: The CMAC' computations here always involve CBC of at least two blocks. As in 4), the requirement is that the initial value of that CBC sequence differs between the two sequences, in a key-dependent way that is unpredictable to an attacker. Whether the following CMAC' inputs to the two sequences are identical or not, the results of the corresponding CBC sequences are different because those first non-zero outputs of the CBC-chaining process differ for the two CBC sequences. Because the second CMAC' computation occurs only when there is Plaintext/ Ciphertext, so that the corresponding input is non-null, the results of the second CMAC' operator always differ from that of the first. This is the weakest justification of this set of six, but a careful analysis of the original security proof of EAX indicates that this optimization, which eliminates the need for time-optimized implementations to store 32 extra bytes per key, does not degrade the security of the EAX mode.

## I.4 EAX' C code example

The EAX' implementation computes the cipher text and MAC of the example #9 in Annex G - Communication Example. This C code is provided strictly as a reference and its use is not required to conform to this standard.

```
/* EaxExample.c */

#include <stdio.h>
#include <string.h>
#include "aes.h"

void CMacStart(unsigned char *, char);
void CMacNext(unsigned char *, int);
void CMacEnd(unsigned char *);
void AesCtr(unsigned char *, unsigned char *, unsigned char *, int);
void Dbl(unsigned char *);
void PrintByteArrayMSB(char *, unsigned char *, int);
void PrintByteArrayLSB(char *, unsigned char *, int);

static unsigned char buffer[16];
static int buffer_idx;
static unsigned char key[16];
static unsigned char d_value[16];
static unsigned char q_value[16];
static unsigned char nonce[16];

/*****
 * main
 *****/
int main()
{
    unsigned char tagN[16], tagC[16], T[16];
    int i;

    unsigned char testkey[] = {0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,
                                0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08};
    unsigned char called_AP_title_uid[] = {0xA2,0x0C,0x06,0x0A,0x60,0x7C,0x86,
                                             0xF7,0x54,0x01,0x16,0x00,0x7B,0x02};
    unsigned char calling_AE_qualifier_element[] = {0xA7,0x03,0x02,0x01,0x04};
    unsigned char calling_AP_invocation_id_element[] = {0xA8,0x03,0x02,0x01,0x02};
    unsigned char calling_authentication_value[] = {0xAC,0x0F,0xA2,0x0D,0xA0,0x0B,0xA1,0x09};
    unsigned char key_id_element[] = {0x80,0x01,0x02};
    unsigned char iv_element[] = {0x81,0x04,0x48,0xF3,0xD2,0xF8};
    unsigned char user_information[] = {0xBE,0x19,0x28,0x17,0x81,0x15};
    unsigned char epsem_control[] = {0x9A};
    unsigned char calling_AP_title_uid[] = {0xA6,0x0D,0x06,0x0B,0x60,0x7C,0x86,0xF7,
                                             0x54,0x01,0x16,0x00,0x7B,0x82,0x11};

    unsigned char key_id[] = {0x02};
    unsigned char iv[] = {0x48,0xF3,0xD2,0xF8};
    unsigned char epsem_data[] = {0x54,0x45,0x4D,0x50,0x0B,0x40,0x00,0x07,
                                   0x00,0x05,0x1A,0x00,0x00,0x02,0x00,0xE4};

    CMacStart(testkey, 'D');
    CMacNext(called_AP_title_uid, sizeof(called_AP_title_uid));
    CMacNext(calling_AE_qualifier_element, sizeof(calling_AE_qualifier_element));
    CMacNext(calling_AP_invocation_id_element, sizeof(calling_AP_invocation_id_element));
    CMacNext(calling_authentication_value, sizeof(calling_authentication_value));
    CMacNext(key_id_element, sizeof(key_id_element));
    CMacNext(iv_element, sizeof(iv_element));
    CMacNext(user_information, sizeof(user_information));
    CMacNext(epsem_control, sizeof(epsem_control));
    CMacNext(calling_AP_title_uid, sizeof(calling_AP_title_uid));
    CMacNext(key_id, sizeof(key_id));
    CMacNext(iv, sizeof(iv));
    CMacEnd(tagN);
    PrintByteArrayMSB("Tag = ", tagN, 16);

    /* Encryption of the payload */
    AesCtr(testkey, tagN, epsem_data, sizeof(epsem_data));

    CMacStart(key, 'Q');
    CMacNext(epsem_data, sizeof(epsem_data));
}
```

```

    CMacEnd(tagC);
    PrintByteArrayMSB("Tag = ", tagC, 16);

    for (i=0; i<16; i++)
        T[i] = tagN[i] ^ tagC[i];

    PrintByteArrayMSB("MAC = (Expected 0x1BD78F32) ", T+12, 4);

    return 0;
}

/*****
 * Description: Initiate a CMAC computation
 *
 * Inputs: key = 16 bytes key
 *         init_with = can be set to either 'D' or 'Q'
 *****/
void CMacStart(unsigned char *_key, char init_with)
{
    buffer_idx = 0;
    memcpy(key, _key, 16);
    memset(d_value, 0, 16);
    AesEncrypt(d_value, key);

    Dbl(d_value);
    memcpy(q_value, d_value, 16);
    Dbl(q_value);

    if (init_with == 'D')
    {
        memcpy(nonce, d_value, 16);
        PrintByteArrayMSB("key = ", key, 16);
        PrintByteArrayMSB("D = ", d_value, 16);
        PrintByteArrayMSB("Q = ", q_value, 16);
    }
    else
        memcpy(nonce, q_value, 16);
}

/*****
 * Description: Add an array of bytes to the CMAC computation
 *
 * Inputs: byte = Array of bytes to be included in the CMAC computation
 *****/
void CMacNext(unsigned char *s, int lgn)
{
    int i, j;

    for (i=0; i<lgn; i++)
    {
        if (buffer_idx == 16)
        {
            PrintByteArrayLSB("S = ", buffer, 16);

            for(j=0; j<16; j++)
                nonce[j] ^= buffer[j];

            AesEncrypt(nonce, key);

            buffer_idx = 0;
        }

        buffer[buffer_idx++] = s[i];
    }
}

/*****
 * Description: Complete the CMAC computation
 *****/
void CMacEnd(unsigned char *s)
{
    int i;

    if (buffer_idx == 16)
    {
        PrintByteArrayLSB("S = ", buffer, 16);
    }
}

```

```

    for(i=0; i<16; i++)
        buffer[i] ^= d_value[i];
    }
    else
    {
        buffer[buffer_idx++] = 0x80;
        for(i = buffer_idx; i < 16; i++)
            buffer[i] = 0;
        PrintByteArrayLSB("S = ", buffer, 16);

        for(i=0; i<16; i++)
            buffer[i] ^= q_value[i];
    }

    for(i=0; i<16; i++)
        nonce[i] ^= buffer[i];

    AesEncrypt(nonce, key);

    memcpy(s, nonce, 16);
}

/*****
* Description: Functions that encrypts or decrypt an array of bytes.
*
* Inputs: data = Pointer to the data which should be encrypted or decrypted
*         data_length = size of the data field in bytes
*         key = The pointer to the 16 bytes key used for the encryption
*****/
void AesCtr(unsigned char *mykey, unsigned char *_nonce, unsigned char *data, int data_length)
{
    int i;
    unsigned char *ptr = data;
    unsigned char mynonce[16], encrypted_nonce[16];

    memcpy(mynonce, _nonce, 16);
    mynonce[1] ^= 0x7F;
    mynonce[3] ^= 0x7F;

    while (data_length > 0)
    {
        memcpy(encrypted_nonce, mynonce, 16);
        AesEncrypt(encrypted_nonce, mykey);

        for (i=0; i < 16; i++, ptr++)
        {
            if (data_length-- <= 0)
                return;
            *ptr ^= encrypted_nonce[i];
        }

        /* Incrementing the nouce */
        mynonce[15]++;
        if (mynonce[15] == 0)
        {
            mynonce[14]++;
            if (mynonce[14] == 0)
            {
                mynonce[13]++;
                if (mynonce[13] == 0)
                    mynonce[12]++;
            }
        }
    }
}

void Db1(unsigned char *x)
{
    int i;
    int carry = (x[15] & 0x80) != 0;

    for (i=15; i>0; i--)
    {
        x[i] <<= 1;
        if ((x[i-1] & 0x80) != 0)
            x[i] |= 0x01;
    }
}

```

```
    }
    x[0] <<= 1;

    if (carry)
        x[0] ^= 0x87;
}

void PrintByteArrayMSB(char *text, unsigned char *mybuffer, int lgn)
{
    int i;

    printf(text);

    for (i=lgn-1; i>=0; i--)
        printf("%02X", mybuffer[i]);
    printf("\n");
}

void PrintByteArrayLSB(char *text, unsigned char *mybuffer, int lgn)
{
    int i;

    printf(text);

    for (i=0; i<lgn; i++)
        printf("%02X", mybuffer[i]);
    printf("\n");
}
```

## 1.4 AES C code example

This annex provides an AES-128 implementation which can be used in conjunction with the EAX' implementation defined in the previous section. This C code is provided strictly as a reference and its use is not required to conform to this standard.

```

/* aes.h */
#ifndef AES_H
#define AES_H
/*
 * encrypt the passed message in place using the passed key and the
 * AES-128 algorithm as defined in FIPS PUB 197
 */
void AesEncrypt(unsigned char msg[16], unsigned char key[16]);
/*
 * decrypt the passed message in place using the passed key and the
 * AES-128 algorithm as defined in FIPS PUB 197
 */
void AesDecrypt(unsigned char msg[16], unsigned char key[16]);
#endif

/* aes.c */
#include "aes.h"
/* AES-128 specifies a 4-byte word size */
#define WORDSIZE 4
/* AES-128 uses 16 byte (4 word) keys */
#define BLKSIZE 16
/* AES-128 uses 10 rounds */
#define NUMROUNDS 10
typedef unsigned char uint8_t;

void AddRoundKey(uint8_t state[BLKSIZE], const uint8_t *w);
void SubBytes(uint8_t s[BLKSIZE]);
void ShiftRows(uint8_t s[BLKSIZE]);
void MixColumns(uint8_t s[BLKSIZE]);
void InvSubBytes(uint8_t s[BLKSIZE]);
void InvShiftRows(uint8_t s[BLKSIZE]);
void InvMixColumns(uint8_t s[BLKSIZE]);
void KeyExpansion(const uint8_t key[BLKSIZE], uint8_t w[(NUMROUNDS+1)*BLKSIZE]);
uint8_t xtime(uint8_t a);
void SubWord(uint8_t a[WORDSIZE]);
void RotWord(uint8_t a[WORDSIZE]);
uint8_t mult(uint8_t a, uint8_t b);
void Cipher(const uint8_t in[BLKSIZE], uint8_t out[BLKSIZE], uint8_t w[(NUMROUNDS+1)*BLKSIZE]);
void Decipher(const uint8_t in[BLKSIZE], uint8_t out[BLKSIZE], uint8_t w[(NUMROUNDS+1)*BLKSIZE]);

static const uint8_t sbox[256] = {
/* 0 1 2 3 4 5 6 7 8 9 a b c d e f
*/
0x63,0x7c,0x77,0x7b,0xf2,0x6b,0x6f,0xc5,0x30,0x01,0x67,0x2b,0xfe,0xd7,0xab,0x76
,0xca,0x82,0xc9,0x7d,0xfa,0x59,0x47,0xf0,0xad,0xd4,0xa2,0xaf,0x9c,0xa4,0x72,0xc0
,0xb7,0xfd,0x93,0x26,0x36,0x3f,0xf7,0xcc,0x34,0xa5,0xe5,0xf1,0x71,0xd8,0x31,0x15
,0x04,0xc7,0x23,0xc3,0x18,0x96,0x05,0x9a,0x07,0x12,0x80,0xe2,0xeb,0x27,0xb2,0x75
,0x09,0x83,0x2c,0x1a,0x1b,0x6e,0x5a,0xa0,0x52,0x3b,0xd6,0xb3,0x29,0xe3,0x2f,0x84
,0x53,0xd1,0x00,0xed,0x20,0xfc,0xb1,0x5b,0x6a,0xcb,0xbe,0x39,0x4a,0x4c,0x58,0xcf
,0xd0,0xef,0xaa,0xfb,0x43,0x4d,0x33,0x85,0x45,0xf9,0x02,0x7f,0x50,0x3c,0x9f,0xa8
,0x51,0xa3,0x40,0x8f,0x92,0x9d,0x38,0xf5,0xbc,0xb6,0xda,0x21,0x10,0xff,0xf3,0xd2
,0xcd,0x0c,0x13,0xec,0x5f,0x97,0x44,0x17,0xc4,0xa7,0x7e,0x3d,0x64,0x5d,0x19,0x73
,0x60,0x81,0x4f,0xdc,0x22,0x2a,0x90,0x88,0x46,0xee,0xb8,0x14,0xde,0x5e,0x0b,0xdb
,0xe0,0x32,0x3a,0x0a,0x49,0x06,0x24,0x5c,0xc2,0xd3,0xac,0x62,0x91,0x95,0xe4,0x79
,0xe7,0xc8,0x37,0x6d,0x8d,0xd5,0x4e,0xa9,0x6c,0x56,0xf4,0xea,0x65,0x7a,0xae,0x08
,0xba,0x78,0x25,0x2e,0x1c,0xa6,0xb4,0xc6,0xe8,0xdd,0x74,0x1f,0x4b,0xbd,0x8b,0x8a
,0x70,0x3e,0xb5,0x66,0x48,0x03,0xf6,0x0e,0x61,0x35,0x57,0xb9,0x86,0xc1,0x1d,0x9e
,0xe1,0xf8,0x98,0x11,0x69,0xd9,0x8e,0x94,0x9b,0x1e,0x87,0xe9,0xce,0x55,0x28,0xdf
,0x8c,0xa1,0x89,0x0d,0xbf,0xe6,0x42,0x68,0x41,0x99,0x2d,0x0f,0xb0,0x54,0xbb,0x16
};

};

static const uint8_t invsbox[256] = {
/* 0 1 2 3 4 5 6 7 8 9 a b c d e f
*/
0x52,0x09,0x6a,0xd5,0x30,0x36,0xa5,0x38,0xbf,0x40,0xa3,0x9e,0x81,0xf3,0xd7,0xfb

```

```

,0x7c,0xe3,0x39,0x82,0x9b,0x2f,0xff,0x87,0x34,0x8e,0x43,0x44,0xc4,0xde,0xe9,0xcb
,0x54,0x7b,0x94,0x32,0xa6,0xc2,0x23,0x3d,0xee,0x4c,0x95,0x0b,0x42,0xfa,0xc3,0x4e
,0x08,0x2e,0xa1,0x66,0x28,0xd9,0x24,0xb2,0x76,0x5b,0xa2,0x49,0x6d,0x8b,0xd1,0x25
,0x72,0xf8,0xf6,0x64,0x86,0x68,0x98,0x16,0xd4,0xa4,0x5c,0xcc,0x5d,0x65,0xb6,0x92
,0x6c,0x70,0x48,0x50,0xfd,0xed,0xb9,0xda,0x5e,0x15,0x46,0x57,0xa7,0x8d,0x9d,0x84
,0x90,0xd8,0xab,0x00,0x8c,0xbc,0xd3,0x0a,0xf7,0xe4,0x58,0x05,0xb8,0xb3,0x45,0x06
,0xd0,0x2c,0x1e,0x8f,0xca,0x3f,0x0f,0x02,0xc1,0xaf,0xbd,0x03,0x01,0x13,0x8a,0x6b
,0x3a,0x91,0x11,0x41,0x4f,0x67,0xdc,0xea,0x97,0xf2,0xcf,0xce,0xf0,0xb4,0xe6,0x73
,0x96,0xac,0x74,0x22,0xe7,0xad,0x35,0x85,0xe2,0xf9,0x37,0xe8,0x1c,0x75,0xdf,0x6e
,0x47,0xf1,0x1a,0x71,0x1d,0x29,0xc5,0x89,0x6f,0xb7,0x62,0x0e,0xaa,0x18,0xbe,0x1b
,0xfc,0x56,0x3e,0x4b,0xc6,0xd2,0x79,0x20,0x9a,0xdb,0xc0,0xfe,0x78,0xcd,0x5a,0xf4
,0x1f,0xdd,0xa8,0x33,0x88,0x07,0xc7,0x31,0xb1,0x12,0x10,0x59,0x27,0x80,0xec,0x5f
,0x60,0x51,0x7f,0xa9,0x19,0xb5,0x4a,0x0d,0x2d,0xe5,0x7a,0x9f,0x93,0xc9,0x9c,0xef
,0xa0,0xe0,0x3b,0x4d,0xae,0x2a,0xf5,0xb0,0xc8,0xeb,0xbb,0x3c,0x83,0x53,0x99,0x61
,0x17,0x2b,0x04,0x7e,0xba,0x77,0xd6,0x26,0xe1,0x69,0x14,0x63,0x55,0x21,0x0c,0x7d
};

void SubWord(uint8_t a[WORDSIZE])
{
    a[0] = sbox[a[0]];
    a[1] = sbox[a[1]];
    a[2] = sbox[a[2]];
    a[3] = sbox[a[3]];
}

void RotWord(uint8_t a[WORDSIZE])
{
    uint8_t temp;
    temp = a[0];
    a[0] = a[1];
    a[1] = a[2];
    a[2] = a[3];
    a[3] = temp;
}

void KeyExpansion(const uint8_t key[BLKSIZE], uint8_t w[(NUMROUNDS+1)*BLKSIZE])
{
    int i,j;
    uint8_t temp[WORDSIZE];
    uint8_t Rcon[WORDSIZE] = { 1, 0, 0, 0};

    for (j=0; j < BLKSIZE; j++)
        w[j] = key[j];

    for (i = BLKSIZE; i < (NUMROUNDS+1)*BLKSIZE; i+=WORDSIZE)
    {
        for (j=0; j < WORDSIZE; j++)
            temp[j] = w[j+i-WORDSIZE];
        if ((i & 0x3*WORDSIZE) == 0)
        {
            RotWord(temp);
            SubWord(temp);
            temp[0] ^= Rcon[0];
            temp[1] ^= Rcon[1];
            temp[2] ^= Rcon[2];
            temp[3] ^= Rcon[3];
            Rcon[0] = xtime(Rcon[0]);
        }
        for (j=0; j < WORDSIZE; j++)
            w[j+i] = temp[j] ^ w[j+i-BLKSIZE];
    }
}

uint8_t xtime(uint8_t a)
{
    if (a & 0x80)
        a = (a << 1) ^ 0x1b;
    else
        a <<= 1;
    return a;
}

uint8_t mult(uint8_t a, uint8_t b)
{
    uint8_t r = 0;
    uint8_t mask;
    for (mask = 0x01; mask; mask <<= 1)

```

```

    {
        if (a & mask)
            r ^= b;
        b = xtime(b);
    }
    return r;
}

void InvMixColumns(uint8_t s[BLKSIZE])
{
    uint8_t a[WORDSIZE];
    int i;
    for (i=0; i<WORDSIZE; i++)
    {
        a[0] = s[0+WORDSIZE*i];
        a[1] = s[1+WORDSIZE*i];
        a[2] = s[2+WORDSIZE*i];
        a[3] = s[3+WORDSIZE*i];
        s[0+WORDSIZE*i] = mult(0x0e, a[0]) ^ mult(0x0b, a[1]) ^ mult(0x0d, a[2]) ^ mult(0x09, a[3]);
        s[1+WORDSIZE*i] = mult(0x0e, a[1]) ^ mult(0x0b, a[2]) ^ mult(0x0d, a[3]) ^ mult(0x09, a[0]);
        s[2+WORDSIZE*i] = mult(0x0e, a[2]) ^ mult(0x0b, a[3]) ^ mult(0x0d, a[0]) ^ mult(0x09, a[1]);
        s[3+WORDSIZE*i] = mult(0x0e, a[3]) ^ mult(0x0b, a[0]) ^ mult(0x0d, a[1]) ^ mult(0x09, a[2]);
    }
}

void MixColumns(uint8_t s[BLKSIZE])
{
    uint8_t a[WORDSIZE];
    int i;
    for (i=0; i<WORDSIZE; i++)
    {
        a[0] = s[0+WORDSIZE*i];
        a[1] = s[1+WORDSIZE*i];
        a[2] = s[2+WORDSIZE*i];
        a[3] = s[3+WORDSIZE*i];
        s[0+WORDSIZE*i] = mult(2, a[0]) ^ mult(3, a[1]) ^ a[2] ^ a[3];
        s[1+WORDSIZE*i] = mult(2, a[1]) ^ mult(3, a[2]) ^ a[3] ^ a[0];
        s[2+WORDSIZE*i] = mult(2, a[2]) ^ mult(3, a[3]) ^ a[0] ^ a[1];
        s[3+WORDSIZE*i] = mult(2, a[3]) ^ mult(3, a[0]) ^ a[1] ^ a[2];
    }
}

void InvShiftRows(uint8_t s[BLKSIZE])
{
    uint8_t temp;
    /* no change to first row */
    /* rotate second row by three */
    temp = s[1];
    s[1] = s[13];
    s[13] = s[9];
    s[9] = s[5];
    s[5] = temp;
    /* rotate third row by two */
    temp = s[2];
    s[2] = s[10];
    s[10] = temp;
    temp = s[6];
    s[6] = s[14];
    s[14] = temp;
    /* rotate fourth row by one */
    temp = s[3];
    s[3] = s[7];
    s[7] = s[11];
    s[11] = s[15];
    s[15] = temp;
}

void ShiftRows(uint8_t s[BLKSIZE])
{
    uint8_t temp;
    /* no change to first row */
    /* rotate second row by one */
    temp = s[1];
    s[1] = s[5];
    s[5] = s[9];
    s[9] = s[13];

```



```

    s[13] = temp;
    /* rotate third row by two */
    temp = s[2];
    s[2] = s[10];
    s[10] = temp;
    temp = s[6];
    s[6] = s[14];
    s[14] = temp;
    /* rotate fourth row by three */
    temp = s[3];
    s[3] = s[15];
    s[15] = s[11];
    s[11] = s[7];
    s[7] = temp;
}

void InvSubBytes(uint8_t s[BLKSIZE])
{
    int i;
    for (i=0; i<BLKSIZE; i++)
        s[i] = invsbox[s[i]];
}

void SubBytes(uint8_t s[BLKSIZE])
{
    int i;
    for (i=0; i<BLKSIZE; i++)
        s[i] = sbbox[s[i]];
}

void AddRoundKey(uint8_t state[BLKSIZE], const uint8_t *w)
{
    int i;
    for (i=0; i < BLKSIZE; i++)
        state[i] ^= w[i];
}

void Cipher(const uint8_t in[BLKSIZE], uint8_t out[BLKSIZE], uint8_t w[(NUMROUNDS+1)*BLKSIZE])
{
    uint8_t state[BLKSIZE];
    int i;
    int round;

    /* copy bytes from in to state */
    for (i=0; i < BLKSIZE; i++)
        state[i] = in[i];

    AddRoundKey(state, &(w[0]));
    for (round = 1; round < 10; round++)
    {
        SubBytes(state);
        ShiftRows(state);
        MixColumns(state);
        AddRoundKey(state, &(w[BLKSIZE*round]));
    }
    SubBytes(state);
    ShiftRows(state);
    AddRoundKey(state, &(w[BLKSIZE*10]));

    /* copy bytes from state to out */
    for (i=0; i < BLKSIZE; i++)
        out[i] = state[i];
}

void Decipher(const uint8_t in[BLKSIZE], uint8_t out[BLKSIZE], uint8_t w[(NUMROUNDS+1)*BLKSIZE])
{
    uint8_t state[BLKSIZE];
    int i;
    int round;

    /* copy bytes from in to state */
    for (i=0; i < BLKSIZE; i++)
        state[i] = in[i];

    AddRoundKey(state, &(w[BLKSIZE*10]));
    for (round = 9; round; round--)

```

## ANSI C12.22-2008

```
{
    InvShiftRows(state);
    InvSubBytes(state);
    AddRoundKey(state, &(w[BLKSIZE*round]));
    InvMixColumns(state);
}
InvShiftRows(state);
InvSubBytes(state);
AddRoundKey(state, &(w[0]));

/* copy bytes from state to out */
for (i=0; i < BLKSIZE; i++)
    out[i] = state[i];
}

void AesEncrypt(unsigned char msg[BLKSIZE], unsigned char key[BLKSIZE])
{
    uint8_t w[(NUMROUNDS+1)*BLKSIZE];

    KeyExpansion(key, w);
    Cipher(msg, msg, w);
}

void AesDecrypt(unsigned char msg[BLKSIZE], unsigned char key[BLKSIZE])
{
    uint8_t w[(NUMROUNDS+1)*BLKSIZE];

    KeyExpansion(key, w);
    Decipher(msg, msg, w);
}
```

## Annex J – Connectionless-ACSE-1 Equivalent Reduced Syntax for C12.22 Message Transmission

(informative)

The following ASN-1 syntax represents a reduced representation of the official Connectionless-ACSE-1 syntax which is sufficient to encode and decode ANSI C12.22 Messages.

```
-- Module Connectionless-ACSE-1 (X.237 bis:09/1998) for transmitting C12.22 Messages
Connectionless-ACSE-1 {joint-iso-itu-t association-control(2) module(2) clacsel(2) version(1)}
DEFINITIONS ::= BEGIN

AUDT-apdu ::= [APPLICATION 0] IMPLICIT SEQUENCE {
    protocol-version          [0] IMPLICIT BIT STRING {version1(0)} DEFAULT {version1},
    aso-context               [1] ASO-context-name OPTIONAL,
    called-AP-title           [2] AP-title OPTIONAL,
    called-AE-qualifier        [3] AE-qualifier OPTIONAL,
    called-AP-invocation-id    [4] AP-invocation-id OPTIONAL,
    called-AE-invocation-id    [5] AE-invocation-id OPTIONAL,
    calling-AP-title           [6] AP-title OPTIONAL,
    calling-AE-qualifier        [7] AE-qualifier OPTIONAL,
    calling-AP-invocation-id   [8] AP-invocation-id OPTIONAL,
    calling-AE-invocation-id   [9] AE-invocation-id OPTIONAL,          -- Not used
    mechanism-name            [11] IMPLICIT Mechanism-name OPTIONAL,
    calling-authentication-value [12] EXPLICIT Authentication-value OPTIONAL,
    p-context                 [14] IMPLICIT Default-P-context OPTIONAL,  -- Not used
    implementation-information [29] IMPLICIT GraphicString OPTIONAL,     -- Not used
    ...,
    ...,
    user-information          [30] IMPLICIT SEQUENCE SIZE (1, ..., 0 | 2..MAX) OF EXTERNAL
}

ASO-context-name ::= OBJECT IDENTIFIER

AP-title ::= CHOICE {
    ap-title-form1  AP-title-form1,          -- Not used
    ap-title-form2  AP-title-form2,
    ...,
    ap-title-form3  AP-title-form3,          -- Not used
    ap-title-form4  [0] IMPLICIT AP-title-form4
}

AP-title-form1 ::= Not-supported              -- Not used, set to null to remove dependencies
AP-title-form2 ::= OBJECT IDENTIFIER
AP-title-form3 ::= PrintableString
AP-title-form4 ::= RELATIVE-OID

AE-qualifier ::= CHOICE {
    ae-qualifier-form1  AE-qualifier-form1,  -- Not used
    ae-qualifier-form2  AE-qualifier-form2,
    ...
}

AE-qualifier-form1 ::= Not-supported          -- Not used, set to null to remove dependencies
AE-qualifier-form2 ::= INTEGER
AP-invocation-id ::= INTEGER
AE-invocation-id ::= INTEGER
Mechanism-name ::= OBJECT IDENTIFIER

Authentication-value ::= CHOICE {
    charstring  [0] IMPLICIT GraphicString,  -- Not used
    bitstring   [1] IMPLICIT BIT STRING,     -- Not used
    external    [2] IMPLICIT EXTERNAL,
    other       [3] IMPLICIT Not-supported    -- Not used
}
Default-P-context ::= NULL                  -- Not used, set to null to remove dependencies
Not-supported ::= SEQUENCE { not-supported [0] NULL }
END
```