# IEC 62056-5-3

Edition 4.0 2023-09

# INTERNATIONAL STANDARD

# NORME INTERNATIONALE

colour inside

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## Electricity metering data exchange – The DLMS®/COSEM suite – Part 5-3: DLMS®/COSEM application layer

## Échange des données de comptage de l'électricité – La suite DLMS®/COSEM – Partie 5-3: Couche application DLMS®/COSEM

IEC 62056-5-3:2023-09(en-fr)

**THIS PUBLICATION IS COPYRIGHT PROTECTED**
**Copyright © 2023 IEC, Geneva, Switzerland**

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester. If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

Droits de reproduction réservés. Sauf indication contraire, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'IEC ou du Comité national de l'IEC du pays du demandeur. Si vous avez des questions sur le copyright de l'IEC ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de l'IEC de votre pays de résidence.

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

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

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

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

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 It also gives information on projects, replaced and withdrawn publications.

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

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

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

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

**A propos de l'IEC**
La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

**A propos des publications IEC**
Le contenu technique des publications IEC est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

**Recherche de publications IEC - webstore.iec.ch/advsearchform**
La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études, …). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

**IEC Just Published - webstore.iec.ch/justpublished**
Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et une fois par mois par email.

**Service Clients - webstore.iec.ch/csc**
Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: sales@iec.ch.

**IEC Products & Services Portal - products.iec.ch**
Découvrez notre puissant moteur de recherche et consultez gratuitement tous les aperçus des publications. Avec un abonnement, vous aurez toujours accès à un contenu à jour adapté à vos besoins.

**Electropedia - www.electropedia.org**
Le premier dictionnaire d'électrotechnologie en ligne au monde, avec plus de 22 300 articles terminologiques en anglais et en français, ainsi que les termes équivalents dans 19 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

IEC 62056-5-3

Edition 4.0 2023-09

# INTERNATIONAL STANDARD

# NORME INTERNATIONALE

colour inside

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## Electricity metering data exchange – The DLMS®/COSEM suite – Part 5-3: DLMS®/COSEM application layer

## Échange des données de comptage de l'électricité – La suite DLMS®/COSEM – Partie 5-3: Couche application DLMS®/COSEM

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 17.220, 35.110, 91.140.50

ISBN 978-2-8322-7223-7

**Warning! Make sure that you obtained this publication from an authorized distributor.**
**Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.**

® Registered trademark of the International Electrotechnical Commission
Marque déposée de la Commission Electrotechnique Internationale

CONTENTS

FOREWORD ..... 12
INTRODUCTION ..... 14
1 Scope ..... 15
2 Normative references ..... 15
3 Terms, definitions, abbreviated terms and symbols ..... 17
3.1 General DLMS®/COSEM definitions ..... 17
3.2 Definitions related to cryptographic security ..... 22
3.3 Definitions and abbreviated terms related to the Galois/Counter Mode ..... 32
3.4 General abbreviated terms ..... 34
3.5 Symbols related to the Galois/Counter Mode ..... 38
3.6 Symbols related the ECDSA algorithm ..... 38
3.7 Symbols related to the key agreement algorithms ..... 39
4 Overview of DLMS®/COSEM ..... 39
4.1 Information exchange in DLMS®/COSEM ..... 39
4.1.1 General ..... 39
4.1.2 Communication model ..... 40
4.1.3 Naming and addressing ..... 41
4.1.4 Connection oriented operation ..... 44
4.1.5 Application associations ..... 45
4.1.6 Messaging patterns ..... 46
4.1.7 Data exchange between third parties and DLMS®/COSEM servers ..... 47
4.1.8 Communication profiles ..... 48
4.1.9 Model of a DLMS®/COSEM metering system ..... 50
4.1.10 Model of DLMS®/COSEM servers ..... 50
4.1.11 Model of a DLMS®/COSEM client ..... 52
4.1.12 Interoperability and interconnectivity in DLMS®/COSEM ..... 53
4.1.13 Ensuring interconnectivity: the protocol identification service ..... 53
4.1.14 System integration and meter installation ..... 53
4.2 DLMS®/COSEM application layer main features ..... 54
4.2.1 General ..... 54
4.2.2 DLMS®/COSEM application layer structure ..... 54
4.2.3 The Association Control Service Element, ACSE ..... 55
4.2.4 The xDLMS application service element ..... 56
4.2.5 Layer management services ..... 64
4.2.6 Summary of DLMS®/COSEM application layer services ..... 64
4.2.7 DLMS®/COSEM application layer protocols ..... 65
5 Information security in DLMS®/COSEM ..... 65
5.1 Overview ..... 65
5.2 The DLMS®/COSEM security concept ..... 65
5.2.1 Overview ..... 65
5.2.2 Identification and authentication ..... 66
5.2.3 Security context ..... 69
5.2.4 Access rights ..... 69
5.2.5 Application layer message security ..... 69
5.2.6 COSEM data security ..... 72
5.3 Cryptographic algorithms ..... 72

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

5.3.1 Overview ..... 72
5.3.2 Hash function ..... 72
5.3.3 Symmetric key algorithms ..... 73
5.3.4 Public key algorithms ..... 80
5.3.5 Random number generation ..... 90
5.3.6 Compression ..... 91
5.3.7 Security suite ..... 91
5.4 Cryptographic keys – overview ..... 92
5.5 Key used with symmetric key algorithms ..... 92
5.5.1 Symmetric keys types ..... 92
5.5.2 Key information with general-ciphering APDU and data protection ..... 94
5.5.3 Key identification ..... 94
5.5.4 Key wrapping ..... 95
5.5.5 Key agreement ..... 95
5.5.6 Symmetric key cryptoperiods ..... 96
5.6 Keys used with public key algorithms ..... 96
5.6.1 Overview ..... 96
5.6.2 Key pair generation ..... 96
5.6.3 Public key certificates and infrastructure ..... 97
5.6.4 Certificate and certificate extension profile ..... 100
5.6.5 Suite B end entity certificate types to be supported by DLMS®/COSEM servers ..... 108
5.6.6 Management of certificates ..... 108
5.7 Applying cryptographic protection ..... 113
5.7.1 Overview ..... 113
5.7.2 Protecting xDLMS APDUs ..... 113
5.7.3 Multi-layer protection by multiple parties ..... 126
5.7.4 HLS authentication mechanisms ..... 127
5.7.5 Protecting COSEM data ..... 130
6 DLMS®/COSEM application layer service specification ..... 131
6.1 Service primitives and parameters ..... 131
6.2 The COSEM-OPEN service ..... 133
6.3 The COSEM-RELEASE service ..... 138
6.4 COSEM-ABORT service ..... 141
6.5 Protection and general block transfer parameters ..... 141
6.6 The GET service ..... 146
6.7 The SET service ..... 149
6.8 The ACTION service ..... 153
6.9 The ACCESS service ..... 156
6.9.1 Overview – Main features ..... 156
6.9.2 Service specification ..... 158
6.10 The DataNotification service ..... 162
6.11 The EventNotification service ..... 164
6.12 The TriggerEventNotificationSending service ..... 165
6.13 Variable access specification ..... 166
6.14 The Read service ..... 166
6.15 The Write service ..... 170
6.16 The UnconfirmedWrite service ..... 173
6.17 The InformationReport service ..... 175

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

6.18 Client side layer management services: the SetMapperTable.request ..... 176
6.19 Summary of services and LN/SN data transfer service mapping ..... 176
7 DLMS®/COSEM application layer protocol specification ..... 177
7.1 The control function ..... 177
7.1.1 State definitions of the client side control function ..... 177
7.1.2 State definitions of the server side control function ..... 179
7.2 The ACSE services and APDUs ..... 181
7.2.1 ACSE functional units, services and service parameters ..... 181
7.2.2 Registered COSEM names ..... 184
7.2.3 APDU encoding rules ..... 187
7.2.4 Protocol for application association establishment ..... 187
7.2.5 Protocol for application association release ..... 193
7.3 Protocol for the data transfer services ..... 196
7.3.1 Negotiation of services and options – the conformance block ..... 196
7.3.2 Confirmed and unconfirmed service invocations ..... 197
7.3.3 Protocol for the GET service ..... 199
7.3.4 Protocol for the SET service ..... 202
7.3.5 Protocol for the ACTION service ..... 205
7.3.6 Protocol for the ACCESS service ..... 207
7.3.7 Protocol of the DataNotification service ..... 208
7.3.8 Protocol for the EventNotification service ..... 211
7.3.9 Protocol for the Read service ..... 212
7.3.10 Protocol for the Write service ..... 215
7.3.11 Protocol for the UnconfirmedWrite service ..... 219
7.3.12 Protocol for the InformationReport service ..... 220
7.3.13 Protocol of general block transfer mechanism ..... 221
7.3.14 Protocol of exception mechanism ..... 243
8 Abstract syntax of ACSE and COSEM APDUs ..... 244
9 COSEM APDU XML schema ..... 263
9.1 General ..... 263
9.2 XML Schema ..... 263
Annex A (normative) Using the DLMS®/COSEM application layer in various communications profiles ..... 285
A.1 General ..... 285
A.2 Targeted communication environments ..... 285
A.3 The structure of the profile ..... 285
A.4 Identification and addressing schemes ..... 285
A.5 Supporting layer services and service mapping ..... 286
A.6 Communication profile specific parameters of the COSEM AL services ..... 286
A.7 Specific considerations / constraints using certain services within a given profile ..... 286
A.8 The 3-layer, connection-oriented, HDLC based communication profile ..... 286
A.9 The TCP-UDP/IP based communication profiles (COSEM_on_IP) ..... 286
A.10 The wired and wireless M-Bus communication profiles ..... 286
A.11 The S-FSK PLC profile ..... 286
Annex B (normative) SMS short wrapper ..... 287
Annex C (normative) Gateway protocol ..... 288
C.1 General ..... 288
C.2 The gateway protocol ..... 289

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

C.3 HES in the WAN/NN acting as Initiator (Pull operation) ..... 290
C.4 End devices in the LAN acting as Initiators (Push operation) ..... 291
C.4.1 General ..... 291
C.4.2 End device with WAN/NN knowledge ..... 291
C.4.3 End devices without WAN/NN knowledge ..... 291
C.5 Security ..... 291
Annex D (informative) AARQ and AARE encoding examples ..... 292
D.1 General ..... 292
D.2 Encoding of the xDLMS InitiateRequest / InitiateResponse APDU ..... 292
D.3 Specification of the AARQ and AARE APDUs ..... 295
D.4 Data for the examples ..... 296
D.5 Encoding of the AARQ APDU ..... 297
D.6 Encoding of the AARE APDU ..... 300
Annex E (informative) Encoding examples: AARQ and AARE APDUs using a ciphered application context ..... 306
E.1 A-XDR encoding of the xDLMS InitiateRequest APDU, carrying a dedicated key ..... 306
E.2 Authenticated encryption of the xDLMS InitiateRequest APDU ..... 307
E.3 The AARQ APDU ..... 308
E.4 A-XDR encoding of the xDLMS InitiateResponse APDU ..... 310
E.5 Authenticated encryption of the xDLMS InitiateResponse APDU ..... 311
E.6 The AARE APDU ..... 312
E.7 The RLRQ APDU (carrying a ciphered xDLMS InitiateRequest APDU) ..... 314
E.8 The RLRE APDU (carrying a ciphered xDLMS InitiateResponse APDU) ..... 315
Annex F (informative) Data transfer service examples ..... 316
F.1 GET / Read, SET / Write examples ..... 316
F.2 ACCESS service example ..... 333
F.3 Compact array encoding example ..... 334
F.3.1 General ..... 334
F.3.2 The specification of compact-array ..... 335
F.3.3 Example 1: Compact array encoding an array of five long-unsigned values ..... 336
F.3.4 Example 2: Compact-array encoding of five octet-string values ..... 337
F.3.5 Example 3: Encoding of the buffer of a Profile generic object ..... 338
F.4 Profile generic IC buffer attribute encoding examples ..... 339
F.4.1 General ..... 339
F.4.2 Get-response with Profile generic normal encoding example ..... 340
F.4.3 Get-response with Profile generic null-data compressed encoding example ..... 342
F.4.4 Get-response with Profile generic compact-array encoding example ..... 345
F.4.5 Get-response with Profile generic null-data and delta-value encoding example ..... 347
F.4.6 Comparison of various encoding methods for Get-response APDU ..... 350
F.4.7 Combination of the various encoding methods and V.44 compression ..... 350
Annex G (normative) NSA Suite B elliptic curves and domain parameters ..... 352
Annex H (informative) Example of an End entity signature certificate using P-256 signed with P-256 ..... 354
H.1 Fields of public key certificates ..... 354
H.2 Example of a Root-CA Certificate using P-256 signed with P-256 ..... 355

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

H.3 Example of an end entity digital signature Certificate using P-256 signed with P-256 ..... 356
Annex I (normative) Use of key agreement schemes in DLMS®/COSEM ..... 357
I.1 Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme ..... 357
I.2 One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme ..... 360
I.3 Static Unified Model C(0e, 2s, ECC CDH) scheme ..... 363
Annex J (informative) Exchanging protected xDLMS APDUs between TP and server ..... 367
J.1 General ..... 367
J.2 Example 1: Protection is the same in the two directions ..... 367
J.3 Example 2: Protection is different in the two directions ..... 368
Annex K (informative) Significant technical changes with respect to IEC 62056-5-3:2017 ..... 370
Bibliography ..... 373

Figure 1 – Client–server model and communication protocols ..... 41
Figure 2 – Naming and addressing in DLMS®/COSEM ..... 42
Figure 3 – A complete communication session in the CO environment ..... 44
Figure 4 – DLMS®/COSEM messaging patterns ..... 47
Figure 5 – DLMS®/COSEM generic communication profile ..... 49
Figure 6 – Model of a DLMS®/COSEM metering system ..... 50
Figure 7 – DLMS®/COSEM server model ..... 51
Figure 8 – Model of a DLMS®/COSEM client using multiple protocol stacks ..... 52
Figure 9 – The structure of the DLMS®/COSEM application layers ..... 54
Figure 10 – The concept of composable xDLMS messages ..... 61
Figure 11 – Summary of DLMS®/COSEM AL services ..... 64
Figure 12 – Authentication mechanisms ..... 67
Figure 13 – Client – server message security concept ..... 70
Figure 14 – End-to-end message security concept ..... 71
Figure 15 – Hash function ..... 73
Figure 16 – Encryption and decryption ..... 74
Figure 17 – Message Authentication Codes (MACs) ..... 75
Figure 18 – GCM functions ..... 77
Figure 19 – Digital signatures ..... 83
Figure 20 – C(2e, 0s) scheme: each party contributes only an ephemeral key pair ..... 85
Figure 21 – C(1e, 1s) schemes: party U contributes an ephemeral key pair, and party V contributes a static key pair ..... 86
Figure 22 – C(0e, 2s) scheme: each party contributes only a static key pair ..... 88
Figure 23 – Architecture of a Public Key Infrastructure (example) ..... 99
Figure 24 – MSC for provisioning the server with CA certificates ..... 109
Figure 25 – MSC for security personalisation of the server ..... 110
Figure 26 – Provisioning the server with the certificate of the client ..... 111
Figure 27 – Provisioning the client / third party with a certificate of the server ..... 112
Figure 28 – Remove certificate from the server ..... 112
Figure 29 – Cryptographic protection of information using AES-GCM ..... 116
Figure 30 – Structure of service-specific global / dedicated ciphering xDLMS APDUs ..... 118

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Figure 31 – Structure of general-glo-ciphering and general-ded-ciphering xDLMS APDUs ..... 119
Figure 32 – Structure of general-ciphering xDLMS APDUs ..... 120
Figure 33 – Structure of general-signing APDUs ..... 126
Figure 34 – Service primitives ..... 131
Figure 35 – Time sequence diagrams ..... 132
Figure 36 – Additional service parameters to control cryptographic protection and GBT ..... 142
Figure 37 – Partial state machine for the client side control function ..... 178
Figure 38 – Partial state machine for the server side control function ..... 180
Figure 39 – MSC for successful AA establishment preceded by a successful lower layer connection establishment ..... 189
Figure 40 – Graceful AA release using the A-RELEASE service ..... 194
Figure 41 – Graceful AA release by disconnecting the supporting layer ..... 195
Figure 42 – Aborting an AA following a PH-ABORT.indication ..... 196
Figure 43 – MSC of the GET service ..... 199
Figure 44 – MSC of the GET service with block transfer ..... 200
Figure 45 – MSC of the GET service with block transfer, long GET aborted ..... 202
Figure 46 – MSC of the SET service ..... 203
Figure 47 – MSC of the SET service with block transfer ..... 203
Figure 48 – MSC of the ACTION service ..... 205
Figure 49 – MSC of the ACTION service with block transfer ..... 207
Figure 50 – Access Service with long response ..... 208
Figure 51 – Access Service with long request and response ..... 208
Figure 52 – MSC for the DataNotification service, case 1) ..... 209
Figure 53 – MSC for the DataNotification service, case 2) ..... 210
Figure 54 – MSC for the DataNotification service, case 3) ..... 211
Figure 55 – MSC of the Read service used for reading an attribute ..... 214
Figure 56 – MSC of the Read service used for invoking a method ..... 214
Figure 57 – MSC of the Read Service used for reading an attribute, with block transfer ..... 215
Figure 58 – MSC of the Write service used for writing an attribute ..... 218
Figure 59 – MSC of the Write service used for invoking a method ..... 218
Figure 60 – MSC of the Write Service used for writing an attribute, with block transfer ..... 219
Figure 61 – MSC of the Unconfirmed Write service used for writing an attribute ..... 220
Figure 62 – Partial service invocations and GBT APDUs ..... 223
Figure 63 – The GBT procedure ..... 226
Figure 64 – Send GBT APDU stream sub-procedure ..... 230
Figure 65 – Process GBT APDU sub-procedure ..... 232
Figure 66 – Check RQ and fill gaps sub-procedure ..... 234
Figure 67 – GET service with GBT, switching to streaming ..... 235
Figure 68 – GET service with partial invocations, GBT and streaming, recovery of 4th block sent in the 2nd stream ..... 236
Figure 69 – GET service with partial invocations, GBT and streaming, recovery of 4th and 5th block ..... 238

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Figure 70 – GET service with partial invocations, GBT and streaming, recovery of last block ..... 239
Figure 71 – SET service with GBT, with server not supporting streaming, recovery of 3rd block ..... 240
Figure 72 – ACTION-WITH-LIST service with bi-directional GBT and block recovery ..... 241
Figure 73 – DataNotification service with GBT with partial invocation ..... 243
Figure B.1 – Short wrapper ..... 287
Figure C.1 – General architecture with gateway ..... 288
Figure C.2 – The fields used for pre-fixing the COSEM APDUs ..... 289
Figure C.3 – Pull message sequence chart ..... 290
Figure C.4 – Push message sequence chart ..... 291
Figure I.1 – MSC for key agreement using the Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme ..... 357
Figure I.2 – Ciphered xDLMS APDU protected by an ephemeral key established using the One-pass Diffie-Hellman (1e, 1s, ECC CDH) scheme ..... 360
Figure I.3 – Ciphered xDLMS APDU protected by an ephemeral key established using the Static Unified Model C(0e, 2s, ECC CDH) scheme ..... 364
Figure J.1 – Exchanging protected xDLMS APDUs between TP and server: example 1 ..... 368
Figure J.2 – Exchanging protected xDLMS APDUs between TP and server: example 2 ..... 369

Table 1 – Client and server SAPs ..... 43
Table 2 – Clarification of the meaning of PDU size for DLMS®/COSEM ..... 63
Table 3 – Elliptic curves in DLMS®/COSEM security suites ..... 81
Table 4 – Ephemeral Unified Model key agreement scheme summary ..... 85
Table 5 – One-pass Diffie-Hellman key agreement scheme summary ..... 87
Table 6 – Static Unified Model key agreement scheme summary ..... 89
Table 7 – *OtherInfo* subfields and substrings ..... 90
Table 8 – Security algorithm ID-s ..... 90
Table 9 – DLMS®/COSEM security suites ..... 91
Table 10 – Symmetric keys types ..... 93
Table 11 – Key information with general-ciphering APDU and data protection ..... 94
Table 12 – Asymmetric keys types and their use ..... 96
Table 13 – X.509 v3 Certificate structure ..... 100
Table 14 – X.509 v3 tbsCertificate fields ..... 101
Table 15 – Naming scheme for the Root-CA instance (informative) ..... 102
Table 16 – Naming scheme for the Sub-CA instance (informative) ..... 102
Table 17 – Naming scheme for the end entity instance ..... 103
Table 18 – X.509 v3 Certificate extensions ..... 105
Table 19 – Key Usage extensions ..... 106
Table 20 – Subject Alternative Name values ..... 106
Table 21 – Issuer Alternative Name values ..... 107
Table 22 – Basic constraints extension values ..... 107
Table 23 – Certificates handled by DLMS®/COSEM end entities ..... 108
Table 24 – Security policy values ("Security setup" version 1) ..... 113

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Table 25 – Access rights values ("Association LN" ver 3 "Association SN" ver 4) ..... 114
Table 26 – Ciphered xDLMS APDUs ..... 115
Table 27 – Security control byte ..... 117
Table 28 – Plaintext and Additional Authenticated Data ..... 117
Table 29 – Use of the fields of the ciphering xDLMS APDUs ..... 121
Table 30 – Example: glo-get-request xDLMS APDU ..... 122
Table 31 – ACCESS service with general-ciphering, One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) key agreement scheme ..... 124
Table 32 – DLMS®/COSEM HLS authentication mechanisms ..... 128
Table 33 – HLS example using authentication-mechanism 5 with GMAC ..... 129
Table 34 – HLS example using authentication-mechanism 7 with ECDSA ..... 130
Table 35 – Codes for AL service parameters ..... 133
Table 36 – Service parameters of the COSEM-OPEN service primitives ..... 134
Table 37 – Service parameters of the COSEM-RELEASE service primitives ..... 138
Table 38 – Service parameters of the COSEM-ABORT service primitives ..... 141
Table 39 – Additional service parameters ..... 143
Table 40 – Security parameters ..... 144
Table 41 – APDUs used with security protection types ..... 145
Table 42 – Service parameters of the GET service ..... 147
Table 43 – GET service request and response types ..... 148
Table 44 – Service parameters of the SET service ..... 150
Table 45 – SET service request and response types ..... 151
Table 46 – Service parameters of the ACTION service ..... 153
Table 47 – ACTION service request and response types ..... 154
Table 48 – Service parameters of the ACCESS service ..... 159
Table 49 – Service parameters of the DataNotification service primitives ..... 163
Table 50 – Service parameters of the EventNotification service primitives ..... 164
Table 51 – Service parameters of the TriggerEventNotificationSending.request service primitive ..... 165
Table 52 – Variable Access Specification ..... 166
Table 53 – Service parameters of the Read service ..... 167
Table 54 – Use of the Variable_Access_Specification variants and the Read.response choices ..... 168
Table 55 – Service parameters of the Write service ..... 171
Table 56 – Use of the Variable_Access_Specification variants and the Write.response choices ..... 172
Table 57 – Service parameters of the UnconfirmedWrite service ..... 174
Table 58 – Use of the Variable_Access_Specification variants ..... 174
Table 59 – Service parameters of the InformationReport service ..... 175
Table 60 – Service parameters of the SetMapperTable.request service primitives ..... 176
Table 61 – Summary of ACSE services ..... 176
Table 62 – Summary of xDLMS services ..... 177
Table 63 – Functional Unit APDUs and their fields ..... 182
Table 64 – COSEM application context names ..... 186

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Table 65 – COSEM authentication mechanism names ..... 186
Table 66 – Cryptographic algorithm ID-s ..... 187
Table 67 – xDLMS Conformance block ..... 197
Table 68 – GET service types and APDUs ..... 199
Table 69 – SET service types and APDUs ..... 202
Table 70 – ACTION service types and APDUs ..... 205
Table 71 – Mapping between the GET and the Read services ..... 212
Table 72 – Mapping between the ACTION and the Read services ..... 213
Table 73 – Mapping between the SET and the Write services ..... 216
Table 74 – Mapping between the ACTION and the Write service ..... 217
Table 75 – Mapping between the SET and the UnconfirmedWrite services ..... 220
Table 76 – Mapping between the ACTION and the UnconfirmedWrite services ..... 220
Table 77 – Mapping between the EventNotification and InformationReport services ..... 221
Table 78 – GBT procedure state variables ..... 228
Table 79 – xDLMS exception mechanism ..... 244
Table B.1 – Reserved Application Processes ..... 287
Table D.1 – Conformance block ..... 293
Table D.2 – A-XDR encoding of the xDLMS InitiateRequest APDU ..... 294
Table D.3 – A-XDR encoding of the xDLMS InitiateResponse APDU ..... 295
Table D.4 – BER encoding of the AARQ APDU ..... 298
Table D.5 – Complete AARQ APDU ..... 300
Table D.6 – BER encoding of the AARE APDU ..... 301
Table D.7 – The complete AARE APDU ..... 305
Table E.1 – A-XDR encoding of the xDLMS InitiateRequest APDU ..... 307
Table E.2 – Authenticated encryption of the xDLMS InitiateRequest APDU ..... 308
Table E.3 – BER encoding of the AARQ APDU ..... 309
Table E.4 – A-XDR encoding of the xDLMS InitiateResponse APDU ..... 311
Table E.5 – Authenticated encryption of the xDLMS InitiateResponse APDU ..... 312
Table E.6 – BER encoding of the AARE APDU ..... 313
Table E.7 – BER encoding of the RLRQ APDU ..... 314
Table E.8 – BER encoding of the RLRE APDU ..... 315
Table F.1 – The objects used in the examples ..... 316
Table F.2 – Example: Reading the value of a single attribute without block transfer ..... 317
Table F.3 – Example: Reading the value of a list of attributes without block transfer ..... 318
Table F.4 – Example: Reading the value of a single attribute with block transfer ..... 320
Table F.5 – Example: Reading the value of a list of attributes with block transfer ..... 322
Table F.6 – Example: Writing the value of a single attribute without block transfer ..... 325
Table F.7 – Example: Writing the value of a list of attributes without block transfer ..... 326
Table F.8 – Example: Writing the value of a single attribute with block transfer ..... 328
Table F.9 – Example: Writing the value of a list of attributes with block transfer ..... 330
Table F.10 – Example: ACCESS service without block transfer ..... 333
Table F.11 – Profile generic buffer – get-response with normal encoding ..... 340
Table F.12 – Profile generic buffer – get-response with null-data compression ..... 342

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Table F.13 – Profile generic buffer – get-response with compact-array encoding ..... 345
Table F.14 – Profile generic buffer – Get-response with null-data and delta-value encoding ..... 348
Table F.15 – Comparison of various encoding methods for get-response APDU ..... 350
Table F.16 – Combination of the various encoding methods and V.44 compression for get-response APDU ..... 351
Table G.1 – ECC_P256_Domain_Parameters ..... 352
Table G.2 – ECC_P384_Domain_Parameters ..... 353
Table H.1 – Fields of public key Certificates using P-256 signed with P-256 ..... 354
Table I.1 – Test vector for key agreement using the Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme ..... 358
Table I.2 – Test vector for key agreement using the One-pass Diffie-Hellman (1e, 1s, ECC CDH) scheme ..... 361
Table I.3 – Test vector for key agreement using the Static-Unified Model (0e, 2s, ECC CDH) scheme ..... 365

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

# ELECTRICITY METERING DATA EXCHANGE – THE DLMS®/COSEM SUITE –

## Part 5-3: DLMS®/COSEM application layer

## FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

IEC 62056-5-3 has been prepared by IEC technical committee 13: Electrical energy measurement and control. It is an International Standard.

This fourth edition cancels and replaces the third edition published in 2017. This edition constitutes a technical revision.

The significant technical changes with respect to the previous edition are listed in Annex K (Informative).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

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

| Draft | Report on voting |
|---|---|
| 13/1890/FDIS | 13/1904/RVD |

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

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

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/publications.

A list of all the parts in the IEC 62056 series, published under the general title *Electricity metering data exchange – The DLMS®/COSEM suite*, can be found on the IEC website.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

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

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# INTRODUCTION

This fourth edition of IEC 62056-5-3 has been prepared by IEC TC13 WG14 with a significant contribution of the DLMS® User Association, its A-type liaison partner.

This edition is in line with DLMS® UA 1000-2, the "Green Book" Ed. 10:2020 and DLMS® UA 1000-2, the "Green Book" Ed. 10 Amendment 1 2021.

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this International Standard may involve the use of a maintenance service concerning the stack of protocols on which the present standard IEC 62056-5-3 is based.

The IEC takes no position concerning the evidence, validity and scope of this maintenance service.

The provider of the maintenance service has assured the IEC that he is willing to provide services under reasonable and non-discriminatory terms and conditions for applicants throughout the world. In this respect, the statement of the provider of the maintenance service is registered with the IEC. Information may be obtained from:

DLMS[1] User Association
www.dlms.com

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

[1] Device Language Message Specification.

# ELECTRICITY METERING DATA EXCHANGE – THE DLMS®/COSEM SUITE –

## Part 5-3: DLMS®/COSEM application layer

## 1 Scope

This part of IEC 62056 specifies the DLMS®/COSEM application layer in terms of structure, services and protocols for DLMS®/COSEM clients and servers, and defines rules to specify the DLMS®/COSEM communication profiles.

It defines services for establishing and releasing application associations, and data communication services for accessing the methods and attributes of COSEM interface objects, defined in IEC 62056-6-2:2021 using either logical name (LN) or short name (SN) referencing.

Annex A (normative) defines how to use the COSEM application layer in various communication profiles. It specifies how various communication profiles can be constructed for exchanging data with metering equipment using the COSEM interface model, and what are the necessary elements to specify in each communication profile. The actual, media-specific communication profiles are specified in separate parts of the IEC 62056 series.

Annex B (normative) specifies the SMS short wrapper.

Annex C (normative) specifies the gateway protocol.

Annex D, Annex E and Annex F (informative) include encoding examples for APDUs.

Annex G (normative) provides NSA Suite B elliptic curves and domain parameters.

Annex H (informative) provides an example of an End entity signature certificate using P-256 signed with P-256.

Annex I (normative) specifies the use of key agreement schemes in DLMS®/COSEM.

Annex J (informative) provides examples of exchanging protected xDLMS APDUs between a third party and a server.

Annex K (informative) lists the main technical changes in this fourth edition.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 2 Normative references

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

IEC 61334-4-41:1996, *Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 41: Application protocols – Distribution line message specification*

IEC 61334-6:2000, *Distribution automation using distribution line carrier systems – Part 6: A-XDR encoding rule*

IEC TR 62051:1999, *Electricity metering – Glossary of terms*

IEC TR 62051-1:2004, *Electricity metering – Data exchange for meter reading, tariff and load control – Glossary of terms – Part 1: Terms related to data exchange with metering equipment using DLMS®/COSEM*

IEC 62056-6-2:2021, *Electricity metering data exchange – The DLMS®/COSEM suite – Part 6-2: COSEM interface classes*

IEC 62056-7-3:2017, *Electricity metering data exchange – The DLMS®/COSEM suite – Part 7-3: Wired and wireless M-Bus communication profiles for local and neighbourhood networks*

IEC 62056-7-6:2013, *Electricity metering data exchange – The DLMS®/COSEM suite – Part 7-6: The 3-layer, connection-oriented HDLC based communication profile*

IEC 62056-8-3:2013, *Electricity metering data exchange – The DLMS®/COSEM suite – Part 8-3: Communication profile for PLC S-FSK neighbourhood networks*

IEC 62056-8-11:–[2] *Electricity metering data exchange – The DLMS®/COSEM suite – Part 8-11: Communication profile for Wi-SUN field area mesh networks*

IEC 62056-8-12:–[3] *Electricity metering data exchange – The DLMS®/COSEM suite – Part 8-12: Communication profile for Low Power Wide Area Networks (LPWAN)*

IEC 62056-9-7:2013, *Electricity metering data exchange – The DLMS®/COSEM suite – Part 9-7: Communication profile for TCP-UDP/IP networks*

ISO/IEC 8824-1:2008, *Information technology – Abstract Syntax Notation One (ASN.1) – Part 1: Specification of basic notation*

ISO/IEC 8825-1:2008, *Information technology – ASN.1 encoding rules – Part 1: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

ISO/IEC 15953:1999, *Information technology – Open Systems Interconnection – Service definition for the Application Service Object Association Control Service Element*

NOTE This standard cancels and replaces ISO/IEC 8649:1996 and its Amd. 1:1997 and Amd. 2:1998, of which it constitutes a technical revision.

ISO/IEC 15954:1999, *Information technology – Open Systems Interconnection – Connection-mode protocol for the Application Service Object Association Control Service Element*

NOTE This standard cancels and replaces ISO/IEC 8650-1:1999 and its Amd. 1:1997 and Amd. 2:1998, of which it constitutes a technical revision.

ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model*

ITU-T X.509:2008, *SERIES X: DATA NETWORKS, OPEN SYSTEM COMMUNICATIONS AND SECURITY – Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks*

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

---

2 Under preparation. Stage at the time of publication: IEC CDV.

3 Under preparation. Stage at the time of publication: 13/1877/CDV:2023.

ITU-T X.693 (11/2008), *Information technology – ASN.1 encoding rules: XML Encoding Rules (XER)*

ITU-T X.693 Corrigendum 1 (10/2011), *Information technology – ASN.1 encoding rules: XML Encoding Rules (XER) Technical Corrigendum 1*

ITU-T X.694 (11/2008), *Information technology – ASN.1 encoding rules: Mapping W3C XML schema definitions into ASN.1*

ITU-T X.694 Corrigendum 1 (10/2011), *Information technology – ASN.1 encoding rules: Mapping W3C XML schema definitions into ASN.1 Technical Corrigendum 1*

FIPS PUB 180-4:2012, *Secure hash standard (SHS)*

FIPS PUB 186-4:2013, *Digital Signature Standard (DSS)*

NIST SP 800-21:2005, *Guideline for Implementing Cryptography in the Federal Government*

NIST SP 800-32:2001, *Introduction to Public Key Technology and the Federal PKI Infrastructure*

NIST SP 800-56A Rev. 2: 2013, *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography*

NIST SP 800-57:2012, *Recommendation for Key Management – Part 1: General* (Revision 3)

NSA2, *Suite B Implementer's Guide to NIST SP800-56A*, 28th July 2009

NSA3, *NSA Suite B Base Certificate and CRL Profile*, 27th May 2008

SEC1:2009, *Standards for Efficient Cryptography: Elliptic Curve Cryptography*. SECG. Version 2.0

RFC 3394, *Advanced Encryption Standard (AES) Key Wrap Algorithm*. Edited by J. Schaad (Soaring Hawk Consulting) and R. Housley (RSA Laboratories) September 2002 http://tools.ietf.org/html/rfc3394

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

RFC 4108, *Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages*, 2005, http://www.ietf.org/rfc/rfc4108

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

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 3 Terms, definitions, abbreviated terms and symbols

For the purposes of this document, the terms and definitions given in IEC TR 62051:1999, IEC TR 62051-1, RFC 4106, and the following apply.

### 3.1 General DLMS®/COSEM definitions

**3.1.1**
**ACSE APDU**
APDU used by the Association Control Service Element (ACSE)

**3.1.2**
**application association**
cooperative relationship between two application entities, formed by their exchange of application protocol control information through their use of presentation services

**3.1.3**
**application context**
set of application service elements, related options and any other information necessary for the interworking of application entities in an application association

**3.1.4**
**application entity**
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

**3.1.5**
**application process**
element within a real open system which performs the information processing for a particular application

[SOURCE: ISO/IEC 7498-1:1994, 4.1.4]

**3.1.6**
**authentication mechanism**
specification of a specific set of authentication-function rules for defining, processing, and transferring authentication-values

[SOURCE: ISO/IEC 15953:1999, 3.5.11]

**3.1.7**
**block**
one portion of an xDLMS APDU; the payload of a GBT APDU

**3.1.8**
**client**
application process running in the data collection system

**3.1.9**
**client/server**
relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfils the request

**3.1.10**
**confirmed GBT procedure**
procedure in which the sender sends streams of GBT APDUs and at the end of each stream the recipient acknowledges the blocks received and attempts recovering any missing blocks

Note 1 to entry: A GBT stream consists of one or more GBT APDUs.

Note 2 to entry: In the case of a confirmed GBT stream the end of the stream is indicated by the streaming bit to set to FALSE (0). In the case of an unconfirmed GBT stream the end of the stream is indicated by the Final bit set to TRUE (1).

**3.1.11**
**COSEM**
Companion Specification for Energy Metering; refers to the COSEM object model

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.1.12**
**COSEM APDU**
comprises ACSE APDUs and xDLMS APDUs

**3.1.13**
**COSEM data**
COSEM object attribute values, method invocation and return parameters

**3.1.14**
**COSEM interface class**
entity with specific set of attributes and methods modelling a certain function on its own or in relation with other COSEM interface classes

**3.1.15**
**COSEM object**
instance of a COSEM interface class

**3.1.16**
**DLMS®/COSEM**
refers to the application layer providing xDLMS services to access COSEM interface object attributes. Also refers to the DLMS®/COSEM Application layer and the COSEM data model together

**3.1.17**
**DLMS® context**
a specification of the service elements of DLMS® and semantics of communication to be used during the lifetime of an application association

[SOURCE: IEC 61334-4-41:1996, 3.3.5]

**3.1.18**
**entity authentication**
corroboration that an entity is the one claimed

[SOURCE: ISO/IEC 9798-1:2010, 3.14]

**3.1.19**
**gap**
empty space i.e. missing blocks in the receive queue RQ

Note 1 to entry: A receive queue RQ may have one or more gaps. In each gap, one or more blocks may be missing.

**3.1.20**
**GBT APDU**
xDLMS APDU with control information that carries a block of another xDLMS APDU or an empty block

**3.1.21**
**GBT exchange**
exchanging GBT APDUs that carry the service primitives of the same service

**3.1.22**
**GBT stream**
a sequence of GBT APDUs

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.1.23**
**general block transfer**
**GBT**
DLMS®/COSEM application layer mechanism that can transfer any other xDLMS APDU that would be otherwise too long to fit into the maximum APDU size negotiated, in several blocks

Note 1 to entry: GBT can be forced by including GBT parameters in the .request service primitive.

**3.1.24**
**logical device**
abstract entity within a physical device, representing a subset of the functionality modelled with COSEM objects

**3.1.25**
**master**
central station – station which takes the initiative and controls the data flow

**3.1.26**
**message**
xDLMS APDU carrying a service primitive in an encoded form, which may also be cryptographically protected

**3.1.27**
**mutual authentication**
entity authentication which provides both entities with assurance of each other's identity

Note 1 to entry: The DLMS®/COSEM HLS authentication mechanism provides mutual authentication.

[SOURCE: ISO/IEC 9798-1:2010, 3.18, modified by adding Note 1]

**3.1.28**
**overflow**
more GBT APDUs received in one stream than the size of the GBT window

**3.1.29**
**physical device**
physical metering equipment, the highest level element used in the COSEM interface model of metering equipment

**3.1.30**
**pull operation**
style of communication where the request for a given transaction is initiated by the client

**3.1.31**
**push operation**
style of communication where the request for a given transaction is initiated by the server

**3.1.32**
**receive queue**
**RQ**
placeholder for the blocks of the APDU received in a GBT stream

**3.1.33**
**server**
application process running in a metering equipment

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.1.34**
**send queue**
**SQ**
placeholder for the blocks of the APDU to be sent

**3.1.35**
**service-specific block transfer**
DLMS®/COSEM application layer mechanism that can transfer an xDLMS APDU corresponding to a specific service primitive, that would be otherwise too long to fit into the maximum APDU size negotiated, in several blocks

**3.1.36**
**streaming window**
number of GBT APDUs that can be received in a stream

**3.1.37**
**slave**
station responding to requests of a master station.

Note 1 to entry: A meter is normally a slave station.

**3.1.38**
**system title**
unique identifier of the system

**3.1.39**
**unconfirmed GBT procedure**
procedure in which the sender sends and the recipient receives a single stream of GBT APDUs, the recipient does not acknowledge the blocks received and does not attempt to recover any blocks lost

Note 1 to entry: This is used to carry unconfirmed service requests from the client to the server or unsolicited service requests from the server to the client.

**3.1.40**
**unilateral authentication**
entity authentication which provides one entity with assurance of the other's identity but not vice versa

Note 1 to entry: The DLMS®/COSEM LLS authentication mechanism provides unilateral authentication.

[SOURCE: ISO/IEC 9798-1:2010, 3.39]

**3.1.41**
**xDLMS**
extended DLMS®; refers to the DLMS® protocol with the extensions specified in this document

**3.1.42**
**xDLMS APDU**
APDU used by the xDLMS Application Service Element (xDLMS ASE)

**3.1.43**
**xDLMS message**
xDLMS APDU exchanged between a client and a server or between a third party and a server

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 3.2 Definitions related to cryptographic security

**3.2.1**
**access control**
restricts access to resources to only privileged entities

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.2**
**asymmetric key algorithm**
see Public key cryptographic algorithm

**3.2.3**
**authentication**
process that establishes the source of information, provides assurance of an entity's identity or provides assurance of the integrity of communications sessions, messages, documents or stored data

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.4**
**authentication code**
a cryptographic checksum based on an approved security function (also known as a Message Authentication Code)

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.5**
**certificate**
see public key certificate

**3.2.6**
**Certification Authority**
**CA**
entity in a Public Key Infrastructure (PKI) that is responsible for issuing public key certificates and exacting compliance to a PKI policy

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.7**
**Certificate Policy**
**CP**
specialized form of administrative policy tuned to electronic transactions performed during certificate management. A Certificate Policy addresses all aspects associated with the generation, production, distribution, accounting, compromise recovery, and administration of digital certificates. Indirectly, a certificate policy can also govern the transactions conducted using a communications system protected by a certificate-based security system. By controlling critical certificate extensions, such policies and associated enforcement technology can support provision of the security services required by particular applications

[SOURCE: NIST SP 800-32:2001]

**3.2.8**
**challenge**
time variant parameter generated by a verifier

[SOURCE: ITU-T X.811:1995, 3.8]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.9**
**ciphering**
authentication and / or encryption using symmetric key algorithms

**3.2.10**
**ciphertext**
data in its encrypted form

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.11**
**cofactor**
order of the elliptic curve group divided by the (prime) order of the generator point (i.e. the base point) specified in the domain parameters

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.12**
**confidentiality**
property that sensitive information is not disclosed to unauthorized entities

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.13**
**cryptographic algorithm**
well-defined computational procedure that takes variable inputs including a cryptographic key and produces an output

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.14**
**cryptographic key**
**key**
parameter used in conjunction with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the key can reproduce or reverse the operation, while an entity without knowledge of the key cannot

Examples include:

1. The transformation of plaintext data into ciphertext data,
2. The transformation of ciphertext data into plaintext data,
3. The computation of a digital signature from data,
4. The verification of a digital signature,
5. The computation of an authentication code from data,
6. The verification of an authentication code from data and a received authentication code,
7. The computation of a shared secret that is used to derive keying material.

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.15**
**cryptoperiod**
time span during which a specific key is authorized for use or in which the keys for a given system or application may remain in effect

[SOURCE: NIST SP 800-57:2012, Part 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.16**
**dedicated key**
in DLMS®/COSEM, a symmetric key used within a single instance of an Application Association. See also session key

**3.2.17**
**deprecated**
not recommended for new implementations

**3.2.18**
**digital signature**
result of a cryptographic transformation of data that, when properly implemented with supporting infrastructure and policy, provides the services of:

1) origin authentication
2) data integrity, and
3) signer non-repudiation

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.19**
**directly trusted CA**
directly trusted CA is a CA whose public key has been obtained and is being stored by an end entity in a secure, trusted manner, and whose public key is accepted by that end entity in the context of one or more applications

[SOURCE: ISO/IEC 15945:2002, 3.4]

**3.2.20**
**directly trusted CA key**
directly trusted CA key is a public key of a directly trusted CA. It has been obtained and is being stored by an end entity in a secure, trusted manner. It is used to verify certificates without being itself verified by means of a certificate created by another CA

Note 1 to entry: Directly trusted CAs and directly trusted CA keys may vary from entity to entity. An entity may regard several CAs as directly trusted CAs.

[SOURCE: ISO/IEC 15945:2002, 3.5]

**3.2.21**
**distribution**
see key distribution

**3.2.22**
**domain parameters**
parameters used with a cryptographic algorithm that are common to a domain of users

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.23**
**encryption**
process of changing plaintext into ciphertext using a cryptographic algorithm and key

[SOURCE: NIST SP 800-57:2012, Part 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.24**
**ephemeral key**
cryptographic key that is generated for each execution of a key establishment process and that meets other requirements of the key type (e.g., unique to each message or session). In some cases ephemeral keys are used more than once, within a single "session (e.g., broadcast applications) where the sender generates only one ephemeral key pair per message and the private key is combined separately with each recipient's public key

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.25**
**global key**
key that is intended for use for a relatively long period of time and is typically intended for use in many instances of a DLMS®/COSEM Application Association, see also static symmetric key

**3.2.26**
**hash function**
function that maps a bit string of arbitrary length to a fixed-length bit string. Approved hash functions satisfy the following properties:

1) One-way: It is computationally infeasible to find any input that maps to any pre-specified output, and
2) Collision resistant: It is computationally infeasible to find any two distinct inputs that map to the same output

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.27**
**hash value**
result of applying a hash function to information

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.28**
**initialization vector**
**IV**
vector used in defining the starting point of a cryptographic process

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.29**
**identification**
process of verifying the identity of a user, process, or device, usually as a prerequisite for granting access to resources in an IT system

[SOURCE: NIST SP 800-47:2002]

**3.2.30**
**key**
see cryptographic key

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.31**
**key agreement**
(pair-wise) key-establishment procedure in which the resultant secret keying material is a function of information contributed by both participants, so that neither party can predetermine the value of the secret keying material independently from the contributions of the other party. Contrast with key-transport

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.32**
**key-confirmation**
procedure to provide assurance to one party (the key-confirmation recipient) that another party (the key-confirmation provider) actually possesses the correct secret keying material and/or shared secret

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.33**
**key-derivation function**
function by which keying material is derived from a shared secret (or a key) and other information

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.34**
**key distribution**
transport of a key and other keying material from an entity that either owns the key or generates the key to another entity that is intended to use the key

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.35**
**key-encrypting key**
cryptographic key that is used for the encryption or decryption of other keys

Note 1 to entry: In DLMS®/COSEM it is the master key.

[SOURCE: NIST SP 800-57:2012, Part 1, modified by adding the Note]

**3.2.36**
**key establishment**
procedure that results in keying material that is shared among different parties

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.37**
**key pair**
public key and its corresponding private key; a key pair is used with a public key algorithm

[SOURCE: NIST SP 800-57:2012, Part 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.38**
**key revocation**
function in the lifecycle of keying material; a process whereby a notice is made available to affected entities that keying material should be removed from operational use prior to the end of the established cryptoperiod of that keying material

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.39**
**key-transport**
(pair-wise) key-establishment procedure whereby one party (the sender) selects a value for the secret keying material and then securely distributes that value to another party (the receiver). Contrast with key agreement

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.40**
**key wrapping**
method of encrypting keying material (along with associated integrity information) that provides both confidentiality and integrity protection using a symmetric key

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.41**
**message authentication code**
**MAC**
cryptographic checksum on data that uses a symmetric key to detect both accidental and intentional modifications of data

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.42**
**message digest**
result of applying a hash function to a message. Also known as "hash value".

[SOURCE: FIPS PUB 186-4:2013]

**3.2.43**
**named curve**
set of ECDH domain parameters is also known as a "curve". A curve is a "named curve" if the domain parameters are well known and defined and can be identified by an Object Identifier; otherwise, it is called a "custom curve"

[SOURCE: RFC 5349]

**3.2.44**
**nonce**
time-varying value that has at most an acceptably small chance of repeating. For example, the nonce may be a random value that is generated anew for each use, a timestamp, a sequence number, or some combination of these

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.45**
**non-repudiation**
service that is used to provide assurance of the integrity and origin of data in such a way that the integrity and origin can be verified by a third party as having originated from a specific entity in possession of the private key of the claimed signatory

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.46**
**password**
string of characters (letters, numbers and other symbols) that are used to authenticate an identity or to verify access authorization or to derive cryptographic keys

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.47**
**plaintext**
intelligible data that has meaning and can be understood without the application of decryption

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.48**
**private key**
cryptographic key, used with a public key cryptographic algorithm, which is uniquely associated with an entity and is not made public. In an asymmetric (public) cryptosystem, the private key is associated with a public key. Depending on the algorithm, the private key may be used, for example, to:

1) Compute the corresponding public key,
2) Compute a digital signature that may be verified by the corresponding public key,
3) Decrypt keys that were encrypted by the corresponding public key, or
4) Compute a shared secret during a key-agreement transaction.

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.49**
**protected**
ciphered and /or digitally signed. Protection may be applied to xDLMS APDUs and/or to COSEM data

**3.2.50**
**public key**
cryptographic key used with a public key cryptographic algorithm that is uniquely associated with an entity and that may be made public. In an asymmetric (public) cryptosystem, the public key is associated with a private key. The public key may be known by anyone and, depending on the algorithm, may be used, for example, to:

1) Verify a digital signature that is signed by the corresponding private key,
2) Encrypt keys that can be decrypted using the corresponding private key, or
3) Compute a shared secret during a key-agreement transaction.

[SOURCE: NIST SP 800-57:2012, Part 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.51**
**public-key certificate**
data structure that contains an entity's identifier(s), the entity's public key (including an indication of the associated set of domain parameters) and possibly other information, along with a signature on that data set that is generated by a trusted party, i.e. a certificate authority, thereby binding the public key to the included identifier(s)

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.52**
**public key (asymmetric) cryptographic algorithm**
cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that determining the private key from the public key is computationally infeasible

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.53**
**Public Key Infrastructure**
**PKI**
framework that is established to issue, maintain and revoke public key certificates

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.54**
**receiver <key-transport>**
party that receives secret keying material via a key-transport transaction. Contrast with sender

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.55**
**revoke a certificate**
to prematurely end the operational period of a certificate effective at a specific date and time

[SOURCE: NIST SP 800-32:2001]

**3.2.56**
**Root Certification Authority**
in a hierarchical Public Key Infrastructure, the Certification Authority whose public key serves as the most trusted datum (i.e., the beginning of trust paths) for a security domain

[SOURCE: NIST SP 800-32:2001]

**3.2.57**
**secret key**
cryptographic key that is used with a secret key (symmetric) cryptographic algorithm that is uniquely associated with one or more entities and is not made public. The use of the term "secret" in this context does not imply a classification level, but rather implies the need to protect the key from disclosure

[SOURCE: NIST SP 800-57:2012, Part 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.58**
**security services**
mechanisms used to provide confidentiality, data integrity, authentication or non-repudiation of information

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.59**
**security strength**
**(also "bits of security")**
number associated with the amount of work (that is, the number of operations) that is required to break a cryptographic algorithm or system

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.60**
**self-signed certificate**
public key certificate whose digital signature may be verified by the public key contained within the certificate. The signature on a self-signed certificate protects the integrity of the data, but does not guarantee authenticity of the information. The trust of self-signed certificates is based on the secure procedures used to distribute them

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.61**
**sender <key-transport>**
party that sends secret keying material to the receiver in a key-transport transaction. Contrast with receiver

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.62**
**session key**
cryptographic key established for use for a relatively short period of time

Note 1 to entry: In DLMS®/COSEM the dedicated key is a session key.

**3.2.63**
**shared secret**
secret value that has been computed using a key agreement scheme and is used as input to a key-derivation function/method

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.64**
**signature generation**
uses a digital signature algorithm and a private key to generate a digital signature on data

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.65**
**signature verification**
uses a digital signature algorithm and a public key to verify a digital signature on data

[SOURCE: NIST SP 800-57:2012, Part 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.66**
**signed data**
data upon which a digital signature has been computed

**3.2.67**
**static symmetric key**
key that dis intended for use for a relatively long period of time and is typically intended for use in many instances of a DLMS®/COSEM Application Association

Note 1 to entry: In DLMS®/COSEM it is known as global key.

**3.2.68**
**static key**
key that is intended for use for a relatively long period of time and is typically intended for use in many instances of a cryptographic key establishment scheme. Contrast with an ephemeral key

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.69**
**Subordinate Certification Authority**
in a hierarchical PKI, a Certification Authority (CA) whose certificate signature key is certified by another CA, and whose activities are constrained by that other CA

[SOURCE: NIST SP 800-32:2001]

**3.2.70**
**symmetric key**
single cryptographic key that is used with a secret (symmetric) key algorithm

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.71**
**symmetric key algorithm**
cryptographic algorithm that uses the same secret key for an operation and its complement (e.g., encryption and decryption)

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.72**
**trust anchor**
public key and the name of a certification authority that is used to validate the first certificate in a sequence of certificates. The trust anchor public key is used to verify the signature on a certificate issued by a trust anchor certification authority. The security of the validation process depends upon the authenticity and integrity of the trust anchor. Trust anchors are often distributed as self-signed certificates

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.73**
**trusted party**
trusted party is a party that is trusted by an entity to faithfully perform certain services for that entity. An entity could be a trusted party for itself

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.74**
**trusted third party**
third party, such as a CA, that is trusted by its clients to perform certain services. (By contrast, in a key establishment transaction, the participants, parties U and V, are considered to be the first and second parties)

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.75**
**X.509 certificate**
X.509 public-key certificate or the X.509 attribute certificate, as defined by the ISO/ITU-T X.509 standard. Most commonly (including in this document), an X.509 certificate refers to the X.509 public-key certificate

[SOURCE: NIST SP 800-57:2012, Part 1]

**3.2.76**
**X.509 public key certificate**
a digital certificate containing a public key for entity and a name for the entity, together with some other information that is rendered unforgeable by the digital signature of the certification authority that issued the certificate, encoded in the format defined in the ISO/ITU-T X.509 standard

[SOURCE: NIST SP 800-57:2012, Part 1]

### 3.3 Definitions and abbreviated terms related to the Galois/Counter Mode

**3.3.1**
**Additional Authenticated Data**
**AAD**
input data to the authenticated encryption function that is authenticated but not encrypted

[SOURCE: NIST SP 800-38D:2007]

**3.3.2**
**authenticated decryption**
function of GCM in which the ciphertext is decrypted into the plaintext, and the authenticity of the ciphertext and the AAD are verified

[SOURCE: NIST SP 800-38D:2007]

**3.3.3**
**authenticated encryption**
function of GCM in which the plaintext is encrypted into the ciphertext and an authentication tag is generated on the AAD and the ciphertext

[SOURCE: NIST SP 800-38D:2007]

**3.3.4**
**authentication tag**
**Tag, T**
cryptographic checksum on data that is designed to reveal both accidental errors and the intentional modification of the data

[SOURCE: NIST SP 800-38D:2007]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.3.5**
**block cipher**
parameterized family of permutations on bit strings of a fixed length; the parameter that determines the permutation is a bit string called the key

[SOURCE: NIST SP 800-38D:2007]

**3.3.6**
**ciphertext**
encrypted form of the plaintext

[SOURCE: NIST SP 800-38D:2007]

**3.3.7**
**fixed field**
in the deterministic construction of IVs, the field that identifies the device or context for the instance of the authenticated encryption function

[SOURCE: NIST SP 800-38D:2007]

**3.3.8**
**fresh**
for a newly generated key, the property of being unequal to any previously used key

[SOURCE: NIST SP 800-38D:2007]

**3.3.9**
**GCM**
Galois/Counter Mode

[SOURCE: NIST SP 800-38D:2007]

**3.3.10**
**initialization Vector**
**IV**
nonce that is associated with an invocation of authenticated encryption on a particular plaintext and AAD

Note 1 to entry: For the purposes of this document, the invocation field is the invocation counter.

[SOURCE: NIST SP 800-38D:2007]

**3.3.11**
**invocation field**
in the deterministic construction of IVs, the field that identifies the sets of inputs to the authenticated encryption function in a particular device or context

[SOURCE: NIST SP 800-38D:2007]

**3.3.12**
**key**
parameter of the block cipher that determines the selection of the forward cipher function from the family of permutations

[SOURCE: NIST SP 800-38D:2007]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.3.13**
**plaintext**
***P***
input data to the authenticated encryption function that is both authenticated and encrypted

[SOURCE: NIST SP 800-38D:2007]

**3.3.14**
**security control byte**
***SC***
byte that provides information on the ciphering applied

**3.3.15**
**security header**
***SH***
concatenation of the security control byte $SC$ and the invocation counter: $SH = SC \text{ II } IC$.

## 3.4 General abbreviated terms

| Abbreviation | Meaning |
|---|---|
| .cnf | .confirm service primitive |
| .ind | .indication service primitive |
| .req | .request service primitive |
| .res | .response service primitive |
| AA | Application Association |
| AARE | A-Associate Response – an APDU of the ACSE |
| AARQ | A-Associate Request – an APDU of the ACSE |
| ACPM | Association Control Protocol Machine |
| ACSE | Association Control Service Element |
| AE | Application Entity |
| AES | Advanced Encryption Standard |
| AL | Application Layer |
| AP | Application Process |
| APDU | Application Layer Protocol Data Unit |
| API | Application Programming Interface |
| ASE | Application Service Element |
| ASO | Application Service Object |
| ATM | Asynchronous Transfer Mode |
| A-XDR | Adapted Extended Data Representation |
| base_name | The short_name corresponding to the first attribute ("logical_name") of a COSEM object |
| BER | Basic Encoding Rules |
| BD | Block Data |
| BN | Block Number |
| BNA | Block Number Acknowledged |
| BS | Bit string |
| BTS | Block Transfer Streaming |
| BTW | Block Transfer Window |
| CA | Certification Authority |
| CF | Control Function |
| CL | Connectionless |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Abbreviation | Meaning |
|---|---|
| class_id | COSEM interface class identification code |
| CMP | Certificate Management Protocol. Refer to RFC 4210. |
| CO | Connection-oriented |
| COSEM | Companion Specification for Energy Metering |
| COSEM_on_IP | The TCP-UDP/IP based COSEM communication profile |
| CRC | Cyclic Redundancy Check |
| CRL | Certificate revocation list. Refer to RFC 5280. |
| CSR | Certificate Signing Request |
| DCE | Data Communication Equipment (communications interface or modem) |
| DCS | Data Collection System |
| DISC | Disconnect (a HDLC frame type) |
| DLMS® | Device Language Message Specification |
| DM | Disconnected Mode (a HDLC frame type) |
| DSA | Digital Signature Algorithm specified in FIPS PUB 186-4:2013 |
| DSAP | Data Link Service Access Point |
| DSO | Energy Distribution System Operator |
| DTE | Data Terminal Equipment (computers, terminals or printers) |
| ECC | Elliptic Curve Cryptography |
| ECDH | Elliptic Curve Diffie-Hellman key agreement protocol |
| ECDSA | Elliptic Curve Digital Signature Algorithm specified in ANSI X9.62 and FIPS PUB 186-4:2013 |
| ECP | Elliptic Curve Point |
| EUI-64 | 64-bit Extended Unique Identifier |
| FCS | Frame Check Sequence |
| FDDI | Fibre Distributed Data Interface |
| FE | Field Element (in relation with public key algorithms) |
| FIPS | Federal Information Processing Standard |
| FRMR | Frame Reject (a HDLC frame type) |
| FTP | File Transfer Protocol |
| GAK | Global Authentication Key |
| GBEK | Global Broadcast Encryption Key |
| GBT | General Block Transfer |
| GCM | Galois/Counter Mode (GCM), an algorithm for authenticated encryption with associated data |
| GMAC | A specialization of GCM for generating a message authentication code (MAC) on data that is not encrypted |
| GMT | Greenwich Mean Time |
| GSM | Global System for Mobile communications |
| GUEK | Global Unicast Encryption Key |
| GW | Gateway |
| HCS | Header Check Sequence |
| HDLC | High-level Data Link Control |
| HES | Head End System, also known as Data Collection System<br>NOTE The HES may be owned by the energy provider or the utility |
| HHU | Hand Held Unit |
| HLS | High Level Security (COSEM) |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Abbreviation | Meaning |
|---|---|
| HMAC | Keyed-Hash Message Authentication Code specified in FIPS 198-1 |
| HSM | Hardware Security Module |
| HTTP | Hypertext Transfer Protocol |
| I | Information (a HDLC frame type) |
| IANA | Internet Assigned Numbers Authority |
| IC | Interface Class |
| IEEE | Institute of Electrical and Electronics Engineers |
| IETF | Internet Engineering Task Force |
| IP | Internet Protocol |
| ISO | International Organization for Standardization |
| IV | Initialization Vector |
| KEK | Key Encrypting Key |
| LAN | Local Area Network |
| LB | Last Block |
| LDN | Logical Device Name |
| LLC | Logical Link Control (Sublayer) |
| LLS | Low Level Security |
| LNAP | Local Network Access Point |
| LPDU | LLC Protocol Data Unit |
| L-SAP | LLC sublayer Service Access Point |
| LSB | Least Significant Bit |
| LSDU | LLC Service Data Unit |
| m | mandatory, used in conjunction with attribute and method definitions |
| MAC | Medium Access Control (sublayer) |
| MAC | Message Authentication Code (cryptography) |
| MIB | Management Information Base |
| MSAP | MAC sublayer Service Access Point (in the HDLC based profile, it is equal to the HDLC address) |
| MSB | Most Significant Bit |
| MSC | Message Sequence Chart |
| MSDU | MAC Service Data Unit |
| N(R) | Receive sequence Number |
| N(S) | Send sequence Number |
| NDM | Normal Disconnected Mode |
| NIST | National Institute of Standards and Technology |
| NNAP | Neighbourhood Network Access Point |
| NRM | Normal Response Mode |
| o | optional, used in conjunction with attribute and method definitions |
| OBIS | Object Identification System |
| OCSP | Online Certificate Status Protocol |
| OID | Object Identifier |
| OOB | Out of Band |
| OS | Octet string |
| OSI | Open System Interconnection |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Abbreviation | Meaning |
|---|---|
| OTA | Over The Air |
| P/F | Poll/Final |
| PAR | Positive Acknowledgement with Retransmission |
| PDU | Protocol data unit |
| PhL | Physical Layer |
| PHSDU | PH SDU |
| PKCS | Public Key Cryptography Standard, established by RSA Laboratories |
| PKI | Public Key Infrastructure |
| PLC | Power line carrier |
| PPP | Point-to-Point Protocol |
| PSDU | Physical layer Service Data Unit |
| PSTN | Public Switched Telephone Network |
| RA | Registration Authority |
| RLRE | A-Release Response – an APDU of the ACSE |
| RLRQ | A-Release Request – an APDU of the ACSE |
| RNG | Random Number Generator |
| RNR | Receive Not Ready (a HDLC frame type) |
| RR | Receive Ready (a HDLC frame type) |
| RSA | Algorithm developed by Rivest, Shamir and Adelman; specified in ANS X9.31 and PKCS #1. |
| SAP | Service Access Point |
| SDU | Service Data Unit |
| SHA | Secure Hash Algorithm; specified in FIPS PUB 180-4:2012. |
| SNMP | Simple Network Management Protocol |
| SNRM | Set Normal Response Mode (a HDLC frame type) |
| STR | Streaming |
| tbsCertificate | To be signed certificate |
| TCP | Transmission Control Protocol |
| TDEA | Triple Data Encryption Algorithm |
| TL | Transport Layer |
| TPDU | Transport Layer Protocol Data Unit |
| TWA | Two Way Alternate |
| UA | Unnumbered Acknowledge (a HDLC frame type) |
| UDP | User Datagram Protocol |
| UI | Unnumbered Information (a HDLC frame type) |
| UNC | Unbalanced operation Normal response mode Class |
| USS | Unnumbered Send Status |
| V(R) | Receive state Variable |
| V(S) | Send state Variable |
| VAA | Virtual Application Association |
| WPDU | Wrapper Protocol Data Unit |
| xDLMS ASE | Extended DLMS® Application Service Element |
| See also list of abbreviations specific to a cryptographic algorithm in the relevant clauses. | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 3.5 Symbols related to the Galois/Counter Mode

| Symbol | Meaning |
|---|---|
| $A$ | Additional Authenticated Data, AAD |
| $AK$ | Authentication key, a parameter that is part of the AAD |
| $C$ | Ciphertext |
| $EK$ | Encryption key, i.e. the block cipher key |
| $IC$ | Invocation counter, part of the initialization vector. See also invocation field. |
| $IV$ | Initialization Vector |
| $\mathrm{len}(X)$ | The bit length of the bit string $X$. |
| $LEN(X)$ | The octet length of the octet string $X$. |
| $P$ | Plaintext |
| $SC$ | Security Control Byte |
| $SH$ | Security Header |
| $Sys\text{-}T$ | System title |
| $T$ | Authentication tag |
| $t$ | The bit length of the authentication tag.<br>NOTE This is the same as $\mathrm{len}(T)$ |
| $X \mid\mid Y$ | Concatenation of two strings, $X$ and $Y$. |

## 3.6 Symbols related the ECDSA algorithm

| Symbol | Meaning |
|---|---|
| $d$ | The ECDSA private key, which is an integer in the interval $[1, n-1]$. |
| $Q = (x_Q, y_Q)$ | An ECDSA public key. The coordinates $x_Q$ and $y_Q$ are integers in the interval $[0, q-1]$, and $Q = dG$. |
| $k$ | The ECDSA per-message secret number, which is an integer in the interval $[1, n-1]$. |
| $r$ | One component of an ECDSA digital signature. It is an integer in $[1, n-1]$. See the definition of $(r, s)$. |
| $s$ | One component of an ECDSA digital signature. It is an integer in $[1, n-1]$. See the definition of $(r, s)$. |
| $(r, s)$ | An ECDSA digital signature, where $r$ and $s$ are the digital signature components. |
| $M$ | The message that is signed using the digital signature algorithm. |
| $\mathrm{Hash}(M)$ | The result of a hash computation (message digest or hash value) on message M using an approved hash function. |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 3.7 Symbols related to the key agreement algorithms

| Symbol | Meaning |
|---|---|
| $d_{e,\,U}$, $d_{e,V}$ | Party U's and Party V's ephemeral private keys. These are integers in the range [1, n-1]. |
| $d_{s,\,U}$, $d_{s,V}$ | Party U's and Party V's static private keys. These are integers in the range [1, n-1]. |
| $ID_{\mathrm{U}}$ | The identifier of Party U (the initiator) |
| $ID_{\mathrm{V}}$ | The identifier of Party V (the responder) |
| $Q_{e,\,U}$, $Q_{e,V}$ | Party U's and Party V's ephemeral public keys. These are points on the elliptic curve defined by the domain parameters. |
| $Q_{s,\,U}$, $Q_{s,V}$ | Party U's and Party V's static public keys. These are points on the elliptic curve defined by the domain parameters. |
| U, V | Represent the two parties in a (pair-wise) key establishment scheme. |
| $Z$ | A shared secret (represented as a byte string) that is used to derive secret keying material using a key derivation method. *Source: NIST SP 800-56A Rev. 2: 2013* |

## 4 Overview of DLMS®/COSEM

### 4.1 Information exchange in DLMS®/COSEM

#### 4.1.1 General

This subclause 4.1 introduces the main concepts of information exchange in DLMS®/COSEM.

The objective of DLMS®/COSEM is to specify a standard for a business domain oriented interface object model for metering devices and systems, as well as services to access the objects. Communication profiles to transport the messages through various communication media are also specified.

The term "metering devices" is an abstraction; consequently "metering device" may be any type of device for which this abstraction is suitable.

The COSEM object model is specified in IEC 62056-6-2:2021. The COSEM objects provide a view of the functionality of metering devices through their communication interfaces.

This document specifies the DLMS®/COSEM application layer and the rules for specifying DLMS®/COSEM communication profiles; see Annex A.

The key characteristics of data exchange using DLMS®/COSEM are the following:

- metering devices can be accessed by various parties: clients and third parties;
- mechanisms to control access to the resources of the metering device are provided; these mechanisms are made available by the DLMS®/COSEM AL and the COSEM objects ("Association SN / LN" object, "Security setup" object);
- security and privacy is ensured by applying cryptographical protection to xDLMS messages and to COSEM data;
- low overhead and efficiency is ensured by various mechanisms including selective access, compact encoding and compression;
- at a metering site, there may be single or multiple metering devices. In the case of multiple metering devices at a metering site, a single access point can be made available;
- data exchange may take place either remotely or locally. Depending on the capabilities of the metering device, local and remote data exchange may be performed simultaneously without interfering with each other;

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- various communication media can be used on local networks (LN), neighbourhood networks (NN) and wide area networks (WAN).

The key element to ensure that the above requirements are met is the Application Association (AA) – determining the contexts of the data exchange – provided by the DLMS®/COSEM AL. For details, see the relevant clauses below.

### 4.1.2 Communication model

DLMS®/COSEM uses the concepts of the Open Systems Interconnection (OSI) model to model information exchange between meters and data collection systems.

NOTE Information in this context comprises xDLMS messages and COSEM data.

Concepts, names and terminology used below relate to the OSI reference model described in ISO/IEC 7498-1:1994. Their use is outlined in this clause and further developed in other clauses.

Application functions of metering devices and data collection systems are modelled by application processes (APs).

Communication between APs is modelled by communication between application entities (AEs). An AE represents the communication functions of an AP. There may be multiple sets of OSI communication functions in an AP, so a single AP may be represented by multiple AEs. However, each AE represents a single AP. An AE contains a set of communication capabilities called application service elements (ASEs). An ASE is a coherent set of integrated functions. These ASEs may be used independently or in combination. See also 4.2.2.

Data exchange between data collection systems and metering devices is based on the client/server model where data collection systems play the role of the client and metering devices play the role of the server. The client sends service requests to the server which sends service responses. In addition the server may initiate unsolicited service requests to inform the client about events or to send data on pre-configured conditions. See also 4.1.6.

In general, the client and the server APs are located in separate devices. Therefore, message exchange takes place via a protocol stack as shown in Figure 1.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

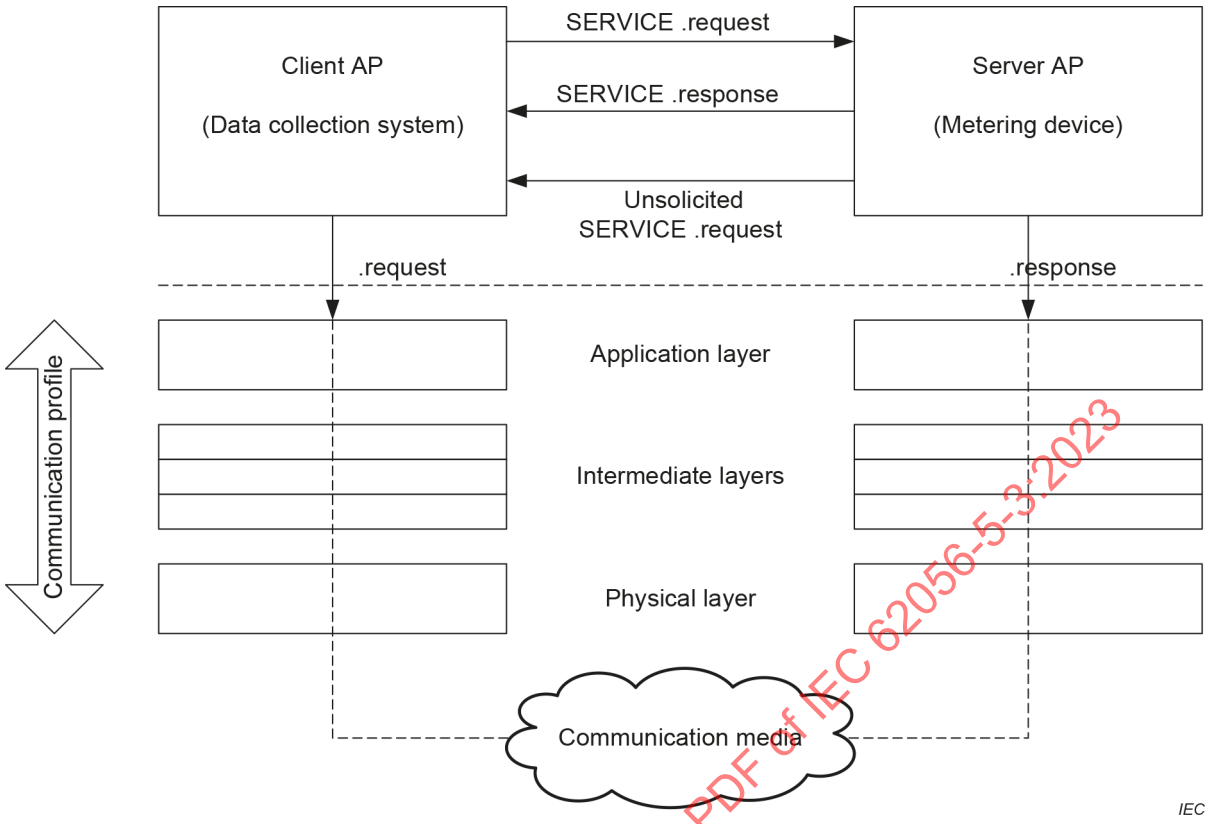



**Figure 1 – Client–server model and communication protocols**

### 4.1.3 Naming and addressing

#### 4.1.3.1 General

Naming and addressing are important aspects in communication systems. A name identifies a communicating entity. An address identifies where that entity can be found. Names are mapped to addresses; this is known as the process of binding. Figure 2 shows the main elements of naming and addressing in DLMS®/COSEM.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

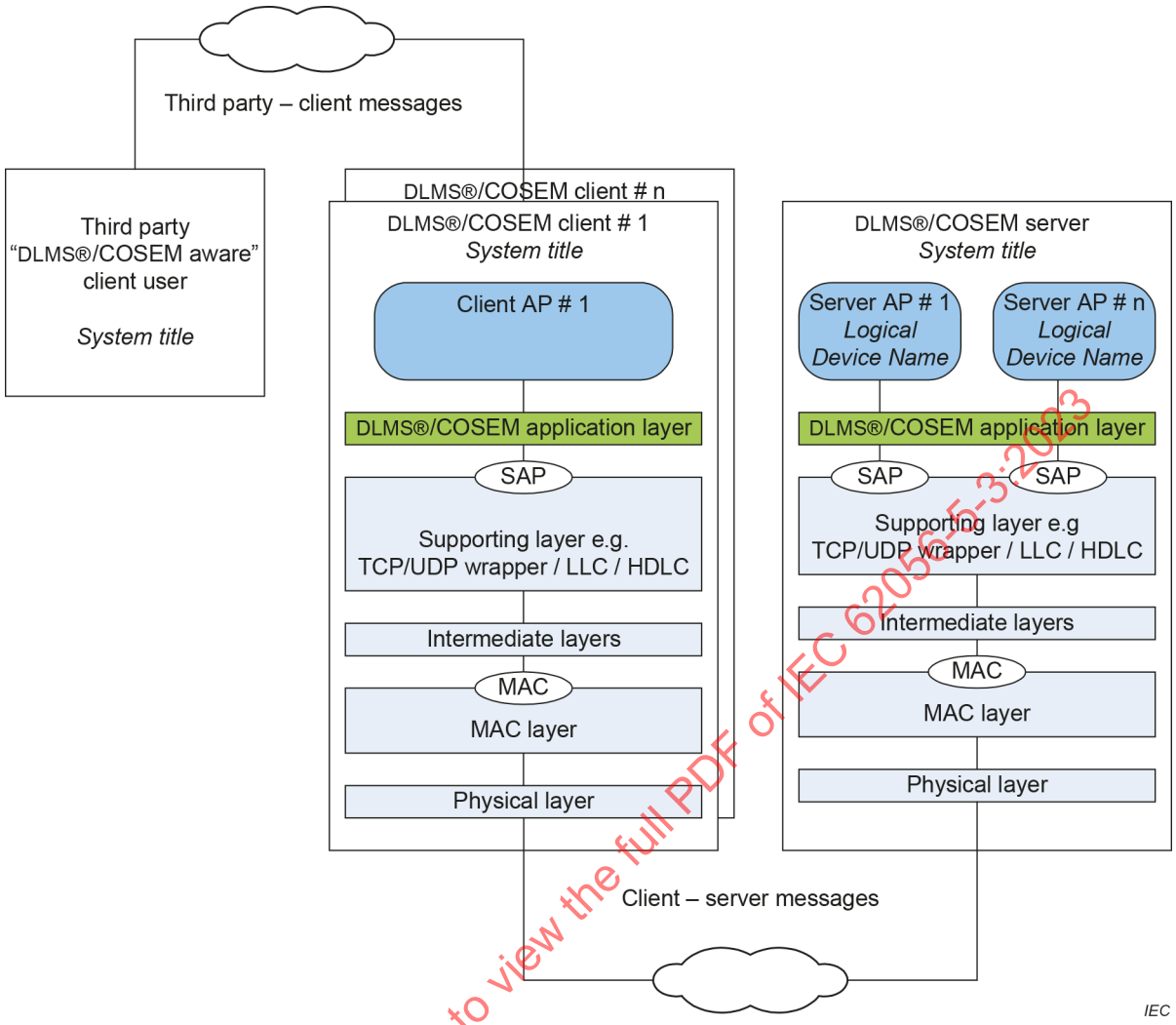



**Figure 2 – Naming and addressing in DLMS®/COSEM**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 4.1.3.2 Naming

DLMS®/COSEM entities, including clients, servers as well as third party systems shall be uniquely named by their *system title*. System titles shall be permanently assigned.

Server physical devices may host one or more logical devices (LDs). LDs shall be uniquely identified by their Logical Device Name (LDN). LDs hosted by the same physical device share the system title. System titles are specified in 4.1.3.4. Logical device names are specified in 4.1.3.5.

#### 4.1.3.3 Addressing

Each physical device shall have an appropriate address. It depends on the communication profile and may be a phone number, a MAC address, an IP network address or a combination of these.

NOTE For example, in the case of the 3-layer, connection-oriented, HDLC based communication profile, the lower HDLC address is the MAC address.

Physical device addresses may be pre-configured or may be assigned during a registration process, which also involves binding between the addresses and the system titles.

Each DLMS®/COSEM client and each server – a COSEM logical device – is bound to a Service Access Point (SAP). The SAPs reside in the supporting layer of the DLMS®/COSEM AL. Depending on the communication profile the SAP may be a TCP-UDP/IP wrapper address, an upper HDLC address, an LLC address etc. On the server side, this binding is modelled by the "SAP Assignment" IC; see IEC 62056-6-2:2021, 4.4.5.

The values of the SAPs on the client and the server side are specified in Table 1. The length of the SAPs depends on the communication profile.

**Table 1 – Client and server SAPs**

| | |
|---|---|
| **Client SAPs** | |
| No-station | 0x00 |
| Client Management Process / CIASE [1] | 0x01 |
| Public Client | 0x10 |
| *Open for client AP assignment* | 0x02…0x0F |
| | 0x11 and up |
| **Server SAPs** | |
| No-station / CIASE [1] | 0x00 |
| Management Logical Device | 0x01 |
| Reserved for future use | 0x02…0x0F |
| *Open for server SAP assignment* | 0x10 and up |
| All-station (Broadcast) | Communication profile specific |
| [1] In the case of the DLMS®/COSEM S-FSK PLC profile, see IEC 62056-8-3. | |
| NOTE Depending on the supporting layer, the SAPs may be represented on one or more bytes. | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 4.1.3.4 System title

The system title *Sys-T* shall uniquely identify each DLMS®/COSEM entity that may be server, a client or a third party that can access servers via clients. The system title:

- shall be 8 octets long;
- shall be unique.

The leading (i.e. the 3 leftmost) octets should hold the three-letter manufacturer ID[4]. This is the same as the leading three octets of the Logical Device Name, see 4.1.3.5. The remaining 5 octets shall ensure uniqueness.

NOTE It can be derived for example from the last 12 digits of the manufacturing number, up to 999 999 999 999. This value converts to 0xE8D4A50FFF. Values above this, up to 0xFFFFFFFFFF (decimal 1 099 511 627 775) can also be used, but these values cannot be mapped to the last 12 digits of the manufacturing number.

Project specific companion specifications may specify a different structure. In that case, the details should be specified by the naming authority designated as such for the project.

The use of the system title in cryptographic protection of xDLMS messages and COSEM data is further specified in 5.3 and 5.7.

---

[4] Administered by the FLAG Association in co-operation with the DLMS UA.

Before the cryptographic security algorithms can be used – this requires a ciphered application context – the peers have to exchange system titles. The following possibilities are available:

- during the communication media specific registration process. For example, when the S-FSK PLC profile is used, system titles are exchanged during the registration process using the CIASE protocol; see IEC 62056-8-3;
- in all communication profiles, system titles may be exchanged during AA establishment using the COSEM-OPEN service, see 6.2, carried the AARQ / AARE APDU. If the system titles sent / received during AA establishment are not the same as the ones exchanged during the registration process, the AA shall be rejected;
- by writing the *client_system_title* attribute and by reading the *server_system_title* attribute of "Security setup" objects, see IEC 62056-6-2:2021, 4.4.7.

In the case of broadcast communication, only the client sends the system title to the server.

#### 4.1.3.5 Logical Device Name

Logical Device Name (LDN) shall be as specified in IEC 62056-6-2:2021, 4.1.4.6.

#### 4.1.3.6 Client user identification

The client user identification mechanism allows a server to distinguish between different users on the client side and to log their activities accessing the meter. It is specified in IEC 62056-6-2:2021, 4.4.2. Naming of client users is outside the scope of this International Standard.

### 4.1.4 Connection oriented operation

The DLMS®/COSEM AL is connection oriented. See also 4.2.3.

A communication session consists of three phases, as it is shown in Figure 3:

first, an application level connection, called Application Association (AA), is established between a client and a server AE; see also 4.2.3. Before initiating the establishment of an AA, the peer PhLs of the client and server side protocol stacks have to be connected. The intermediate layers may have to be connected or not. Each layer, which needs to be connected, may support one or more connections simultaneously;

- once the AA is established, message exchange can take place;
- at the end of the data exchange, the AA is released.

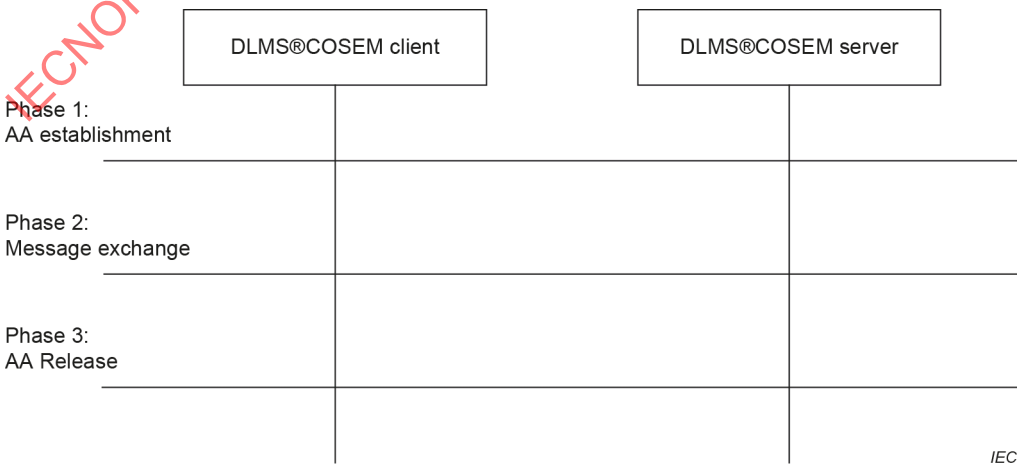



**Figure 3 – A complete communication session in the CO environment**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

For the purposes of very simple devices, one-way communicating devices, and for multicasting and broadcasting pre-established AAs are also allowed. For such AAs the full communication session may include only the message exchange phase: it can be considered that the connection establishment phase has been already done somewhere in the past. Pre-established AAs cannot be released. See also 7.2.4.4.

### 4.1.5 Application associations

#### 4.1.5.1 General

Application Associations (AAs) are logical connections between a client and a server AE. AAs may be established on the request of a client using the services of the connection-oriented ACSE of the AL or may be pre-established. They may be confirmed or unconfirmed. See also 4.2.3.

NOTE 1 A pre-established AA can be considered to have been established in the past.

NOTE 2 Servers cannot initiate the establishment of an AA.

A COSEM logical device may support one or more AAs, each with a different client. Each AA determines the contexts in which information exchange takes place.

A confirmed AA is proposed by the client and accepted by the server provided that:

- the user of the client is known by the server, see 4.1.3.6;
- the application context proposed by the client – see 4.1.5.2 – is acceptable for the server;
- the authentication mechanism proposed by the client – see 4.1.5.3 – is acceptable for the server and the authentication is successful;
- the elements of the xDLMS context – see 4.1.5.4 – can be successfully negotiated between the client and the server.

An unconfirmed AA is also proposed by a client with the assumption that the server will accept it. No negotiation takes place. Unconfirmed AAs are useful for sending broadcast messages from the client to servers.

AAs are modelled by COSEM "Association SN / LN" objects that hold the SAPs identifying the associated partners, the name of the *application context*, the name of the *authentication mechanism*, and the *xDLMS context.*

The "Association SN / LN" objects also determine a specific set of access rights to COSEM object attributes and methods and they point to (reference) a "Security setup" object that hold the elements of the security context. The access rights and the security context may be different in each AA. These objects are specified in IEC 62056-6-2:2021, 4.4.3, 4.4.4.

#### 4.1.5.2 Application context

The application context determines:

- the set of Application Service Elements (ASEs) present in the AL;
- the referencing style of COSEM object attributes and methods: short name (SN) referencing or logical name (LN) referencing. See also 4.2.4.3.1;
- the transfer syntax;
- whether ciphering is used or not.

Application contexts are identified by names, see 7.2.2.2.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 4.1.5.3 Authentication

In communication systems entity authentication is a fundamentally important security service. The goal of entity authentication is to establish whether the claimant of a certain identity is in fact who it claims to be. In order to achieve this goal, there should be a pre-existing relation which links the entity to a secret.

In DLMS®/COSEM, authentication takes place during AA establishment.

In confirmed AAs either the client (unilateral authentication) or both the client and the server (mutual authentication) can authenticate itself.

In an unconfirmed AA, only the client can authenticate itself.

In pre-established AAs, authentication of the communicating partners is not available.

Once the AA is established, COSEM object attributes and methods can be accessed using xDLMS services subject to the prevailing security context and access rights in the given AA.

The authentication mechanisms are specified in 5.2.2.2. The authentication mechanisms are identified by names, see 7.2.2.3.

#### 4.1.5.4 xDLMS context

The xDLMS context determines the set of xDLMS services and capabilities that can be used in a given AA. See 4.2.4.

#### 4.1.5.5 Security context

The security context is relevant when the application context stipulates ciphering. It comprises the security suite, the security policy, the security keys and other security material. See also 5.2.3. It is managed by "Security setup" objects.

#### 4.1.5.6 Access rights

Access rights determine the rights of the client(s) to access COSEM object attributes and methods within an AA. The set of access rights depend on the role of the client and is pre-configured in the server. See also 5.2.4.

NOTE The roles and the related access rights are subject to project specific companion specifications. Examples for roles are meter reader, meter service / communication service / energy provider, manufacturer, end user, etc.

### 4.1.6 Messaging patterns

The messaging patterns available between a DLMS®/COSEM client and server are shown in Figure 4.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

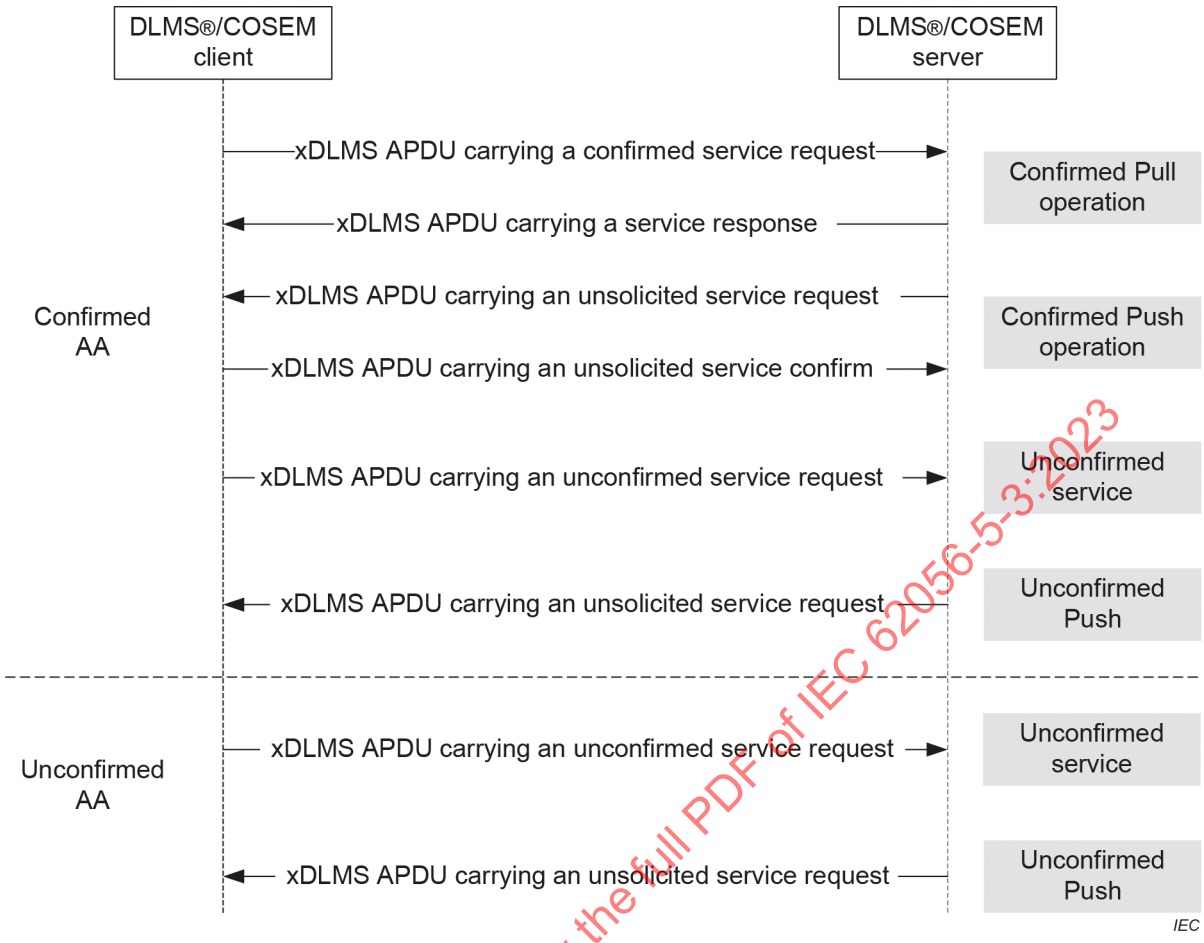



**Figure 4 – DLMS®/COSEM messaging patterns**

In confirmed AAs:

- the client can send confirmed service requests and the server responds: *pull operation*;
- the client can send unconfirmed service requests. The server does not respond;
- the server can send unsolicited service requests to the client: *push operation*.

NOTE The unsolicited services may be InformationReport (with SN referencing), EventNotification (with LN referencing) or DataNotification (used both with SN and LN referencing).

In unconfirmed AAs:

- only the client can initiate service requests and only unconfirmed ones. The server cannot respond and it cannot initiate service requests.

### 4.1.7 Data exchange between third parties and DLMS®/COSEM servers

Third parties – that are outside the DLMS®/COSEM client-server relationship – may also exchange information with servers, using a client as a broker. To support end-to-end security, such third parties shall be "DLMS®/COSEM aware" meaning that they shall be able to send messages to the client that contain properly formatted xDLMS APDUs carrying properly formatted COSEM data, and that they shall be able to process messages received from the server via the client. See also 5.2.5, Figure 14.

Messages from the server to the third party may be solicited or unsolicited.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 4.1.8 Communication profiles

Communication profiles specify how the DLMS®/COSEM AL and the COSEM data model modelling the Application Process (AP) are supported by the lower, communication media specific protocol layers.

Communication profiles comprise a number of protocol layers. Each layer has a distinct task and provides services to its upper layer and uses services of its supporting layer(s). The client and server COSEM APs use the services of the highest protocol layer, that of the DLMS®/COSEM AL. This is the only protocol layer containing COSEM specific element(s): the xDLMS ASE; see 4.2.4. It may be supported by any layer capable of providing the services required by the DLMS®/COSEM AL. The number and type of lower layers depend on the communication media used.

A given set of protocol layers with the DLMS®/COSEM AL and the COSEM object model on top constitutes a particular DLMS®/COSEM communication profile. Each profile is characterized by the protocol layers included and their parameters.

Figure 5 shows a generic DLMS®/COSEM communication profile, including:

- the COSEM object model modelling the Application Process. For each communication media, media-specific setup interface classes are specified;
- the DLMS®/COSEM application layer;
- the DLMS®/COSEM transport layer, present in internet capable profiles;
- the convergence layers that bind the MAC layer to the DLMS®/COSEM AL either directly or through the DLMS®/COSEM transport layer;
- the media specific physical and MAC layers; and
- the connection managers.

A single physical device may support more than one communication profile to allow data exchange using various communication media. In such cases it is the task of the client side AP to decide which communication profile should be used.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

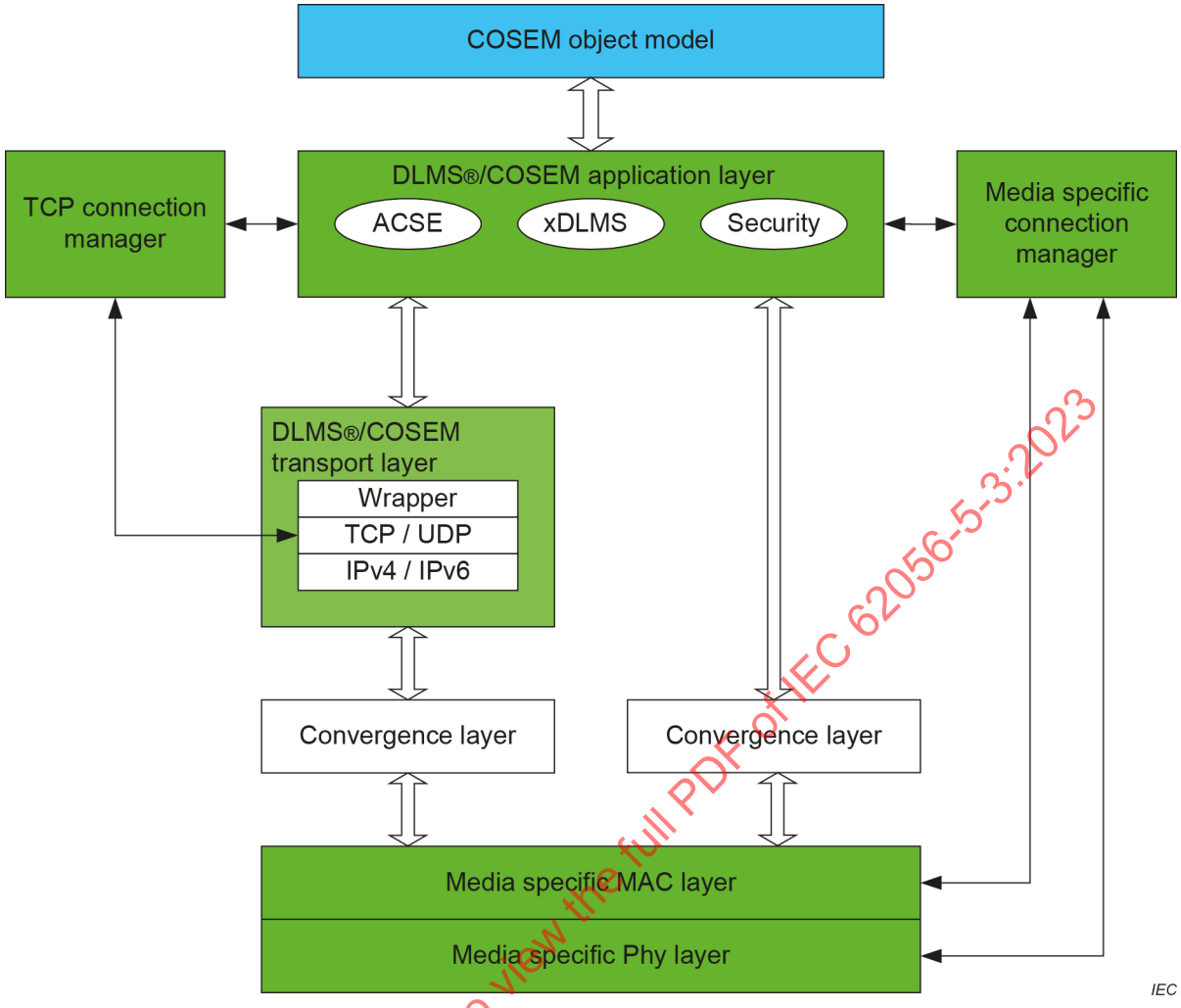



**Figure 5 – DLMS®/COSEM generic communication profile**

Using the DLMS®/COSEM application layer in various communications profiles Communication is specified in Annex A. Communication profile standards are specified in other parts of the IEC 62056 DLMS®/COSEM suite:

- the 3-layer, connection-oriented, HDLC based communication profile, is specified in IEC 62056-7-6;
- the TCP-UDP/IP based communication profiles (COSEM_on_IP), is specified in IEC 62056-9-7;
- the S-FSK PLC profile, is specified in IEC 62056-8-3.
- the wired and wireless M-Bus profile is specified in IEC 62056-7-3:2017.
- the LPWAN profile that can be used over LoRaWAN networks, specified in IEC 62056-8-11:–;
- the Wi-SUN profile specified in IEC 62056-8-12:–.

NOTE Further communication profiles may be specified in the future.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 4.1.9 Model of a DLMS®/COSEM metering system

Figure 6 shows a model of a DLMS®/COSEM metering system.

Metering equipment are modelled as a set of logical devices, hosted in a single physical device. Each logical device represents a server AP and models a subset of the functionality of the metering equipment as these are seen through its communication interfaces. The various functions are modelled using COSEM objects.

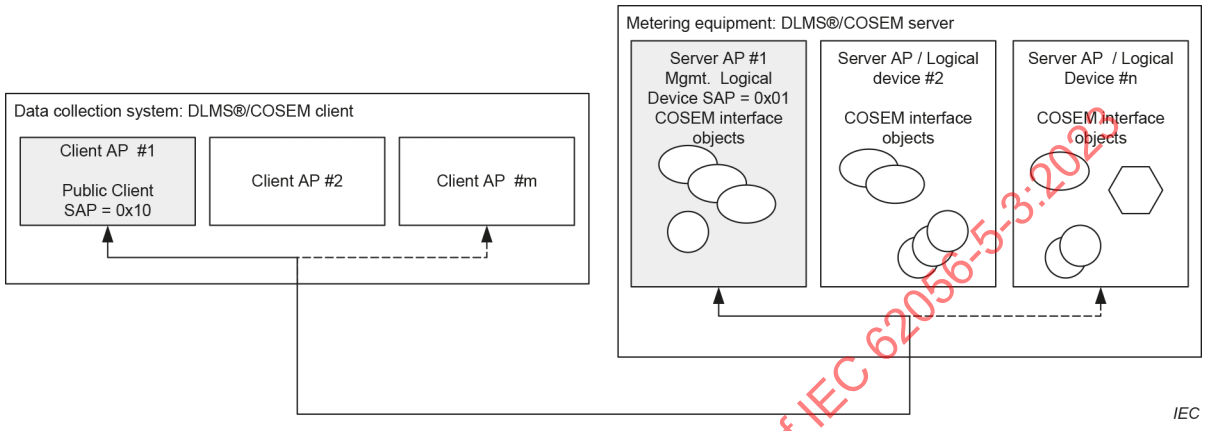



**Figure 6 – Model of a DLMS®/COSEM metering system**

Data collection systems are modelled as a set of client APs that may be hosted by one or several physical devices. Each client AP may have different roles and access rights, granted by the metering equipment.

The Public Client and the Management Logical Device APs have a special role and they shall be always present.

See more in IEC 62056-6-2:2021, 4.1.7 and 4.1.8.

### 4.1.10 Model of DLMS®/COSEM servers

Figure 7 shows the model of two DLMS®/COSEM servers as an example. One of them uses a 3-layer, CO, HDLC based communication profile, and the other one uses a TCP-UDP/IP based communication profile.

The metering equipment on the left hand side comprises "n" logical devices and supports the 3-layer, CO, HDLC based communication profile.

The DLMS®/COSEM AL is supported by the HDLC based data link layer. Its main role is to provide a reliable data transfer between the peer layers. It also provides addressing of the logical devices in such a way, that each logical device is bound to a single HDLC address. The Management Logical Device is always bound to the address 0x01. To allow creating a local network so that several metering devices at a given metering site can be reached through a single access point, another address, the physical address is also provided by the data link layer. The logical device addresses are referred to as upper HDLC addresses, while the physical device address is referred to as a lower HDLC address. See also IEC 62056-7-6.

The PhL supporting the data link layer provides serial bit transmission between physical devices hosting the client and server applications. This allows using various interfaces, like RS 232, RS 485, 20 mA current loop, etc., to transfer data locally through PSTN and GSM networks etc.

The metering equipment on the right hand side comprises "m" logical devices.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The DLMS®/COSEM AL is supported by the DLMS®/COSEM TL, comprising the internet TCP or UDP layer and a wrapper. The main role of the wrapper is to adapt the OSI-style service set, provided by the DLMS®/COSEM TL to and from TCP and UDP function calls. It also provides addressing for the logical devices, binding them to a SAP called wrapper port. The Management Logical Device is always bound to wrapper port 0x01. Finally, the wrapper provides information about the length of the APDUs transmitted, to help the peer to recognise the end of the APDU. This is necessary due the streaming nature of TCP.

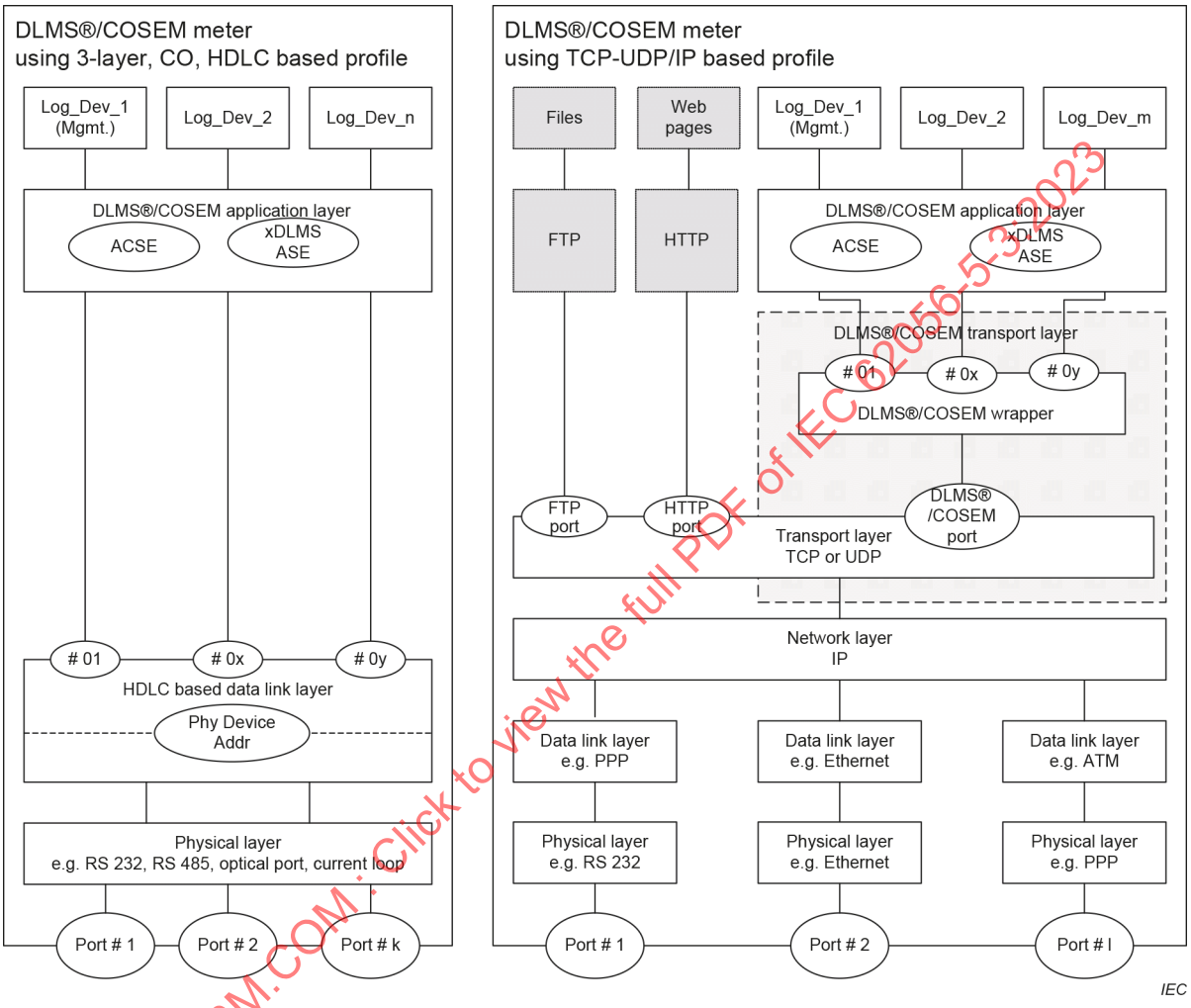



**Figure 7 – DLMS®/COSEM server model**

Through the wrapper, the DLMS®/COSEM AL is bound to a TCP or UDP port number, which is used for the DLMS®/COSEM application. The presence of the TCP and UDP layers allows incorporating other internet applications, like FTP or HTTP, bound to their standard ports respectively.

The TCP layer is supported by the IP layer, which is in turn may be supported by any set of lower layers depending on the communication media to be used (for example Ethernet, PPP, IEEE 802, or IP-capable PLC lower layers, etc.).

Obviously, in a single server it is possible to implement several protocol stacks, with the common DLMS®/COSEM AL being supported by distinct sets of lower layers. This allows the server to exchange data via various communication media with clients in different AAs. Such a structure would be similar to the structure of a DLMS®/COSEM client show below.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 4.1.11 Model of a DLMS®/COSEM client

Figure 8 shows the model of a DLMS®/COSEM client as an example.

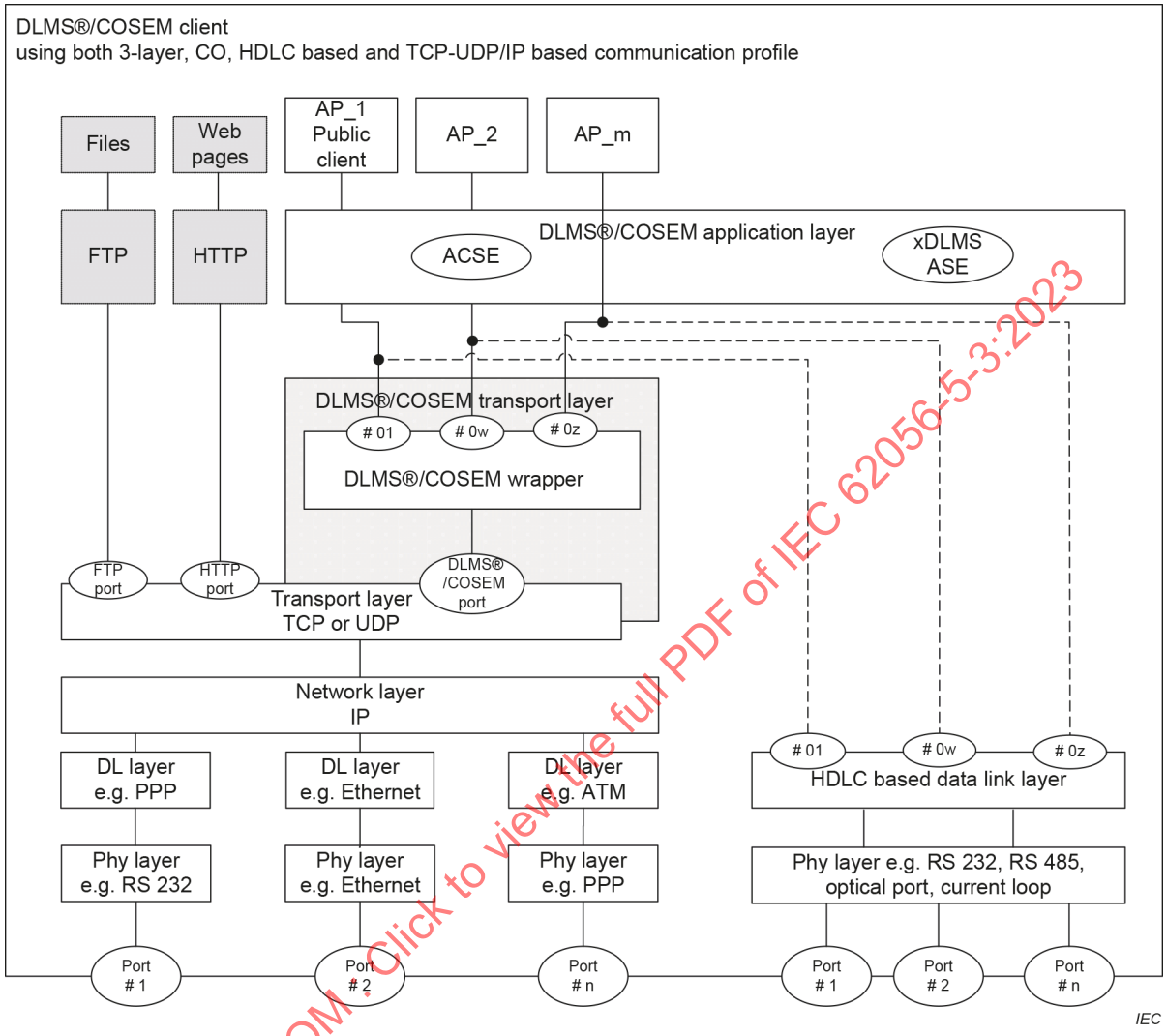



**Figure 8 – Model of a DLMS®/COSEM client using multiple protocol stacks**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The model of the client – obviously – is very similar to the model of the servers:

- in this particular model, the DLMS®/COSEM AL is supported either by the HDLC based data link layer or the DLMS®/COSEM TL, meaning that the AL uses the services of one or the other as determined by the APs. In other words, the APDUs are received from or sent through the appropriate supporting layer, which in turn use the services of its supporting layer respectively;
- unlike on the server side, the addressing provided by the HDLC layer has a single level only, that of the Service Access Points (SAP) of each Application Process (AP).

As explained, client APs and server APs are identified by their SAPs. Therefore, an AA between a client and a server side AP can be identified by a pair of client and server SAPs.

The DLMS®/COSEM AL may be capable to support one or more AAs simultaneously. Likewise, lower layers may be capable of supporting more than one connection with their peer layers. This allows data exchange between clients and servers simultaneously via different ports and communication media.

### 4.1.12 Interoperability and interconnectivity in DLMS®/COSEM

In the DLMS®/COSEM environment, interoperability and interconnectivity is defined between client and server AEs. A client and a server AE shall be interoperable and interconnectable to ensure data exchange between the two systems.

Using the COSEM object model to model metering of all kinds of energy, over all communication media ensures semantic interoperability, i.e. an unambiguous, shared meaning between clients and servers using different communication media. The semantic elements are the COSEM objects, their logical name i.e. the OBIS code, the definition of their attributes and methods and the data types that can be used.

Using the DLMS®/COSEM AL over all communication media ensures syntactic interoperability, which is a pre-requisite of semantic interoperability. Syntactic interoperability comprises the ability to establish AAs between clients and server using various application contexts, authentication mechanisms, xDLMS contexts and security contexts as well as the standard structure and encoding of all messages exchanged.

Interconnectivity is a protocol level notion: in order to be able to exchange messages, the client and the server AEs should be interconnectable and interconnected.

Before the two AEs can establish an AA, they shall be interconnected. The two AEs are interconnected, if each peer protocol layer of both sides, which needs to be connected, is connected. In order to be interconnected, the client and server AEs should be interconnectable and shall establish the required connections. Two AEs are interconnectable if they use the same communication profile.

With this, interconnectivity in DLMS®/COSEM is ensured by the ability of the DLMS®/COSEM AE to establish a connection between all peer layers, which need to be connected.

### 4.1.13 Ensuring interconnectivity: the protocol identification service

In DLMS®/COSEM, AA establishment is always initiated by the client AE. However, in some cases, it may not have knowledge about the protocol stack used by an unknown server device (for example when the server has initiated the physical connection establishment). In such cases, the client AE needs to obtain information about the protocol stack implemented in the server.

A specific, application level service is available for this purpose: the protocol identification service. It is an optional application level service, allowing the client AE to obtain information – after establishing a physical connection – about the protocol stack implemented in the server. The protocol identification service uses directly the data transfer services (PH-DATA.request /.indication) of the PhL; it bypasses the other protocol layers. It is recommended to support it in all communication profiles that have access to the PhL.

### 4.1.14 System integration and meter installation

System integration is supported by DLMS®/COSEM in a number of ways.

A possible process is described here.

As shown in Figure 6, the presence of a Public Client (bound to address 0x10 in any profile) is mandatory in each client system. Its main role is to reveal the structure of an unknown – for example newly installed – metering equipment. This takes place within a mandatory AA between the Public Client and the Management Logical Device, with no security precautions. Once the structure is known, data can be accessed with using the proper authentication mechanisms and cryptographic protection of the xDLMS messages and COSEM data.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

When a new meter is installed in the system, it may generate an event report to the client. Once this is detected, the client can retrieve the internal structure of the meter, and then send the necessary configuration information (for example tariff schedules and installation specific parameters) to the meter. With this, the meter is ready to use.

System integration is also facilitated by the availability of the DLMS®/COSEM conformance testing, described in the Yellow Book, DLMS® UA 1001-1. With this, correct implementation of the specification in metering equipment can be tested and certified.

## 4.2 DLMS®/COSEM application layer main features

### 4.2.1 General

This subclause 4.2 provides an overview of the main features provided by the DLMS®/COSEM AL.

### 4.2.2 DLMS®/COSEM application layer structure

The structure of the client and server DLMS®/COSEM application layers is shown in Figure 9.

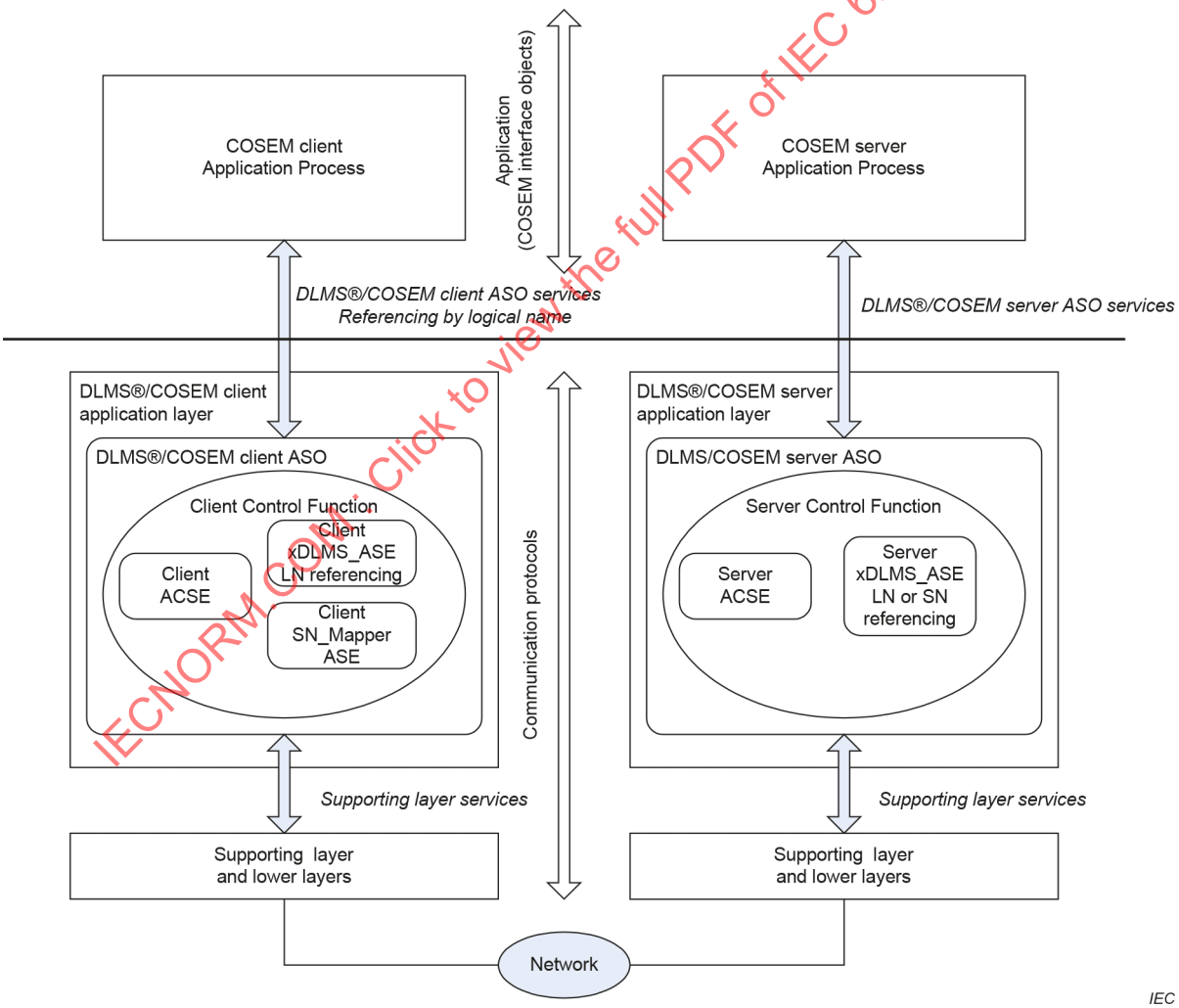



**Figure 9 – The structure of the DLMS®/COSEM application layers**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The main component of the DLMS®/COSEM AL is the Application Service Object (ASO). It provides services to its service user, the COSEM Application Process (APs) and uses services provided by the supporting layer. It contains three mandatory components both on the client and on the server side:

- the Association Control Service Element, ACSE;
- the extended DLMS® Application Service Element, xDLMS ASE;
- the Control Function, CF.

On the client side, there is a fourth, optional element, called the Client SN_Mapper ASE.

The ACSE provides services to establish and release application associations (AAs). See 4.2.3.

The xDLMS ASE provides services to transport data between COSEM APs. See 4.2.4.

The Control Function (CF) element specifies how the ASO services invoke the appropriate service primitives of the ACSE, the xDLMS ASE and the services of the supporting layer. See also 7.1.

Both the client and the server DLMS®/COSEM ASO may contain other, optional application protocol components.

The optional Client SN_Mapper ASE is present in the client side AL ASO, when the server uses SN referencing. It provides mapping between services using LN and SN referencing. See 4.2.5.

The DLMS®/COSEM AL performs also some functions of the OSI presentation layer:

- encoding and decoding the ACSE APDUs and the xDLMS APDUs, see also 0;
- alternatively, generating and using XML documents representing ACSE and xDLMS APDUs;
- applying compression and decompression;
- applying, verifying and removing cryptographic protection.

### 4.2.3 The Association Control Service Element, ACSE

For the purposes of DLMS®/COSEM connection oriented (CO) communication profiles, the CO ACSE, specified in ISO/IEC 15953:1999 and ISO/IEC 15954:1999 is used.

The services provided for application association establishment and release are the following:

- COSEM-OPEN;
- COSEM-RELEASE;
- COSEM-ABORT.

The COSEM-OPEN service is used to establish AAs. It is based on the ACSE A-ASSOCIATE service. It causes the start of use of an AA by those ASE procedures identified by the value of the Application_Context_Name, Security_Mechanism_Name and xDLMS context parameters. AAs may be established in different ways:

- confirmed AAs are established via a message exchange – using the COSEM-OPEN service – between the client and the server to negotiate the contexts. Confirmed AAs can be established between a single client and a single server;
- unconfirmed AAs are established via a message sent – using the COSEM-OPEN service – from the client to the server, using the parameters of the contexts supposed to be supported by the server. Unconfirmed AAs can be established between a client and one or multiple servers;

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- pre-established AAs may pre-exist. In this case, the COSEM-OPEN service is not used. The client has to be aware of the contexts supported by the server. A pre-established AA can be confirmed or unconfirmed.

The COSEM-RELEASE service is used to release AAs. If successful, it causes the completion of the use of the AA without loss of information in transit (graceful release). In some communication profiles – for example in the TCP-UDP/IP based profile – the COSEM-RELEASE service is based on the ACSE A-RELEASE service. In some other communication profiles – for example in the 3-layer, CO, HDLC based profile – there is a one-to-one relationship between a confirmed AA and the supporting protocol layer connection. In such profiles AAs can be released simply by disconnecting the corresponding supporting layer connection. Pre-established AAs cannot be released.

The COSEM-ABORT service causes the abnormal release of an AA with the possible loss of information in transit. It does not rely on the ACSE A-ABORT service.

The COSEM-OPEN service is specified in 6.2, the COSEM-RELEASE service in 6.3 and the COSEM-ABORT service in 6.4.

#### 4.2.4 The xDLMS application service element

##### 4.2.4.1 Overview

To access attributes and methods of COSEM objects, the services of the xDLMS ASE are used. It is based on the DLMS® standard, IEC 61334-4-41:1996. This International Standard specifies a range of extensions to extend functionality while maintaining backward compatibility. The extensions comprise the following:

- additional services, see 4.2.4.3;
- additional mechanisms, see 4.2.4.4;
- additional data types, see 4.2.4.5;
- new DLMS® version number, see 4.2.4.6;
- new conformance block, see 4.2.4.7;
- clarification of the meaning of the PDU size, see 4.2.4.8.

##### 4.2.4.2 The xDLMS initiate service

To establish the xDLMS context the xDLMS Initiate service — specified in IEC 61334-4-41:1996, 5.2 — is used. This service is integrated in the COSEM-OPEN service, see 6.2.

##### 4.2.4.3 COSEM object related xDLMS services

###### 4.2.4.3.1 General

COSEM object related xDLMS services are used to access COSEM object attributes and methods.

IEC 62056-6-2:2021, 4.1.2 specifies two referencing methods:

- Logical Name (LN) referencing; and
- Short Name (SN) referencing.

For more information on referencing methods, see 4.2.4.4.2.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Therefore, two distinct xDLMS service sets are specified: one exclusively using Logical Name (LN) referencing and the other exclusively using short name (SN) referencing. It can be considered that there are two different xDLMS ASEs: one providing services with LN referencing and the other with SN referencing. The client ASO always uses the xDLMS ASE with LN referencing. The server ASO may use either the xDLMS ASE with LN referencing or the xDLMS ASE with SN referencing or both.

These services may be:

- requested / solicited by the client; or
- unsolicited: these are always initiated by the server without a previous request from the client.

Services requested by the client may be also (see 7.3.2):

- confirmed: in this case, the server provides a response to the request;
- unconfirmed: in this case, the server does not provide a response to the request.

Unsolicited DataNotification from the server may be also (see 9.3.10):

- confirmed: in this case, the client provides a response to acknowledge the receipt of the unsolicited DataNotification
- unconfirmed: in this case, the client does not provide a response to the unsolicited DataNotification.

The additional services – which are not based on DLMS® services specified in IEC 61334-4-41:1996 – are:

- the GET, SET, ACTION and ACCESS used to access COSEM object attributes and methods using LN referencing;
- the DataNotification service used by the server to push data to the client;
- the EventNotification service used by the server to notify the client about events that occur in the server.

#### 4.2.4.3.2 xDLMS services used by the client with LN referencing

In the case of LN referencing, COSEM object attributes and methods are referenced via the identifier of the COSEM object instance to which they belong. For this referencing method, the following additional services are specified:

- the GET service is used by the client to request the server to return the value of one or more attributes, see 6.6;
- the SET service is used by the client to request the server to replace the content of one or more attributes, see 6.7;
- the ACTION service is used by the client to request the server to invoke one or more methods. Invoking methods may imply sending method invocation parameters and receiving return parameters, see 6.8;
- the ACCESS service, a unified service which can be used by the client to access multiple attributes and/or methods with a single .request; see 6.9.

These services can be invoked by the client in a confirmed or unconfirmed manner.

#### 4.2.4.3.3 xDLMS services used by the client with SN referencing

In the case of SN referencing, COSEM object attributes and methods are mapped to DLMS® named variables specified in IEC 61334-4-41:1996, 10.1.2.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The xDLMS services using SN referencing are based on the DLMS® variable access services, specified in IEC 61334-4-41:1996, 10.4 to 10.6 and they are the following:

- the Read service is used by the client to request the server to return the value of one or more attributes or to invoke one or more methods when return parameters are expected. It is a confirmed service. See 6.14;
- the Write service is used by the client to request the server to replace the content of one or more attributes or to invoke one or more methods when no return parameters are expected. It is a confirmed service. See 6.15;
- the UnconfirmedWrite service is used by the client to request the server to replace the content of one or more attributes or to invoke one or more methods when no return parameters are expected. It is an unconfirmed service. See 6.16.

New variants of the Variable_Access_Specification service parameter (see 6.13), the Read.response and the Write.response services have been added to support selective access – see 4.2.4.3.5 – and block transfer, see 4.2.4.4.5.

#### 4.2.4.3.4 Unsolicited services

Unsolicited services are initiated by the server, on pre-defined conditions, e.g. schedules, triggers or events, to inform the client of the value of one or more attributes, as though they had been requested by the client.

To support push operation, the DataNotification service is available, see 6.10. It can be used in application contexts using either SN referencing or LN referencing.

NOTE The DataNotification service is used in conjunction with "Push setup" COSEM objects, see IEC 62056-6-2:2021, 4.4.8.

To support event notification, the following unsolicited services are available:

- with LN referencing, the EventNotification service, see 6.11;
- with SN referencing, the InformationReport service, see 6.17. This service is based on IEC 61334-4-41:1996, 10.7.

#### 4.2.4.3.5 Selective access

In the case of some COSEM interface classes, selective access to attributes is available, meaning that either the whole attribute or a selected portion of it can be accessed as required. For this purpose, access selectors and parameters are specified as part of the specification of the relevant attributes.

To use this possibility, attribute-related services can be invoked with access selection parameters. In the case of LN referencing, this feature is called Selective access; see 6.6 and 6.7. It is a negotiable feature; see 7.3.1. In the case of SN referencing, this feature is called Parameterized access; see 6.14, 6.15 and 6.16. It is a negotiable feature; see 7.3.1.

#### 4.2.4.3.6 Multiple references

In a COSEM object related service invocation, it is possible to reference one or several named variables, attributes and/or methods. Using multiple references is a negotiable feature. See 7.3.1.

#### 4.2.4.3.7 Attribute_0 referencing

With the GET, SET and ACCESS services a special feature, Attribute_0 referencing is available. By convention, attributes of COSEM objects are numbered from 1 to n, where Attribute_1 is the logical name of the COSEM object. Attribute_0 has a special meaning: it references all attributes with positive index (public attributes). The use of Attribute_0 referencing with the GET service is explained in 6.6, with the SET service in 6.7 and with the ACCESS service in 6.9.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

NOTE As specified in IEC 62056-6-2:2021, 4.1.2, manufacturers may add proprietary methods and/or attributes to any object, using negative numbers.

Attribute_0 referencing is a negotiable feature, see 7.3.1.

#### 4.2.4.4 Additional mechanisms

##### 4.2.4.4.1 Overview

xDLMS specifies several new mechanisms – compared to DLMS® as specified in IEC 61334-4-41:1996 – to improve functionality, flexibility and efficiency. The additional mechanisms are:

- referencing using logical names;
- identification of service invocations;
- priority handling;
- transferring long application messages;
- composable xDLMS messages;
- compression and decompression;
- general cryptographic protection;
- general block transfer.

##### 4.2.4.4.2 Referencing methods and service mapping

To access COSEM object attributes and methods with the xDLMS services, they have to be referenced. As already mentioned in 4.2.4.3.1, IEC 62056-6-2:2021, 4.1.2 specifies two referencing methods:

- Logical Name (LN) referencing; and
- Short Name (SN) referencing.

In the case of LN referencing, COSEM object attributes and methods are referenced via the logical name (COSEM_Object_Instance_ID) of the COSEM object instance to which they belong. In the case of SN referencing, COSEM object attributes and methods are mapped to DLMS® named variables.

Accordingly, there are two xDLMS ASEs specified: one using xDLMS services with LN referencing and one using xDLMS services with SN referencing.

On the client side, in order to handle the different referencing methods transparently for the AP, the AL uses the xDLMS ASE with LN referencing. Using a unique, standardized service set between COSEM client APs and the communication protocol – hiding the particularities of DLMS®/COSEM servers using different referencing methods – allows specifying an Application Programming Interface, API. This is an explicitly specified interface corresponding to this service set for applications running in a given computing environment (for example Windows, UNIX, etc.) Using this – public – API specification, client applications can be developed without knowledge about particularities of a given server.

On the server side, either the xDLMS ASE with LN referencing or the xDLMS ASE with SN referencing or both xDLMS ASEs can be used.

In the case of confirmed AAs, the referencing method is negotiated during the AA establishment phase via the COSEM application context. It shall not change during the lifetime of the AA established. Using LN or SN services within a given AA is exclusive.

In the case of unconfirmed and pre-established AAs, the client AL is expected to know the referencing method supported by the server.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

When the server uses LN referencing, the services are the same on both sides. When the server uses SN referencing the Client SN_Mapper ASE in the client maps the SN referencing into LN referencing or vice versa. See 4.2.2 and 4.2.5.

#### 4.2.4.4.3 Identification of service invocations: the Invoke_Id parameter

In the client/server model, requests are sent by the client and responses are sent by the server. The client is allowed to send several requests before receiving the response to the previous ones, provided that this is allowed by the lower layers.

Therefore – to be able to identify which response corresponds to each request – it is necessary to include a reference in the request.

The Invoke_Id parameter is used for this purpose. The value of this parameter is assigned by the client so that each request carries a different Invoke_Id. The server shall copy the Invoke_Id into the corresponding response.

In the ACCESS and the DataNotification service – see 6.9 and 6.10 – the Long-Invoke-Id parameter is used instead of the Invoke_Id parameter.

The EventNotification service does not contain the Invoke_Id parameter.

This feature is available only with LN referencing.

#### 4.2.4.4.4 Priority handling

For data transfer services using LN referencing, two priority levels are available: normal (FALSE) and high (TRUE). This feature allows receiving a response to a new request before the response to a previous request is completed.

Normally, the server serves incoming service requests in the order of reception (FIFS, First In, First Served). However, a request with the priority parameter set to high (TRUE) is served before the previous requests with priority set to normal (FALSE). The response carries the same priority flag as that of the corresponding request. Managing priority is a negotiable feature; see 7.3.1.

NOTE 1 As service invocations are identified with an Invoke_Id, services with the same priority can be served in any order.

NOTE 2 If the feature is not supported, requests with HIGH priority are served with NORMAL priority.

This feature is not available with services using SN referencing. The server treats the services on a FIFS basis.

#### 4.2.4.4.5 Transferring long messages

The xDLMS service primitives are carried in an encoded form by xDLMS APDUs. This encoded form may be longer than the Client / Server Max Receive PDU Size negotiated. To transfer such ‘long’ messages, there are two mechanisms available:

a) the general block transfer (GBT) mechanism specified in 4.2.4.4.9;

b) the service-specific block transfer mechanism. This mechanism is available with the GET, SET, ACTION, Read and Write services. In this case, the service primitive invocations contain only one part – one block – of the data (e.g. attribute values), so that the encoded form fits in a single APDU.

Using the general or the service-specific block transfer mechanism is a negotiable feature, see 7.3.1.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

An APDU that fits in the Client / Server Max Receive PDU Size negotiated may be too long to fit in a single frame / packet of the supporting protocol layer. Such APDUs may be transported if the supporting protocol layer provide(s) segmentation.

Service-specific block transfer mechanism is available with:

- confirmed services using LN referencing: GET, SET, ACTION;
- confirmed services using SN referencing: Read, Write.

and the related APDUs.

Service-specific block transfer is not available with:

- unconfirmed services;
- unsolicited services (DataNotification, EventNotification and InformationReport);
- the ACCESS service

and the related APDUs.

With service-specific block transfer, the reception of each block has to be acknowledged before the next block can be sent. A recovery mechanism for lost blocks is not available. When global key or dedicated key ciphering is required, it is applied to the APDU carrying a block – or an acknowledgement of a block – of the long message. This creates considerable computing and transport overhead. Service specific digital signature is not available.

Conversely, the GBT mechanism can be used with any xDLMS APDU including the general-ciphering and general-signing APDUs. It provides bidirectional block transfer, streaming and lost block recovery. When cryptographic protection is required, it is applied to the complete APDU and then the protected APDU is transferred in blocks, as specified in 4.2.4.4.6.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 4.2.4.4.6 Composable xDLMS messages

The three important aspects of dealing with xDLMS messages are encoding / decoding, applying, verifying / removing cryptographic protection and block transfer.

The concept of composable xDLMS messages separates the three aspects, as shown in Figure 10. See also Figure 36.

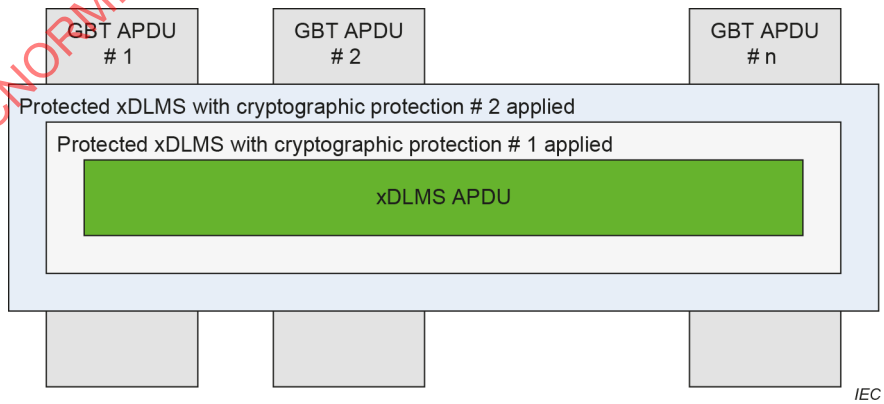



**Figure 10 – The concept of composable xDLMS messages**

Once the APDU corresponding to the service primitive invoked by the AP is built by the AL, the general protection mechanism can be used to apply cryptographic protection. When an unprotected or a protected APDU is too long to fit in the negotiated APDU size, then the general block transfer mechanism can be applied.

These mechanisms can be applied with all xDLMS APDUs.

NOTE 1 With the GET, SET, ACTION, EventNotification, Read and Write, UnconfirmedWrite and InformationReport APDUs, service-specific cryptographic protection is available using specific service protection types and APDUs.

NOTE 2 With the GET, SET, ACTION, Read, Write, and UnconfirmedWrite and APDUs, service-specific block transfer is available using specific service request / response types and APDUs.

#### 4.2.4.4.7 Compression and decompression

In order to optimize the use of communication media, it is possible to compress xDLMS APDUs to be sent and decompress xDLMS APDUs received. For details, see 5.3.6.

#### 4.2.4.4.8 General protection

This mechanism can be used to apply cryptographic protection to any xDLMS APDU and this allows applying multiple layers of protection between the client and the server or between a third party and the server. See also 5.2.5

For this purpose, the following APDUs are available; see 5.7.2.3:

- the general-ded-ciphering and the general-glo-ciphering APDUs;
- the general-ciphering APDUs;
- the general-signing APDU.

Using the general protection mechanism is a negotiable feature, see 7.3.1.

#### 4.2.4.4.9 General block transfer (GBT)

The GBT mechanism can be used to transfer any – short or long – xDLMS APDU in blocks. With GBT, the blocks are carried by general-block-transfer APDUs instead of service-specific “with-datablock” APDUs.

The GBT mechanism supports bi-directional block transfer, streaming and lost block recovery:

- bi-directional block transfer means that while one party is sending blocks, the other party not only confirms the blocks received but if it has blocks to send it can send them as well while it is still receiving blocks;

  NOTE Bi-directional block transfer is useful when long service parameters need to be transported in both directions.

- streaming means that several blocks may be sent – streamed – by one party without an acknowledgement of each block from the other party;
- lost block recovery means that if the reception of a block sent is not confirmed, it can be sent again. If streaming is used, lost block recovery takes place at the end of each streaming window.

The GBT mechanism is managed by the AL using the block transfer streaming parameters specified in 6.5.

The protocol of the general block transfer mechanism is specified in 7.3.13.

The GBT mechanism supports the following use cases:

a) short request by the client that leads to a long response from the server: the request may be sent with or without using GBT. If it is sent using GBT then the client may advertise its preferred GBT window size when the exchange is started. The server responds using GBT;

b) long request by the client: the client sends the request and the server sends the response using GBT, whatever is the length of the response;

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

c) long unconfirmed request by the client: the client sends the request using GBT;

d) long unsolicited request from the server: the server sends the unsolicited message using GBT.

#### 4.2.4.5 Additional data types

The additional data types are specified in Clause 8 and in Clause 9.

#### 4.2.4.6 xDLMS version number

The new DLMS® version number, corresponding to the first version of the xDLMS ASE is 6.

#### 4.2.4.7 xDLMS conformance block

The xDLMS conformance block enables optimised DLMS®/COSEM server implementations with extended functionality. It can be distinguished from the DLMS® conformance block by its tag "Application 31". See 7.3.1, Clause 8 and Clause 9.

The xDLMS conformance block is part of the xDLMS context.

In the case of confirmed AAs, the conformance block is negotiated during the AA establishment phase via the xDLMS context. It shall not change during the lifetime of the AA established.

In the case of unconfirmed and pre-established AAs, the client AL is expected to know the conformance block supported by the server.

#### 4.2.4.8 Maximum PDU size

To clarify the meaning of the maximum PDU size usable by the client and the server, the modifications shown in Table 2 have been made. The xDLMS Initiate service uses these names for PDU sizes.

**Table 2 – Clarification of the meaning of PDU size for DLMS®/COSEM**

| was: | new: |
|---|---|
| **IEC 61334-4-41:1996, 5.2.2, Table 3** | |
| Proposed Max PDU Size | Client Max Receive PDU Size |
| Negotiated Max PDU Size | Server Max Receive PDU Size |
| **IEC 61334-4-41:1996, 5.2.3, 7th paragraph** | |
| The Proposed Max PDU Size parameter, of type Unsigned16, proposes a maximum length expressed in bytes for the exchanged DLMS® APDUs. The value proposed in an Initiate request shall be large enough to always permit the Initiate Error PDU transmission | The Client Max Receive PDU Size parameter, of type Unsigned16, contains the maximum length expressed in bytes for a DLMS® APDU that the server may send. The client will discard any received PDUs that are longer than this maximum length. The value shall be large enough to always permit the AARE APDU transmission.<br>Values below 12 are reserved. The value 0 indicates that there is no limit on the PDU size. |
| **IEC 61334-4-41:1996, 5.2.3, last paragraph** | |
| The Negotiated Max PDU Size parameter, of type Unsigned16, contains a maximum length expressed in bytes for the exchanged DLMS® APDUs. A PDU that is longer than this maximum length will be discarded. This maximum length is computed as the minimum of the Proposed Max PDU Size and the maximum PDU size than the VDE-handler may support. | The Server Max Receive PDU Size parameter, of type Unsigned16, contains the maximum length expressed in bytes for a DLMS® APDU that the client may send. The server will discard any received PDUs that are longer than this maximum length.<br>Values below 12 are reserved. The value 0 indicates that there is no limit on the PDU size. |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 4.2.5 Layer management services

Layer management services have local importance only. Therefore, specification of these services is not within the Scope of this document.

The specific SetMapperTable service is defined in 6.18.

### 4.2.6 Summary of DLMS®/COSEM application layer services

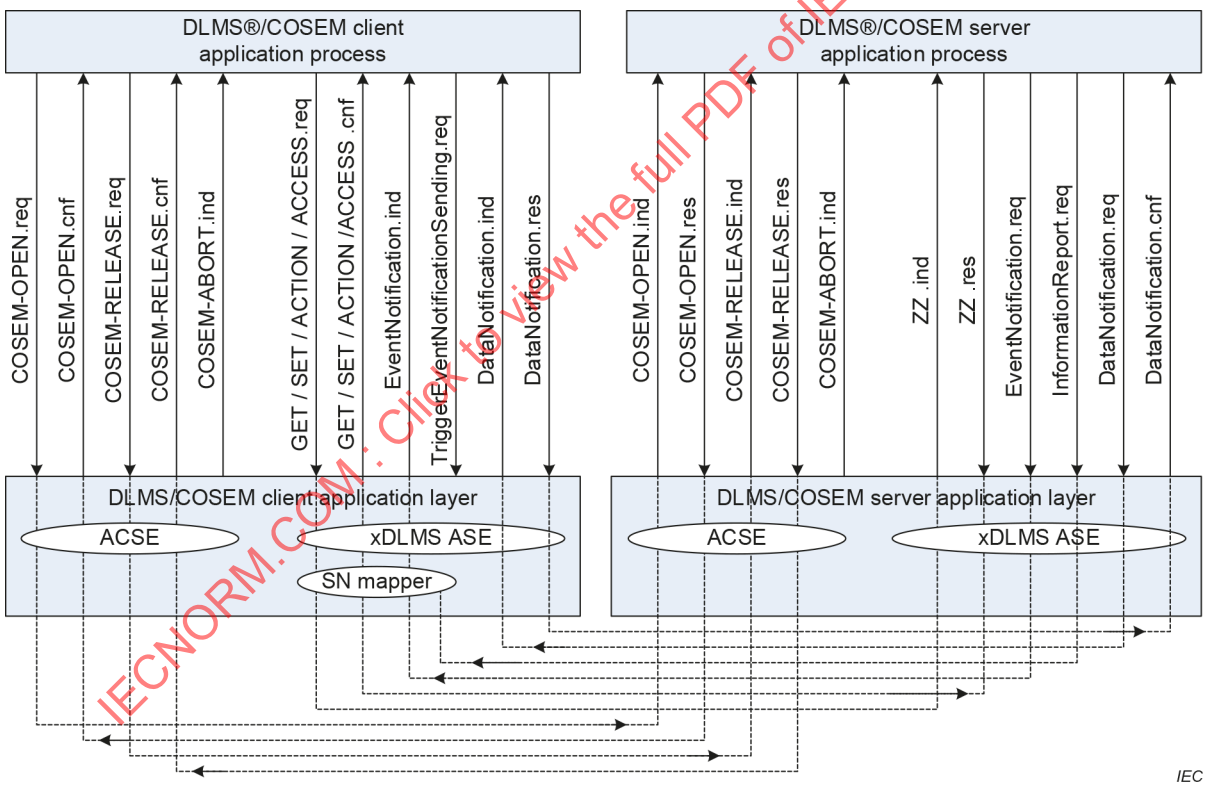
A summary of the services available at the top of the DLMS®/COSEM AL is shown in Figure 11. Layer management services are not shown. Although the service primitives are different on the client and server side, the APDUs are the same.

NOTE 1 For example, when the client AP invokes a GET.request service primitive the client AL builds a GET-Request APDU. When this is received by the server AL, it invokes a GET.ind service primitive.

The DLMS®/COSEM AL services are specified in 6. The DLMS®/COSEM AL protocol is specified in Clause 7. The abstract syntax of the ACSE and xDLMS APDUs is specified in 7.3.13. The XML schema is defined in Clause 9.

Encoding examples are provided in Annex D, Annex E, and Annex F.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




NOTE 1 The client AP always uses LN referencing. If the server uses SN referencing then a mapping is performed by the Client SN_Mapper ASE. Consequently, the service primitives ZZ.ind and ZZ.res may be LN or SN service primitives. LN/SN service mapping is specified in Clause 8.

NOTE 2 The ACCESS service cannot be mapped to services using SN referencing.

**Figure 11 – Summary of DLMS®/COSEM AL services**

#### 4.2.7 DLMS®/COSEM application layer protocols

The DLMS®/COSEM AL protocols specify the procedures for information transfer for AA control and authentication using connection-oriented ACSE procedures, and for data transfer between COSEM clients and servers using xDLMS procedures. Therefore, the DLMS®/COSEM AL protocol is based on the ACSE standard as specified in ISO/IEC 15954:1999 and the DLMS® standard, as specified in IEC 61334-4-41:1996, with the extensions for DLMS®/COSEM. The procedures are defined in terms of:

- the interactions between peer ACSE and xDLMS protocol machines through the use of services of the supporting protocol layer;
- the interactions between the ACSE and xDLMS protocol machines and their service user.

The DLMS®/COSEM AL protocols are specified in Clause 7.

## 5 Information security in DLMS®/COSEM

### 5.1 Overview

This Clause 5 describes and specifies:

- the DLMS®/COSEM security concept, see 5.2;
- the cryptographic algorithms selected, see 5.3;
- the security keys, see 5.4, 5.5 and 5.6;
- the use of the cryptographic algorithms for entity authentication, xDLMS APDU protection and COSEM data protection, see 5.7.

### 5.2 The DLMS®/COSEM security concept

#### 5.2.1 Overview

The resources of DLMS®/COSEM servers – COSEM object attributes and methods – can be accessed by DLMS®/COSEM clients within Application Associations, see also 4.1.5.

During an AA establishment the client and the server have to identify themselves. The server may also require that the *user* of a client identifies itself. Furthermore, the server may require that the client authenticates itself and the client may also require that the server authenticates itself. The identification and authentication mechanisms are specified in 5.2.2.

Once an AA is established, xDLMS services can be used to access COSEM object attributes and methods, subject to the security context and access rights. See 5.2.3 and 5.2.4.

The xDLMS APDUs carrying the services primitives can be cryptographically protected. The required protection is determined by the security context and the access rights. To support end-to-end security between third parties and servers, such third parties can also access the resources of a server using a client as a broker. The concept of message protection is further explained in 5.2.5.

Moreover, COSEM data carried by the xDLMS APDUs can be cryptographically protected; see 5.2.6.

As these security mechanisms are applied on the application process / application layer level, they can be used in all DLMS®/COSEM communication profiles.

NOTE Lower layers may provide additional security.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.2.2 Identification and authentication

#### 5.2.2.1 Identification

As specified in 4.1.3.3, DLMS®/COSEM AEs are bound to Service Access Points (SAPs) in the protocol layer supporting the AL. These SAPs are present in the PDUs carrying the xDLMS APDUs within an AA.

The client user identification mechanism enables the server to distinguish between different users on the client side— that may be operators or third parties — to log their activities accessing the meter. See also 4.1.3.6.

#### 5.2.2.2 Authentication mechanisms

##### 5.2.2.2.1 Overview

The authentication mechanisms determine the protocol to be used by the communication entities to authenticate themselves during AA establishment. There are three different authentication mechanisms available with different authentication security levels:

- no security (Lowest Level Security) authentication; see 5.2.2.2.2;
- Low Level Security (LLS) authentication, see 5.2.2.2.3;

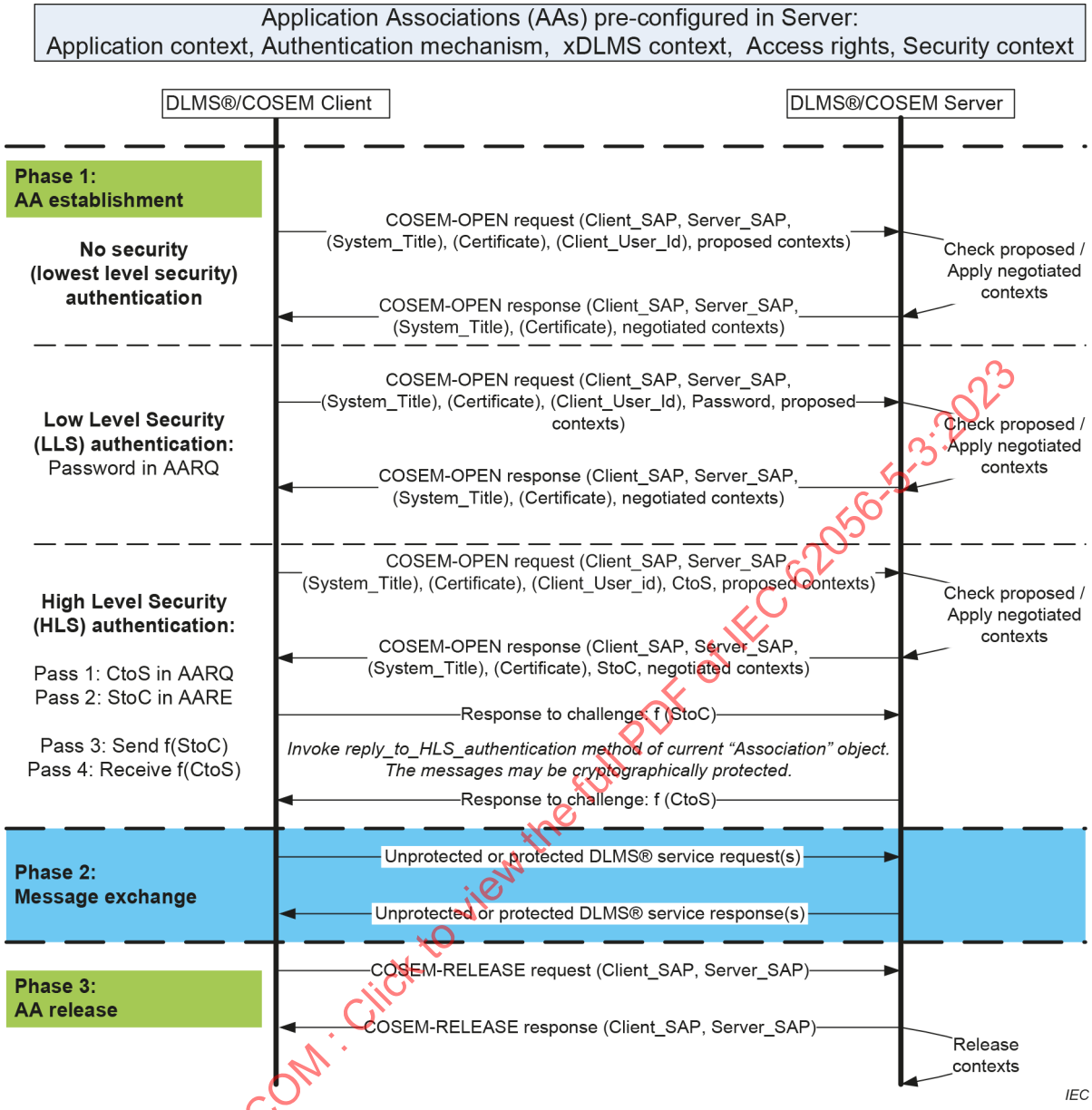
  NOTE 1 In ITU-T X.811 this is known as unilateral authentication, class 0 mechanism.

- High Level Security (HLS) authentication, see 5.2.2.2.4.

  NOTE 2 In ITU-T X.811 this is known as mutual authentication using challenge mechanisms.

They are shown in Figure 12. Authentication mechanisms are identified by names, see 7.2.2.3.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

NOTE 1 The COSEM-OPEN service primitives are carried by AARQ / AARE APDUs. The COSEM-RELEASE service primitives are carried by RLRQ / RLRE APDUs (when used). See 6.2 and 6.3.

NOTE 2 The elements (*System_Title*), (*Certificate*) and (*Client_User_Id*) are optional.

NOTE 3 In pre-established AAs no authentication takes place.

NOTE 4 The COSEM-RELEASE service can be cryptographically protected by including a ciphered xDLMS Initiate .request / .response APDU in the RLRQ.

**Figure 12 – Authentication mechanisms**

The security of the message exchange (in Phase 2) is independent of the client-server authentication during AA establishment (Phase 1). Even in the case where no client-server authentication takes place, cryptographically protected APDUs can be used to ensure message security.

#### 5.2.2.2.2 No security (Lowest Level Security) authentication

The purpose of No security (Lowest Level Security) authentication is to allow the client to retrieve some basic information from the server. This authentication mechanism does not require any authentication; the client can access the COSEM object attributes and methods within the security context and access rights prevailing in the given AA.

#### 5.2.2.2.3 Low Level Security (LLS) authentication

In this case, the server requires that the client authenticates itself by supplying a password that is known by the server. The password is held by the current "Association SN / LN" object modelling the AA to be established. The "Association SN / LN" objects provide means to change the secret.

If the password supplied is accepted, the AA can be established, otherwise it shall be rejected.

LLS authentication is supported by the COSEM-OPEN service – see 6.2 – as follows:

- the client transmits a "secret" (a password) to the server, using the COSEM-OPEN.request service primitive;
- the server checks if the "secret" is correct;
- if yes, the client is authenticated and the AA can be established. From this moment, the negotiated contexts are valid;
- if not, the AA shall be rejected;
- the result of establishing the AA shall be sent back by the server using the COSEM-OPEN.response service primitive, together with diagnostic information.

#### 5.2.2.2.4 High Level Security (HLS) authentication

In this case, both the client and the server have to successfully authenticate themselves to establish an AA. HLS authentication is a four-pass process that is supported by the COSEM-OPEN service and the *reply_to_HLS_authentication* method of the "Association SN / LN" interface class:

- Pass 1: The client transmits a "challenge" $CtoS$ and – depending on the authentication mechanism – additional information to the server;
- Pass 2: The server transmits a "challenge" $StoC$ and – depending on the authentication mechanism – additional information to the client;

If $StoC$ is the same as $CtoS$, the client shall reject it and shall abort the AA establishment process.

- Pass 3: The client processes $StoC$ and the additional information according to the rules of the HLS authentication mechanism valid for the given AA and sends the result to the server. The server checks if f($StoC$) is the result of correct processing and – if so – it accepts the authentication of the client;
- Pass 4: The server processes then $CtoS$ and the additional information according to the rules of the HLS authentication mechanism valid for the given AA and sends the result to the client. The client checks if f($CtoS$) is the result of correct processing and – if so – it accepts the authentication of the server.

Pass 1 and Pass 2 are supported by the COSEM-OPEN service.

After Pass 2 – provided that the proposed application context and xDLMS context are acceptable – the server grants access to the method *reply_to_HLS_authentication* of the current "Association SN / LN" object using the application context negotiated.

Pass 3 and Pass 4 are supported by the method *reply_to_HLS_authentication* of the "Association SN / LN" object(s). If both passes 3 and 4 are successfully executed, then the AA is established with the application context and xDLMS context negotiated.

The dedicated-key, if transferred, can be used from this moment.

Otherwise, either the client or the server aborts.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

There are several HLS authentication mechanisms available. These are further specified in 5.7.4.

In some HLS authentication mechanisms, the processing of the challenges involves the use of an HLS secret.

The "Association SN / LN" interface class provides a method to change the HLS "secret": *change_HLS_secret*.

REMARK After the client has issued the change_HLS_secret () – or change_LLS_secret () – method, it expects a response from the server acknowledging that the secret has been changed. It is possible that the server transmits the acknowledgement, but due to communication problems, the acknowledgement is not received at the client side. Therefore, the client does not know if the secret has been changed or not. For simplicity reasons, the server does not offer any special support for this case; i.e. it is left to the client to cope with this situation.

### 5.2.3 Security context

The security context defines security attributes relevant for cryptographic transformations and includes the following elements:

- the security suite, determining the security algorithms available, see 5.3.7;
- the security policy, determining the kind(s) of protection to be applied generally to all xDLMS APDUs exchanged within an AA. The possible security policies are specified in 5.7.2.2;
- the security material, relevant for the given security algorithms, that includes security keys, initialization vectors, public key certificates and the like. As the security material is specific for each security algorithm, the elements are specified in detail in the relevant clauses.

The security context is managed by "Security setup" objects; see IEC 62056-6-2:2021, 4.4.7.

### 5.2.4 Access rights

Access rights to attributes may be: *no_access*, *read_only*, *write_only*, or *read_and_write*. Access rights to methods may be *no_access* or *access*.

In addition, access rights may stipulate cryptographic protection to be applied to xDLMS APDUs carrying the service primitives used to access a particular COSEM object attribute / method. The protection required on the .request and on the .response can be independently configured.

Access rights are held by the relevant "Association SN / LN" objects; see IEC 62056-6-2:2021, 4.4.3 and 4.4.4. The possible access rights are specified in 5.7.2.2.

The protection to be applied shall meet the stronger of the requirement stipulated by the security policy and the access rights.

### 5.2.5 Application layer message security

DLMS®/COSEM ensures AL level security by providing means to cryptographically protect xDLMS APDUs. The protection may be any combination of authentication, encryption and digital signature and can be applied in a multi-layer fashion by multiple parties. The protection is applied by the originator and is verified and removed by the recipient.

A request or response received shall be processed only if the protection on the message carrying the request or response could be successfully verified and removed.

Project specific companion specifications may specify additional criteria for accepting and processing messages.

The concept of message protection between a client and a server is shown in Figure 13.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

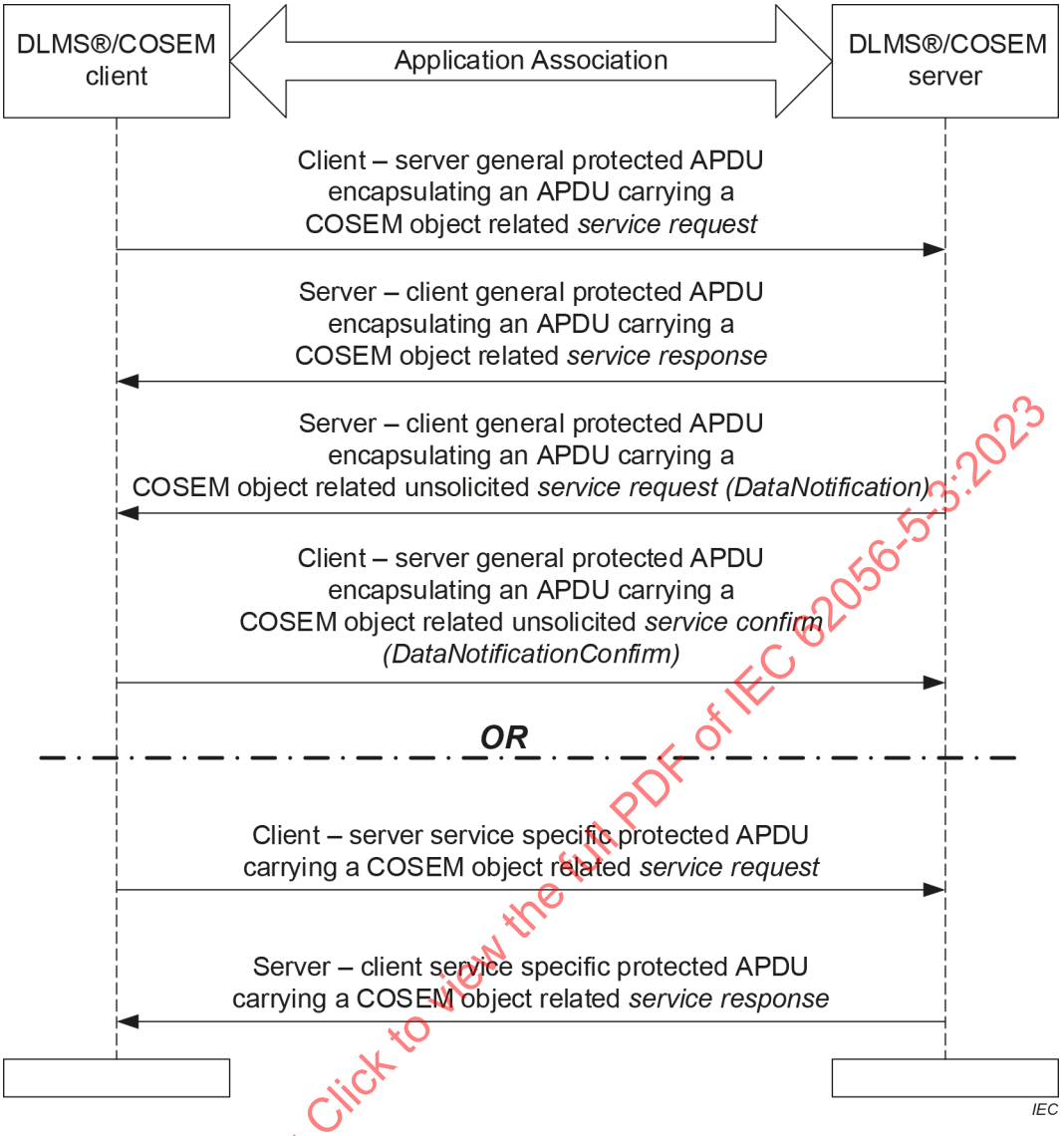



**Figure 13 – Client – server message security concept**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

To ensure end-to-end message security, third parties have to be able to exchange protected xDLMS service requests with DLMS®/COSEM servers. In this case, the client acts as a broker, meaning that a third party is a user of one of the AAs between a client and a server the third party wants to reach.

The concept of message protection between a third party and a server is shown in Figure 14.

The third party:

- is DLMS®/COSEM aware i.e. it can generate and process messages encapsulating xDLMS APDUs carrying COSEM object related service requests and responses;
- it is able to apply its own protection to the xDLMS APDU carrying the request;
- it is able to verify protection applied by the server and / or the client on the response.

The DLMS®/COSEM client:

- acts as a broker between the third party and the server;
- makes an appropriate AA available for use by the third party, based on information included in the TP – client message;

- verifies that the TP has the right to use that AA;

NOTE The way to verify this is outside the Scope of this document.

- it may verify the protection applied by the third party;
- encapsulates the third party – client message into a general protected xDLMS APDU;
- it may verify the protection applied by the server on the APDU encapsulating the COSEM object related service response or unsolicited service request; (in the case of Push operation);
- it may apply its own protection to the protected xDLMS APDUs sent to the TP.

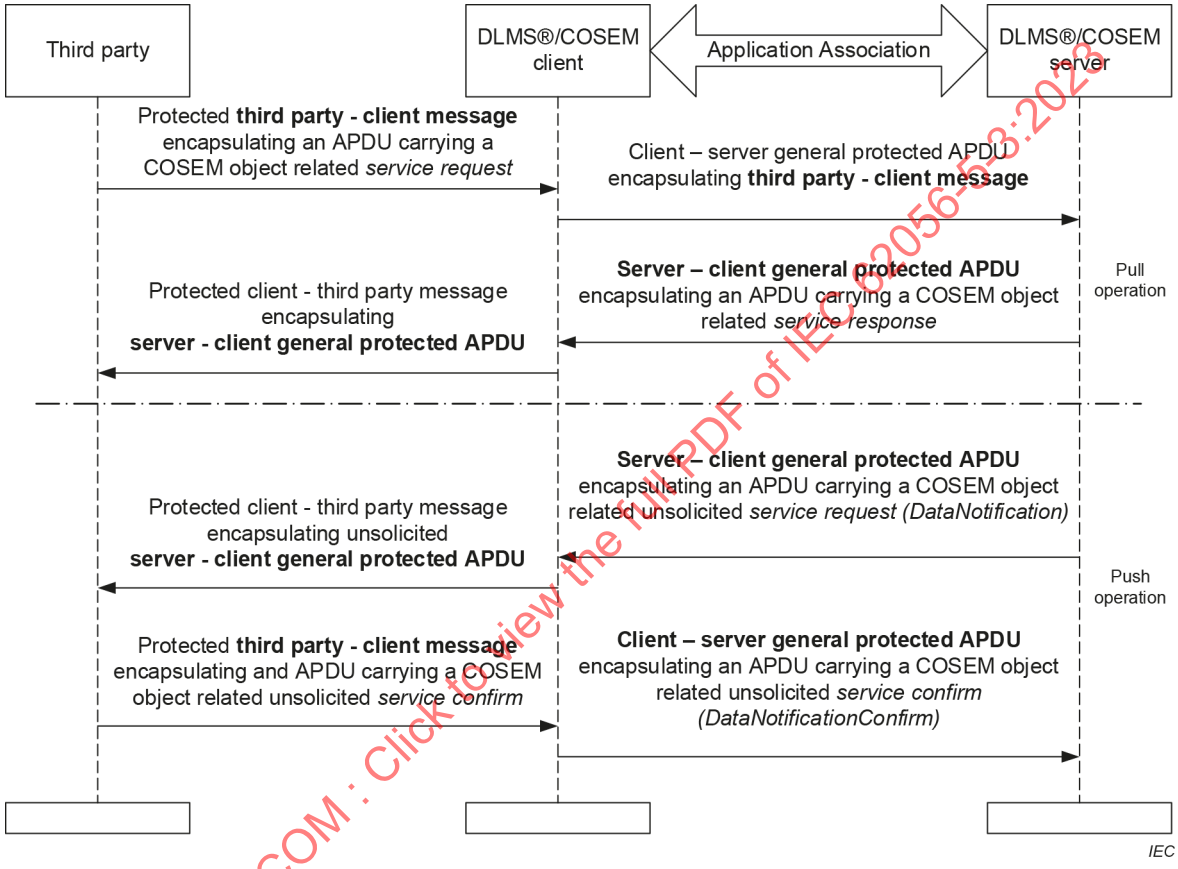



**Figure 14 – End-to-end message security concept**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The server:

- shall (pre-)establish an AA with the client used by the third party;
- it may check the identity of the third party using the AA;
- it shall provide access to COSEM object attributes and methods as determined by the security policy and access rights once the protection(s) applied by the client and/or the third party have been successfully verified;
- it shall prepare the response – or, in the case of Push operation an unsolicited service request – and apply the protection determined by the protection applied on the incoming request, the access rights and the security policy.

The application of cryptographic protection on xDLMS APDUs is specified in 5.7.2.

### 5.2.6 COSEM data security

COSEM data i.e. values of COSEM object attributes, method invocation parameters and return parameters can be also cryptographically protected. When this is required, the attributes and methods concerned are accessed indirectly, via "Data protection" objects, that apply and verify / remove protection on COSEM data; see IEC 62056-6-2:2021, 4.4.9.

See also 5.7.5.

## 5.3 Cryptographic algorithms

### 5.3.1 Overview

DLMS®/COSEM applies cryptography to protect the information.

NOTE The following text is quoted from NIST SP 800-21:2005, 3.1.

Cryptography is a branch of mathematics that is based on the transformation of data and can be used to provide several security services: confidentiality, data integrity, authentication, authorization and non-repudiation. Cryptography relies upon two basic components: an *algorithm* (or cryptographic methodology) and a *key*. The algorithm is a mathematical function, and the key is a parameter used in the transformation.

A cryptographic algorithm and key are used to apply cryptographic protection to data (e.g., encrypt the data or generate a digital signature) and to remove or check the protection (e.g., decrypt the encrypted data or verify the digital signature). There are three basic types of approved cryptographic algorithms:

- cryptographic hash functions that do not require keys (although they can be used in a mode in which keys are used). A hash function is often used as a component of an algorithm to provide a security service. See 5.3.2;
- symmetric key algorithms (often called secret key algorithms) that use a single key – shared by a sender and a receiver – to both apply the protection and to remove or check the protection. Symmetric key algorithms are relatively easy to implement and provide a high throughput. See 5.3.3;
- asymmetric key algorithms (often called public key algorithms) that use two keys (i.e., a key pair): a public key and a private key that are mathematically related to each other. Compared to symmetric key algorithms, implementation of asymmetric key algorithms is complex and requires much more computation. See 5.3.4.

In order to use cryptography, cryptographic keys shall be "in place", i.e., keys shall be established for parties using cryptography. See 5.4.

### 5.3.2 Hash function

NOTE The following text is quoted from NIST SP 800-21:2005, 3.2.

A hash function produces a short representation of a longer message. A good hash function is a one-way function: it is easy to compute the hash value from a particular input; however, backing up the process from the hash value back to the input is extremely difficult. With a good hash function, it is also extremely difficult to find two specific inputs that produce the same hash value. Because of these characteristics, hash functions are often used to determine whether or not data has changed.

A hash function takes an input of arbitrary length and outputs a fixed length value. Common names for the output of a hash function include *hash value* and *message digest*. Figure 15 depicts the use of a hash function.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

A hash value (H1) is computed on data (M1). M1 and H1 are then saved or transmitted. At a later time, the correctness of the retrieved or received data is checked by labelling the received data as M2 (rather than M1) and computing a new hash value (H2) on the received value. If the newly computed hash value (H2) is equal to the retrieved or received hash value (H1), then it can be assumed that the retrieved or received data (M2) is the same as the original data (M1) (i.e., M1 = M2).

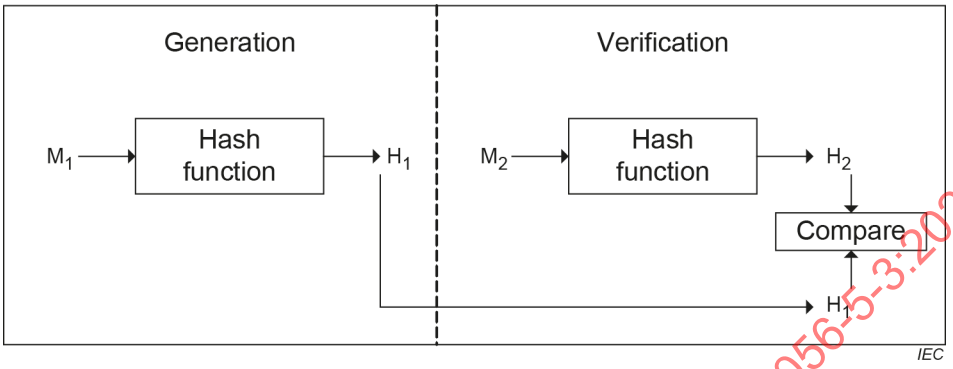



**Figure 15 – Hash function**

Hash algorithms are used in DLMS®/COSEM for the following purposes:

- digital signature, see 5.3.4.4;
- key agreement, see 5.3.4.6; and
- HLS authentication. The algorithm to be used depends on the authentication mechanism, see 5.7.4.

For digital signature and key agreement the algorithm shall be as stipulated by the security suite, see Table 9.

### 5.3.3 Symmetric key algorithms

#### 5.3.3.1 General

Symmetric key algorithms are used in DLMS®/COSEM for the following purposes:

- authentication of communicating partners using HLS authentication mechanisms, see 5.7.4;
- authentication and encryption of xDLMS messages, see 5.7.2;
- authentication and encryption of COSEM data, see 5.7.5.

NOTE The following text is quoted from NIST SP 800-21:2005, 3.3.

Symmetric key algorithms (often called secret key algorithms) use a single key to both apply the protection and to remove or check the protection. For example, the key used to encrypt data is also used to decrypt the encrypted data. This key shall be kept secret if the data is to retain its cryptographic protection. Symmetric key algorithms are used to provide confidentiality via encryption, or an assurance of authenticity or integrity via authentication, or are used during key establishment.

Keys used for one purpose shall not be used for other purposes. (See NIST SP 800-57:2012).

#### 5.3.3.2 Encryption and decryption

NOTE The following text is quoted from NIST SP 800-21:2005, 3.3.1.

Encryption is used to provide confidentiality for data. The data to be protected is called *plaintext*. Encryption transforms the data into *ciphertext*. Ciphertext can be transformed back into plaintext using decryption.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Plaintext data can be recovered from ciphertext only by using the same key that was used to encrypt the data. Unauthorized recipients of the ciphertext who know the cryptographic algorithm but do not have the correct key should not be able to decrypt the ciphertext. However, anyone who has the key and the cryptographic algorithm can easily decrypt the ciphertext and obtain the original plaintext data.

Figure 16 depicts the encryption and decryption processes. The plaintext (P) and a key (K) are used by the encryption process to produce the ciphertext (C). To decrypt, the ciphertext (C) and the same key (K) are used by the decryption process to recover the plaintext (P).

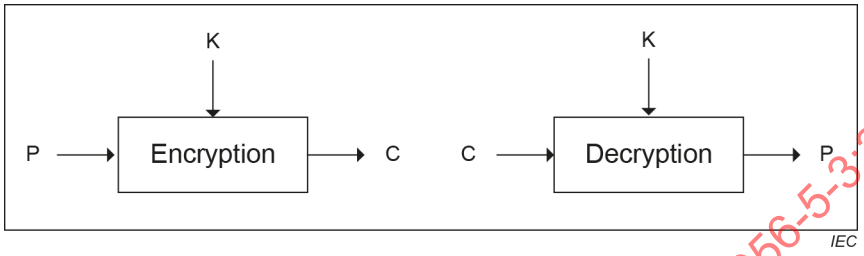



**Figure 16 – Encryption and decryption**

With symmetric key block cipher algorithms, the same plaintext block and key will always produce the same ciphertext block. This property does not provide acceptable security. Therefore, cryptographic modes of operation have been defined to address this problem (see 5.3.3.4).

#### 5.3.3.3 Advanced Encryption Standard

For the purposes of DLMS®/COSEM, the Advanced Encryption Standard (AES) as specified in FIPS PUB 197:2001 shall be used. AES operates on blocks (chunks) of data during an encryption or decryption operation. For this reason, AES is referred to as a block cipher algorithm.

NOTE The following text is quoted from NIST SP 800-21:2005, 3.3.1.3.

AES encrypts and decrypts data in 128-bit blocks, using 128, 192 or 256 bit keys. All three key sizes are adequate.

AES offers a combination of security, performance, efficiency, ease of implementation, and flexibility. Specifically, the algorithm performs well in both hardware and software across a wide range of computing environments. Also, the very low memory requirements of the algorithm make it very well suited for restricted-space environments.

#### 5.3.3.4 Encryption Modes of Operation

NOTE The following text is quoted from NIST SP 800-21:2005, 3.3.1.4.

With a symmetric key block cipher algorithm, the same plaintext block will always encrypt to the same ciphertext block when the same symmetric key is used. If the multiple blocks in a typical message (data stream) are encrypted separately, an adversary could easily substitute individual blocks, possibly without detection. Furthermore, certain kinds of data patterns in the plaintext, such as repeated blocks, would be apparent in the ciphertext.

Cryptographic modes of operation have been defined to address this problem by combining the basic cryptographic algorithm with variable initialization values (commonly known as initialization vectors) and feedback rules for the information derived from the cryptographic operation.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

NIST SP 800-38D:2007 specifies the Galois/Counter Mode (GCM), an algorithm for authenticated encryption with associated data, and its specialization, GMAC, for generating a message authentication code (MAC) on data that is not encrypted. GCM and GMAC are modes of operation for an underlying approved symmetric key block cipher. See 5.3.3.3.

#### 5.3.3.5 Message Authentication Code

NOTE The following text is quoted from NIST SP 800-21:2005, 3.3.2.

Message Authentication Codes (MACs) provide an assurance of authenticity and integrity. A MAC is a cryptographic checksum on the data that is used to provide assurance that the data has not changed or been altered and that the MAC was computed by the expected party (the sender). Typically, MACs are used between two parties that share a secret key to authenticate information exchanged between those parties.

Figure 17 depicts the use of message authentication codes (MACs).

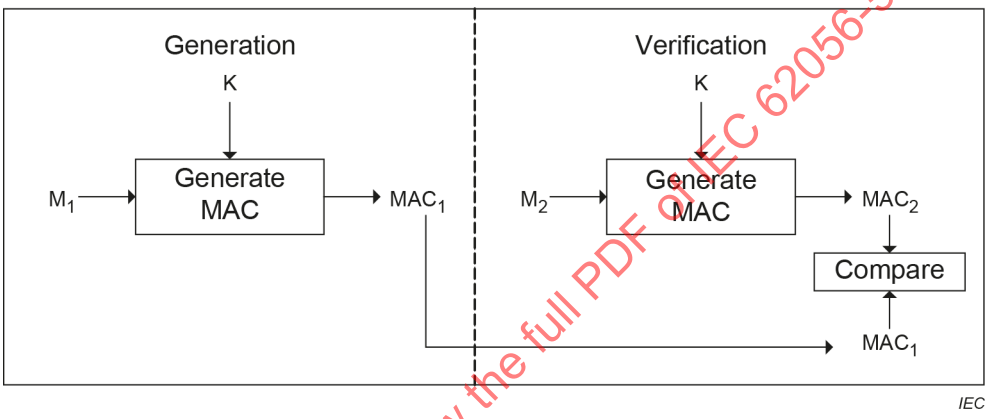



**Figure 17 – Message Authentication Codes (MACs)**

A MAC (MAC1) is computed on data (M1) using a key (K). M1 and MAC1 are then saved or transmitted. At a later time, the authenticity of the retrieved or received data is checked by labelling the retrieved or received data as M2 and computing a MAC (MAC2) on it using the same key (K). If the retrieved or received MAC (MAC1) is the same as the newly computed MAC (MAC2), then it can be assumed that the retrieved or received data (M2) is the same as the original data (M1) (i.e., M1 = M2). The verifying party also knows who the sending party is because no one else knows the key.

Typically, MACs are used to detect data modifications that occur between the initial generation of the MAC and the verification of the received MAC. They do not detect errors that occur before the MAC is originally generated.

Message integrity is frequently provided using non-cryptographic techniques known as error detection codes. However, these codes can be altered by an adversary to the adversary's benefit. The use of an approved cryptographic mechanism, such as a MAC, addresses this problem. That is, the integrity provided by a MAC is based on the assumption that it is not possible to generate a MAC without knowing the cryptographic key. An adversary without knowledge of the key will be unable to modify data and then generate an authentic MAC on the modified data. It is therefore crucial that MAC keys be kept secret.

For the purposes of DLMS®/COSEM, the GMAC algorithm as specified in 5.3.3.7.2 shall be used.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.3.3.6 Key wrapping

NOTE The following text is quoted from NIST SP 800-21:2005, 3.3.3.

Symmetric key algorithms may be used to wrap (i.e., encrypt) keying material using a key-wrapping key (also known as a key encrypting key). The wrapped keying material can then be stored or transmitted securely. Unwrapping the keying material requires the use of the same key-wrapping key that was used during the original wrapping process.

Key wrapping differs from simple encryption in that the wrapping process includes an integrity feature. During the unwrapping process, this integrity feature detects accidental or intentional modifications to the wrapped keying material. For the purposes of DLMS®/COSEM the AES key wrap algorithm shall be used; see 5.3.3.8.

#### 5.3.3.7 Galois/Counter Mode

##### 5.3.3.7.1 General

NOTE The following text is taken from NIST SP 800-38D:2007, Clause 3.

Galois/Counter Mode (GCM) is an algorithm for authenticated encryption with associated data. GCM is constructed from an approved symmetric key block cipher with a block size of 128 bits, such as the Advanced Encryption Standard (AES) algorithm, see FIPS PUB 197. Thus, GCM is a mode of operation of the AES algorithm.

GCM provides assurance of the confidentiality of data using a variation of the Counter mode of operation for encryption.

GCM provides assurance of the authenticity of the confidential data (up to about 64 gigabytes per invocation) using a universal hash function that is defined over a binary Galois (i.e., finite) field (GHASH). GCM can also provide authentication assurance for additional data (of practically unlimited length per invocation) that is not encrypted.

If the GCM input is restricted to data that is not to be encrypted, the resulting specialization of GCM, called GMAC, is simply an authentication mode on the input data.

GCM provides stronger authentication assurance than a (non-cryptographic) checksum or error detecting code; in particular, GCM can detect both 1) accidental modifications of the data and 2) intentional, unauthorized modifications.

In DLMS®/COSEM, it is also possible to use GCM to provide confidentiality only: in this case, the authentication tags are simply not computed and checked.

##### 5.3.3.7.2 GCM functions

NOTE The following is based on NIST SP 800-38D:2007, 5.2.

The two functions that comprise GCM are called authenticated encryption and authenticated decryption; see Figure 18.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

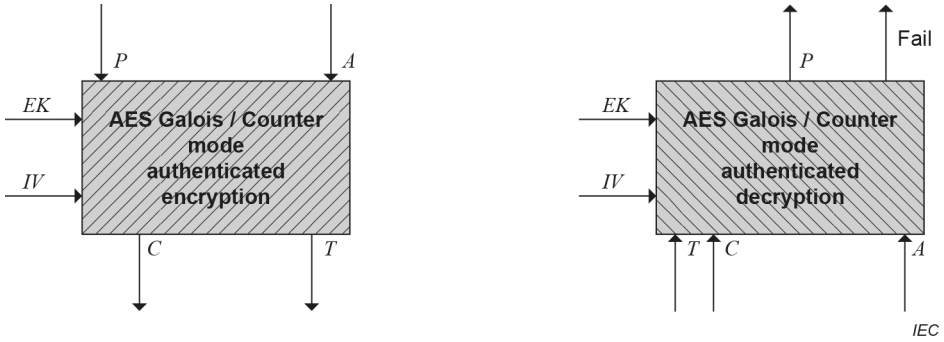



**Figure 18 – GCM functions**

The authenticated encryption function encrypts the confidential data and computes an authentication tag on both the confidential data and any additional, non-confidential data. The authenticated decryption function decrypts the confidential data, contingent on the verification of the tag.

When the input is restricted to non-confidential data, the resulting variant of GCM is called GMAC. For GMAC, the authenticated encryption and decryption functions become the functions for generating and verifying an authentication tag on the non-confidential data.

Finally, if authentication is not required, the authenticated encryption function encrypts the confidential data, but the authentication tag is not computed. The authenticated decryption function decrypts the confidential data, but no authentication tag is computed and verified.

In DLMS®/COSEM, the use of authentication and encryption is indicated by bit 4 and bit 5 of the Security Control Byte, specified in 5.7.2.4.

a) The authenticated encryption function – given the selection of a block cipher key $EK$ – has three input strings:
   - a plaintext, denoted $P$;
   - Additional Authenticated Data (AAD), denoted $A$;
   - an initialization vector (IV) denoted $IV$.

The plaintext and the AAD are the two categories of data that GCM protects. GCM protects the authenticity of the plaintext and the AAD; GCM also protects the confidentiality of the plaintext, while the AAD is left in the clear.

The IV is essentially a nonce, i.e. a value that is unique within the specified (security) context, which determines an invocation of the authenticated encryption function on the input data to be protected. See 5.3.3.7.3.

The bit lengths of the input strings to the authenticated encryption function shall meet the following requirements:

- len($P$) < $2^{39}$-256;
- len($A$) < $2^{64}$-1;
- 1 ≤ len($IV$) ≤ $2^{64}$-1.

The bit lengths of $P$, $A$ and $IV$ shall all be multiples of 8, so that these values are byte strings.

There are two outputs:

- a ciphertext, denoted $C$ whose bit length is the same as that of the plaintext P;
- an authentication tag, or tag, for short, denoted $T$.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

b) The authenticated decryption function – given the selection of a block cipher key $EK$ – has four input strings:

- the initialization vector, denoted $IV$;
- the ciphertext, denoted $C$;
- the Additional Authenticated Data (AAD), denoted $A$;
- the authentication tag, denoted $T$.

The output is one of the following:

- the plaintext $P$ that corresponds to the ciphertext $C$, or
- a special error code denoted $FAIL$ in this International Standard.

The output $P$ indicates that $T$ is the correct authentication tag for $IV$, $A$, and $C$; otherwise, the output is $FAIL$.

#### 5.3.3.7.3 The initialization vector, *IV*

In DLMS®/COSEM, for the construction of the initialization vector $IV$ deterministic construction as specified in NIST SP 800-38D:2007, 8.2.1 shall be used: the $IV$ is the concatenation of two fields, called the fixed field and the invocation field. The fixed field shall identify the physical device, or, more generally, the (security) context for the instance of the authenticated encryption function. The invocation field shall identify the sets of inputs to the authenticated encryption function in that particular device.

For any given key, no two distinct physical devices shall share the same fixed field, and no two distinct sets of inputs to any single device shall share the same invocation field.

The length of the $IV$ shall be 96 bits (12 octets): $\mathrm{len}(IV) = 96$. Within this:

- the leading (i.e. the leftmost) 64 bits (8 octets) shall hold the fixed field. It shall contain the system title, see 4.1.3.4;
- the trailing (i.e. the rightmost) 32 bits shall hold the invocation field. The invocation field shall be an integer counter.

Each encryption key (*EK*) has two invocation counters (*IC*) associated with it, one for the authenticated encryption function and the other for the authenticated decryption function. The *EK* is for block cyphering. The following rules apply:

- when the key is established, the corresponding *IC's* are reset to 0;
- when the authenticated encryption function is used, the corresponding *IC* is used after which it is incremented by 1. If the maximum value of the *IC* has been reached, any further invocations of the authenticated encryption function shall return an error and the *IC* shall not be incremented.
- When the authenticated decryption function is used, the value of the *IC* is verified. The value shall be equal to or greater than the lowest acceptable value.

  If the value being verified satisfies this requirement, the lowest acceptable is set to the *IC* value verified plus 1 following the use of the authentication decryption function.

  If the value being verified is less than the lowest acceptable value, the verification fails and so does the authenticated decryption function.

  If the value being verified is equal to the maximum value, the authenticated decryption function shall return an error.

  NOTE The maximimum number of invocations is $2^{32}$-1.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The bit length of the fixed field limits the number of distinct physical devices that can implement the authenticated encryption function for the given key to $2^{64}$. The bit length of the invocation field limits the number of invocations of the authenticated encryption function to $2^{32}$ with any given input sets without violating the uniqueness requirement.

#### 5.3.3.7.4 The encryption key, *EK*

GCM uses a single key, the block cipher key. In DLMS®/COSEM, this is known as the encryption key, denoted *EK*. Its size depends on the security suite – see 5.3.7 – and shall be:

- for security suite 0 and 1, 128 bits (16 octets): $\text{len}(EK) = 128$;
- for security suite 2, 256 bits (32 octets): $\text{len}(EK) = 256$;

The key shall be generated uniformly at random, or close to uniformly at random, i.e., so that each possible key is (nearly) equally likely to be generated. Consequently, the key will be fresh, i.e., unequal to any previous key, with high probability. The key shall be secret and shall be used exclusively for GCM with the chosen block cipher AES. Additional requirements on the establishment and management of keys are discussed in NIST SP 800-38D 2007, 8.1.

#### 5.3.3.7.5 The authentication key, *AK*

In DLMS®/COSEM, for additional security, an authentication key denoted *AK* is also specified. When present, it shall be part of the Additional Authenticated Data, AAD. For its length and its generation, the same rules apply as for the encryption key.

#### 5.3.3.7.6 Length of the authentication tag

The bit length of the authentication tag, denoted $t$, is a security parameter. In security suites 0, 1 and 2 its value shall be 96 bits.

### 5.3.3.8 AES key wrap

For wrapping key data DLMS®/COSEM has selected the AES key wrap algorithm specified in RFC 3394. The algorithm is designed to wrap or encrypt key data. It operates on blocks of 64 bits. Before being wrapped, the key data is parsed into *n* blocks of 64 bits. The only restriction the key wrap algorithm places on *n* is that *n* has to be at least two.

The AES key wrap can be configured to use any of the three key sizes supported by the AES codebook: 128, 192, 256.

The two algorithms are key wrap and key unwrap.

The inputs to the key wrapping process are the Key Encrypting Key *KEK* and the plaintext to be wrapped. The plaintext consists of n 64-bit blocks, containing the key data being wrapped. The output is the ciphertext, (n+1) 64 bit values.

The inputs to the unwrap process are the *KEK* and (n+1) 64-bit blocks of ciphertext consisting of previously wrapped key. It returns n blocks of plaintext consisting of the n 64-bit blocks of the decrypted key data.

In DLMS®/COSEM, the size of *KEK* depends on the security suite – see 5.3.7 – and shall be:

- for security suite 0 and 1, 128 bits (16 octets): $\text{len}(KEK) = 128$;
- for security suite 2, 256 bits (32 octets): $\text{len}(KEK) = 256$.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.3.4 Public key algorithms

#### 5.3.4.1 General

In general, public key cryptography systems use hard-to-solve problems as the basis of the algorithm. The RSA algorithm is based on the prime factorization of very large integers. Elliptic Curve Cryptography (ECC) is based on the difficulty of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP). ECC provides similar levels of security compared to RSA but with significantly reduced key sizes. ECC is particularly suitable for embedded devices and therefore it has been selected for use in DLMS®/COSEM.

Public key algorithms are used in DLMS®/COSEM for the following purposes:

- authentication of communicating partners;
- digital signature of xDLMS APDUs and COSEM data;
- key agreement.

NOTE 1 The following text is quoted from NIST SP 800-21:2005, 3.4.

Asymmetric key algorithms (often called public key algorithms) use two keys: a public key and a private key, which are mathematically related to each other. The public key may be made public; the private key shall remain secret if the data is to retain its cryptographic protection. Even though there is a relationship between the two keys, the private key cannot be determined from the public key. Which key to be used to apply versus remove or check the protection depends on the service to be provided. For example, a digital signature is computed using a private key and the signature is verified using the public key; for those algorithms also capable of encryption, the encryption is performed using the public key, and the decryption is performed using the private key.

NOTE 2 Not all public key algorithms are capable of multiple functions, e.g., generating digital signatures and encryption. Asymmetric key algorithms are not used for encryption in DLMS®/COSEM.

Asymmetric key algorithms are used primarily as data integrity, authentication, and non-repudiation mechanisms (i.e., digital signatures), as well as for key establishment.

Some asymmetric key algorithms use domain parameters, which are additional values necessary for the operation of the cryptographic algorithm. These values are mathematically related to each other. Domain parameters are usually public and are used by a community of users for a substantial period of time.

The secure use of asymmetric key algorithms requires that users obtain certain assurances:

- assurance of domain parameter validity provides confidence that the domain parameters are mathematically correct;
- assurance of public key validity provides confidence that the public key appears to be a suitable key; and
- assurance of private key possession provides confidence that the party that is supposedly the owner of the private key really has the key.

Some asymmetric key algorithms may be used for multiple purposes (e.g., for both digital signatures and key establishment). Keys used for one purpose shall not be used for other purposes.

#### 5.3.4.2 Elliptic curve cryptography

##### 5.3.4.2.1 General

Elliptic curve cryptography involves arithmetic operations on an elliptic curve over a finite field. Elliptic curves can be defined over any field of numbers (i.e., real, integer, complex) although they are most often used over finite prime fields for applications in cryptography.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

An elliptic curve on a prime field consists of the set of real numbers $(x, y)$ that satisfy the equation:

$$y^2 = x^3 + ax + b$$

The set of all of the solutions to the equation forms the elliptic curve. Changing $a$ and $b$ changes the shape of the curve, and small changes in these parameters can result in major changes in the set of (x, y) solutions.

#### 5.3.4.2.2 NIST recommended elliptic curves

FIPS PUB 186-4:2013 recommends five prime field elliptic curves over a prime field $GF(p)$. Of these, the curves P-256 and P-384 have been selected for DLMS®/COSEM as shown in Table 3.

**Table 3 – Elliptic curves in DLMS®/COSEM security suites**

| DLMS® security suite | Curve name in FIPS PUB 186-4:2013 | ASN.1 Object Identifier |
|---|---|---|
| Suite 0 | – | – |
| Suite 1 | NIST curve P-256 | 1.2.840.10045.3.1.7 |
| Suite 2 | NIST curve P-384 | 1.3.132.0.34 |
| NOTE The ASN.1 Object Identifier appears in the Certificate under AlgorithmIdentifier: Parameters. See 5.6.4.2. | | |

### 5.3.4.3 Data conversions

#### 5.3.4.3.1 Overview

This subclause describes the data conversion primitives that shall be used to convert between different data types used to specify public key algorithms: octet strings (OS), bit strings (BS), integers (I), field elements (FE) and elliptic curve points (ECP). DLMS®/COSEM uses octet strings to represent elements of public key algorithms and uses conversion primitives between these data types from and to octet strings. The octet string $M_{d-1}\ M_{d-2}\ \ldots\ M_0$ of length d is encoded as A-XDR OCTET STRING where the leftmost octet $M_{d-1}$ corresponds to first octet of the encoded value of the OCTET STRING.

#### 5.3.4.3.2 Conversion between Bit Strings and Octet Strings (BS2OS)

The data conversion primitive that converts a bit string to an octet string is called the Bit String to Octet String Conversion Primitive, or BS2OS. It takes the bit string as input and outputs the octet string. The bit string $_{bl-1}\ b_{l-2}\ \ldots\ b_0$ of length $l$ shall be converted to an octet string $M_{d-1}\ M_{d-2}\ \ldots\ M_0$ of length $d = \lceil l/8 \rceil$.

The conversion pads enough zeroes on the left to make the number of bits multiple of eight, and then breaks it into octets.

More precisely, conversion shall be as follows:

- for $0 \le i < d - 1$, let the octet $M_i = b_{8i+7}\ b_{8i+6}\ \ldots\ b_{8i},$;
- the leftmost octet $M_{d-1}$ shall have its leftmost $8d - l$ bits set to zero;
- its rightmost $8 - (8d - l)$ bits shall be $b_{l-1}\ b_{l-2}\ \ldots\ b_{8d-8}$.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.3.4.3.3 Conversion between Octet Strings and Bit Strings (OS2BS)

The data conversion primitive that converts an octet string to a bit string is called the Octet String to Bit String Conversion Primitive, or OS2BS. It takes the octet string as input and outputs the bit string. The octet string $M_{d-1}\ M_{d-2}\ \ldots\ M_0$ of length d shall be converted to a bit string $b_{l-1}\ b_{l-2}\ \ldots\ b_0$ of desired length $l$. where $d = \lceil l/8 \rceil$ and the leftmost $8d\text{-}l$ bits of the leftmost octet are zero.

More precisely conversion shall be as follows:

- for $0 \le i < l - 1$, let the bits $b_{8i+7}\ b_{8i+6}\ \ldots\ b_{8i} = M_i$;
- its leftmost $(8d - l)$ bits of the leftmost octet shall be zero.

#### 5.3.4.3.4 Conversion between Integers and Octet Strings (I2OS)

The data conversion primitive that converts an integer to an octet string is called the Integer to Octet String Conversion Primitive, or I2OS. It takes a non-negative integer $x$ and the desired length $d$ of the octet string as input. The length $d$ has to satisfy $256^d > x$, otherwise it shall output "error". I2OS outputs the corresponding octet string.

The integer $x$ shall be written in its unique $l$-digit representation base 256:

- $x = x^{d-1} \cdot 256^{d-1} + x^{d-2} \cdot 256^{d-2} + \cdots + x_1 \cdot 256 + x_0$,;
- where $0 \le x_i < 256$ for $0 \le i \le d\text{-}1$;
- $M_i = x_i$, for $0 \le i \le d\text{-}1$.

The output octet string shall be $M_{d-1}\ M_{d-2}\ \ldots\ M_0$.

#### 5.3.4.3.5 Conversion between Octet Strings and Integers (OS2I)

The data conversion primitive that converts an octet string to an integer is called the Octet String to Integer Conversion Primitive, or OS2I. It takes the octet string $M_{d-1}\ M_{d-2}\ \ldots\ M_0$ of length $d$ as an input and outputs the corresponding integer $x$. In the case of the octet string of length zero, the conversion outputs integer 0.

Each octet is interpreted as a non-negative integer to the base 256. More precisely, conversion shall be as follows:

- $x_i = M_i$, for $0 \le i \le d\text{-}1$;
- $x = x_{d-1} \cdot 256^{d-1} + x_{d-2} \cdot 256^{d-2} + \cdots + x_1 \cdot 256 + x_0$.

#### 5.3.4.3.6 Conversion between Field Elements and Octet Strings (FE2OS)

The data conversion primitive that converts a field element to an octet string is called the Field Element to Octet String Conversion Primitive, or FE2OS. It takes a field element as input and outputs the corresponding octet string. A field element $x \in F_p$ is converted to an octet string $M_{d-1}\ M_{d-2}\ \ldots\ M_0$ of length $d = \lceil log256\ p \rceil$ by applying I2OS conversion primitive with parameter $l$, where

- FE2OS($x$) = I2OS($x$, $l$).

#### 5.3.4.3.7 Conversion between Octet Strings and Field Elements (OS2FE)

The data conversion primitive that converts an octet string to a field element is called the Octet String to Field Element Conversion Primitive, or OS2FE. It takes an octet string as input and outputs the corresponding field element. An octet string $M_{d-1}\ M_{d-2}\ \ldots\ M_0$ of length $d$ is converted to field element $x \in F_p$ by applying OS2I conversion primitive where:

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- OS2FE($x$) = OS2I($x$) mod $p$.

#### 5.3.4.4 Digital signature

NOTE The following text is quoted from NIST SP 800-21:2005, 3.4.1.

A digital signature is an electronic analogue of a written signature that can be used in proving to the recipient or a third party that the message was signed by the originator (a property known as non-repudiation). Digital signatures may also be generated for stored data and programs so that the integrity of the data and programs may be verified at a later time.

Digital signatures authenticate the integrity of the signed data and the identity of the signatory. A digital signature is represented in a computer as a string of bits and is computed using a digital signature algorithm that provides the capability to generate and verify signatures. Signature generation uses a private key to generate a digital signature. Signature verification uses the public key that corresponds to, but is not the same as, the private key to verify the signature. Each signatory possesses a private and public key pair. Signature generation can be performed only by the possessor of the signatory's private key. However, anyone can verify the signature by employing the signatory's public key. The security of a digital signature system is dependent on maintaining the secrecy of a signatory's private key. Therefore, users shall guard against the unauthorized acquisition of their private keys.

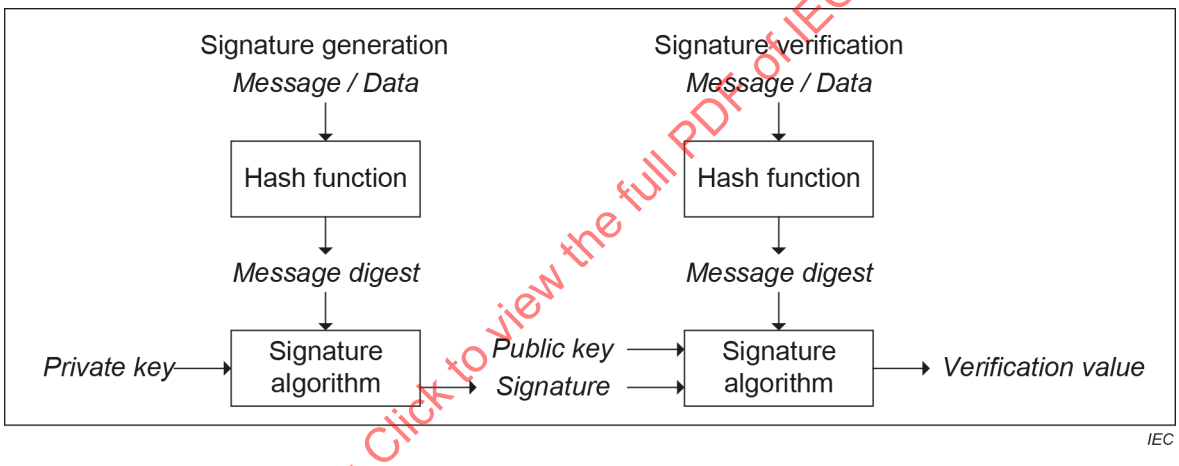



**Figure 19 – Digital signatures**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Figure 19 depicts the digital signature process. A hash function (see 5.3.2) is used in the signature generation process to obtain a condensed version of data to be signed, called a message digest or hash value. The message digest is then input to the digital signature algorithm to generate the digital signature. The digital signature is sent to the intended verifier along with the signed data (often called the message). The verifier of the message and signature verifies the signature by using the signatory's public key. The same hash function and digital signature algorithm shall also be used in the verification process. Similar procedures may be used to generate and verify signatures for stored as well as transmitted data.

#### 5.3.4.5 Elliptic curve digital signature (ECDSA)

For DLMS®/COSEM the elliptic curve digital signature (ECDSA) algorithm as specified in FIPS PUB 186-4:2013 has been selected. NSA1 provides an implementation guide.

In DLMS®/COSEM the elliptic curves and algorithms used shall be:

- in the case of Security Suite 1, the elliptic curve P-256 with the SHA-256 hash algorithm;
- in the case of Security Suite 2, the elliptic curve P-384 with the SHA-384 hash algorithm.

The inputs to ECDSA digital signature generation are the following:

- the message $M$ to be signed;

  NOTE 1 In the DLMS®/COSEM context "Message" may be an xDLMS APDU, see 5.7.2 or COSEM data, see 5.7.5.

- the private key of the signatory, $d$.

The output is the ECDSA signature $(r, s)$ over M.

In DLMS®/COSEM the plain format shall be used: the signature $(r, s)$ is encoded as octet string $R$ II $S$, i.e. as concatenation of the octet strings $R$ = I2OS($r$, $l$) and $S$ = I2OS($s$, $l$) with $l = [log_{256}\, n]$. Thus, the signature has a fixed length of 2 $l$ octets.

NOTE 2 Here, $n$ is the order of the base point $G$ of the elliptic curve. I2OS is the Integer to Octet String Conversion Primitive. See 5.3.4.3.

The inputs to the verification of the ECDSA digital signature generation are the following:

- the signed message $M'$;
- the received ECDSA signature ($r'$, $s'$);
- the authentic public key of the signatory, $Q$.

The process of generating and verifying the signatures shall be as specified in NSA1, 3.4.

#### 5.3.4.6 Key agreement

##### 5.3.4.6.1 Overview

Key agreement allows two entities to jointly compute a shared secret and derive secret keying material from it. See also NIST SP 800-56A Rev. 2: 2013.

For DLMS®/COSEM three elliptic curve key agreement schemes have been selected from NIST SP 800-56A Rev. 2: 2013:

- the Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme;
- the One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme;
- the Static Unified Model C(0e, 2s, ECC CDH) scheme.

NOTE For NSA Suite B the first two schemes have been selected.

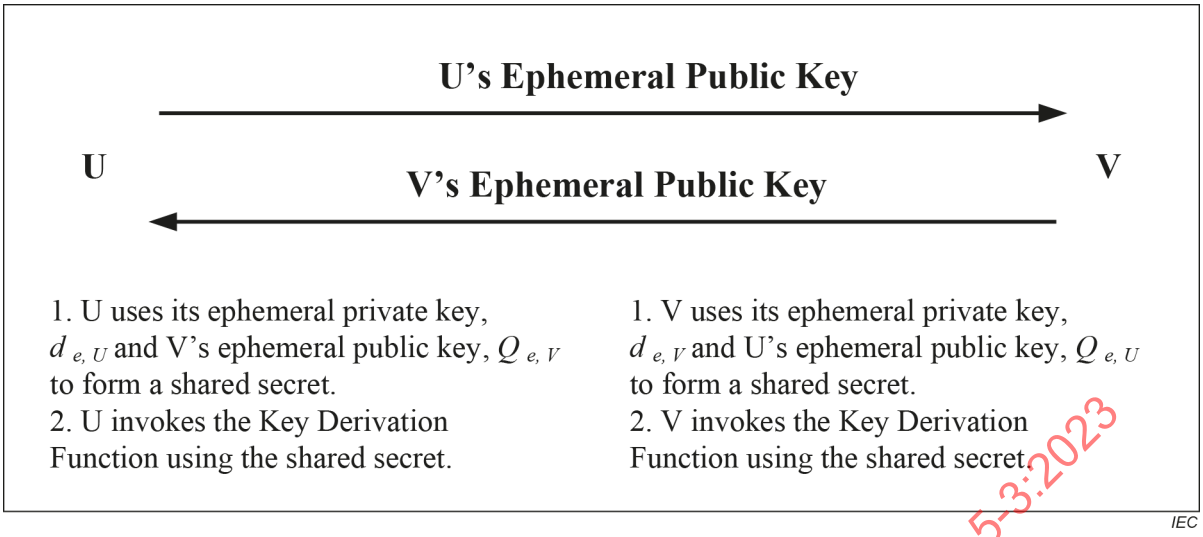
##### 5.3.4.6.2 The Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme

This scheme is for use between a DLMS®/COSEM client and a server to agree on the master key, on global encryption keys and/or on the authentication key. The client plays the role of party U and the server plays the role of party V. The process is supported by the methods of the "Security setup" interface class; see IEC 62056-6-2:2021, 4.4.7.

The parties generate an ephemeral key pair from the domain parameters $D$. The parties exchange ephemeral public keys and then compute the shared secret $Z$ using the domain parameters, their ephemeral private key and the ephemeral public key of the other party. The secret keying material is derived using the key derivation function specified in 5.3.4.6.5 from the shared secret $Z$ and other input.

The process is specified in detail in NIST SP 800-56A Rev. 2: 2013, 6.1.2.2 and NSA2:2009, 3.1 and it is shown in Figure 20 and Table 4.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




NOTE This figure reproduces NSA2:2009, Figure 1.

**Figure 20 – C(2e, 0s) scheme: each party contributes only an ephemeral key pair**

Prerequisites:

a) each party has an authentic copy of the same set of domain parameters, $D$. $D$ shall be selected from one of the two sets of domain parameters, see Annex G;

b) the parties have agreed on using the NIST Concatenation KDF; see 5.3.4.6.5. SHA-256 is the hash function to use with the domain parameters for P-256 and SHA-384 is the hash function to use with the domain parameters for P-384;

c) prior to or during the key agreement process, the parties obtain the identifier associated with the other party during the key agreement scheme.

NOTE See also NIST SP 800-56A Rev. 2: 2013 and NSA2 for additional information on assurance on the validity of the domain parameters, the validity of the private and public key and the assurance of the possession of the private key.

**Table 4 – Ephemeral Unified Model key agreement scheme summary**

| | Party U | Party V |
|---|---|---|
| Domain parameters | $(q, FR, a, b\{,SEED\}, G, n, h)$<br>as determined by the security suite | $(q, FR, a, b\{,SEED\}, G, n, h)$<br>as determined by the security suite |
| Static data | N/A | N/A |
| Ephemeral data | Ephemeral private key, $d_{e,U}$<br>Ephemeral public key, $Q_{e,U}$ | Ephemeral private key, $d_{e,V}$<br>Ephemeral public key, $Q_{e,V}$ |
| Computation | Compute $Z$ by calling ECC CDH using $d_{e,U}$ and $Q_{e,V}$ | Compute $Z$ by calling ECC CDH using $d_{e,V}$ and $Q_{e,U}$ |
| Derive secret keying material | 1. Compute kdf($Z$, $OtherInput$)<br>2. Destroy $Z$ | 1. Compute kdf($Z$, $OtherInput$)<br>2. Destroy $Z$ |
| NOTE This table is based on NIST SP 800-56A Rev. 2: 2013, Table 18 and NSA2:2009, Table 1. | | |

The rationale for choosing a C(2e, 0s) scheme is specified in NIST SP 800-56A Rev. 2: 2013, 8.2.

The use of this scheme in DLMS®/COSEM is further explained in Clause I.1.

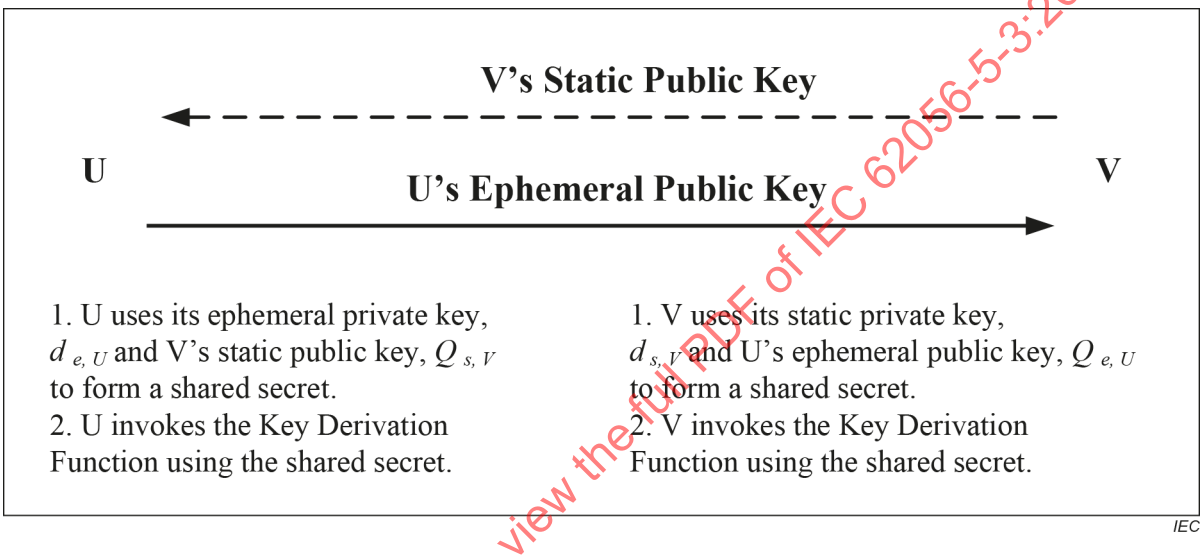
IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.3.4.6.3 The One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme

This scheme is for use by a DLMS®/COSEM server and another party to agree on an ephemeral encryption key to protect xDLMS APDUs or COSEM data. The party sending the message (the originator) plays the role of party U and the other party (the recipient) plays the role of party V.

NOTE 1 The terms originator and recipient refer to fields of the general-ciphering APDU.

For this scheme, party U generates an ephemeral key pair; party V has only a static key pair. Party U obtains Party V's static public key in a trusted manner (for example, from a certificate signed by a trusted CA) and sends its ephemeral public key to party V. Each party derives the shared secret $Z$ by using its own private key and the other party's public key. Each party derives secret keying material using the key derivation method specified in 5.3.4.6.5 from the shared secret $Z$ and other input.




NOTE This figure reproduces NSA2:2009, Figure 2.

**Figure 21 – C(1e, 1s) schemes: party U contributes an ephemeral key pair, and party V contributes a static key pair**

The process is specified in detail in NIST SP 800-56A Rev. 2: 2013, 6.2.2.2 and NSA2:2009, 3.2 and is shown in Figure 21 and Table 5.

Prerequisites:

a) each party shall have an authentic copy of the same set of domain parameters, $D$. $D$ shall be selected from one of the two sets of domain parameters, see Annex G;

b) party V shall have been designated as the owner of a static key pair that was generated as specified in 5.6.2 using the set of domain parameters, $D$;

c) the parties have agreed on using the NIST Concatenation KDF, see 5.3.4.6.5. SHA-256 is the hash function to use with the domain parameters for P-256 and SHA-384 is the hash function to use with the domain parameters for P-384;

d) prior to or during the key agreement process, the parties obtain the identifier associated with the other party during the key agreement scheme. Party U shall obtain the static public key that is bound to party V's identifier. This static public key shall be obtained in a trusted manner (e.g., from a certificate signed by a trusted CA).

NOTE 2 See also NIST SP 800-56A Rev. 2: 2013 and NSA2 for additional information on assurance on the validity of the domain parameters, the validity of the private and public key and the assurance of the possession of the private key.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 5 – One-pass Diffie-Hellman key agreement scheme summary**

| | Party U | Party V |
|---|---|---|
| **Domain Parameters** | $(q, FR, a, b\{,SEED\}, G, n, h)$<br>As determined by the security suite | $(q, FR, a, b\{,SEED\}, G, n, h)$<br>As determined by the security suite |
| **Static Data** | N/A | Static private key, $d_{s,\,V}$<br>Static public key, $Q_{s,\,V}$ |
| **Ephemeral Data** | Ephemeral private key, $d_{e,\,U}$<br>Ephemeral public key, $Q_{e,\,U}$ | N/A |
| **Computation** | Compute $Z$ by calling ECC CDH using $d_{e,\,U}$ and $Q_{s,\,V}$ | Compute $Z$ by calling ECC CDH using $d_{s,\,V}$ and $Q_{e\,,U}$ |
| **Derive Secret Keying Material** | 1. Compute kdf($Z$, $OtherInput$)<br>2. Destroy $Z$ | 1. Compute kdf($Z$, $OtherInput$)<br>2. Destroy $Z$ |
| NOTE This table is based on NIST SP 800-56A Rev. 2: 2013 Table 24 and NSA2:2009, Table 2. | | |

The rationale for choosing this scheme is specified in NIST SP 800-56A Rev. 2: 2013, 8.4.

The use of this scheme in DLMS®/COSEM is further explained in Clause I.2.

#### 5.3.4.6.4 The Static Unified Model C(0e, 2s, ECC CDH) scheme

This scheme is for use by a DLMS®/COSEM server and another party to agree on an ephemeral encryption key to protect xDLMS APDUs or COSEM data. The party sending the message (the originator) plays the role of party U and the other party (the recipient) plays the role of party V.

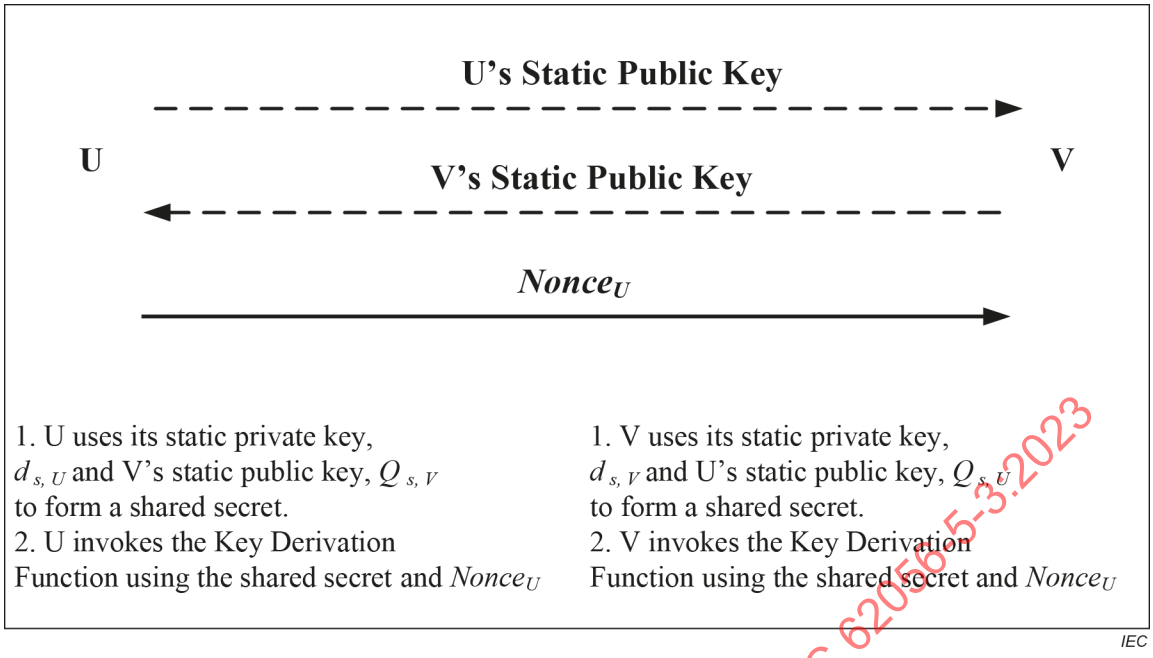
NOTE 1 The terms originator and recipient refer to fields of the general-ciphering APDU.

In this case, the parties use only static key pairs. Each party obtains the other party's static public key. A nonce, $Nonce_U$, is sent by party U to party V to ensure that the derived keying material is different for each key-establishment transaction.

The parties derive the shared secret $Z$ using their own static private key and the other party's static public key. Secret keying material is derived using the key-derivation function specified in 5.3.4.6.5 the shared secret $Z$, U and V's identifier and the nonce.

The process is shown in Figure 22 and Table 6.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




NOTE This figure is based on NIST SP 800-56A Rev. 2: 2013, Figure 15.

**Figure 22 – C(0e, 2s) scheme: each party contributes only a static key pair**

Prerequisites:

a) each party shall have an authentic copy of the same set of domain parameters, $D$. $D$ shall be selected from one of the two sets of domain parameters, see Annex G;

b) each party has been designated as the owner of a static key pair that was generated as specified in 5.6.2 using the set of domain parameters, $D$;

c) the parties have agreed on using the NIST Concatenation KDF, see 5.3.4.6.5. SHA-256 is the hash function to use with the domain parameters for P-256 and SHA-384 is the hash function to use with the domain parameters for P-384;

d) prior to or during the key agreement process, each party shall obtain the identifier associated with the other party during the key agreement scheme;

e) both parties shall obtain the static public key of the other party that is bound to the identifier. These static public keys shall be obtained in a trusted manner (e.g., from a certificate signed by a trusted CA.

NOTE 2 See also NIST SP 800-56A Rev. 2: 2013 for additional information on assurance on the validity of the domain parameters, the validity of the private and public key and the assurance of the possession of the private key.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 6 – Static Unified Model key agreement scheme summary**

| | Party U | Party V |
|---|---|---|
| **Domain Parameters** | *(q, FR, a, b{,SEED}, G, n, h)*<br>As determined by the security suite | *(q, FR, a, b{,SEED}, G, n, h)*<br>As determined by the security suite |
| **Static Data** | Static private key, $d_{s,\,U}$<br>Static public key, $Q_{s,\,U}$ | Static private key, $d_{s,\,V}$<br>Static public key, $Q_{s,\,V}$ |
| **Ephemeral Data** | $Nonce_U$ | |
| **Computation** | Compute $Z$ by calling ECC CDH using $d_{s,\,U}$ and $Q_{s,\,V}$ | Compute $Z$ by calling ECC CDH using $d_{s,\,V}$ and $Q_{s\,,U}$ |
| **Derive Secret Keying Material** | 1. Compute *DerivedKeyingMaterial* using $Nonce_U$<br>2. Destroy $Z$ | 1. Compute *DerivedKeyingMaterial* using $Nonce_U$<br>2. Destroy $Z$ |
| NOTE 1 This table is based on NIST SP 800-56A Rev. 2: 2013, Table 26.<br>NOTE 2 The value of $Z$ is the same in all C(0e, 2s) key-establishment transactions between the same two parties, therefore if it is ever compromised, then all of the keying material derived in past, current, and future C(0e, 2s) key-agreement transactions between these same two entities that employ these same static key pairs may be compromised as well. Any shared secret $Z$ that is not 'zeroized' shall be stored and used with the same security protections as private keys. | | |

The rationale for choosing this scheme is specified in NIST SP 800-56A Rev. 2: 2013, 8.5.

The use of this scheme in DLMS®/COSEM is further explained in Clause I.3.

#### 5.3.4.6.5 Key Derivation Function – The NIST Concatenation KDF

The NIST Concatenation KDF as specified in NIST SP 800-56A Rev. 2: 2013, 5.8.1.1 and NSA2, Clause 5 shall be used. It is as follows.

**Function call**: kdf(*Z, OtherInput*)

where *OtherInput* consists of *keydatalen* and *OtherInfo*.

**Implementation-Dependent Parameters:**

a) *hashlen*: an integer that indicates the length (in bits) of the output of the hash function, *hash*, used to derive blocks of secret keying material. This will be 256 (for SHA-256) in the case of security suite 1 and 384 (for SHA-384) in the case of security suite 2;

b) *max_hash_inputlen*: an integer that indicates the maximum length (in bits) of the bit string(s) input to the hash function.

The length shall be less than 264 bits for SHA-256 and less than 2128 bits for SHA-384.

**Function:** H: a hash function: SHA-256 in the case of security suite 1 and SHA-384 in the case of security suite 2.

**Input:**

c) *Z*: a byte string that represents the shared secret *z*;

d) *keydatalen*: an integer that indicates the length (in bits) of the secret keying material to be generated: 128 bit for security suite 1 and 256 bit for security suite 2;

e) *OtherInfo:* A bit string equal to the following concatenation:

*AlgorithmID* || *PartyUInfo* || *PartyVInfo* {||*SuppPubInfo*}{||*SuppPrivInfo*}

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

where the subfields are defined as follows:

f) *AlgorithmID*: A bit string that indicates how the derived secret keying material will be parsed and for which algorithm(s) the derived secret keying material will be used. See Table 8;

g) *PartyUInfo*: A bit string containing public information that is required by the application using this KDF to be contributed by Party U to the key derivation process;

h) *PartyVInfo*: A bit string containing public information that is required by the application using this KDF to be contributed by Party V to the key derivation process;

i) (Optional) *SuppPubInfo*: A bit string containing additional, mutually known public information. Not used in DLMS®/COSEM;

j) (Optional) *SuppPrivInfo*: A bit string containing additional, mutually known private information (for example, a shared secret symmetric key that has been communicated through a separate channel). Not used in DLMS®/COSEM.

The format and content of each subfield and substring is specified in Table 7.

**Table 7 – *OtherInfo* subfields and substrings**

| ***Subfield*** *Substring* | **Key agreement scheme** | | | **Length in octets** | **Value** |
|---|---|---|---|---|---|
| | **C(2e, 0s)** | **C(1e, 1s)** | **C(0e, 2s)** | | |
| | **Format** | | | | |
| ***AlgorithmID*** | Fixed | Fixed | Fixed | 7 | See Table 8 |
| ***PartyUinfo*** | Fixed | Fixed | Variable | 8+n | – |
| $ID_U$ | Fixed | Fixed | Fixed | 8 | originator-system-title |
| *NonceU* | – | – | Variable | n | *Datalen* = length of transaction-id, shall be 1 octet<br>*Data* = value of transaction-id |
| ***PartyVInfo*** | Fixed | Fixed | Fixed | 8 | – |
| $ID_V$ | Fixed | Fixed | Fixed | 8 | recipient-system-title |
| "originator-system-title", transaction-id" and "recipient-system-title" are the fields of the general-ciphering APDU | | | | | |

In DLMS®/COSEM, key derivation delivers a single key for a given purpose. The length is as determined by the security suite. *AlgorithmId* shall be as specified in Table 8. See also 7.2.2.4.

**Table 8 – Security algorithm ID-s**

| **Algorithm** | **COSEM cryptographic algorithm ID** | **Encoded value** |
|---|---|---|
| AES-GCM-128 | 2.16.756.5.8.3.0 | 60 85 74 05 08 03 00 |
| AES-GCM-256 | 2.16.756.5.8.3.1 | 60 85 74 05 08 03 01 |
| AES-WRAP-128 | 2.16.756.5.8.3.2 | 60 85 74 05 08 03 02 |
| AES-WRAP-256 | 2.16.756.5.8.3.3 | 60 85 74 05 08 03 03 |

### 5.3.5 Random number generation

Strong random number generator (RNG) shall be provided to generate the random numbers required for the various algorithms used in DLMS®/COSEM. The RNG shall be preferably non-deterministic. If a non-deterministic RBG is not available, the system shall make use of sufficient entropy to create a good quality seed for a deterministic RNG.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.3.6 Compression

NOTE Compression does not involve cryptography, but it is a transformation of the xDLMS APDU. For this reason, and as it is controlled together with symmetric key ciphering it is specified here.

Compression can be applied to COSEM data or xDLMS APDUs. This process can be combined with symmetric key ciphering. See 5.7.2.4.7 and 5.7.2.4.8. The compression algorithm shall be as specified in ITU-T V.44: 2000. This algorithm has been selected for to meet the following requirements:

- low processing load;
- low memory requirements;
- low latency.

The use of compression is indicated by bit 7 of the Security Control Byte, specified in 5.7.2.4.

### 5.3.7 Security suite

A security suite determines the set of cryptographic algorithms available for the various cryptographic primitives and the key sizes.

The DLMS®/COSEM security suites – see Table 9 – are based on NSA Suite B and include cryptographic algorithms for authentication, encryption, key agreement, digital signature and hashing specifically:

- authentication and encryption: the Advanced Encryption Standard (AES) shall be used as specified in FIPS PUB 197, with key sizes of 128 and 256 bits. AES shall be used with the Galois/Counter Mode (GCM) of operation specified in NIST SP 800-38D:2007;
- digital signature: the Elliptic Curve Digital Signature Algorithm (ECDSA) shall be used as specified FIPS PUB 186-4:2013 and in NSA1, using the curves P-256 or P-384;
- key agreement:
  - the Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme, see 5.3.4.6.2;
  - the One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme, see 5.3.4.6.3; and
  - the Static Unified Model C(0e, 2s, ECC CDH) scheme, see 5.3.4.6.4

  shall be used using the elliptic curves P-256 or P-384.
- hashing: the Secure Hash Algorithms (SHA) SHA-256 and SHA-384 shall be used as specified in FIPS PUB 180-4:2012.

In addition, a key wrapping and a compression algorithm are available.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 9 – DLMS®/COSEM security suites**

| Security Suite Id | Security suite name | Authenticated encryption | Digital signature | Key agreement | Hash | Key-transport | Compression |
|---|---|---|---|---|---|---|---|
| 0 | AES-GCM-128 | AES-GCM-128 | – | – | – | AES-128 key wrap | – |
| 1 | ECDH-ECDSA-AES-GCM-128-SHA-256 | AES-GCM-128 | ECDSA with P-256 | ECDH with P-256 | SHA-256 | AES-128 key wrap | V.44 |
| 2 | ECDH-ECDSA-AES-GCM-256-SHA-384 | AES-GCM-256 | ECDSA with P-384 | ECDH with P-384 | SHA-384 | AES-256 key wrap | V.44 |
| All other reserved | – | – | – | – | – | – | – |

### 5.4 Cryptographic keys – overview

A cryptographic key is a parameter used in conjunction with a cryptographic algorithm that determines its operation in such a way that an entity with knowledge of the key can reproduce or reverse the operation, while an entity without knowledge of the key cannot. In DLMS®/COSEM, examples of operations include:

- the transformation of plaintext into ciphertext;
- the transformation of ciphertext into plaintext;
- the computation and verification of an authentication code (MAC);
- key wrapping;
- applying and verifying digital signature;
- key agreement.

Keys used with symmetric key algorithms are specified in 5.5.

Keys used with public key algorithms are specified in 5.6.

### 5.5 Key used with symmetric key algorithms

#### 5.5.1 Symmetric keys types

Symmetric keys are classified according to:

a) their purpose:
   1) a key encrypting key (KEK) is used to encrypt / decrypt other symmetric keys; see 5.5.4. In DLMS®/COSEM this is the master key;
   2) an encryption key is used as the block cipher key of the AES-GCM algorithm, see also 5.3.3.7.4;
   3) an authentication key is used as Additional Authenticated Data (AAD) in the AES-GCM algorithm, see also 5.3.3.7.5.

b) their lifetime:
   1) static keys that are intended to be used for a relatively long period of time. In DLMS®/COSEM these may be:
      - a global key that may be used over several AAs established repeatedly between the same partners. A global key may be a unicast encryption key (GUEK), a broadcast encryption key (GBEK) or an authentication key (GAK);
      - a dedicated key that may be used repeatedly during a single AA established between two partners. Therefore, its lifetime is the same as the lifetime of the AA. A dedicated key can be only a unicast encryption key.
   2) ephemeral keys used generally for a single exchange within an AA.

For generation and distribution of symmetric keys, see NIST SP 800-57:2012, 8.1.5.2.

A master key and global keys are established between each DLMS®/COSEM client – server pair using one of the methods shown in Table 10. They should be renewed in appropriate intervals, see 5.3.3.7.3 and 5.5.6.

Dedicated keys are generated by the DLMS®/COSEM client and transported to the server in the dedicated-key field of the xDLMS InitiateRequest APDU, carried by the user-information field of the AARQ APDU. When the dedicated key is present, the xDLMS InitiateRequest APDU shall be authenticated and encrypted using the AES-GCM-128 / 256 algorithm, the global unicast encryption key and – if in use, see 5.3.3.7.5 – the authentication key. The xDLMS InitiateResponse APDU, carried by the user-information field of the AARE APDU shall be also

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

encrypted and authenticated the same way. When the dedicated key is used, the key-set bit of the security control byte, see Table 27 – is not relevant and shall be set to zero.

NOTE The AARQ and the AARE APDUs themselves are not protected.

Table 10 summarizes the symmetric key types, their purpose, the methods to establish them and their use with the different APDUs and between different entities.

**Table 10 – Symmetric keys types**

| Key type | Purpose | Key establishment | Use |
|---|---|---|---|
| Master key, KEK [1)] | Key Encrypting Key (KEK) for:<br>– (new) master key;<br>– global encryption or authentication keys;<br>– ephemeral encryption keys. | Out of band<br>Key wrapping<br>Key agreement [3)] | Can be identified as the KEK in general-ciphering APDUs between client-server, see Table 11 |
| Global unicast encryption key, GUEK [2)] | Block cipher key for unicast<br>– xDLMS APDUs and / or<br>– COSEM Data | Key wrapping<br>Key agreement [3)] | – service-specific global ciphering APDU client-server<br>– general-glo-ciphering APDU client-server<br>– general-ciphering APDU client-server<br>– "Data protection" object protection parameters |
| Global broadcast encryption key, GBEK [2)] | Block cipher key for broadcast<br>– xDLMS APDUs and / or<br>– COSEM Data | Key wrapping | |
| (Global) Authentication key, GAK [2)] | Part of AAD to the ciphering process of xDLMS APDUs and / or COSEM data | Key wrapping<br>Key agreement [3)] | All APDUs between client-server and third party-server |
| Dedicated key (unicast) | Block cipher key of unicast xDLMS APDUs, within an established AA | Key-transport in xDLMS Initiate.request APDU | – service-specific dedicated ciphering APDU client-server<br>– general-ded-ciphering APDU client-server<br>during the lifetime of an AA |
| Ephemeral encryption key | Block cipher key for:<br>– xDLMS APDUs and/or<br>– COSEM data | Key wrapping | – general-ciphering APDU client-server<br>– "Data protection" object protection parameters |
| | Block cipher key for:<br>– xDLMS APDUs and/or<br>– COSEM data | Key agreement [4)] | – general-ciphering APDU client-server or third-party server<br>– "Data protection" object protection parameters |

1) Held by a "Security setup" object. Different AAs may use the same or different "Security setup" objects.

2) Held by a "Security setup" object. Different AAs may use the same or different "Security setup" objects. The use of the GUEK or the GBEK can be identified by:
   – the key-set bit of the Security Control Byte, see Table 27; or
   – by the key-id parameter of the general-ciphering APDU, see Table 11;
   – or by the protection parameters of the "Data protection " IC, see IEC 62056-6-2:2021, 4.4.9.

3) Established using the Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme, see 5.3.4.6.2

4) Established using the One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme or the Static Unified Model C(0e, 2s, ECC CDH) scheme. See 5.3.4.6.3 and 5.3.4.6.4.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.5.2 Key information with general-ciphering APDU and data protection

When the general-ciphering APDU is used to protect xDLMS APDUs or when COSEM data is protected, the sender sends the necessary information on the key that has been / shall be used to cipher / decipher the xDLMS APDU / COSEM data, together with the ciphered xDLMS APDU / COSEM data.

The key information required is summarized in Table 11 and further specified in 5.5.3, 5.5.4 and 5.5.5.

**Table 11 – Key information with general-ciphering APDU and data protection**

| Key information choices | | Comment |
|---|---|---|
| key-Info | | |
| **identified-key** | S | The EK is identified |
| key-id | M | |
| global-unicast-encryption-key | S | GUEK |
| global-broadcast-encryption-key | S | GBEK |
| **wrapped-key** | S | The EK is transported using key wrapping |
| kek-id | M | |
| master-key | M | Identifies the key used for wrapping the key-ciphered-data.<br>0 = Master Key (KEK) |
| key-ciphered-data | M | Randomly generated key wrapped with KEK |
| **agreed-key** | S | The key is agreed by the parties using either:<br>– the One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme see 5.3.4.6.3; or<br>– the Static Unified Model C(0e, 2s, ECC CDH) scheme see 5.3.4.6.4. |
| key-parameters | M | Identifier of the key agreement scheme:<br>0x01: C(1e, 1s ECC CDH)<br>0x02: C(0e, 2s ECC CDH)<br>All other reserved. |
| key-ciphered-data | M | – In the case of the C(1e, 1s, ECC CDH) scheme: the public key $Q_{e,\,U}$, of the ephemeral key agreement key pair of party U, signed with the private digital signature key of party U.<br>– In the case of the C(0e, 2s, ECC CDH) scheme: an octet-string of length zero. In this case party U has to provide a nonce, $Nonce_U$. See 5.3.4.6.4 and 5.3.4.6.5. |
| M: Mandatory (part of a SEQUENCE)<br>S: Selectable (part of a CHOICE)<br>For the ASN.1 specification, see Clause 8. | | |
| NOTE Using key identification restricts exchanging protected xDLMS APDUs / COSEM data between a client and a server because the GUEK and the GBEK shall not be disclosed to any party other than the client and the server. | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.5.3 Key identification

The key identified may be the Global Unicast Encryption Key (GUEK) or the Global Broadcast Encryption Key (GBEK). In this case, the key-set bit of the security control byte – see Table 27 – is not relevant and shall be set to zero.

### 5.5.4 Key wrapping

Key wrapping can be used to establish static or ephemeral symmetric keys.

The algorithm is the AES key wrap algorithm specified in 5.3.3.8. The KEK is the master key. Consequently, this method can be used only between parties sharing the master key, i.e. between a client and a server.

The static keys that can be established using key wrapping may be:

- the master key, KEK; and/or
- the global unicast encryption key GUEK; and/or
- the global broadcast encryption key GBEK; and/or
- the (global) authentication key, GAK.

To establish these static keys using key wrap, the key shall be first generated by the client, then it shall be transferred to the server by invoking the *key_transfer* method of the "Security setup" object, see IEC 62056-6-2:2021, 4.4.7. The method invocation parameter shall carry the key_id(s) and the wrapped key(s). The APDU carrying the service that invokes the method and the method invocation parameters shall be protected as required by the security policy and the access rights.

NOTE The required level of protection can be specified in project specific companion specifications.

To establish an ephemeral key using key wrapping, the originator of the xDLMS APDU or the COSEM data randomly generates an ephemeral key. This key shall be wrapped using the AES key wrap algorithm and the KEK and shall be sent to the recipient together with the xDLMS APDU or the COSEM data that have been ciphered using the ephemeral key. The recipient shall unwrap the key then it shall use it to decipher the xDLMS APDU / COSEM data received.

### 5.5.5 Key agreement

Key agreement can be used to establish static keys between a server and a client or ephemeral keys between a server and a client or a third party. Different key agreement schemes are available to establish different keys.

The Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme can be used by the client and the server to agree on the:

- master key, KEK; and/or
- global unicast encryption key GUEK; and/or
- global broadcast encryption key GBEK; and/or
- (global) authentication key, GAK.

This scheme is supported by the *key_agreement* method of the "Security setup" interface class, see IEC 62056-6-2:2021, 4.4.7. The method invocation parameters carry the necessary parameters as specified in 5.3.4.6.2. The APDUs carrying the service that invokes the method as well as the method invocation parameters shall be protected as required by the security policy and the access rights. See also Annex I.

NOTE The required level of protection can be specified in project specific companion specifications.

To establish an ephemeral encryption key – used as the block cipher key – using key agreement, two schemes are available:

- the One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme specified in 5.3.4.6.3;

  Unless specified otherwise in a project specific companion specification, the C(1e, 1s ECC CDH) scheme shall be used.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- the Static Unified Model C(0e, 2s, ECC CDH) scheme specified in 5.3.4.6.4.

### 5.5.6 Symmetric key cryptoperiods

Symmetric key cryptoperiods should be determined in project specific companion specifications. Recommendations are given in NIST SP 800-57:2012, Part 1, 5.3.5 *Symmetric Key Usage Periods and Cryptoperiods* and 5.3.6 *Cryptoperiod Recommendations for Specific Key Types.*

## 5.6 Keys used with public key algorithms

### 5.6.1 Overview

Asymmetric keys – see Table 12 – are classified according to:

- their purpose: digital signature key or key agreement key;
- by their lifetime: static keys or ephemeral keys.

**Table 12 – Asymmetric keys types and their use**

| Digital signature | | | |
|---|---|---|---|
| **Key type** | **Signatory** | **Verifier** | **Use** |
| Digital signature key pair | Private key | Public key | Signatory uses private key to compute digital signature:<br>– on an xDLMS APDUs; and/ or<br>– on COSEM data; or<br>– on an ephemeral public key agreement key.<br>Verifier uses public key to verify digital signature<br>– on an xDLMS APDUs; and/or<br>– on COSEM data; or<br>– on an ephemeral public key agreement key<br>received. |
| **Key agreement** | | | |
| **Key type** | **Party U** | **Party V** | **Use** |
| Ephemeral key agreement key pair | Private key<br>Public key | Private key<br>Public key | In the case of the Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme both parties have an ephemeral key pair. See 5.3.4.6.2.<br>In the case of the One-Pass Diffie-Hellman C(1e,1s, ECC CDH) scheme only party U has an ephemeral key pair. See 5.3.4.6.3 |
| Static key agreement key pair | Private key<br>Public key | Private key<br>Public key | In the case of the One-Pass Diffie-Hellman C(1e,1s, ECC CDH) scheme only party V has a static key pair. See 5.3.4.6.3.<br>In the case of the Static Unified Model, C(0e, 2s, ECC CDH) scheme both the party U and party V have a static key pair. See 5.3.4.6.4. |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.6.2 Key pair generation

An ECC key pair $d$ and $Q$ is generated for a set of domain parameters $(q, FR, a, b \{, domain_parameter_seed\}, G, n, h)$. Two methods are provided for the generation of the ECC private key $d$ and public key $Q$; one of these two methods shall be used to generate $d$ and $Q$.

Prior to generating ECDSA key pairs, assurance of the validity of the domain parameters $(q, FR, a, b \{, domain_parameter_seed\}, G, n, h)$ shall have been obtained.

For details, see FIPS PUB 186-4:2013, Annex B.4.

### 5.6.3 Public key certificates and infrastructure

#### 5.6.3.1 Overview

This subclause 5.6.3 describes the public key certificates for the purposes of DLMS®/COSEM and an example PKI infrastructure to manage them. It is based on the following documents:

- NIST SP 800-21:2005 and NIST SP 800-32:2001, providing a general description of public key cryptography and public key infrastructures;
- ITU-T X.509:2008, specifying public key and attribute certificate frameworks;
- RFC 5280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile;
- NSA3 specifying the NSA Suite B Base Certificate and CRL Profile.

The trust model is described in 5.6.3.2.

A PKI architecture – as an example – is described in 5.6.3.3.

The certificate and certificate extension profile is specified in 5.6.4.

The public key certificates to be held by DLMS®/COSEM servers are specified in 5.6.5.

The management of certificates is specified in 5.6.6.

#### 5.6.3.2 Trust model

For DLMS®/COSEM based meter data exchange systems using public key cryptography various public-private key pairs and public key certificates should be in place.

A public key certificate binds a public key to an identity: the subject. A certificate is digitally signed by a Certification Authority.

To provide and manage the certificates, some form of Public Key Infrastructure is required. A PKI consists of Certification Authorities issuing certificates and end entities using these certificates. For a PKI example, see 5.6.3.3.

In its simplest form, a certification hierarchy consists of a single CA. However, the hierarchy usually contains multiple CAs that have clearly defined parent-child relationships. It is also possible to deploy multiple hierarchies.

The PKI needs a trust anchor that is used to validate the first certificate in a sequence of certificates. The trust anchor may be a Root-CA certificate, a Sub-CA certificate or a directly trusted key.

NOTE 1 Trust anchors are Certificates or directly trusted keys marked as such. However, this marking is out of the scope of this document.

DLMS®/COSEM servers shall be provisioned with one or more trust anchors during manufacturing using a trusted Out of Band (OOB) process.

NOTE 2 Provisioning clients and third parties with trust anchors is out of the scope of this document.

DLMS®/COSEM servers may also be provisioned with their own certificates and certificates of CAs, DLMS®/COSEM clients and third parties. This may also happen using a trusted OOB process or through the "Security setup" object.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The "Security setup" interface class – see IEC 62056-6-2:2021, 4.4.7 – provides:

- an attribute that provides information on the certificates stored on the server;
- a method to generate server key pairs and a method to generate Certificate Signing Request (CSR) information on the server to be sent by the client to a CA;
- methods to import, export and remove certificates.

These attributes / methods can be used to manage the certificates by the client or by third parties via the client acting as a broker. Messages accessing the attributes and methods of "Security setup" objects and the data included shall be suitably protected.

Certificates generally have a validity period. However, certificates issued to DLMS®/COSEM servers may be indefinitely valid. Certificates may be replaced when they expire.

Before a server uses a Certificate, it has to be verified. Verification includes:

- checking syntactic validity of the certificate;
- checking the attributes included in the certificate;
- checking that the certificate validity period has not expired;
- checking the certification path to the trust anchor;
- checking the signature of the issuer of the certificate.

It is assumed that the trust anchor, other CA-certificates, as well as the certificates of DLMS®/COSEM clients and third parties held by the server are all valid. It is the responsibility of the system to replace / remove any certificates the validity of which have expired or that have been revoked.

Clients and third parties also have to verify server certificates before using them. They may have capabilities to verify the status of certificates they are using. However this is outside the Scope of this International Standard.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.6.3.3 PKI architecture – informative

##### 5.6.3.3.1 General

NOTE 1 The introductory text is quoted from NIST SP 800-21:2005, 3.7.

A PKI is a security infrastructure that creates and manages public key certificates to facilitate the use of public key (i.e., asymmetric key) cryptography. To achieve this goal, a PKI needs to perform two basic tasks:

a) Generate and distribute public key certificates to bind public keys to other information after validating the accuracy of the binding; and

b) Maintain and distribute certificate status information for unexpired certificates.

For DLMS®/COSEM a hierarchical PKI is foreseen, comprising the following components as shown in Figure 23.

NOTE 2 The actual structure of the PKI is left to project specific companion specifications to meet the operators' needs.

- Root Certification Authority (Root-CA): see 5.6.3.3.2;
- Certification Authority / Subordinate Authority (Sub-CA): see 5.6.3.3.3;
- End entities: entities that do not issue certificates: DLMS®/COSEM clients, DLMS®/COSEM servers and third parties: see 5.6.3.3.4.

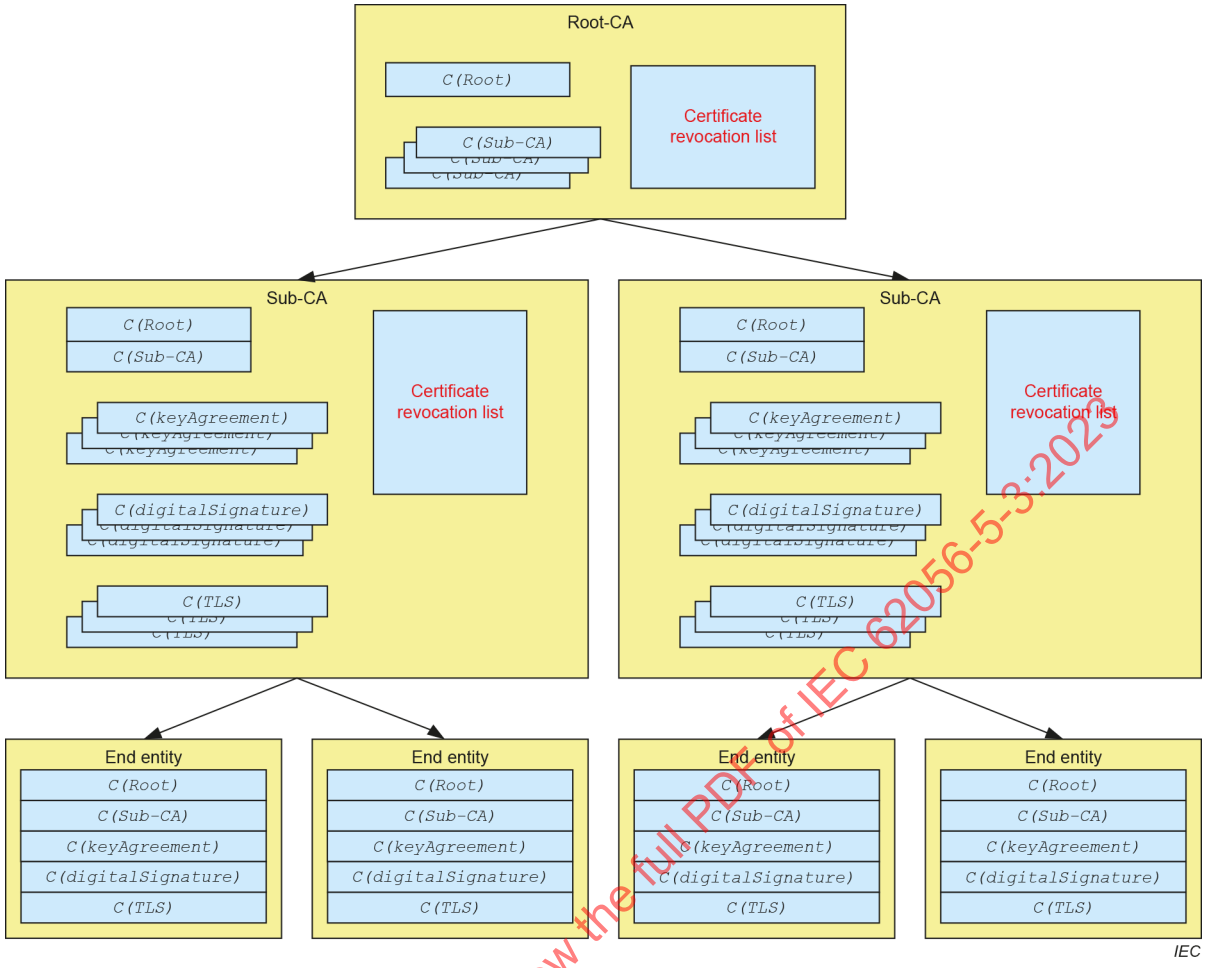



**Figure 23 – Architecture of a Public Key Infrastructure (example)**

#### 5.6.3.3.2 Root-CA

The Root-CA provides the trust anchor of the PKI. It issues certificates for Sub-CAs and maintains a certificate revocation list (CRL). The Root-CA Certificate Policy defines the rules for handling the issuance of certificates.

The Root-CA owns the Root certificate `C(Root)`. The Certificate of the Root-CA is self-signed with the Root-CA private key. Sub-CA certificates are also signed with the Root-CA private key.

#### 5.6.3.3.3 Sub-CA

A Sub-CA is an organization that issues certificates for end entities. Each Sub-CA is authorised by the Root-CA to do so.

NOTE Sub-CAs may be independent organizations, or may be meter market participants, meter operators, manufacturers.

Each Sub-CA shall handle a Certificate Policy, which shall comply with the Root-CA Certificate Policy.

Sub-CAs also maintain the list of certificates issued to end entities and a Certification Revocation List.

A sub-CA owns a certificate *C(Sub-CA)*. This certificate is issued by the Root-CA. The private key of the Sub-CA is used for signing end entity certificates.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.6.3.3.4 End entities

In the PKI infrastructure, an end entity is a user of PKI certificates and/or an end user system that is the subject of a certificate. The following end entity certificates are defined:

- digital signature key certificate *C(digitalSignature)*, used for digital signature;
- static key agreement key certificate *C(keyAgreement)*, used for key agreement;
- [optional] TLS-Certificate *C(TLS)*, used for performing authentication between a DLMS®/COSEM client and a DLMS®/COSEM server prior the establishment of a TLS secure channel.

See also 5.6.5.

### 5.6.4 Certificate and certificate extension profile

#### 5.6.4.1 General

This subclause 5.6.4 specifies the certificate and certification extension profile as required for DLMS®/COSEM systems using public key cryptography.

All certificates shall have the structure specified for X.509 version 3 certificates.

For the tables presenting the fields of the certificate and certificate extensions, the following notation applies:

- m (mandatory): the field shall be filled;
- o (optional): the field is optional;
- x (do not use): the field shall not be used.

Each certificate extension is designated either as critical or non-critical. A certificate-using system shall reject the certificate if it encounters a critical extension it does not recognize or a critical extension that contains information that it cannot process. A non-critical extension may be ignored if it is not recognized, but shall processed if it is recognized.

This document specifies minimum requirements. Project specific companion specifications may specify more strict requirements, for example a field specified in this document as optional may be made mandatory or an extension designated as non-critical may be designated as critical.

#### 5.6.4.2 The X.509 v3 Certificate

An X.509 v3 certificate is a SEQUENCE of three required fields as shown in Table 13.

**Table 13 – X.509 v3 Certificate structure**

| Certificate field | RFC 5280 reference | m/x/o | Value |
|---|---|---|---|
| Certificate | 4.1.1 | | |
| tbsCertificate | 4.1.1.1 | m | See below |
| signatureAlgorithm | 4.1.1.2 | m | See below |
| signatureValue | 4.1.1.3 | m | See below |

The tbsCertificate field contains the names of the subject and issuer, a public key associated with the subject, a validity period, and other associated information. The fields of the tbsCertificate are summarized in Table 14. The tbsCertificate usually includes extensions; these are described in 5.6.4.4.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The signatureAlgorithm field contains the identifier for the cryptographic algorithm used by the CA to sign this certificate.

```
AlgorithmIdentifier::= SEQUENCE {
            algorithm               OBJECT IDENTIFIER
            parameters              ANY DEFINED BY algorithm OPTIONAL }
```

The two algorithm identifiers used in DLMS®/COSEM are:

- ecdsa-with-SHA256, OID 1.2.840.10045.4.3.2 in the case of security suite 1;
- ecdsa-with-SHA384, OID 1.2.840.10045.4.3.3 in the case of security suite 2;

The signatureValue contains a digital signature computed upon the ASN.1 DER encoded tbsCertificate. The ASN.1 DER encoded tbsCertificate is used as the input to the signature function.

By generating this signature, a CA certifies the validity of the information in the tbsCertificate field. In particular, the CA certifies the binding between the public key material and the subject of the certificate.

### 5.6.4.3 tbsCertificate

#### 5.6.4.3.1 Overview

The fields of the tbsCertificate are shown in Table 14.

**Table 14 – X.509 v3 tbsCertificate fields**

| Certificate field | RFC 5280 reference | Clause | m/x/o | Comment |
|---|---|---|---|---|
| tbsCertificate | 4.1.2 | 5.6.4.2 | | |
| Version | 4.1.2.1 | – | m | 'v3' (value is 2) |
| Serial Number | 4.1.2.2 | 5.6.4.3.2 | m | Certificate serial number assigned by the CA (not longer than 20 octets) |
| Signature | 4.1.2.3 | – | m | Same algorithm identifier as the signatureAlgorithm in the Certificate |
| Issuer | 4.1.2.4 | 5.6.4.3.3 | m | Distinguished name (DN) of the certificate issuer. |
| Validity | 4.1.2.5 | 5.6.4.3.4 | m | Validity of the certificate. |
| Subject | 4.1.2.6 | 5.6.4.3.3 | m | Distinguished name (DN) of the certificate subject. |
| Subject Public Key Info | 4.1.2.7 | 5.6.4.3.5 | m | |
| Issuer Unique ID | 4.1.2.8 | | x | Not applicable |
| Subject Unique ID | 4.1.2.8 | 5.6.4.3.6 | o | |
| Extensions | 4.1.2.9 | 5.6.4.4 | m | |

#### 5.6.4.3.2 Serial number

As specified in RFC 5280:2008, 4.1.2.2 the serial number shall be a positive integer assigned by the CA to each certificate. It shall be unique for each certificate issued by a given CA (i.e., the issuer name and serial number identify a unique certificate).

Certificate users shall be able to handle serialNumber values up to 20 octets. Conforming CAs shall not use serialNumber values longer than 20 octets.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.6.4.3.3 Issuer and Subject

The `Issuer` field identifies the entity that has signed and issued the certificate.

The `Subject` field identifies the entity associated with the public key stored in the subject public key field. The subject name MAY be carried in the `Subject` field and/or the `subjectAltName` extension. If subject naming information is present only in the `subjectAltName` extension then the subject name shall be an empty sequence and the `subjectAltName` extension shall be critical. See 5.6.4.4.6.

The naming scheme of the various entities of the PKI is shown in the following tables. The names shall be inserted in the `Issuer` or the `Subject` field of the `tbsCertificate` as applicable. See Table 15, Table 16 and Table 17.

**Table 15 – Naming scheme for the Root-CA instance (informative)**

| Attribute | Abbreviation | m/x/o | Name | Comment |
|---|---|---|---|---|
| `Common Name` | CN | m | <Root-CA> | Name of Root-CA |
| `Organization` | O | o | <PKI-Name> | Name of PKI |
| `Organizational Unit` | OU | o | | Name of organizational unit |
| `Country` | C | o | | ISO 3166 country code |
| NOTE Values within the less-than – greater-than signs "< >" are to be assigned by the PKI or the CA as applicable. | | | | |

**Table 16 – Naming scheme for the Sub-CA instance (informative)**

| Attribute | Abbrev. | m/x/o | Name | Comment |
|---|---|---|---|---|
| `Common Name` | CN | m | <XXX-CA> | Name of sub-CA<br>The CN shall finish with "CA" so that the CA function is recognized. |
| `Organization` | O | o | <PKI-Name> | Name of PKI |
| `Organizational Unit` | OU | o | | Name of organizational unit |
| `Country` | C | o | | ISO 3166 country code |
| `Locality` | L | o | <Locality> | Locality where the Sub-CA is located |
| `State` | ST | o | <State> | |
| NOTE Values within the less-than – greater-than signs "< >" are to be assigned by the PKI or the CA as applicable. | | | | |

The format of the elements of the naming scheme of Root-CA and Sub-CA instances is left to project specific companion specifications.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 17 – Naming scheme for the end entity instance**

| Attribute | Abbrev. | m/x/o | Name | Comment |
|---|---|---|---|---|
| Common Name | CN | m | <System-title> | DLMS®/COSEM System title: 8 bytes represented as a 16 characters:<br>Example: "4D4D4D0000BC614E" |
| Organization | O | o | <PKI-Name> | Name of PKI |
| Organizational Unit | OU | o | | Name of organizational unit |
| Country | C | o | | ISO 3166 country code |
| Locality | L | x | <Locality> | Locality where the entity is located |
| State | S | x | <State> | |
| NOTE Values within the less-than – greater-than signs "< >" are to be assigned by the PKI or the CA as applicable. | | | | |

#### 5.6.4.3.4 Validity period

The certificate validity period is the time interval during which the CA warrants that it will maintain information about the status of the certificate. The field is represented as a SEQUENCE of two dates:

- the date on which the certificate validity period begins (`notBefore`); and
- the date on which the certificate validity period ends (`notAfter`).

In the case of CA certificates, Sub-CA certificates and end-entities other than DLMS®/COSEM servers `notBefore` and `notAfter` shall be well defined dates.

DLMS®/COSEM servers may be given certificates for which no good expiration date can be assigned; such a certificate is intended to be used for the entire lifetime of the device.

To indicate that a certificate has no well-defined expiration date, the `notAfter` should be assigned the GeneralizedTime value of 99991231235959Z.

For details, see RFC 5280:2008, 4.1.2.5.

#### 5.6.4.3.5 SubjectPublicKeyInfo

The field `SubjectPublicKeyInfo` shall have the following structure:

```
SubjectPublicKeyInfo::= SEQUENCE {
      Algorithm                                AlgorithmIdentifier,
      subjectPublicKey                    BIT STRING }
```

An algorithm identifier is defined by the following ASN.1 structure:

```
AlgorithmIdentifier::= SEQUENCE {
      algorithm                                OBJECT IDENTIFIER,
      parameters                          ANY DEFINED BY algorithm OPTIONAL }
```

The algorithm identifier is used to identify a cryptographic algorithm. The OBJECT IDENTIFIER component identifies the algorithm (such as DSA with SHA-1). The contents of the optional parameters field will vary according to the algorithm identified. This field shall contain the same algorithm identifier as the signature field in the sequence `tbsCertificate`; see 5.6.4.2.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The algorithm `OBJECT IDENTIFIER` shall contain the value:

- OID value: 1.2.840.10045.2.1;
- OID description: ECDSA and ECDH Public Key.

The value of the `parameter` shall be 1.2.840.10045.3.1.7 for the curve NIST P-256 and 1.3.132.0.34 for the curve NIST P-384.

The subjectPublicKey from SubjectPublicKeyInfo is the ECC public key. ECC public keys have the following syntax:

ECPoint::= OCTET STRING

Implementations of Elliptic Curve Cryptography according to this document shall support the uncompressed form and may support the compressed form of the ECC public key. The hybrid form of the ECC public key from [X9.62] shall not be used.

As specified in SEC1:2009:

- The elliptic curve public key (a value of type ECPoint that is an OCTET STRING) is mapped to a subjectPublicKey (a value of type BIT STRING) as follows: the most significant bit of the OCTET STRING value becomes the most significant bit of the BIT STRING value, and so on; the least significant bit of the OCTET STRING becomes the least significant bit of the BIT STRING. Conversion routines are found in SEC1:2009, 2.3.1 and 2.3.2;
- The first octet of the OCTET STRING indicates whether the key is compressed or uncompressed. The uncompressed form is indicated by 0x04 and the compressed form is indicated by either 0x02 or 0x03 (see 2.3.3 in SEC1:2009. The public key shall be rejected if any other value is included in the first octet.

#### 5.6.4.3.6 Subject Unique ID

Subject unique IDs may be optionally used in end device certificates other than server certificates. The use of this field is left to project specific companion specifications.

### 5.6.4.4 Certificate extensions

#### 5.6.4.4.1 Overview

The extensions defined for X.509 v3 certificates provide methods for associating additional attributes with users or public keys and for managing relationships between CAs. Each extension in a certificate is designated as either critical (TRUE) or non-critical (FALSE).

The extension fields have to be completed according to the type of Certificate used, as specified in Table 18.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 18 – X.509 v3 Certificate extensions**

| | Attributes | RFC 5280:2008 reference | Clause | CAs | | End entities | | |
|---|---|---|---|---|---|---|---|---|
| | | | | *C(Root)* | *C (Sub-CA)* | *C(TLS)* | *C (Key Agree)* | *C (Data Sign)* |
| 1 | AuthorityKeyIdentifier | 4.2.1.1 | 5.6.4.4.2 | o | m | m | m | m |
| 2 | SubjectkeyIdentifier | 4.2.1.2 | 5.6.4.4.3 | m | m | o | o | o |
| 3 | KeyUsage | 4.2.1.3 | 5.6.4.4.4 | m | m | m | m | m |
| 4 | CertificatePolicies | 4.2.1.4 | 5.6.4.4.5 | o | m | m | o | o |
| 5 | SubjectAltNames | 4.2.1.6 | 5.6.4.4.6 | o | o | o | o | o |
| 6 | IssuerAltNames | 4.2.1.7 | 5.6.4.4.7 | o | o | x | x | x |
| 7 | BasicConstraints | 4.2.1.9 | 5.6.4.4.8 | m | m | x | x | x |
| 8 | ExtendedKeyUsage | 4.2.1.12 | 5.6.4.4.9 | x | x | m | x | x |
| 9 | cRLDistributionPoints | 4.2.1.13 | 5.6.4.4.10 | o | o | x | x | x |
| C(Root): Certificate of the Root CA<br>C(Sub-CA): Certificate of a Sub-CA<br>C(TLS): Certificate for Transport Layer Security<br>C(KeyAgree): Certificate an ECDH capable key establishment key<br>C(DataSign): Certificate of an ECDSA capable signing key | | | | | | | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.6.4.4.2 Authority Key Identifier

- Extension-ID (OID): 2.5.29.35;
- Critical: FALSE;
- Description: the AuthorityKeyIdentifier extension provides a means of identifying the public key corresponding to the private key used to sign a certificate;
- Value: the `AuthorityKeyIdentifier` extension shall include the `keyIdentifier` field.

The value of the `keyIdentifier` field needs to be computed either:

- with the method 1 defined in RFC 5280:2008, 4.2.1.2 i.e. the `keyIdentifier` is composed of the 160-bit SHA-1 hash of the value of the BIT STRING `SubjectPublicKey` (excluding the tag, length, and number of unused bits); or
- with the method 2 defined in RFC 5280:2008, 4.2.1.2 i.e. the keyIdentifier is composed of a four-bit type field with the value 0100 followed by the least significant 60 bits of the SHA-1 hash of the value of the BIT STRING `subjectPublicKey` (excluding the tag, length, and number of unused bits).

NOTE The choice of the method is left to project specific companion specifications.

#### 5.6.4.4.3 SubjectKeyIdentifier

- Extension-ID (OID): 2.5.29.14;
- Critical: FALSE;
- Description: the `SubjectKeyIdentifier` extension provides a means of identifying certificates that contain a particular public key;
- Value: the `SubjectKeyIdentifier` extension shall include the `keyIdentifier` field.

For the method of calculating the `keyIdentifier` see 5.6.4.4.2.

#### 5.6.4.4.4 KeyUsage

- Extension-ID (OID): 2.5.29.15;
- Critical: TRUE;
- Description: the KeyUsage extension defines the purpose of the key contained in the certificate;
- Value: The bits that shall be set are shown in Table 19.

**Table 19 – Key Usage extensions**

| Certificate | *C(Root)* | *C(Sub-CA)* | *C(TLS)* | *C (KeyAgree)* | *C (DataSign)* |
|---|---|---|---|---|---|
| Bits to be set | keyCertSign, cRLSign | keyCertSign, cRLSign | digitalSignature keyAgreement | keyAgreement | digitalSignature |

For details, see RFC 5280:2008, 4.2.1.3 and NSA3.

#### 5.6.4.4.5 CertificatePolicies

- Extension-ID (OID): 2.5.29.32;
- Critical: FALSE;
- Description: the certificate policies extension contains a sequence of one or more policy information terms, each of which consists of an object identifier (OID) and optional qualifier. For details, see RFC 5280, 4.2.1.4;
- Value: contains the OID for the applicable certificate policy.

#### 5.6.4.4.6 SubjectAltNames

- Extension-ID (OID): 2.5.29.17;
- Critical: TRUE if the "subject" field of the certificate is empty (an empty sequence), else FALSE.

Description: this extension allows identities to be bound to the subject of the certificate. These identities may be included in addition to or in place of the identity in the subject field of the certificate. If the subject name is an empty sequence, then the `subjectAltName` extension shall be added in the End Entity Signature and Key Establishment Certificates and shall be marked as critical. The `subjectAltName` extension is OPTIONAL otherwise and if included, shall be marked as critical.

The `SubjectAltName` extension when used shall contain a single GeneralName of type OtherName that is further sub-typed as a HardwareModuleName (id-on-HardwareModuleName) as defined in RFC 4108. The hwSerialNum field shall be set to the system title.

- Value: See Table 20.

**Table 20 – Subject Alternative Name values**

| Certificate | *C(Root)* | *C(Sub-CA)* | *C(TLS)* | *C (KeyAgree)* | *C (DataSign)* |
|---|---|---|---|---|---|
| rfc822Name | `<E-Mail Address>` | `<E-Mail Address>` | – | – | – |
| uRI | `<Web site>` | `<Web site>` | – | – | – |
| otherName | – | – | – | `<otherName>` | `<otherName>` |
| NOTE Values within the less-than – greater-than signs "< >" are to be assigned by the PKI or the CA as applicable. | | | | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.6.4.4.7 IssuerAltName

- Extension-ID (OID): 2.5.29.18;
- Critical: FALSE;
- Description: this extension is used to associate Internet style identities with the certificate issuer. For details see RFC 5280:2008, 4.2.1.7;
- Value: See Table 21.

**Table 21 – Issuer Alternative Name values**

| Certificate | *C(Root)* | *C(Sub-CA)* | *C(TLS)* | *C (KeyAgree)* | *C (dataSign)* |
|---|---|---|---|---|---|
| rfc822Name | <E-Mail Address> | <E-Mail Address> | – | – | – |
| URI | <Web site> | <Web site> | – | – | – |
| NOTE Values within the less-than – greater-than signs "< >" are to be assigned by the PKI or the CA as applicable. | | | | | |

#### 5.6.4.4.8 Basic constraints

- Extension-ID (OID): 2.5.29.19;
- Critical: TRUE;
- Description: the basic constraints extension identifies whether the subject of the certificate is a CA and the maximum depth of valid certification paths that include this certificate. See RFC 5280:2008, 4.2.1.9;
- Value: See Table 22.

**Table 22 – Basic constraints extension values**

| Certificate | *C(Root)* | *C(Sub-CA)* | *C(TLS)* | *C (KeyAgree)* | *C (dataSign)* |
|---|---|---|---|---|---|
| cA | TRUE | TRUE | – | – | – |
| pathLenConstraint | See Note 1 | See Note 1 | – | – | – |
| NOTE The value of the –optional – pathLenConstraint depends on the structure of the PKI. | | | | | |

#### 5.6.4.4.9 Extended Key Usage

- Extension-ID (OID): 2.5.29.37;
- Critical: FALSE;
- Description: Indicates that a certificate can be used as an TLS server certificate;
  - TLS server authentication OID: 1.3.6.1.5.5.7.3.1;
  - TLS client authentication OID: 1.3.6.1.5.5.7.3.2.

#### 5.6.4.4.10 cRLDistributionPoints

- Extension-ID (OID): 2.5.29.31;
- Critical: FALSE;
- Description: The CRL distribution points extension identifies how CRL information is obtained;
- This extension is not used in DLMS®/COSEM server certificates.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.6.4.4.11 Other extensions

All other extensions not described in this profile should be considered OPTIONAL; their inclusion or exclusion and their values will depend upon the particular application or PKI profile.

### 5.6.5 Suite B end entity certificate types to be supported by DLMS®/COSEM servers

Every DLMS®/COSEM server shall use X.509 v3 format and contain either:

- a P-256 or P-384 ECDSA-capable signing key; or
- a P-256 or P-384 ECDH-capable key agreement key.

Every certificate shall be signed using ECDSA. The signing CA's key shall be P-256 or P-384 if the certificate contains a key on P-256. The signing CA's key shall be P-384 if the certificate contains a key on P-384.

Depending on the security policy, the following X.509 v3 certificates shall be handled by DLMS®/COSEM end entities, see Table 23.

**Table 23 – Certificates handled by DLMS®/COSEM end entities**

| Security suite 1 | Security suite 2 | Role |
|---|---|---|
| Root Certification Authority (Root-CA) Self-Signed Certificate using P-256 signed with P-256 | Root Certification Authority (Root-CA) Self-Signed Certificate using P-384 signed with P-384 | Trust anchor; there may be more than one. |
| Subordinate CA Certificate (Sub-CA) using P-256 signed with P-256 | Subordinate CA Certificate (Sub-CA) using P-384 signed with P-384 | Certificate of an issuing CA.<br>Subordinate CAs may be also used as trust anchors. |
| – | Subordinate CA Certificate (Sub-CA) using P-256 signed with P-384 | |
| End-Entity Signature Certificate using P-256 signed with P-256 | End-Entity Signature Certificate using P-384 signed with P-384 | Public key for ECDSA signature generation and verification |
| – | End-Entity Signature Certificate using P-256 signed with P-384 | |
| End-Entity Key Establishment Certificate using P-256 signed with P-256 | End-Entity Key Establishment Certificate using P-384 signed with P-384 | Used with the One-Pass Diffie-Hellman C(1e, 1s) scheme or with the Static Unified Model C(0e, 2s, ECC CDH) scheme |
| – | End-Entity Key Establishment Certificate using P-256 signed with P-384 | |
| End-Entity TLS Certificate using P-256 signed with P-256 | End-Entity TLS Certificate using P-384 signed with P-384 | |
| – | End-Entity TLS Certificate using P-256 signed with P-384 | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

An example Certificate is given in Annex H.

### 5.6.6 Management of certificates

#### 5.6.6.1 Overview

This subclause 5.6.6 applies only to the management of public key certificates in DLMS®/COSEM servers, including:

- provisioning the server with trust anchors, see 5.6.6.2;
- provisioning the server with further CA certificates, see 5.6.6.3;
- security personalisation of the server, see 5.6.6.4;
- provisioning servers with certificates of clients and third parties, see 5.6.6.5;

- provisioning clients and third parties with certificates of servers, see 5.6.6.6;
- removing certificates, see 5.6.6.7.

NOTE Management of public key certificates in DLMS®/COSEM clients and in third party systems is out of the Scope of this document.

#### 5.6.6.2 Provisioning servers with trust anchors

Before starting steady state operations, servers shall be provisioned with trust anchors that will be used to validate the certificates. Trust anchors may be Root-CA (i.e. self-signed) certificates, Sub-CA certificates or directly trusted CA keys. A server may be provisioned with more than one trust anchor.

Trust anchors shall be placed in the server out of band (OOB).

Trust anchor certificates are stored together with other certificates.

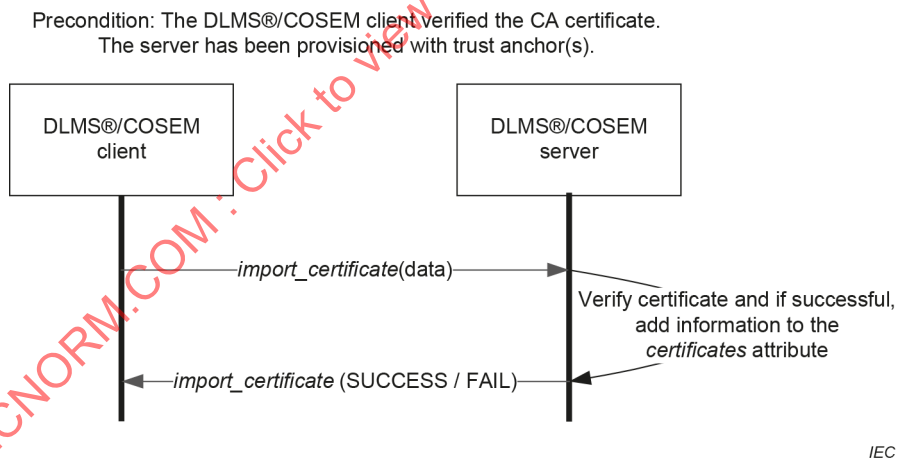
They can be exported, but they cannot be imported or removed.

Directly trusted CA keys cannot be exported.

#### 5.6.6.3 Provisioning the server with further CA certificates

The server may be provisioned with further CA certificates that will be used to verify digital signatures on end device certificates.

For this purpose the *import_certificate* method of the "Security setup" object is available. The process is shown in Figure 24.




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

NOTE When a third party is responsible for managing CA certificates, then the *import_certificate* method may be invoked by that third party via the client acting as a broker.

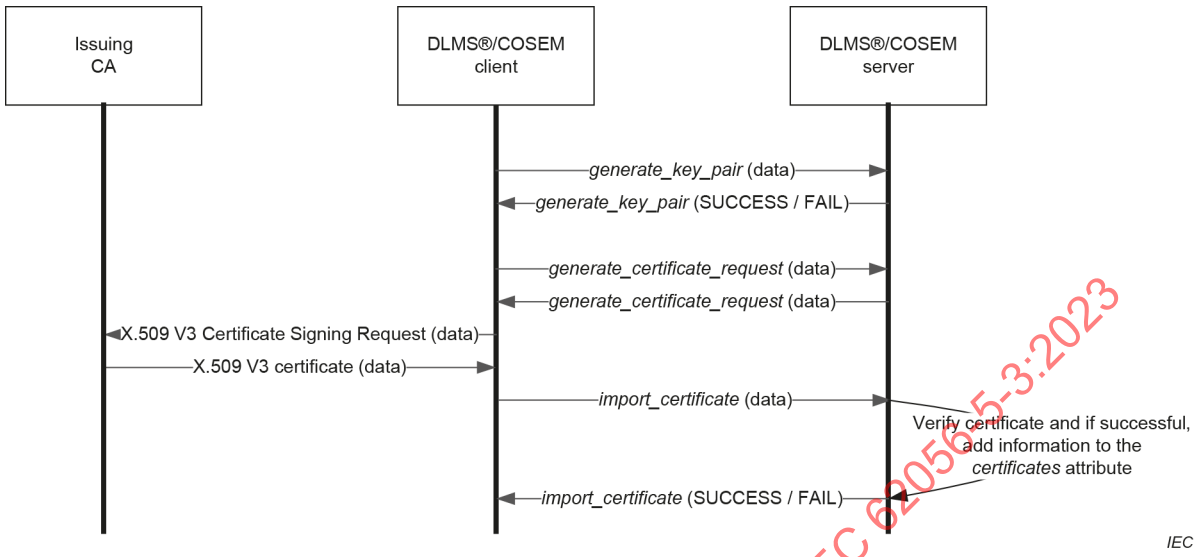
**Figure 24 – MSC for provisioning the server with CA certificates**

#### 5.6.6.4 Security personalisation of the server

Security personalisation means the provision of the server with asymmetric key pairs and the corresponding public key certificates. This can take place either:

- using the security primitives provided by the manufacturer to inject the private key and the public key certificates. The private keys have to be securely stored in the server and shall never be exposed; or

- using the appropriate methods of a "Security setup" object. This process can be used during the manufacturing process and whenever a new key pair and the related public key certificate need to be generated. This process is shown in Figure 25 and described below.




NOTE The security personalisation may be carried out by a third party instead of the client. In that case, the methods of the "Security setup" object may be invoked by that third party via the client acting as a broker.

**Figure 25 – MSC for security personalisation of the server**

- Step 1: the client invokes the *generate_key_pair* method. The method invocation parameters specify the key pair to be generated: digital signature, key agreement or TLS key pair;

  NOTE 1 The new key pair can be used in transactions once its certificate will have been imported and successfully verified.

- Step 2: the client invokes the *generate_certificate_request* method. The method invocation parameters identify the key pair for which the Certificate Signing Request (CSR) will be generated. The return parameters include the X.509 v3 CSR, signed by the private key of the newly generated key pair;
- Step 3: The client sends the X.509 v3 CSR to the CA. This message shall encapsulate the return parameters resulting from the invocation of the *generate_certificate_request* method. The CA – provided that the necessary conditions are met – issues the certificate and sends it to the client;

  NOTE 2 The format of the messages between the client and the issuing CA is out of the scope of this document.

- Step 4: The client invokes the *import_certificate* method. The method invocation parameters contain the certificate. The server verifies the certificate and if successful adds information on the certificate to the *certificates* attribute. If the verification fails, the certificate shall be discarded.

There may be only one key pair and certificate present for the same purpose (digital signature, key agreement, TLS). Therefore when the new certificate is successfully imported the old certificate is removed. From this point, the new key pair can be used for transactions.

For the details of method invocation and return parameters, see IEC 62056-6-2:2021, 4.4.7.

Parties using the server's certificates can obtain these either:

- out of band;
- using the *export_certificate* method of the "Security setup" objects, see 5.6.6.6; or
- as part of the AARE (during HLS authentication), see 5.7.4.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

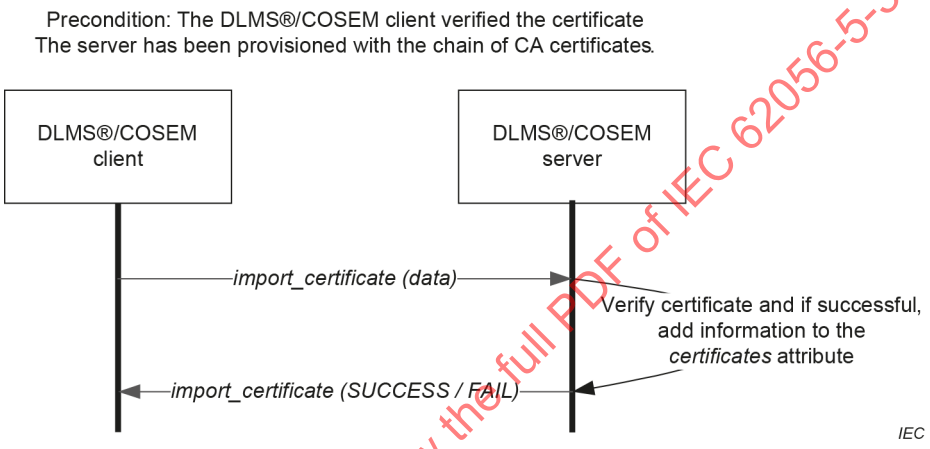
#### 5.6.6.5 Provisioning servers with certificates of clients and third parties

To verify digital signatures, to perform key agreement using a scheme that uses static key agreement keys, or to establish a TLS connection, the server needs to have the appropriate public key certificates of the other party.

If, at the time of manufacturing, the client and/or third parties are already known, their public keys certificates may be injected into the server by the manufacturer.

NOTE Distribution of public key certificates to the manufacturer is out of Scope of this document.

Otherwise, the servers can be provisioned with certificates of clients and third parties using the *import_certificate* method of the "Security setup" object. The method invocation parameter is the certificate to be placed in the server. For the details of method invocation and return parameters, see IEC 62056-6-2:2021, 4.4.7. The process is shown in Figure 26.




NOTE The *import_certificate* (data) method may be also invoked by a third party, using the client as a broker.

**Figure 26 – Provisioning the server with the certificate of the client**

In the case of HLS authentication mechanism using ECDSA, the public key certificate of the clients' digital signature key can be carried by the calling-AE-qualifier field of the AARQ. See 5.7.4.

#### 5.6.6.6 Provisioning clients and third parties with certificates of servers

To verify digital signatures applied by the server, to perform key agreement that involves a static key agreement key, or to establish a TLS connection the client or third party needs to have the appropriate public key certificate of the server.

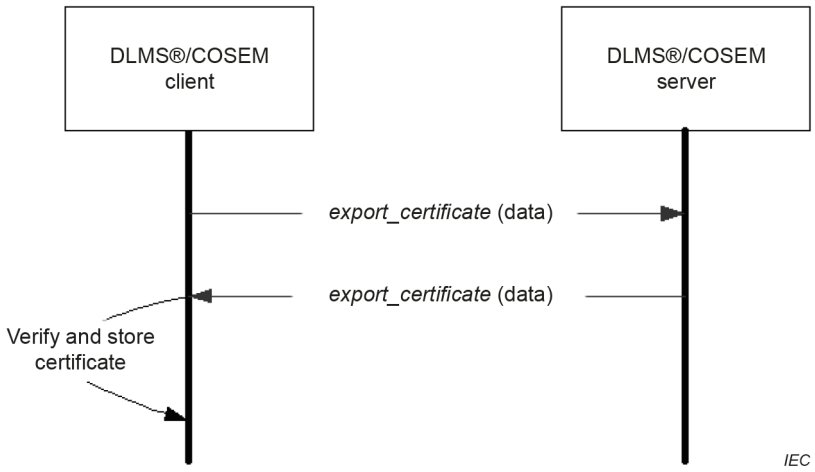
The certificate may be delivered with the server and inserted in clients / third parties OOB.

Alternatively, the client or third party can request the certificate from the server using the *export_certificate* method of the "Security setup" object. The method invocation parameter identifies the certificate requested.

NOTE In the case of HLS authentication using ECDSA – this is authentication_mechanism 7 – the public key certificate of the server for digital signature is transported in the AARE.

The return parameters – in the case of success – include the certificate. For the details of method invocation and return parameters, see IEC 62056-6-2:2021, 4.4.7. The process is shown in Figure 27.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




NOTE The *export_certificate* (data) method may be also invoked by a third party, using the client as a broker.

**Figure 27 – Provisioning the client / third party with a certificate of the server**

### 5.6.6.7 Certificate removal from the server

It is sometimes necessary to remove a public key certificate stored by the server.

NOTE 1 This may relate to certificates that belong to the server or certificates that belong to a client or third party.

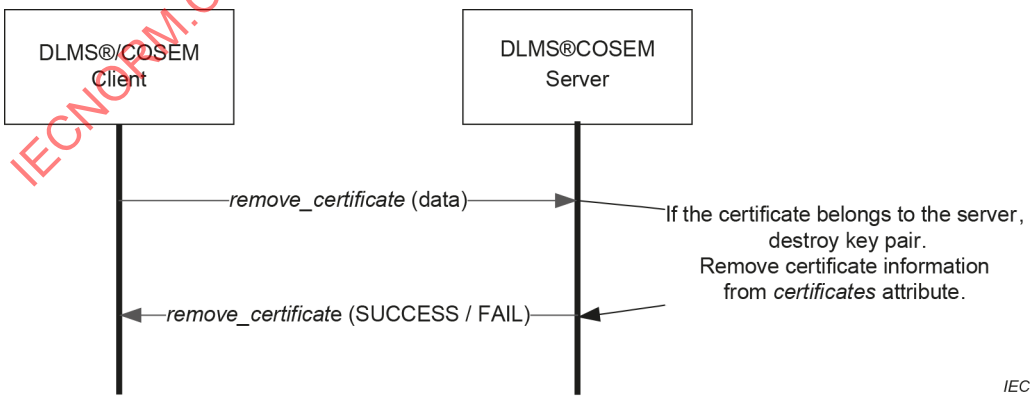
NOTE 2 The conditions when removal of a public key certificate is necessary are out of the scope of this document.

When a certificate that belongs to the server is removed, the private key associated with the public key shall be destroyed.

The information on the certificate removed shall be also removed from the *certificates* attribute of the "Security setup" object.

The key pair the public key certificate of which has been removed cannot be used any more for transactions.

The process is shown in Figure 28.




NOTE The *remove_certificate* (data) method may be also invoked by a third party, using the client as a broker.

**Figure 28 – Remove certificate from the server**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 5.7 Applying cryptographic protection

### 5.7.1 Overview

The cryptographic algorithms specified in 5.3 can be applied:

- to protect the xDLMS APDUs see 5.7.2;
- to process the challenges during HLS authentication, see 5.7.4;
- to protect COSEM data, see 5.7.5.

### 5.7.2 Protecting xDLMS APDUs

#### 5.7.2.1 Overview

This subclause 5.7.2 specifies how the cryptographic algorithms specified in 5.3.3 and 5.3.4 can be used to protect xDLMS APDUs:

- 5.7.2.2 specifies the possible values of security policy and access rights;
- 5.7.2.3 presents the types of ciphered APDUs;
- 5.7.2.4 specifies the use of the AES-GCM algorithm for authentication and encryption;
- 5.7.2.5 specifies the use of the ECDSA algorithm for digital signature.

When a COSEM object attribute or method related xDLMS service primitive is invoked by the AP, the service parameters include the Security_Options parameter. This parameter informs the AL on the kind of ciphered APDU to be used, on the protection to be applied, and includes the necessary security material. The AL builds first the APDU corresponding to the service primitive then it builds the ciphered APDU.

When the AL receives a ciphered APDU from a remote partner, it deciphers it and restores the original, unciphered APDU. When this is successfully done, it invokes the appropriate service primitive. The additional service parameters include the Security_Status and the Protection_Element parameters that inform the AP about the kind of ciphered APDU used, on the protection that has been verified and removed, and may include security material. See also 6.5.

#### 5.7.2.2 Security policy and access rights values

In the case of "Security setup" version 1 – see IEC 62056-6-2:2021, 4.4.7 – the enum type shall be interpreted as an unsigned 8 type. The meaning of each bit is as shown in Table 24.

**Table 24 – Security policy values ("Security setup" version 1)**

| Bit | Security policy |
|---|---|
| 0 | unused, shall be set to 0 |
| 1 | unused, shall be set to 0 |
| 2 | authenticated request |
| 3 | encrypted request |
| 4 | digitally signed request |
| 5 | authenticated response |
| 6 | encrypted response |
| 7 | digitally signed response |
| NOTE In the case of "Security policy" version 0 the possible security policy values are specified in IEC 62056-6-2:2021, 5.3.8. For both "Security policy" version 0 and 1 the value (0) means that no cryptographic protection is required. | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Access rights are held by the *object_list* attribute of "Association LN" or the *access_rights_list* of "Association SN" objects. The access_mode element of the access_rights determines the access kind and stipulates the cryptographic protection. It is an *enum* data type.

In the case of "Association LN" version 3 and "Association SN" version 4 – see IEC 62056-6-2:2021, 4.4.4 and 4.4.3 – the *enum* value is interpreted as an unsigned8: The meaning of each bit is as shown in Table 25.

For older versions, see their specification in IEC 62056-6-2:2021.

**Table 25 – Access rights values ("Association LN" ver 3 "Association SN" ver 4)**

| Bit | Attribute access | Method access |
|---|---|---|
| 0 | read-access | access |
| 1 | write-access | not-used |
| 2 | authenticated request | authenticated request |
| 3 | encrypted request | encrypted request |
| 4 | digitally signed request | digitally signed request |
| 5 | authenticated response | authenticated response |
| 6 | encrypted response | encrypted response |
| 7 | digitally signed response | digitally signed response |
| Examples | enum (3): read-write<br>enum (6) write access with authenticated request<br>enum (255): read-write access with authenticated, encrypted and digitally signed request and response | enum (1): access<br>enum (2): not used<br>enum (5): access with authenticated request<br>enum (253): access with authenticated, encrypted and digitally signed request and response |

Access rights to COSEM object attributes and/or methods may require authenticated, encrypted and / or signed xDLMS APDUs. For this reason, APDUs with more protection than required by the security policy are always allowed. APDUs with less protection than required by the security policy and the access rights shall be rejected.

More protection in this context means that more kinds of protection are applied on the xDLMS APDU than what is requested by the security policy: for example, security policy requires that all APDUs are authenticated but access rights require that the APDU is encrypted and authenticated i.e. a higher protection.

#### 5.7.2.3 Ciphered xDLMS APDUs

The different kind of ciphered xDLMS APDUs are shown in Table 26. See also 6.5. Ciphered xDLMS APDUs can be used in a ciphered application context only. On the other hand, in a ciphered application context, both ciphered and unciphered APDUs may be used.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 26 – Ciphered xDLMS APDUs**

| APDU type | Parties | Type of ciphering | Security services | Key used | Com-pression |
|---|---|---|---|---|---|
| Service-specific glo-ciphering or ded-ciphering | Client – Server | Symmetric key | Authentication<br>Encryption | Block cipher key:<br>– Dedicated key [1]<br>– Global unicast / broadcast key *established* [2] outside the *exchange* [3], *identified* by the SC byte<br>Authentication key: global, *established* [2] outside the exchange [3] | – |
| general-glo-ciphering<br>general-ded-ciphering | | | | | Yes [5] |
| general-ciphering | Third party or Client – Server | Symmetric key | Authentication<br>Encryption | Block cipher key:<br>– Global unicast / broadcast, established [2] outside the exchange [3], identified as part of the exchange, or<br>– Established [2] as part of the exchange [4]<br>Authentication key: global, established [2] outside the exchange [3] | Yes [5] |
| general-signing | | Asymmetric key | Digital signature | Signing key | No |

[1] Transported by the AARQ;

[2] Key establishment may be key wrapping or key agreement;

[3] In the server, these keys are held by the Security setup objects;

[4] Key data is transported in the protected APDU;

[5] The use of compression is controlled by the Security Control byte.

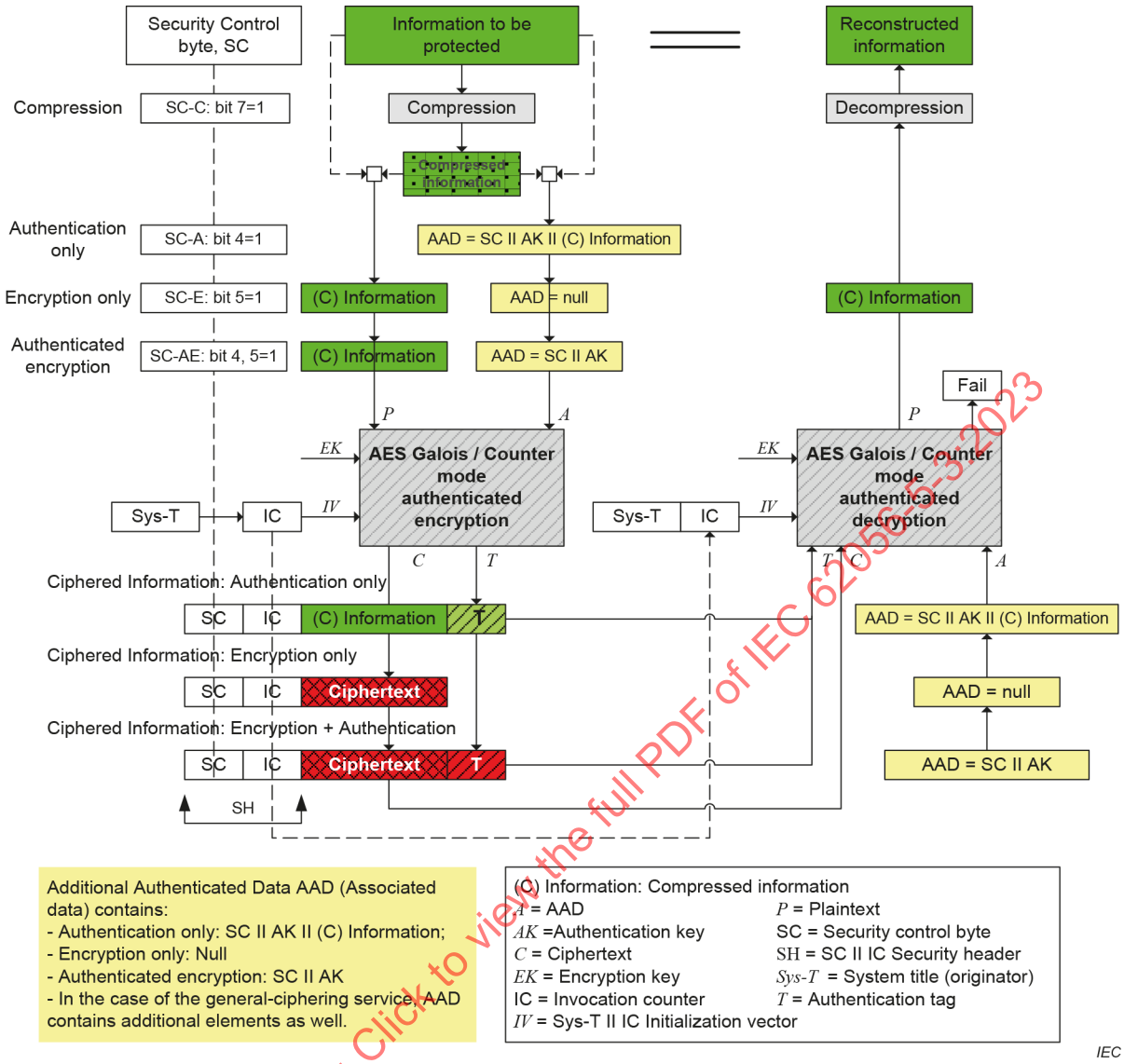
IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.7.2.4 Encryption, authentication and compression

##### 5.7.2.4.1 Overview

Encryption and authentication to protect information using the AES-GCM algorithm is shown in Figure 29. See also 5.3.3.7. This algorithm can be combined with compression.

In the case of message protection, the information to be protected is an xDLMS APDU. In the case of COSEM data protection, the information to be protected is COSEM data, i.e. COSEM attribute value(s) or method invocation / return parameter(s).




NOTE In the case of general-ciphering, AAD also includes additional fields, see Table 28.

**Figure 29 – Cryptographic protection of information using AES-GCM**

The security material required is specified in 5.7.2.4.2 through 5.7.2.4.5.

#### 5.7.2.4.2 The security header

The security header SH includes the security control byte concatenated with the invocation counter: SH = SC II IC. The security control byte is shown in Table 27 where:

- Bit 3…0: Security_Suite_Id, see 5.3.7;
- Bit 4: "A" subfield: indicates that authentication is applied;
- Bit 5: "E" subfield: indicates that encryption is applied;
- Bit 6: Key_Set subfield: 0 = Unicast,
  1 = Broadcast;
- Bit 7: Indicates the use of compression.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 27 – Security control byte**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3..0 |
|---|---|---|---|---|
| Compression | Key_Set | E | A | Security_Suite_Id |
| The Key_Set bit is not relevant and shall be set to 0 when the service-specific dedicated ciphering, the general-ded-ciphering or the general-ciphering APDUs are used. | | | | |

#### 5.7.2.4.3 Plaintext and Additional Authenticated Data

The plaintext denoted $P$ and the Additional Authenticated Data denoted $A$ depend on the kind of protection. They are shown in Table 28, where $SC$ is the security control byte, $AK$ is the authentication key and $I$ is the information, i.e. the xDLMS APDU or the COSEM data to be protected.

**Table 28 – Plaintext and Additional Authenticated Data**

| Security control, *SC* | | Protection | *P* | *A, Additional Authenticated Data* | |
|---|---|---|---|---|---|
| **E field** | **A field** | | | **Service-specific glo-ciphering**<br>**Service-specific ded-ciphering**<br>**general-glo-ciphering**<br>**general-ded-ciphering** | **general-ciphering** |
| 0 | 0 | None | – | – | – |
| 0 | 1 | Authenticated only | – | $SC$ II $AK$ II $(C)$ $I$ | $SC$ II $AK$ II<br>transaction-id II [1]<br>originator-system-title II [1]<br>recipient-system-title II [1]<br>date-time II [1]<br>other-information II [1]<br>$(C)$ $I$ |
| 1 | 0 | Encrypted only | $(C)$ $I$ | – | – |
| 1 | 1 | Encrypted and authenticated | $(C)$ $I$ | $SC$ II $AK$ | $SC$ II $AK$ II<br>transaction-id II [1]<br>originator-system-title II [1]<br>recipient-system-title II [1]<br>date-time II [1]<br>other-information [1] |
| [1] The fields transaction-id ….other-information are A-XDR encoded OCTET STRINGs. The length and the value of each field is included in the AAD. | | | | | |

#### 5.7.2.4.4 Encryption key and authentication key

These keys used by AES-GCM are specified in 5.3.3.7.4 and 5.3.3.7.5 respectively. The various keys used in DLMS®/COSEM and their establishment are specified in 5.5.

#### 5.7.2.4.5 Initialization vector

See 5.3.3.7.3.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.7.2.4.6 Service-specific ciphering xDLMS APDUs

For certain xDLMS APDUs – see 4.2.4.4.6 and 7.3.13 – a ciphered variant using the global key and one using the dedicated key is available. These ciphered APDUs can be used between a client and a server. The structure of the service-specific ciphering APDUs is shown in Figure 30. See also Table 28 and Table 29.

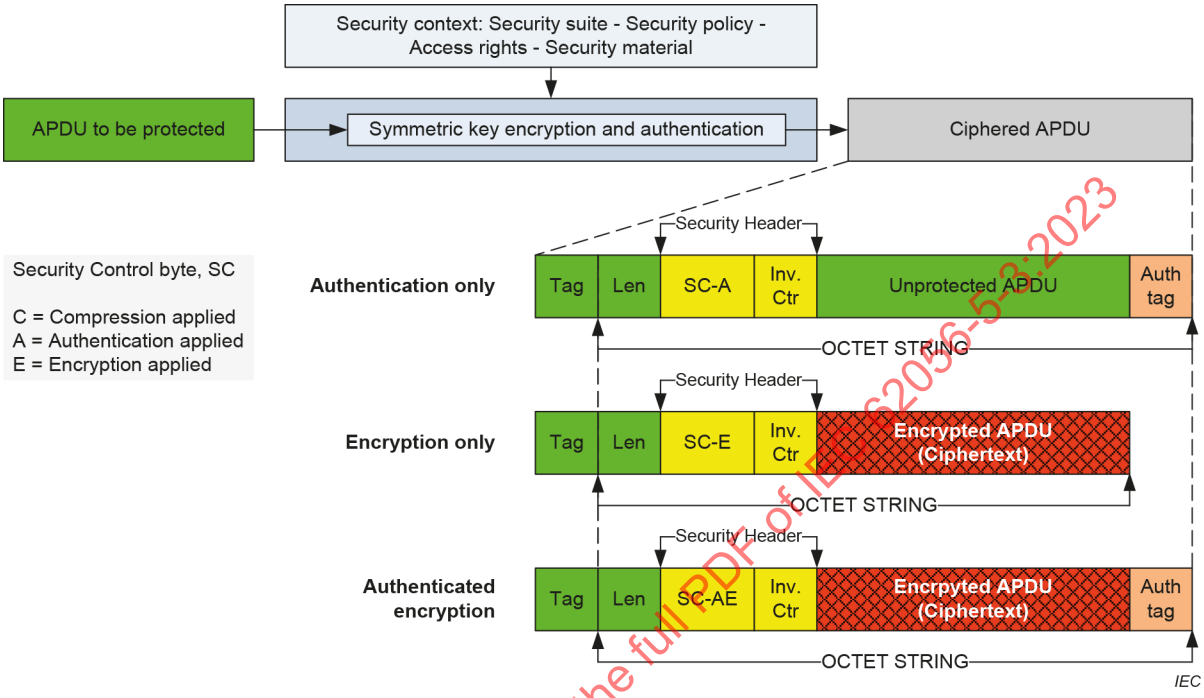



**Figure 30 – Structure of service-specific global / dedicated ciphering xDLMS APDUs**

#### 5.7.2.4.7 The general-glo-ciphering and general-ded-ciphering xDLMS APDUs

These APDUs can be used to cipher other xDLMS APDUs using the global key or the dedicated key. They can be used between a client and a server. Their structure is shown in Figure 31. See also Table 28 and Table 29.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

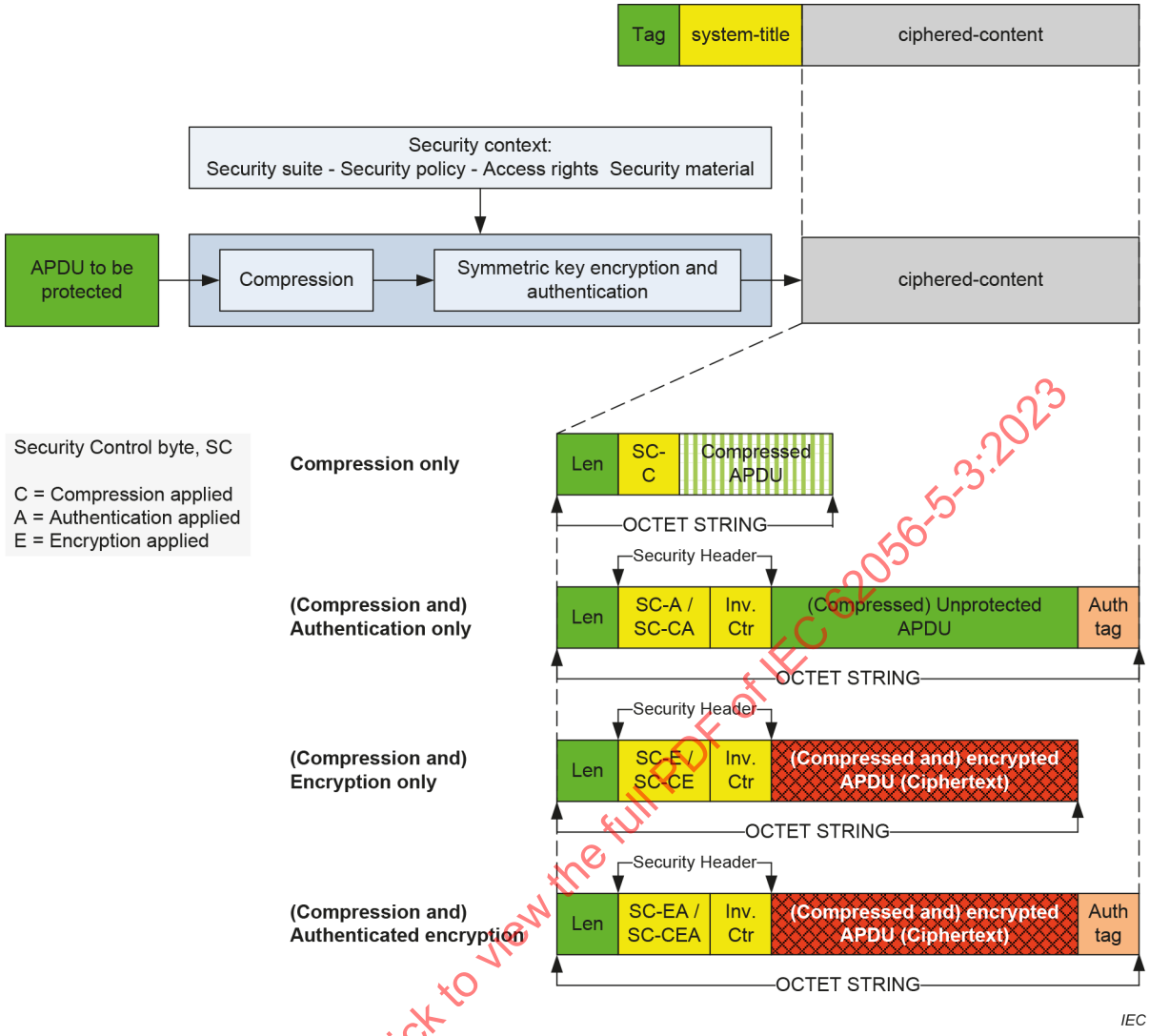



**Figure 31 – Structure of general-glo-ciphering and general-ded-ciphering xDLMS APDUs**

#### 5.7.2.4.8 The general-ciphering APDU

The general-ciphering APDU can be used between a client and a server or between a third party and the server. These APDUs carry also the necessary information on the key to be used. The structure of the general-ciphering APDU is shown in Figure 32. See also Table 28 and Table 29.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

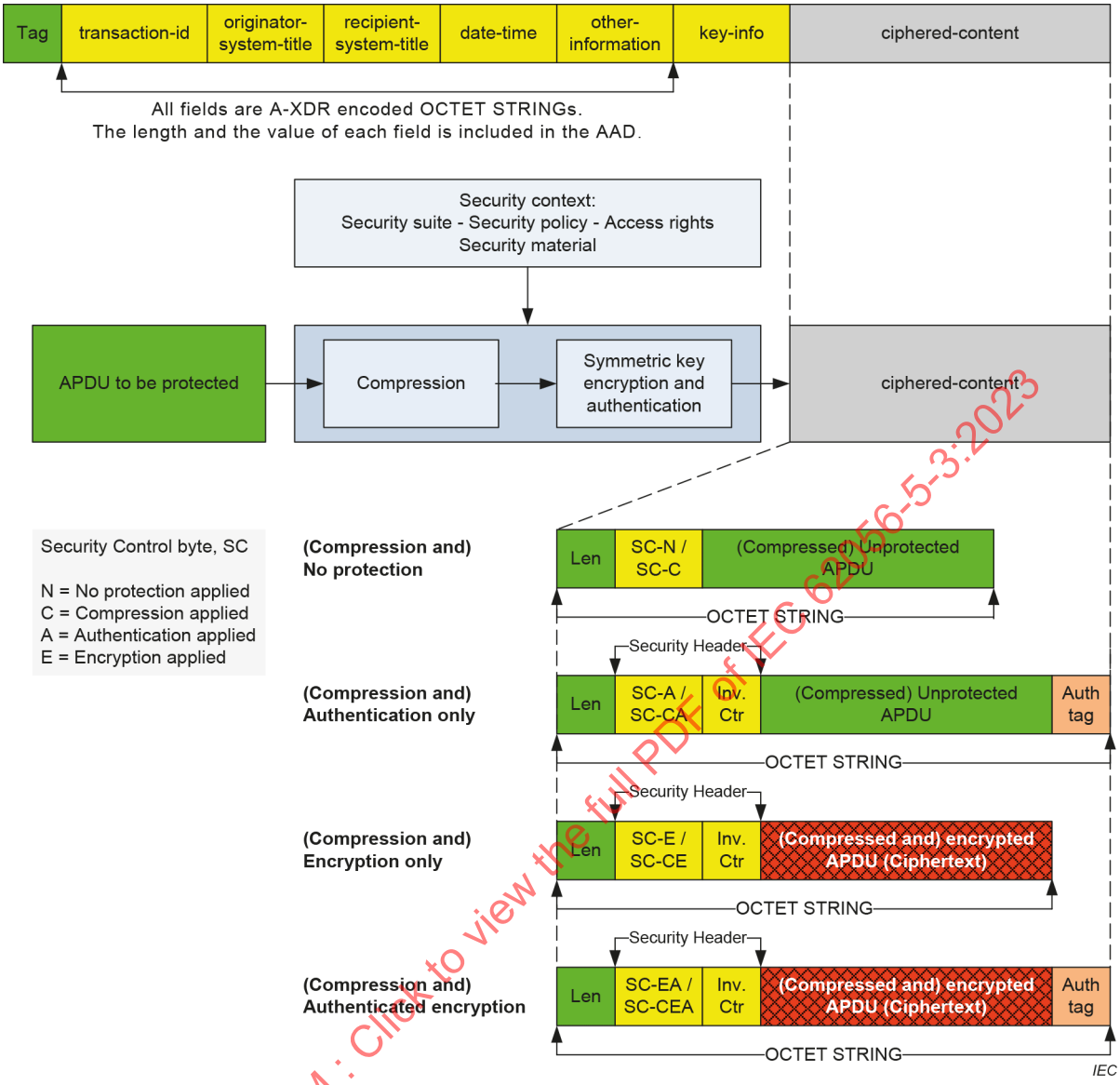



**Figure 32 – Structure of general-ciphering xDLMS APDUs**

#### 5.7.2.4.9 Use of the fields of the ciphering xDLMS APDUs

The use of the fields of the ciphering xDLMS APDUs is summarized in Table 29.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 29 – Use of the fields of the ciphering xDLMS APDUs**

| APDU field | Service-specific global / dedicated ciphering | general-glo-/ general-ded-ciphering | general-ciphering | Meaning |
|---|---|---|---|---|
| tag | Service specific | [219] [220] | [221] | The tag of the ciphering APDU; see 7.3.13 |
| system-title | – | + | – | See 6.5. |
| transaction-id | – | – | + | |
| originator-system-title | – | – | + | |
| recipient-system-title | – | – | + | |
| date-time | – | – | + | |
| other-information | – | – | + | |
| key-info | – | – | + | |
| security control byte | + | + | + | Provides information on the protection applied, the key-set and the security suite used. See Table 27. [1)] |
| Invocation counter | + | + | + | The invocation field of the initialization vector. It is an integer counter which increments upon each invocation of the authenticated encryption function using the same key.<br><br>When a new key is established the related invocation counter shall be reset to 0. |
| unprotected APDU | + | + | + | The unprotected APDU (same as the APDU to be protected). |
| encrypted APDU | + | + | + | The encrypted APDU i.e. the ciphertext. |
| authentication tag | + | + | + | Calculated by the AES-GCM algorithm, see 5.3.3.7. |

[1)] In the case of the general-ciphering APDU, the key-set bit of the security control byte is not relevant and shall be set to zero.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.7.2.4.10 Encoding example: global-get-request xDLMS APDU

Table 30 shows an encoding example of a service-specific global ciphering xDLMS APDU: glo-get-request.

**Table 30 – Example: glo-get-request xDLMS APDU**

| | *X* | Contents | | | *LEN (X)* bytes | *Len (X)* bits |
|---|---|---|---|---|---|---|
| **Security material** | | | | | | |
| Security suite | | GCM-AES-128 | | | | |
| System Title | *Sys-T* | **4D4D4D**0000BC614E(here, the five last octets contain the manufacturing number in hexa) | | | 8 | 64 |
| Invocation counter | *IC* | 01234567 | | | 4 | 32 |
| Initialization Vector | *IV* | *Sys-T* II IC<br>4D4D4D0000BC614E01234567 | | | 12 | 96 |
| Block cipher key (global) | *EK* | 000102030405060708090A0B0C0D0E0F | | | 16 | 128 |
| Authentication Key | *AK* | D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF | | | 16 | 128 |
| **Security applied** | | **Authentication** | **Encryption** | **Authenticated encryption** | | |
| Security control byte (with unicast key) | *SC* | *SC-A*<br>10 | *SC-E*<br>20 | *SC-AE*<br>30 | 1 | 8 |
| Security header | *SH* | *SH = SC-A* II IC<br>1001234567 | *SH = SC-E* II IC<br>2001234567 | *SH = SC-AE* II IC<br>3001234567 | 5 | 40 |
| **Inputs** | | **Authentication** | **Encryption** | **Authenticated encryption** | | |
| xDLMS APDU to be protected | *APDU* | C0010000080000010000FF0200<br>(Get-request, attribute 2 of the Clock object) | | | 13 | 104 |
| Plaintext | *P* | Null | C0010000080000010000FF0200 | C0010000080000010000FF0200 | 13 | 104 |
| Associated data | *A* | *SC* II *AK* II *APDU* | – | *SC* II *AK* | | |
| Associated Data – Authentication | *A-A* | 10D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDFC0010000080000010000FF0200 | – | – | 30 | 240 |
| Associated Data – Encryption | *A-E* | – | – | – | 0 | 0 |
| Associated Data – Authenticated encryption | *A-AE* | – | – | 30D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF | 17 | 136 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| | *X* | Contents | | | *LEN (X)* bytes | *Len (X)* bits |
|---|---|---|---|---|---|---|
| **Outputs** | | **Authentication** | **Encryption** | **Authenticated encryption** | | |
| Ciphertext | *C* | NULL | 411312FF935A47566827C467BC | 411312FF935A47566827C467BC | 13 | 104 |
| Authentication tag | *T* | 06725D910F9221D263877516 | – | 7D825C3BE4A77C3FCC056B6B | 12 | 96 |
| The complete Ciphered APDU | | *TAG* ‖ *LEN* ‖ *SH* ‖ *APDU* ‖ *T* | *TAG* ‖ *LEN* ‖ *SH* ‖ *C* | *TAG* ‖ *LEN* ‖ *SH* ‖ *C* ‖ *T* | – | – |
| Authenticated APDU | | C81E1001234567C001000000800000100 00FF020006725D910F9221D263877516 | – | – | 32 | 256 |
| Encrypted APDU | | – | C8122001234567411312FF935A47566827C467BC | – | 20 | 160 |
| Authenticated and encrypted APDU | | – | – | C81E3001234567411312FF935A47566827C467BC7D825C3BE4A77C3FCC056B6B | 32 | 256 |
| NOTE In this example the value of the invocation counter is 01234567. In the real life, the value shall be incremented with each invocation of the AES-GCM algorithm. | | | | | | |

Table 31 shows an example where the ACCESS.request and ACCESS.response APDUs shown in Table F.10 are protected using authenticated encryption. The general-ciphering APDU specified in 5.7.2.4.8 is used. The encryption key is agreed on using the One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) key agreement scheme, see 5.3.4.6.3. The authentication key is the same as in Table 30.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 31 – ACCESS service with general-ciphering, One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) key agreement scheme**

| Message Elements | Contents | LEN (Bytes) |
|---|---|---|
| **General-Ciphering** | DD | 1 |
| **transaction-id** | | 0 |
| **length** | 08 | 1 |
| **value** | 0102030405060708 | 8 |
| **originator-system-title** | | 0 |
| **length** | 08 | 1 |
| **value** | 4D4D4D0000BC614E | 8 |
| **recipient-system-title** | | 0 |
| **length** | 08 | 1 |
| **value** | 4D4D4D0000000001 | 8 |
| **date-time** | | 0 |
| **length** | 00 | 1 |
| **value** | | 0 |
| **other-information** | | 0 |
| **length** | 00 | 1 |
| **value** | | 0 |
| **key-info OPTIONAL** | 01 | 1 |
| **agreed-key CHOICE** | 02 | 1 |
| **key-parameters** | | 0 |
| **length** | 01 | 1 |
| **value** | 01 | 1 |
| **key-ciphered-data** | | 0 |
| **length** | 8180 | 2 |
| **value** | C323C2BD45711DE4688637D919F92E9D<br>B8FB2DFC213A88D21C9DC8DCBA917D81<br>70511DE1BADB360D50058F794B0960AE<br>11FA28D392CFF907A62D13E3357B1DC0<br>B51BE089D0B682863B2217201E73A1A9<br>031968A9B4121DCBC3281A69739AF874<br>29F5B3AC5471E7B6A04A2C0F2F8A25FD<br>772A317DF97FC5463FEAC248EB8AB8BE | 128 |
| **ciphered-content** | | 0 |
| **length** | 81EB | 2 |
| **value**<br><br>ACCESS.request with authenticated encryption<br><br>SC II IC II ciphertext II auth. Tag | 3100000000F435069679270C5BF4425E<br>E5777402A6C8D51C620EED52DBB18837<br>8B836E2857D5C053E6DDF27FA87409AE<br>F502CD9618AE47017C010224FD109CC0<br>BEB21E742D44AB40CD11908743EC90EC<br>8C40E221D517F72228E1A26E827F43DC<br>18ED27B5F458D66508B05A2A4CC6FED1<br>78C881AFC3BC67064689BE8BB41C80AB<br>B3C114A31F4CB03B8B64C7E0B4CE77B2<br>399C93347858888F92239713B38DF01C<br>4858245827A92EF334172EA636B31CBB<br>DF2A96AD5D035F66AA38F1A2D97D4BBA<br>99622E6B5F18789CECB2DFB3937D9F3E<br>17F8B472098E6563238F375283748098<br>36002AEA6E7012D2ADFAA7 | 235 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Message Elements | Contents | LEN (Bytes) |
|---|---|---|
| **general-ciphering(encoded)** | DD080102030405060708084D4D4D0000<br>BC614E084D4D4D00000000100000102<br>01018180C323C2BD45711DE4688637D9<br>19F92E9DB8FB2DFC213A88D21C9DC8DC<br>BA917D8170511DE1BADB360D50058F79<br>4B0960AE11FA28D392CFF907A62D13E3<br>357B1DC0B51BE089D0B682863B221720<br>1E73A1A9031968A9B4121DCBC3281A69<br>739AF87429F5B3AC5471E7B6A04A2C0F<br>2F8A25FD772A317DF97FC5463FEAC248<br>EB8AB8BE81EB3100000000F435069679<br>270C5BF4425EE5777402A6C8D51C620E<br>ED52DBB188378B836E2857D5C053E6DD<br>F27FA87409AEF502CD9618AE47017C01<br>0224FD109CC0BEB21E742D44AB40CD11<br>908743EC90EC8C40E221D517F72228E1<br>A26E827F43DC18ED27B5F458D66508B0<br>5A2A4CC6FED178C881AFC3BC67064689<br>BE8BB41C80ABB3C114A31F4CB03B8B64<br>C7E0B4CE77B2399C93347858888F9223<br>9713B38DF01C4858245827A92EF33417<br>2EA636B31CBBDF2A96AD5D035F66AA38<br>F1A2D97D4BBA99622E6B5F18789CE0B2<br>DFB3937D9F3E17F8B472098E6563238F<br>37528374809836002AEA6E7012D2ADFA<br>A7 | 401 |
| **General-Ciphering** | DD | 1 |
| **transaction-id** | | 0 |
| **length** | 08 | 1 |
| **value** | 0123456789012345 | 8 |
| **originator-system-title** | | 0 |
| **length** | 08 | 1 |
| **value** | 4D4D4D0000000001 | 8 |
| **recipient-system-title** | | 0 |
| **length** | 08 | 1 |
| **value** | 4D4D4D0000BC614E | 8 |
| **date-time OPTIONAL** | | 0 |
| **length** | 00 | 1 |
| **value** | | 0 |
| **other-information** | | 0 |
| **length** | 00 | 1 |
| **value** | | 0 |
| **key-info OPTIONAL** | 01 | 1 |
| **agreed-key CHOICE** | 02 | 1 |
| **key-parameters** | | 0 |
| **length** | 01 | 1 |
| **value** | 01 | 1 |
| **key-ciphered-data** | | 0 |
| **length** | 8180 | 2 |
| **value** | 6439724714B47CD9CB988897D8424AB9<br>46DCD083D37A954637616011B9C23787<br>73295F0F850D8DAFD1BBE9FE666E53E4<br>F097CD10B38B69622152724A90987444<br>E1FF47974A1F6931A6502F58147463F0<br>E8CC517D47F55B0AC56DD8AC5C9D0E48<br>1934F2D90F9893016BD82B6E3FFE21FF<br>1588F3278B4E9D98EB4FB62ADD64B380 | 128 |
| **ciphered-content** | | 0 |
| **length** | 3D | 1 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Message Elements | Contents | LEN (Bytes) |
|---|---|---|
| **value**<br><br>ACCESS.response with authenticated encryption<br><br>SC II IC II ciphertext II auth. tag | 3100000000B3FFCAA594642D8319CEC6<br>B2A233E2BF4621D6991B97E4565B986E<br>8CCBE9A299D8E7869723638FF6BB20E6<br>6E175E6F2D762CFD26B3D58733 | 61 |
| **general-ciphering (encoded)** | DD08012345678901234508 4D4D4D0000<br>000001084D4D4D0000BC614E00000102<br>0101818064397247 14B47CD9CB988897<br>D8424AB946DCD083D37A954637616011<br>B9C2378773295F0F850D8DAFD1BBE9FE<br>666E53E4F097CD10B38B69622152724A<br>90987444E1FF47974A1F6931A6502F58<br>147463F0E8CC517D47F55B0AC56DD8AC<br>5C9D0E481934F2D90F9893016BD82B6E<br>3FFE21FF1588F3278B4E9D98EB4FB62A<br>DD64B3803D3100000000B3FFCAA59464<br>2D8319CEC6B2A233E2BF4621D6991B97<br>E4565B986E8CCBE9A299D8E786972363<br>8FF6BB20E66E175E6F2D762CFD26B3D5<br>8733 | 226 |

### 5.7.2.5 Digital signature

The algorithm is the elliptic curve digital signature algorithm (ECDSA) as specified in 5.3.4.5.

The structure of the general-signing APDU is shown in Figure 33. For the additional fields, see Table 29 and 6.5.

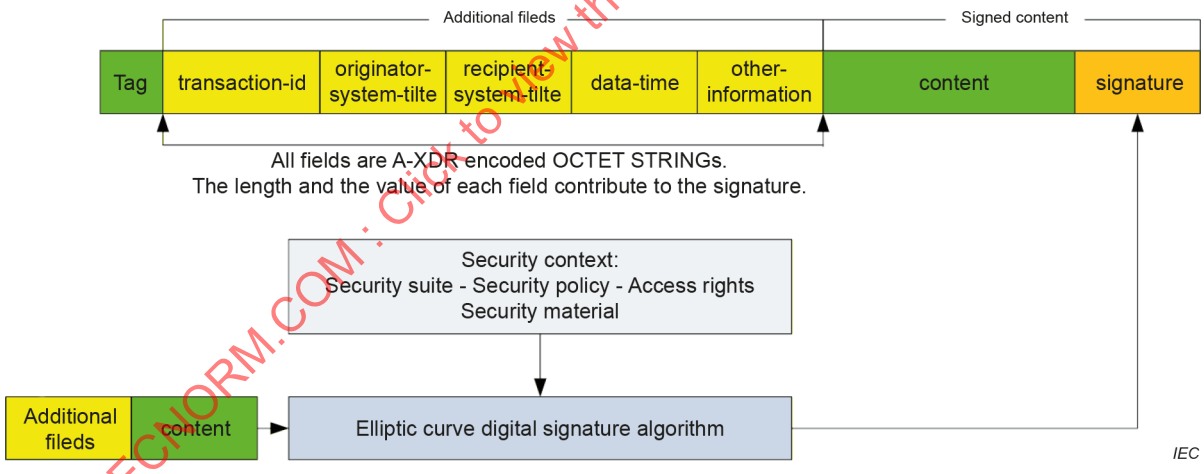



**Figure 33 – Structure of general-signing APDUs**

### 5.7.3 Multi-layer protection by multiple parties

Cryptographic protection can be applied by multiple parties. Generally the parties are:

- a server;
- a client;
- a third party.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Each party can apply one or multiple layers of protection:

- to apply encryption, authentication or authenticated encryption, the ciphering APDUs are used. A third party shall use the general-ciphering APDU. The client can use any of the ciphering APDUs. Authenticated encryption is considered to be a single layer of protection;
- to apply digital signature, the general-signing APDU is used.

If both ciphering and digital signature is applied by the same party for the same party, then normally the digital signature is applied first.

Both the general-ciphering and general-signing APDUs include the Originator_System_Title and the Recipient_System_Title, identifying the party that applied the protection and the party that shall check / remove the protection.

The protection to be applied on the response depends on the security policy and the access rights on the response and on the protection applied on the request. If a kind of protection has been applied on the request by a party, then the same kind of protection will be applied for the same party in the response. However if a kind of protection which was applied on the request is not required on the response, than no protection will be applied on the response for that party.

Example 1 If the request was digitally signed by the third party and authenticated by the client, and the required protection on the response is authentication and digital signature, then the response will be authenticated for the client and digitally signed for the third party.

Example 2 If the request was digitally signed by the third party and authenticated by the client, and the required protection on the response is authentication only, then the response will be authenticated for the client and no protection will be applied to the third party. (The TP will receive a general-ciphering APDU without any protection applied.)

If a protection is required on the response that was not applied on the request, then the server cannot determine from the request which party has the response to be protected for. Therefore it shall apply the protection for all parties.

See also Annex J.

### 5.7.4 HLS authentication mechanisms

HLS authentication requires cryptographic processing of the challenges exchanged by the client and the server. The HLS authentication mechanisms, the information exchanged and the formulae to process the challenges are shown in Table 32.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 32 – DLMS®/COSEM HLS authentication mechanisms**

| **Authentication mechanism** | **Pass 1:** C →S | **Pass 2:** S →C | **Pass 3:** C →S f(StoC) | **Pass 4:** S→C f(CtoS) |
|---|---|---|---|---|
| | **Carried by** | | | |
| | AARQ | AARE | XX.request<br>reply_to_HLS authentication | XX.response<br>reply_to_HLS authentication |
| mechanism_id(2)<br>HLS man. Spec. | *CtoS*: Random string 8-64 octets | *StoC*: Random string 8-64 octets | Man. Spec. | Man. Spec. |
| mechanism_id(3)<br>HLS MD5 [1)] | | | **MD5**(StoC \|\| HLS Secret) | **MD5**(CtoS \|\| HLS Secret) |
| mechanism_id(4)<br>HLS SHA-1 [1] | | | **SHA-1**(StoC \|\| HLS Secret) | **SHA-1**(CtoS \|\| HLS Secret) |
| mechanism_id(5)<br>HLS GMAC | *CtoS*:<br>Random string 8-64 octets<br>Optionally:<br>System-Title-C in calling-AP-title | *StoC*:<br>Random string 8-64 octets<br>Optionally:<br>System-Title-S in responding-AP-title | SC II IC II **GMAC** (SC \|\| AK \|\| StoC) | SC II IC II **GMAC** (SC \|\| AK \|\| CtoS) |
| mechanism_id(6)<br>HLS SHA-256 | | | **SHA-256** (HLS_Secret \|\| SystemTitle-C \|\| SystemTitle-S \|\| StoC II CtoS) | **SHA-256** (HLS_Secret \|\| SystemTitle-S \|\| SystemTitle-C \|\| CtoS \|\| StoC) |
| mechanism_id(7)<br>HLS ECDSA | *CtoS*: Random string 32 to 64 octets<br>Optionally:<br>System-Title-C in calling-AP-title,<br>Cert-Sign-Client in calling-AE-qualifier | *StoC*: Random string 32 to 64 octets<br>Optionally:<br>System-Title-S in responding-AP-title,<br>Cert-Sign-Server responding-AE-qualifier | **ECDSA**( SystemTitle-C \|\| SystemTitle-S \|\| StoC II CtoS) | **ECDSA**( SystemTitle-S \|\| SystemTitle-C \|\| CtoS II StoC) |

Legend:

– C: Client, S: Server, CtoS: Challenge client to server, StoC: Challenge server to client

– IC: Invocation counter

– xx.request / .response: xDLMS service primitives used to access the *reply_to_HLS* authentication method of the "Association SN / LN" object.

[1] The use of authentication mechanisms 3 and 4 are not recommended for new implementations.

NOTE The system titles and the Certificates have to be sent only if not already known by the other party.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Where the system titles and the certificates for the digital signature key are also needed, these may be transported in the AARQ / AARE APDUs, carrying the COSEM-OPEN service .request / .response, see Figure 12. The System_Title and Cert-Sign may be already known; in this case they do not have to be transported. If these elements are not available, the result of the processing of the challenge fails and the AA shall not be established.

Table 33 provides a test vector for HLS authentication-mechanism 5 with GMAC.

**Table 33 – HLS example using authentication-mechanism 5 with GMAC**

| Security material | X | Contents | | LEN(*X*) bytes | len(*X*) bits |
|---|---|---|---|---|---|
| Security suite | | GCM-AES-128 | | | |
| System Title | *Sys-T* | Client | Server | | |
| | | **4D4D4D**0000000001 | **4D4D4D**0000BC614E | | |
| | | (here, the five last octets contain the manufacturing number in hexa) | | 8 | 64 |
| Invocation counter | *IC* | 00000001 | 01234567 | 4 | 32 |
| Initialization Vector | *IV* | *Sys-T* II IC | | 12 | 96 |
| | | Client | Server | | |
| | | 4D4D4D0000000001 00000001 | 4D4D4D0000BC614E 01234567 | | |
| Block cipher key (global) | *EK* | 000102030405060708090A0B0C0D0E0F | | 16 | 128 |
| Authentication Key | *AK* | D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF | | 16 | 128 |
| Security control byte | SC | 10 | | 1 | 8 |
| | | **Pass 1: Client sends challenge to server** | | | |
| CtoS | | 4B35366956616759 "K56iVagY" | | 8 | 64 |
| | | **Pass 2: Server sends challenge to client** | | | |
| StoC | | 503677524A323146 "P6wRJ21F" | | 8 | 64 |
| | | **Pass 3: Client processes StoC** | | | |
| SC II AK II StoC | | 10D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF503677524A323146 | | | |
| T = **GMAC** (SC II AK II StoC) | | 1A52FE7DD3E72748973C1E28 | | 12 | 96 |
| f(StoC) = SC II IC II T | | 10000000011A52FE7DD3E72748973C1E28 | | 17 | 136 |
| | | **Pass 4: Server processes CtoS** | | | |
| SC II AK II CtoS | | 10D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF4B35366956616759 | | | |
| T (SC II AK II CtoS) | | FE1466AFB3DBCD4F9389E2B7 | | 12 | 96 |
| f(CtoS) = SC II IC II T | | 1001234567FE1466AFB3DBCD4F9389E2B7 | | 17 | 136 |

Table 34 provides a test vector for HLS authentication-mechanism 7 with ECDSA.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 34 – HLS example using authentication-mechanism 7 with ECDSA**

| Security Material | X | Contents | LEN (Bytes) |
|---|---|---|---|
| Security Suite | | ECDH-ECDSA-AES-128-GCM | |
| Curve | | P-256 | |
| Domain Parameters | D | See Table G.1. | |
| System Title Client | Sys-TC | 4D4D4D0000BC614E | 8 |
| System Title Server | Sys-TS | 4D4D4D0000000001 | 8 |
| Private Key Client | Pri-KC | E9A045346B2057F1820318AB125493E9AB36CE590011C0FF30090858A118DD2E | 32 |
| Private Key Server | Pri-KS | B582D8C910018302BA3131BAB9BB6838108BB9408C30B2E49285985256A59038 | 32 |
| Public Key Client | Pub-KC | 917DBFECA43307375247989F07CC23F53D4B963AF8026C749DB33852011056DFDBE8327BD69CC149F018A8E446DDA6C55BCD78E596A56D403236233F93CC89B3 | 64 |
| Public Key Server | Pub-KS | E4D07CEB0A5A6DA9D2228B054A1F5E295E1747A963974AF75091A0B0BC2FB92DA7D2ABD9FDD41579F36A1C8171A0CB638221DF1949FD95C8FAE148896920450D | 64 |
| Challenge Client To Server | CtoS | 2CA1FC2DE9CD03B5E8E234CEA16F2853F6DC5F54526F4F4995772A50FB7E63B3 | 32 |
| Challenge Server To Client | StoC | 18E95FFE3AD0DCABDC5D0D141DC987E270CB0A395948D4231B09DE6579883657 | 32 |
| ECDSA(SystemTitle-C \|\| SystemTitle-S \|\|StoC \|\| CtoS)<br>(calculated with Pri-KC) | f(StoC) | C5C6D6620BDB1A39FCE50F4D64F0DB712D6FB57A64030B0C297F1250DC859660D3B1FA334AD80411807369F5DD3BC17B59894C9E9C11C59376580D15A2646D16 | 64 |
| ECDSA(SystemTitle-S \|\| SystemTitle-C \|\| CtoS \|\| StoC)<br>(calculated with Pri-KS) | f(CtoS) | 946C2E3E4F18291571F4A45ACB7086100574694A3BAF67D2D147FE8F92481A5AB2186C5CBC3F80E94482D9388B85C6A73E5FD687F09773C1F615AA2A905ED057 | 64 |
| NOTE The values of the public keys are represented here as FE2OS($x_p$)II FE2OS($y_p$). | | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.7.5 Protecting COSEM data

The cryptographic algorithms applied to xDLMS APDUs can be also applied to COSEM data, i.e. attribute values and method invocation / return parameters. This is achieved by accessing attributes and/or methods of other COSEM objects indirectly through instances of the "Data protection" interface class, see IEC 62056-6-2:2021, 4.4.9.

The list of data to be protected, the required protection and the protection parameters are determined by the "Data protection" objects.

"Data protection" objects allow applying or removing protection when reading or writing a list of attributes, or when invoking methods of COSEM objects. Protection to be applied / removed may include any combination of authentication, encryption and digital signature.

The APDUs carrying the service invocations to access the attributes and methods of "Data protection" objects are protected as required by the prevailing security policy and the access rights of the "Data protection" object.

# 6 DLMS®/COSEM application layer service specification

## 6.1 Service primitives and parameters

In general, the services of a layer (or sublayer) are the capabilities it offers to a user in the next higher layer (or sublayer). In order to provide its service, a layer builds its functions on the services it requires from the next lower layer. Figure 34 illustrates this notion of service hierarchy and shows the relationship of the two correspondent N-users and their associated N-layer peer protocol entities.

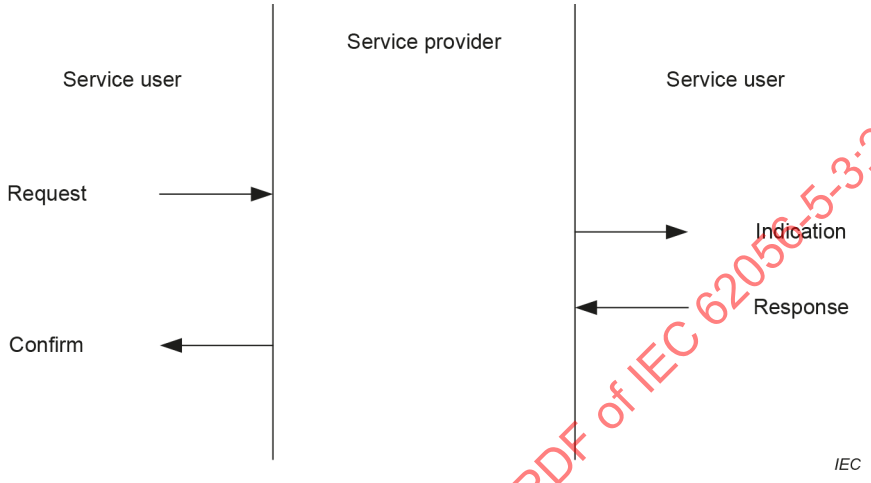



**Figure 34 – Service primitives**

Services are specified by describing the information flow between the N-user and the N-layer. This information flow is modelled by discrete, instantaneous events, which characterize the provision of a service. Each event consists of passing a service primitive from one layer to the other through an N-layer service access point associated with an N-user. Service primitives convey the information required in providing a particular service. These service primitives are an abstraction in that they specify only the service provided rather than the means by which the service is provided. This definition of service is independent of any particular interface implementation.

Services are specified by describing the service primitives and parameters that characterize each service. A service may have one or more related primitives that constitute the activity that is related to the particular service. Each service primitive may have zero or more parameters that convey the information required to provide the service. Primitives are of four generic types:

- REQUEST: The request primitive is passed from the N-user to the N-layer to request that a service be initiated;
- INDICATION: The indication primitive is passed from the N-layer to the N-user to indicate an internal N-layer event that is significant to the N-user. This event may be logically related to a remote service request, or may be caused by an event internal to the N-layer;
- RESPONSE: The response primitive is passed from the N-user to the N-layer to complete a procedure previously invoked by an indication primitive;
- CONFIRM: The confirm primitive is passed from the N-layer to the N-user to convey the results of one or more associated previous service request(s).

Possible relationships among primitive types are illustrated by the time-sequence diagrams shown in Figure 35. The figure also indicates the logical relationship of the primitive types. Primitive types that occur earlier in time and are connected by dotted lines in the diagrams are the logical antecedents of subsequent primitive types.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

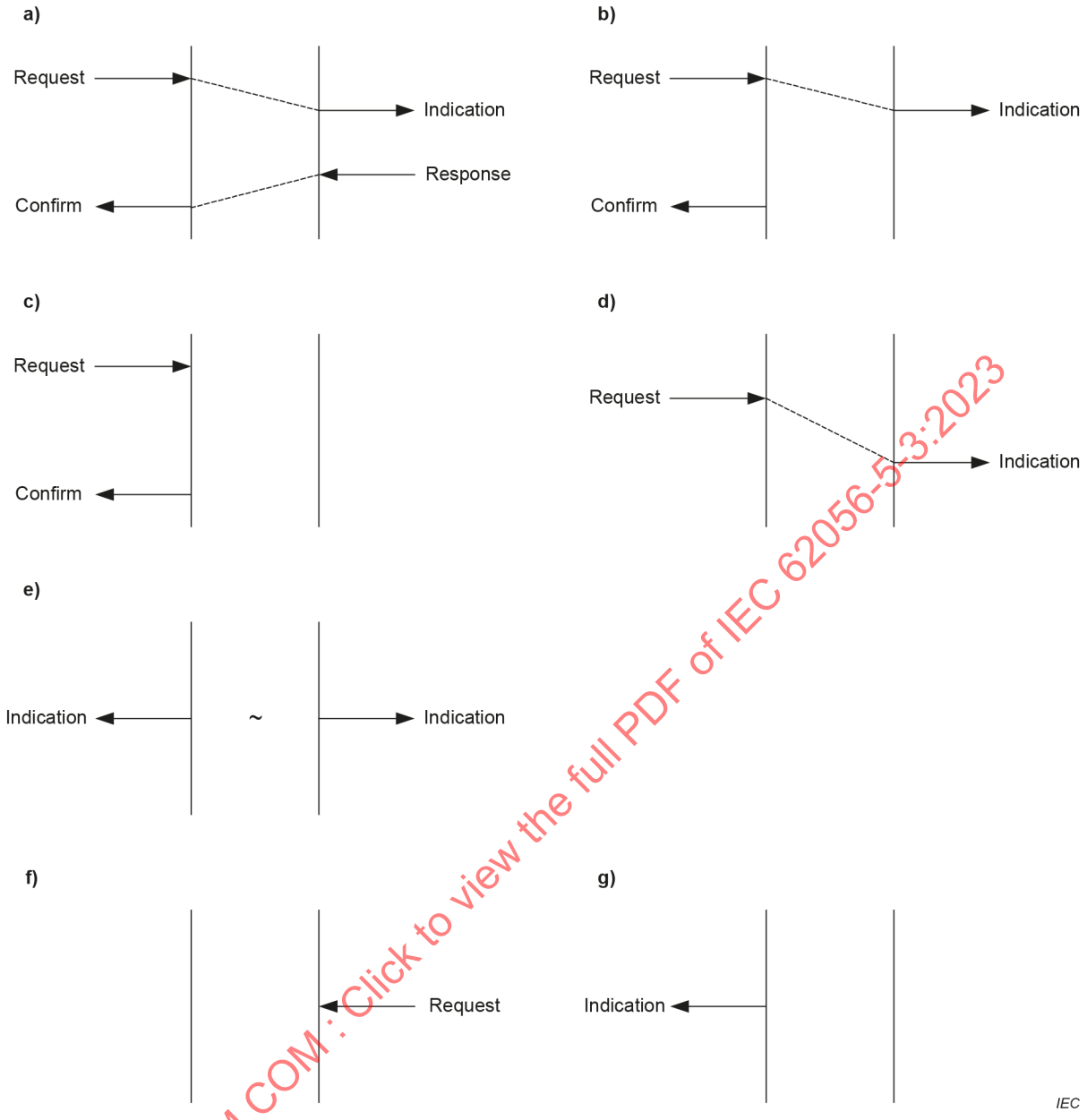



**Figure 35 – Time sequence diagrams**

IEC 1119/13

The service parameters of the COSEM AL service primitives are presented in a tabular format. Each table consists of two to five columns describing the service primitives and their parameters. In each table, one parameter – or a part of it – is listed on each line. In the appropriate service primitive columns, a code is used to specify the type of usage of the parameter. The codes used are listed in Table 35.

Some parameters may contain sub-parameters. These are indicated by labelling of the parameters as M, U, S or C, and indenting all sub-parameters under the parameter. Presence of the sub-parameters is always dependent on the presence of the parameter that they appear under. For example, an optional parameter may have sub-parameters; if the parameter is not supplied, then no sub-parameters may be supplied.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 35 – Codes for AL service parameters**

| | |
|---|---|
| M | The parameter is mandatory for the primitive. |
| U | The parameter is a user option, and may or may not be provided depending on dynamic usage by the ASE user. |
| S | The parameter is selected among other S-parameters as internal response of the server ASE environment. |
| C | The parameter is conditional upon other parameters or the environment of the ASE user. |
| (–) | The parameter is never present. |
| = | The " (=) " code following one of the M, U, S or C codes indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table. For instance, an " M (=) " code in the .indication service primitive column and an "M" in the .request service primitive column means that the parameter in the .indication primitive is semantically equivalent to the one in the .request primitive. |

Throughout this standard, the following rules are observed regarding the naming of terms:

- the name of ACSE services and the data transfer services using LN referencing is written in uppercase. Examples are: COSEM-OPEN, GET;
- the name of the data transfer services using SN referencing is written in title case. Examples are: Read, Write;
- camel notation is used in the following cases: DataNotification, EventNotification, TriggerEventNotificationSending, UnconfirmedWrite, InformationReport;
- the types of the LN service primitives may be mentioned in two alternative forms. Examples: "GET.request service primitive of Request_Type == NORMAL" or "GET-REQUEST-NORMAL service primitive";
- service parameter name elements are capitalized and joined with an underscore to signify a single entity: Examples are Protocol_Connection_Parameters and COSEM_Attribute_Descriptor;
- when the same parameter may occur several times, this is indicated by repeating the parameter in curly brackets. Example: Data { Data };
- in the data transfer service specifications, parameters used with block transfer only are shown in bold. Example: **DataBlock_G**;
- direct reference to a service parameter uses the capitalized form, while indirect (non-specific) reference uses the normal text without underscore joining. A direct reference example is: "The COSEM_Attribute_Descriptor parameter references a COSEM object attribute." An indirect (non-specific) reference example is: "A GET-REQUEST-NORMAL service primitive contains a single COSEM attribute descriptor";
- the names of COSEM data transfer APDUs using LN referencing are capitalized and joined with a dash to signify a single entity. Example: Get-Request-Normal;
- the names of COSEM data transfer APDUs using SN referencing use the camel notation. Example: ReadRequest.

## 6.2 The COSEM-OPEN service

*Function*

The function of the COSEM-OPEN service is to establish an AA between peer COSEM APs. It uses the A-ASSOCIATE service of the ACSE. The COSEM-OPEN service provides only the framework for *transporting* this information. To provide and verify that information is the job of the appropriate COSEM AP.

*Semantics of the service primitives*

The COSEM-OPEN service primitives shall provide parameters as shown in Table 36.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 36 – Service parameters of the COSEM-OPEN service primitives**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Protocol_Connection_Parameters | M | M (=) | M | M (=) |
| ACSE_Protocol_Version | U | U (=) | U | U (=) |
| Application_Context_Name | M | M (=) | M | M (=) |
| Called_AP_Title | U | U (=) | – | – |
| Called_AE_Qualifier | U | U(=) | – | – |
| Called_AP_Invocation_Identifier | U | U (=) | – | – |
| Called_AE_Invocation_Identifier | U | U (=) | – | – |
| Calling_AP_Title | C | C (=) | – | – |
| Calling_AE_Qualifier | U | U (=) | – | – |
| Calling_AP_Invocation_Identifier | U | U (=) | – | – |
| Calling_AE_Invocation_Identifier | U | U (=) | – | – |
| Local_Or_Remote | – | – | – | M |
| Result | – | – | M | M |
| Failure_Type | – | – | M | M |
| Responding_AP_Title | – | – | C | C (=) |
| Responding_AE_Qualifier | – | – | U | U (=) |
| Responding_AP_Invocation_Identifier | – | – | U | U (=) |
| Responding_AE_Invocation_Identifier | – | – | U | U (=) |
| ACSE_Requirements | U | U (=) | U | U (=) |
| Security_Mechanism_Name | C | C (=) | C | C (=) |
| Calling_Authentication_Value | C | C (=) | – | – |
| Responding_Authentication_Value | – | – | C | C (=) |
| Implementation_Information | U | U (=) | U | U (=) |
| Proposed_xDLMS_Context | M | M (=) | – | – |
| Dedicated_Key | C | C (=) | – | – |
| Response_Allowed | C | C (=) | | |
| Proposed_DLMS_Version_Number | M | M (=) | – | – |
| Proposed_DLMS_Conformance | M | M (=) | – | – |
| Client_Max_Receive_PDU_Size | M | M (=) | – | – |
| Negotiated_xDLMS_Context | – | – | S | S (=) |
| Negotiated_DLMS_Version_Number | – | – | M | M (=) |
| Negotiated_DLMS_Conformance | – | – | M | M (=) |
| Server_Max_Receive_PDU_Size | – | – | M | M (=) |
| VAA_Name | | | M | M (=) |
| xDLMS_Initiate_Error | | | S | S (=) |
| User_Information | U | C (=) | – | – |
| Service_Class | M | M (=) | – | – |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The service parameters of the COSEM-OPEN.request service primitive, except the Protocol_Connection_Parameters, the User_Information parameter and – depending on the communication profile – the Service_Class parameter are carried by the fields of the AARQ APDU sent by the client.

The service parameters of the COSEM-OPEN.response service primitive, except the Protocol_Connection_Parameters is carried by the fields of the AARE APDU sent by the server.

The A-ASSOCIATE service and the AARQ and AARE APDUs are specified in 7.2. Encoding examples are given in Annex D and Annex E.

The Protocol_Connection_Parameters parameter is mandatory. It contains all information necessary to use the layers of the communication profile, including the communication profile (protocol) identifier and the addresses required. It identifies the participants of the AA. The elements of this parameter are passed to the entities managing lower layer connections and to the lower layers as appropriate.

The ACSE_Protocol_Version parameter is optional. If present, the default value shall be used.

The Application_Context_Name parameter is mandatory. In the request primitive, it holds the value proposed by the client. In the response primitive, it holds the same value or the value supported by the server.

The use of the Called_AP_Title, Called_AE_Qualifier, Called_AP_Invocation_Identifier, Called_AE_Invocation_Identifier parameters is optional. Their use is not specified in this International Standard.

The use of the Calling_AP_Title parameter is conditional. When the Application_Context_Name indicates an application context using ciphering, it may carry the client system title specified in 4.1.3.4.

The use of the Calling_AE_Qualifier parameter is conditional. When the Application_Context_Name indicates an application context using ciphering, it may carry the public digital signature key certificate of the client.

The use of the Calling_AP_Invocation_Identifier is optional. Its use is not specified in this document.

The use of the Calling_AE_Invocation_Identifier parameter is optional. When present, it carries the identifier of the client-side user of the AA.

NOTE 1 The client user identification mechanism is specified in IEC 62056-6-2:2021, 4.4.2.

The Local_or_Remote parameter is mandatory. It indicates the origin of the COSEM-OPEN.confirm primitive. It is set to Remote if the primitive has been generated following the reception of an AARE APDU from the server. It is set to Local if the primitive has been locally generated.

The Result parameter is mandatory. In the case of remote confirmation, it indicates whether the Server accepted the proposed AA or not. In the case of local confirmation, it indicates whether the client side protocol stack accepted the request or not.

The Failure_type parameter is mandatory. In the case of remote confirmation, it carries the information provided by the server. In the case of local and negative confirmation, it indicates the reason for the failure.

The use of the Responding_AP_Title parameter is conditional. When the Application_Context_Name parameter indicates an application context using ciphering, it may carry the server system title specified in 4.1.3.4.

The use of the Responding_AE_Qualifier is conditional. When the Application_Context_Name indicates an application context using ciphering, it may carry the public digital signature key certificate of the server.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The use of the Responding_AP_Invocation_Identifier and Responding_AE_Invocation_ Identifier parameters is optional. Their use is not specified in this International Standard.

The ACSE_Requirements parameter is optional. It is used to select the optional authentication functional unit of the A-Associate service for the association; see 7.2.1.

The presence of the ACSE_Requirements parameter depends on the authentication mechanism used:

- in the case of Lowest Level Security authentication it shall not be present; only the Kernel functional unit will be used;
- in the case of Low Level Security (LLS) authentication it shall be present in the .request primitive and it may be present in the .response service primitive and it shall indicate authentication (bit 0 set);
- in the case of High Level Security (HLS) authentication, it shall be present both in the .request and the .response service primitives and it shall indicate authentication (bit 0 set).

The Security_Mechanism_Name parameter is conditional. It is present only if the authentication functional unit has been selected. If present, the .request primitive holds the value proposed by the client and the .response primitive holds the value required by the server, i.e. the one to be used by the client.

The Calling_Authentication_Value parameter and the Responding_Authentication_Value parameters are conditional. They are present only if the authentication functional unit has been selected. They hold the client authentication value / server authentication value respectively, appropriate for the Security_Mechanism_Name.

The Implementation_Information parameter is optional. Its use is not specified in this international standard.

The Proposed_xDLMS_Context parameter holds the elements of the proposed xDLMS context carried by the xDLMS InitiateRequest APDU, placed in the user-information field of the AARQ APDU.

The Dedicated_Key element is conditional. It may be present only, when the Application_Context_Name parameter indicates an application context using ciphering. The dedicated key is used for dedicated ciphering of xDLMS APDUs exchanged within the AA established.

When the dedicated key is present, the xDLMS InitiateRequest APDU shall be authenticated and encrypted using the AES-GCM algorithm, the global unicast encryption key and the authentication key (if in use). In addition it shall also be digitally signed if required by the security policy.

The xDLMS InitiateRequest APDU shall be protected the same way as described above, when the dedicated key is not present, but it is necessary to protect the RLRQ APDU by including the protected xDLMS InitiateRequest in its user-information field. See 6.3.

The use of Response_Allowed element is conditional. It indicates if the server is allowed to respond with an AARE APDU, i.e. if the AA to be established is confirmed (Response_Allowed == TRUE) or not confirmed (Response_Allowed == FALSE).

The Proposed_DLMS_Version_Number element holds the proposed DLMS® version number; see 4.2.4.

The Proposed_DLMS_Conformance element holds the proposed conformance block; see 7.3.1.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The Client_Max_Receive_PDU_Size element holds the maximum length of the xDLMS APDUs the client can receive; see Table 2.

If the xDLMS context proposed by the client is acceptable for the server, then the response service primitive shall contain the Negotiated_xDLMS_Context parameter. It holds the elements of the negotiated xDLMS context, carried by the xDLMS InitiateResponse APDU, placed in the user-information field of the AARE APDU. If the xDLMS InitiateRequest APDU has been ciphered, the xDLMS InitiateResponse APDU shall be also ciphered the same way.

The Negotiated_DLMS_Version_Number element holds the negotiated DLMS® version number. See 4.2.4.

The Negotiated_DLMS_Conformance element holds the negotiated conformance block. See 7.3.1.

The Server_Max_Receive_PDU_Size element carries the maximum length of the xDLMS APDUs the server can receive; see Table 2.

The VAA_name element carries the dummy value of 0x0007 in the case of LN referencing, and the base_name of the current Association object, 0xFA00, in the case of SN referencing.

If the xDLMS context proposed by the client is not acceptable for the server, then the response service primitive shall carry the xDLMS_Initiate_Error parameter. It is carried by the ConfirmedServiceError APDU, with appropriate diagnostic elements, placed in the user-information field of the AARE APDU.

The User_Information parameter is optional. If present, it shall be passed on to the supporting layer, provided it is capable to carry it. The indication primitive shall then contain the user-specific information carried by the supporting lower protocol layer(s); see Annex A.

NOTE 2 The User_Information parameter of the COSEM-OPEN service is not to be confused with the user-information field of the AARQ / AARE APDUs.

The Service_Class parameter is mandatory. It indicates whether the service shall be invoked in a confirmed or in an unconfirmed manner. The handling of this parameter may depend on the communication profile; see Annex A.

*Use*

Possible logical sequences of the COSEM-OPEN service primitives are illustrated in Figure 35:

- for confirmed AA – successful or unsuccessful – establishment, item a);
- for unconfirmed AA establishment, item b);
- in the case of a pre-established AA or an unsuccessful attempt due to a local error, item c).

The .request primitive is invoked by the client AP to request the establishment of a confirmed or an unconfirmed AA with a server AP.

Before the invocation of the COSEM-OPEN.request primitive, the physical layers shall be connected. Depending on the communication profile, the invocation of this primitive may also imply the connection of other lower layers.

Upon reception of the request invocation, the AL constructs and sends an AARQ APDU to the server.

The .indication primitive is generated by the server AL when a correctly formatted AARQ APDU is received.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The .response primitive is invoked by the server AP to indicate to the AL whether the proposed AA is accepted or not. It is invoked only if the proposed AA is confirmed. The AL constructs then an AARE APDU and sends it to its peer, containing the service parameters received from the AP.

The .confirm primitive is generated by the client AL to indicate to the client AP whether the AA previously requested is accepted or not:

- remotely, when an AARE APDU is received;
- locally, if the requested AA already exists; this includes pre-established AAs;
- locally, if the corresponding .request primitive has been invoked with Service_Class == Unconfirmed;
- locally, if the requested AA is not allowed;
- locally, if an error is detected: missing or not correct parameters, failure during the establishment of the requested lower layer connections, missing physical connection, etc.

The protocol for establishing an AA is specified in 7.2.4. Communication profile specific rules are specified in Annex A.

### 6.3 The COSEM-RELEASE service

*Function*

The function of the COSEM-RELEASE service is to gracefully release an existing AA. Depending on the way it is invoked, it uses the A-RELEASE service of the ACSE or not.

*Semantics of the service primitives*

The COSEM-RELEASE service primitives shall provide parameters as shown in Table 37.

**Table 37 – Service parameters of the COSEM-RELEASE service primitives**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Use_RLRQ_RLRE | U | C(=) | C(=) | - |
| Reason | U | U (=) | U | U (=) |
| Proposed_xDLMS_Context | C | C (=) | – | – |
| Negotiated_xDLMS_Context | – | – | C | C (=) |
| Local_Or_Remote | – | – | – | M |
| Result | – | – | M | M |
| Failure_Type | – | – | – | C |
| User_Information | U | C (=) | U | C (=) |

The Use_RLRQ_RLRE parameter in the .request primitive is optional. If present, its value may be FALSE (default) or TRUE. It indicates whether the ACSE A-RELEASE service – involving an RLRQ / RLRE APDU exchange – should be used or not. The A-RELEASE service and the RLRQ / RLRE APDUs are specified in 7.2. The Use_RLRQ_RLRE parameter in the .response primitive is conditional. If it was present in the .indication primitive and if its value was TRUE, it shall also be present and its value shall be TRUE. Otherwise, it shall not be present or its value shall be FALSE.

If the value of the Use_RLRQ_RLRE parameter is FALSE, then the AA can be released by disconnecting the supporting layer of the AL.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The Reason parameter is optional. It may be present only if the value of the Use_RLRQ_RLRE is TRUE. It is carried by the reason field of the RLRQ / RLRE APDU respectively.

When used on the .request primitive, this parameter identifies the general level of urgency of the request. It takes one of the following symbolic values:

- normal;
- urgent (not available in DLMS®/COSEM); or
- user defined.

When used on the .response primitive, this parameter identifies information about why the acceptor accepted or rejected the release request. Note that in DLMS®/COSEM, the server cannot reject the release request. It takes one of the following symbolic values:

- normal;
- not finished; or
- user defined.

NOTE 1 The value "not finished" is used in the .response primitive when the acceptor is forced to release the association but wishes to give a warning that it has additional information to send or receive.

The Proposed_xDLMS_Context parameter is conditional. It is present only if the value of the Use_RLRQ_RLRE is TRUE and the AA to be released has been established with an application context using ciphering. This option allows securing the COSEM-RELEASE service, and avoiding thereby a denial-of-service attack that may be carried out by unauthorized releasing of the AA.

In the .request primitive, the Proposed_xDLMS_Context parameter shall be the same as in the COSEM-OPEN.request service primitive, having established the AA to be released. It is carried by the xDLMS InitiateRequest APDU, protected the same way as in the AARQ and placed in the user-information field of the RLRQ APDU.

If the xDLMS InitiateRequest APDU can be successfully deciphered, then the .response primitive shall carry the same Negotiated_xDLMS_Context parameter as in the COSEM-OPEN.response primitive. It is carried by the xDLMS InitiateResponse APDU, protected the same way as in the AARE and placed in the user-information field of the RLRE APDU.

Otherwise, the RLRQ APDU is silently discarded.

The Local_or_Remote parameter is mandatory. It indicates the origin of the COSEM-RELEASE.confirm primitive.

It is set to Remote if either:

- a RLRE APDU has been received from the server; or
- a disconnect confirmation service primitive has been received.

It is set to Local if the primitive has been locally generated.

The Result parameter is mandatory. In the .response primitive, it indicates whether the server AP can accept the request to release the AA or not. As servers cannot refuse such requests, its value should normally be SUCCESS unless the AA referenced does not exist.

The Failure_Type parameter is conditional. It is present if Result == ERROR. In this case, it indicates the reason for the failure. It is a locally generated parameter on the client side.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The User_Information parameter in the .request primitive is optional. If present, it is passed to the supporting layer, provided it is able to carry it. The .indication primitive contains then the user-specific information carried by the supporting lower protocol layer(s). Similarly, it is optional in the .response primitive. If present, it is passed to the supporting layer. In the .confirm primitive, it may be present only when the service is remotely confirmed. It contains then the user-specific information carried by the supporting lower protocol layer(s).

NOTE 2 The User_Information parameter of the COSEM-RELEASE service is not to be confused with the user-information field of the RLRQ / RLRE APDUs.

The specification of the content of the User_Information parameter is not within the scope of this international standard. See also Annex A.

*Use*

Possible logical sequences of the COSEM-RELEASE service primitives are illustrated in Figure 35:

- for successful release of a confirmed AA , item a);
- for release of an unconfirmed AA , item b); and
- for an unsuccessful attempt due to a local error, item c).

The use of the COSEM-RELEASE service depends on the value of the Use_RLRQ_RLRE parameter. When it is invoked with Use_RLRQ_RLRE == TRUE, the service is based on the ACSE A-RELEASE service. Otherwise, the invocation of the service leads to the disconnection of the supporting layer.

The .request primitive is invoked by the client AP to request the release of a confirmed or an unconfirmed AA with a server AP. Upon reception of the request invocation with Use_RLRQ_RLRE == TRUE, the AL constructs and sends an RLRQ APDU to the server. Otherwise, it sends an XX-DISCONNECT.request primitive (where XX is the supporting lower protocol layer).

The .indication primitive is generated by the server AL if:

- a RLRQ APDU is received. If a deciphering error occurs, the RLRQ APDU is silently discarded (no .indication primitive is generated); or
- an XX-DISCONNECT.request is received.

The .response primitive is invoked by the server AP, but only if the AA to be released is confirmed. Note that the server AP cannot refuse this request. Upon reception of the .response service primitive, the server AL:

- sends a RLRE APDU, if the Use_RLRQ_RLRE parameter is TRUE; or
- sends an XX-DISCONNECT.response otherwise.

The .confirm primitive is generated by the client AL to indicate to the client AP whether the requested release of the AA is accepted or not:

- remotely, when an XX-DISCONNECT.cnf primitive is received. The supporting layer is disconnected; or
- remotely, when a RLRE APDU is received. The supporting layer is not disconnected; or
- locally, upon the expiry of a time-out on waiting for an RLRE APDU; or
- locally, when an RLRQ APDU to release an unconfirmed AA is sent out; or
- locally, when a local error is detected: missing or incorrect parameters, or communication failure at lower protocol layer level etc.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

If the RLRE APDU received contains a ciphered xDLMS InitiateResponse APDU but it cannot be deciphered, then the RLRE APDU shall be discarded. It is left to the client to cope with the situation.

The protocol for releasing an AA is specified in 7.2.5. Communication profile specific rules are specified in Annex A. See also 7.2.1.

### 6.4 COSEM-ABORT service

*Function*

The function of the COSEM-ABORT service is to indicate an unsolicited disconnection of the supporting layer.

*Semantics*

The COSEM-ABORT service primitives shall provide parameters as shown in Table 38.

**Table 38 – Service parameters of the COSEM-ABORT service primitives**

| | .indication |
|---|---|
| Diagnostics | U |

The Diagnostics parameter is optional. It shall indicate the possible reason for the disconnection, and may carry lower protocol layer dependent information as well. Specification of the contents of this parameter is not within the scope of this document.

*Use*

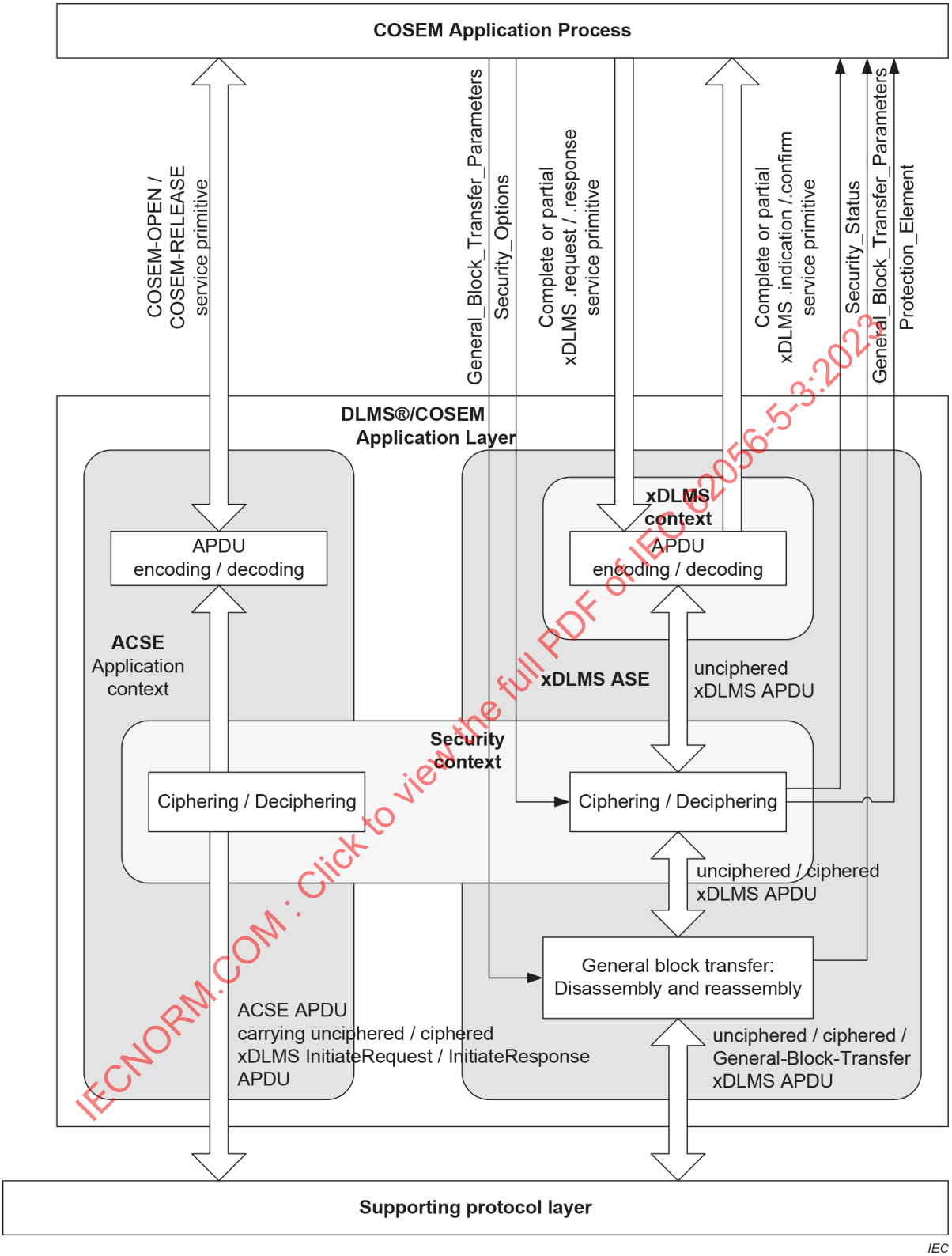
The COSEM-ABORT.indication primitive is locally generated both on the client and on the server side to indicate to the COSEM AP that a lower layer connection closed in an unsolicited manner. The origin of such an event can be an external event (for example the physical line is broken), or an action of a supporting layer connection manager AP, present in some profiles, when the supporting layer connection is not managed by the COSEM AL. This shall cause the COSEM APs to abort any existing AAs, except the pre-established ones on the server side.

The protocol for the COSEM-ABORT service is specified in 7.2.5.3.

### 6.5 Protection and general block transfer parameters

To control cryptographic protection of xDLMS APDUs and the GBT mechanism, additional service parameters are passed between the AL and the AP as shown in Figure 36 and Table 39.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

NOTE For services initiated by the client, the service primitives are .request, .indication, .response and .confirm. For unsolicited services – initiated by the server – the service primitives are .request and .indication.

**Figure 36 – Additional service parameters to control cryptographic protection and GBT**

**Table 39 – Additional service parameters**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Additional_Service_Parameters | U | – | U (=) | |
| Invocation_Type | M | – | M (=) | – |
| Security_Options | C | – | C (=) | – |
| General_Block_Transfer_Parameters | C | – | C (=) | – |
| Block_Transfer_Streaming | M | – | M (=) | – |
| Block_Transfer_Window | M | – | M (=) | – |
| Service_Parameters | M | – | M (=) | – |
| Additional_Service_Parameters | – | U | – | U (=) |
| Invocation_Type | – | U | – | U (=) |
| Security_Status | – | C | – | C (=) |
| General_Block_Transfer_Parameters | – | C | – | C (=) |
| Block_Transfer_Window | – | M | – | M (=) |
| Service_Parameters | – | M | – | M (=) |
| Protection_Element | – | C | – | C (=) |
| NOTE The service primitives available depend on the kind of the service. | | | | |

The Additional_Service_Parameters are present only if ciphering or GBT is used.

The Invocation_Type parameter is mandatory: it indicates if the service invocation is complete or partial. Possible values: COMPLETE, FIRST-PART, ONE-PART and LAST-PART.

NOTE 1 Partial service invocations may be useful when the service parameters are long. However, there is no direct relationship between the partial service invocations and the general-block-transfer APDUs.

The Security_Options parameter is conditional: it is present only if the application context is a ciphered one, the .request / .response service primitive has to be ciphered and Invocation_Type = COMPLETE or FIRST-PART. It determines the protection to be applied by the AL. See also Table 40, 7.3.13 and Figure 62.

The General_Block_Transfer_Parameters parameter is conditional: it is present only if general block transfer (GBT) is used and Invocation_Type = COMPLETE or FIRST-PART. It provides information on the GBT streaming capabilities:

- the Block_Transfer_Streaming parameter is present only in .request and .response service primitives. It is passed by the AP to the AL to indicate if the AL is allowed to send general-block-transfer APDUs using streaming (TRUE) or not (FALSE);
- the Block_Transfer_Window parameter indicates the window size supported, i.e. the maximum number of blocks that can be received in a window.

The streaming process itself is managed by the AL. See 7.3.13.

The Service_Parameters are mandatory: they include the parameters of xDLMS service invocations. If Invocation_Type != COMPLETE, then it includes a part of the service parameters.

The Security_Status parameter is conditional: it is present only if cryptographic protection has been applied. It carries information on the protection that has been verified / removed by the AL. It may be present in all type of service invocations. See Table 40.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The Protection_Element parameter is conditional: it is present only if the APDU has been authenticated or signed. See Table 40.

**Table 40 – Security parameters**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Security_Options | C | – | C (=) | – |
| Security_Options_Element {Security_Options_Element} | M | – | M (=) | – |
| Security_Protection_Type | M | – | M (=) | – |
| Glo_Ciphering | S | – | S (=) | – |
| Ded_Ciphering | S | – | S (=) | – |
| General_Glo_Ciphering | S | – | S (=) | – |
| General_Ded_Ciphering | S | – | S (=) | – |
| General_Ciphering | S | – | S (=) | – |
| General_Signing | S | – | S (=) | – |
| *With General_Glo_Ciphering and General_Ded_Ciphering* | | | | |
| System_Title | U | | U (=) | – |
| *With General_Ciphering and General_Signing* | | | | |
| Transaction_Id | U | – | U (=) | – |
| Originator_System_Title | U | – | U (=) | – |
| Recipient_System_Title | U | – | U (=) | – |
| Date_Time | U | – | U (=) | – |
| Other_Information | U | – | U (=) | – |
| *With General_Ciphering* | | | | |
| Key_Info_Options | C | – | C (=) | – |
| Identified_Key_Options | S | – | S (=) | – |
| Wrapped_Key_Options | S | – | S (=) | – |
| Agreed_Key_Options | S | – | S (=) | – |
| *With Glo_Ciphering, Ded_Ciphering, General_Glo_Ciphering, General_Ded_Ciphering, General_Ciphering* | | | | |
| Security_Control | M | – | M (=) | – |
| Security_Status | – | C | – | C (=) |
| Security_Status_Element {Security_Status_Element} | – | M | – | M (=) |
| Security_Protection_Type | | M | | M (=) |
| Glo_Ciphering | – | S | – | S (=) |
| Ded_Ciphering | – | S | – | S (=) |
| General_Glo_Ciphering | – | S | – | S (=) |
| General_Ded_Ciphering | – | S | – | S (=) |
| General_Ciphering | – | S | – | S (=) |
| General_Signing | – | S | – | S (=) |
| *With General_Glo_Ciphering and General_Ded_Ciphering* | | | | |
| System_Title | – | U | – | U (=) |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| *With General_Ciphering and General_Signing* | | | | |
| Transaction_Id | – | U | – | U (=) |
| Originator_System_Title | – | U | – | U (=) |
| Recipient_System_Title | – | U | – | U (=) |
| Date_Time | – | U | – | U (=) |
| Other_Information | – | U | – | U (=) |
| *With General_Ciphering* | | | | |
| Key_Info_Status | – | C | – | C (=) |
| Identified_Key_Status | – | S | – | S (=) |
| Wrapped_Key_Status | – | S | – | S (=) |
| Agreed_Key_Status | – | S | – | S (=) |
| *With Glo_Ciphering, Ded_Ciphering, General_Ded_Ciphering, General_Glo_Ciphering, General_Ciphering* | | | | |
| Security_Control | – | M | – | M (=) |
| *The protection element is present when authentication or digital signature is applied* | | | | |
| Protection_Element {Protection_Element} | – | C | – | C (=) |
| Invocation_Counter | – | C | – | C (=) |
| Authentication_Tag | – | C | – | C (=) |
| Signature | – | C | – | C (=) |

The Security_Options parameter contains one Security_Options_Element parameter for each kind of protection to be applied. Similarly, the Security_Status parameter contains one Security_Status_Element parameter for each kind of protection that has been applied. See also 5.7.3.

The Security_Options_Element and Security_Status_Element parameters include the following sub-parameters:

- the Security_Protection_Type sub-parameter is mandatory: it identifies the ciphered APDU to be used; see Table 41;
- the System_Title subparameter is optional. When present, it holds the system title of the sender. It can be present only with General_Glo_Ciphering and General_Ded_Ciphering;

NOTE 2 The purpose to include system-title of the sender is to allow the other party to build the initialization vector where the system-title has not been exchanged during the media specific registration process or during the AARQ / AARE exchange.

**Table 41 – APDUs used with security protection types**

| Security_Protection_Type | APDU |
|---|---|
| Glo_Ciphering | Service-specific glo-ciphering |
| Ded_Ciphering | Service-specific ded-ciphering |
| General_Glo_Ciphering | general-glo-ciphering |
| General_Ded_Ciphering | general-ded-ciphering |
| General_Ciphering | general-ciphering |
| General_Signing | general-signing |
| See also Table 26. | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The following five parameters are optionally present with General_Ciphering and General_Signing:

- Transaction_Id: identifies the transaction between two parties;
- Originator_System_Title: indicates the system title of the originator of the protected APDU;
- Recipient_System_Title: indicates the system title of the recipient, i.e. the entity which will verify / remove the protection that has been applied to the APDU. In the case of broadcast, the Recipient_System_Title shall be an empty string;
- The Date_Time parameter is optional. When present, it indicates the date and time of the invocation of the .request / .response service primitive. Unless otherwise specified in a project specific companion specification, the Date_Time parameter in the response shall be present if it was present in the request and shall not be present in the response if it was not present in the request;

  If any of the four parameters above is not used, then an octet-string of length zero shall be included.
- the Other_Information parameter is optional. When present, it holds additional information concerning the protection. Its content may be specified in project specific companion specifications;

The Key_Info_Options parameter is conditional: when protection has to be applied, it carries information on the symmetric key that has been used by the originator / is to be used by the recipient. The key information is sent / received as part of the ciphered APDU:

- Identified_Key_Options (see 5.5.3): it can be used when the partners share the key; this may be the global unicast encryption key or the global broadcast encryption key;
- Wrapped_Key_Options (see 5.5.4): in this case, a wrapped key is sent;
- Agreed_Key_Options (see 5.5.5): in this case, the partners use a Diffie-Hellman key agreement scheme to agree on the key;

Security_Control: contains the Security Control byte, see Table 27.

The Protection_Element parameter is conditional: it shall be present if the APDU has been authenticated or digitally signed. It may be present in all type of service invocations, but it may be empty if it is not yet available (this may occur in the case when general block transfer is used). It contains:

- in the case of General_Ciphering, the Invocation_Counter, holding the invocation field of the initialization vector, see 5.3.3.7.3;
- in the case when the APDU has been authenticated, the authentication tag;
- in the case when the APDU has been signed, the digital signature.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 6.6 The GET service

*Function*

The GET service is used with LN referencing. It can be invoked in a confirmed or unconfirmed manner. Its function is to read the value of one or more COSEM interface object attributes. The result can be delivered in a single response or – if it is too long to fit in a single response – in multiple responses, with block transfer.

*Semantics*

The GET service primitives shall provide parameters as shown in Table 42.

**Table 42 – Service parameters of the GET service**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Invoke_Id | M | M (=) | M (=) | M (=) |
| Priority | M | M (=) | M (=) | M (=) |
| Service_Class | M | M (=) | M (=) | M (=) |
| Request_Type | M | M (=) | – | – |
| COSEM_Attribute_Descriptor<br>{ COSEM_Attribute_Descriptor } | C | C (=) | – | – |
| COSEM_Class_Id | M | M (=) | | |
| COSEM_Object_Instance_Id | M | M (=) | | |
| COSEM_Object_Attribute_Id | M | M (=) | | |
| Access_Selection_Parameters | U | U (=) | | |
| Access_Selector | M | M (=) | | |
| Access_Parameters | M | M (=) | | |
| **Block_Number** | C | C (=) | – | – |
| Response_Type | – | – | M | M (=) |
| Result | – | – | M | M (=) |
| Get_Data_Result { Get_Data_Result } | – | – | S | S (=) |
| Data | | | S | S (=) |
| Data_Access_Result | | | S | S (=) |
| **DataBlock_G** | – | – | S | S (=) |
| Last_Block | | | M | M (=) |
| Block_Number | | | M | M (=) |
| Result | | | M | M (=) |
| Raw_Data | | | S | S (=) |
| Data_Access_Result | | | S | S (=) |
| NOTE For security parameters see Table 40. | | | | |

The Invoke_Id parameter identifies the instance of the service invocation.

The Priority parameter indicates the priority level associated to the instance of the service invocation: normal (FALSE) or high (TRUE).

The Service_Class parameter indicates whether the service is confirmed or unconfirmed. The handling of this parameter depends on the communication profile; see Annex A.

The use of the Request_Type and Response_Type parameters is shown in Table 43.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 43 – GET service request and response types**

| Request type | | Response type | |
|---|---|---|---|
| NORMAL | The value of a single attribute is requested. | NORMAL | The complete result is delivered. |
| | | ONE-BLOCK | One block of the result is delivered. |
| NEXT | The next data block is requested. | ONE-BLOCK | As above. |
| | | LAST-BLOCK | The last block of the result is delivered. |
| WITH-LIST | The value of a list of attributes is requested. | WITH-LIST | The complete result is delivered. |
| | | ONE-BLOCK | As above. |
| NOTE The same Response_Type can be present more than once, to show the possible responses to each kind of request. | | | |

The COSEM_Attribute_Descriptor parameter references a COSEM object attribute. It is present when Request_Type == NORMAL or WITH-LIST. It is a composite parameter:

- the (COSEM_Class_Id, COSEM_Object_Instance_Id) doublet non-ambiguously references one and only one COSEM object instance;
- the COSEM_Object_Attribute_Id element identifies the attribute(s) of the object instance. COSEM_Object_Attribute_Id == 0 references all public attributes of the object (Attribute_0 feature; see 4.2.4.3.7);
- the Access_Selection_Parameters is present only when COSEM_Object_Attribute_Id != 0 and if selective access to the given attribute is available; see 4.2.4.3.5. The Access_Selector and Access_Parameters sub-parameters are defined in the COSEM interface object definitions; see IEC 62056-6-2:2021.

A GET-REQUEST-NORMAL service primitive contains a single COSEM attribute descriptor. A GET-REQUEST-WITH-LIST service primitive contains a list of COSEM attribute descriptors; their number is limited by the server-max-receive-pdu-size: a GET.request service primitive shall always fit in a single APDU.

The Block_Number parameter is used in the GET-REQUEST-NEXT service primitive. It carries the number of the latest data block received correctly.

If the encoded form of the response fits in a single APDU, the Result is of type Get_Data_Result:

- the Data choice carries the value of the attribute at the time of access; or
- the Data_Access_Result choice carries the reason for the read to fail for this attribute.

A GET-RESPONSE-NORMAL service primitive carries a single Get_Data_Result parameter. A GET-RESPONSE-WITH-LIST service primitive carries a list of Get_Data_Result parameters; their number and order shall be the same as that of the COSEM_Attribute_Descriptor parameters in the request.

If COSEM_Object_Attribute_Id == 0 (Attribute_0), the Data shall be a structure containing the value of all public attributes in the order of their appearance in the given object specification. For attributes to which no access right is granted within the given AA, or which cannot be accessed for any other reason, null-data shall be returned.

If the encoded form of the response does not fit in a single APDU, it can be transported in data blocks using either the service-specific or the general block transfer mechanism.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

If the service-specific block transfer mechanism is used, the Result is of type DataBlock_G. It carries block transfer control information and raw-data:

- the Last_Block element indicates whether the current block is the last one (TRUE) or not (FALSE);
- the Block_Number element carries the number of the actual block sent;
- the (inner) Result element carries either Raw_Data or Data_Access_Result. Within this:
  - if the value of a single attribute was requested, Raw_Data carries a part of the value of the attribute (Data). If the Data cannot be delivered, the response shall be GET-RESPONSE-NORMAL with Data_Access_Result;
  - if the value of a list of attributes was requested, Raw_Data carries a part of the list of Get_Data_Results, either Data or Data_Access_Result for each attribute;
  - if the raw data cannot be delivered, Data_Access_Result shall carry the reason. For error cases, see 7.3.3.

*Use*

Possible logical sequences of the GET service primitives are illustrated in Figure 35:

- for a successful confirmed GET, item a);
- for an unconfirmed GET, item d); and
- for an unsuccessful attempt due to a local error, item c).

The GET.request primitive is invoked by the client AP to read the value of one or all attributes of one or more COSEM objects of the server AP. The first request shall always be of Request_Type == NORMAL or WITH-LIST. A GET-REQUEST-NEXT service primitive is invoked only when the server could not deliver the complete data in a single response, i.e. following the reception of a .confirm primitive of Response_Type == ONE-BLOCK. Upon reception of the .request primitive, the client AL builds the Get-Request APDU appropriate for the request type and sends it to the server.

The GET.indication primitive is generated by the server AL upon reception of a Get-Request APDU.

The GET.response primitive is invoked by the server AP, if Service_Class == Confirmed, to send a response to an .indication primitive received. If the complete data requested fits in a single APDU, the .response primitive is invoked with Response_Type == NORMAL or WITH-LIST as appropriate. Otherwise, it is invoked with Response_Type == ONE-BLOCK and finally with LAST-BLOCK.

The GET.confirm primitive is generated by the client AL to indicate the reception of a Get-Response APDU.

The protocol for the GET service is specified in 7.3.3.

### 6.7 The SET service

*Function*

The SET service is used with LN referencing. It can be invoked in a confirmed or unconfirmed manner. Its function is to write the value of one or more COSEM interface object attributes. The data to be written can be sent in a single request or – if it is too long to fit in a single request – in multiple requests, with block transfer.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

*Semantics*

The SET service primitives shall provide parameters as shown in Table 44.

**Table 44 – Service parameters of the SET service**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Invoke_Id | M | M (=) | M (=) | M (=) |
| Priority | M | M (=) | M (=) | M (=) |
| Service_Class | M | M (=) | M (=) | M (=) |
| Request_Type | M | M (=) | – | – |
| COSEM_Attribute_Descriptor<br>{ COSEM_Attribute_Descriptor } | C | C (=) | – | – |
| COSEM_Class_Id | M | M (=) | | |
| COSEM_Object_Instance_Id | M | M (=) | | |
| COSEM_Object_Attribute_Id | M | M (=) | | |
| Access_Selection_Parameters | U | U (=) | | |
| Access_Selector | M | M (=) | | |
| Access_Parameters | M | M (=) | | |
| Data {Data } | C | C (=) | – | – |
| **DataBlock_SA** | C | C (=) | – | – |
| Last_Block | M | M (=) | | |
| Block_Number | M | M (=) | | |
| Raw_Data | M | M (=) | | |
| Response_Type | – | – | M | M (=) |
| Result { Result } | – | – | C | C (=) |
| **Block_Number** | – | – | C | C (=) |
| NOTE For security parameters, see Table 40. | | | | |

The Invoke_Id parameter identifies the instance of the service invocation.

The Priority parameter indicates the priority level associated to the instance of the service invocation: normal (FALSE) or high (TRUE).

The Service_Class parameter indicates whether the service is confirmed or unconfirmed. The handling of this parameter depends on the communication profile; see Annex A.

The use of the Request_Type and Response_Type parameters is shown in Table 45.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 45 – SET service request and response types**

| Request type | | Response type | |
|---|---|---|---|
| NORMAL | The reference of a single attribute and the complete data to be written is sent. | NORMAL | The result is delivered. |
| FIRST-BLOCK | The reference of a single attribute and the first block of the data to be written is sent. | ACK-BLOCK | The correct reception of the block is acknowledged. |
| ONE-BLOCK | One block of the data to be written is sent. | | |
| LAST-BLOCK | The last block of the data to be written is sent. | LAST-BLOCK | The correct reception of the last block is acknowledged and the result is delivered. |
| | | LAST-BLOCK-WITH-LIST | The correct reception of the last block is acknowledged and the list of results is delivered. |
| WITH-LIST | The reference of a list of attributes and the complete data to be written is sent. | WITH-LIST | The list of results is delivered. |
| FIRST-BLOCK-WITH-LIST | The reference of a list of attributes and the first block of data to be written is sent. | ACK-BLOCK | The correct reception of the block is acknowledged. |
| NOTE The same Response_Type can be present more than once, to show the possible responses to each request. | | | |

The COSEM_Attribute_Descriptor parameter references a COSEM object attribute. It is present when Request_Type == NORMAL, FIRST-BLOCK, WITH-LIST and FIRST-BLOCK-WITH-LIST. It is a composite parameter:

- the (COSEM_Class_Id, COSEM_Object_Instance_Id) doublet non-ambiguously references one and only one COSEM object instance;
- the COSEM_Object_Attribute_Id identifies the attribute(s) of the object instance. COSEM_Object_Attribute_Id == 0 references all public attributes of the object (Attribute_0 feature; see 4.2.4.3.7);
- the Access_Selection_Parameters is present only when COSEM_Object_Attribute_Id != 0 and if selective access to the given attribute is available; see 4.2.4.3.5. The Access_Selector and the Access_Parameters sub-parameters are defined in the COSEM interface object definitions; IEC 62056-6-2:2021.

A SET-REQUEST-NORMAL or SET-REQUEST-WITH-FIRST-BLOCK service primitive contains a single COSEM attribute descriptor. A SET-REQUEST-WITH-LIST or a SET-REQUEST-FIRST-BLOCK-WITH-LIST service primitive contains a list of COSEM attribute descriptors; their number is limited by the server-max-receive-pdu-size: all COSEM attribute descriptors – together with (a part of) the data to be written – shall fit in a single APDU.

The Data parameter contains the data necessary to write the value of the referenced attributes. The number and the order of the Data parameters shall be the same as that of the COSEM_Attribute_Descriptor parameters.

If COSEM_Object_Attribute_Id == 0 (Attribute_0), the Data sent shall be a structure, containing, for each public attribute, in the order of their appearance in the given object specification, either a value or null-data, meaning that the given attribute need not be set.

If the encoded form of the Data parameter does not fit in a single APDU, it can be transported in blocks using either the service-specific or the general block transfer mechanism.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

If the service-specific block transfer mechanism is used, the DataBlock_SA parameter carries block transfer control information and raw-data:

- the Last_Block element indicates whether the current block is the last one (TRUE) or not (FALSE);
- the Block_Number element carries the number of the actual block sent;
- the Raw_Data element carries a part of the data to be written.

The Result parameters are present in the .response primitive when Response_Type != ACK-BLOCK. Their number and order shall be the same as that of the COSEM_Attribute_Descriptor parameters in the request. Each Result shall contain either the information "success" or a reason for failing to write the attribute referenced (Data_Access_Result). When in the .request primitive COSEM_Object_Attribute_Id == 0 (Attribute_0), the Result shall carry a list of results, either the information "success" or a reason for failing to write the attribute (Data_Access_Result), for each public attribute, in the order of their appearance in the given object specification.

The Block_Number parameter shall be present when Response_Type == ACK-BLOCK, LAST-BLOCK, or LAST-BLOCK-WITH-LIST. It carries the number of the latest data block received correctly.

*Use*

Possible logical sequence of the SET service primitives are illustrated in Figure 35:

- for a successful confirmed SET, item a);
- for an unconfirmed SET, item d); and
- for an unsuccessful attempt due to a local error, item c).

The SET.request primitive is invoked by the client AP to write the value of one or all attributes of one or more COSEM objects of the server AP. If the complete data to be sent fits in a single APDU, the .request primitive shall be invoked with Request_Type == NORMAL or WITH-LIST as appropriate. Otherwise, it shall be invoked with Request_Type == FIRST-BLOCK or FIRST-BLOCK-WITH-LIST, then with Request_Type == ONE-BLOCK and finally with LAST-BLOCK as appropriate. Upon reception of the .request primitive, the client AL builds the Set-Request APDU appropriate for the Request_Type and sends it to the server.

The SET.indication primitive is generated by the server AL upon reception of a Set-Request APDU.

The SET.response primitive is invoked by the server AP, if Service_Class == Confirmed, to send a response to an .indication primitive received. If the data were sent in a single APDU, the .response primitive is invoked with Response_Type == NORMAL or WITH-LIST as appropriate. Otherwise, it is invoked with Response_Type == ACK-BLOCK, and finally with LAST-BLOCK or LAST-BLOCK-WITH-LIST as appropriate.

The SET.confirm primitive is generated by the client AL to indicate the reception of a Set-Response APDU.

The protocol for the SET service is specified in 7.3.4.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 6.8 The ACTION service

*Function*

The ACTION service is used with LN referencing. It can be invoked in a confirmed or unconfirmed manner. Its function is to invoke one or more COSEM interface objects methods. It comprises two phases:

- in the first phase, the client sends the reference(s) of the method(s) to be invoked, with the method invocation parameters necessary;
- in the second phase, after invoking the methods, the server sends back the result and the return parameters generated by the invocation of the method(s), if any.

If the method invocation parameters are too long to fit in a single request, they are sent in multiple requests (block transfer from the client to the server). If the result and the return parameters are too long to fit in a single response, they are returned in multiple responses (block transfer from the server to the client.)

*Semantics*

The ACTION service primitives shall provide parameters as shown in Table 46.

**Table 46 – Service parameters of the ACTION service**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Invoke_Id | M | M (=) | M (=) | M (=) |
| Priority | M | M (=) | M (=) | M (=) |
| Service_Class | M | M (=) | M (=) | M (=) |
| Request_Type | M | M (=) | – | – |
| COSEM_Method_Descriptor<br>{ COSEM_Method_Descriptor } | C | C (=) | – | – |
| COSEM_Class_Id | M | M (=) | | |
| COSEM_Object_Instance_Id | M | M (=) | | |
| COSEM_Object_Method_Id | M | M (=) | | |
| Method_Invocation_Parameters<br>{ Method_Invocation_Parameters } | U | U (=) | – | – |
| Response_Type | - | - | M | M (=) |
| Action_Response { Action_Response } | – | – | M | M (=) |
| Result | | | M | M (=) |
| Response_Parameters | | | U | U (=) |
| Data | | | S | S (=) |
| Data_Access_Result | | | S | S (=) |
| **DataBlock_SA** | C | C (=) | C | C (=) |
| Last_Block | M | M (=) | M | M (=) |
| Block_Number | M | M (=) | M | M (=) |
| Raw_Data | M | M (=) | M | M (=) |
| **Block_Number** | C | C (=) | C | C (=) |
| NOTE For security parameters, see Table 40. | | | | |

The Invoke_Id parameter identifies the instance of the service invocation.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The Priority parameter indicates the priority level associated to the instance of the service invocation: normal (FALSE) or high (TRUE).

The Service_Class parameter indicates whether the service is confirmed or unconfirmed. The handling of this parameter depends on the communication profile; see Annex A.

The use of the Request_Type and Response_Type parameters is shown in Table 47.

**Table 47 – ACTION service request and response types**

| Request type | | Response type | |
|---|---|---|---|
| NORMAL | The reference of a single method and the complete method invocation parameter is sent. | NORMAL | The result and the complete return parameter are sent. |
| | | ONE-BLOCK | One block of the result and of the return parameter is sent. |
| NEXT | The next data block is requested. | ONE-BLOCK | As above. |
| | | LAST-BLOCK | The last block of the result(s) and of the return parameter(s) is sent. |
| FIRST-BLOCK | The reference of a single method and the first block of the method invocation parameters is sent. | NEXT | Correct reception of the block is acknowledged. |
| ONE-BLOCK | One block of the method invocation parameters is sent. | | |
| LAST-BLOCK | The last block of the method invocation parameters is sent. | NORMAL | As above. |
| | | ONE-BLOCK | As above. |
| WITH-LIST | The reference of a list of methods and the complete list of method invocation parameters is sent. | WITH-LIST | The complete list of results and return parameters is sent. |
| | | ONE-BLOCK | See above. |
| WITH-LIST-AND-FIRST-BLOCK | The reference of a list of methods and the first block of the method invocation parameters is sent. | NEXT | The correct reception of the block is acknowledged. |
| NOTE The same Response_Type can be present more than once, to show the possible responses to each request. | | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The COSEM_Method_Descriptor parameter references a COSEM object method. It is present if Request_Type == NORMAL, FIRST-BLOCK, WITH-LIST and WITH-LIST-AND-FIRST-BLOCK. It is a composite parameter:

- the (COSEM_Class_Id, COSEM_Object_Instance_Id) doublet non-ambiguously references one and only one COSEM object instance;
- the COSEM_Method_Id identifies one method of the COSEM object referenced.

An ACTION-REQUEST-NORMAL or ACTION-REQUEST-FIRST-BLOCK service primitive shall contain a single COSEM method descriptor. An ACTION-REQUEST-WITH-LIST or ACTION-REQUEST-WITH-LIST-AND-FIRST-BLOCK service primitive shall contain a list of COSEM method descriptors; their number is limited by the server-max-receive-pdu-size: all COSEM method references – together with (a part of) the method invocation parameters – shall fit in a single APDU.

The Method_Invocation_Parameter parameter carries the parameter(s) necessary for the invocation of the method(s) referenced.

- if Request_Type == NORMAL, the Method_Invocation_Parameter parameter is optional;
- if Request_Type == WITH-LIST, the service primitive shall contain a list of Method_Invocation_Parameters. The number and the order of the method invocation parameters shall be the same as that of the COSEM_Method_Descriptor-s. If the invocation of any of the methods does not require additional parameters, it shall be nevertheless present, but it shall be null data.

If the encoded form of the COSEM method descriptor(s) and method invocation parameter(s) does not fit in a single APDU, it can be transported in blocks using either the service-specific or the general block transfer mechanism.

If the service-specific block transfer mechanism is used the DataBlock_SA parameter carries block transfer control information and raw-data:

- the Last_Block element indicates whether the current block is the last one (TRUE) or not (FALSE);
- the Block_Number element carries the number of the actual block sent;
- the Raw_Data element carries a part of the method invocation parameters.

The Action_Response parameters are present in the .response primitive when Response_Type == NORMAL, or WITH-LIST. Their number and the order shall be the same as that of the of COSEM method descriptors. It consists of two elements:

- the Result parameter. It contains either the information "success" or a reason for failing to invoke the method referenced (Action-Result);
- the Response_Parameter(s). Each response parameter shall contain either the Data returned as a result of invoking the method, or a reason for returning the parameters to fail (Data-Access-Result). If the invocation of any of the methods does not return parameters, null data shall be returned.

If the response does not fit in a single APDU, it can be transported in blocks using either the service-specific or the general block transfer mechanism.

If the service-specific block transfer mechanism is used, the DataBlock_SA parameter carries block transfer control information and raw-data:

- the Last_Block element indicates whether the current block is the last one (TRUE) or not (FALSE);
- the Block_Number element carries the number of the actual block sent;
- the Raw_Data element carries a part of the response:
  - if a single method was invoked, Raw_Data carries the result and the Response_Parameters if any. If no Response_Parameters are returned, the response shall be of type ACTION-RESPONSE-NORMAL;
  - if a list of methods was invoked, Raw_Data carries a part of the list of Action_Responses and optional data.

The Block_Number parameter in an ACTION-REQUEST-NEXT service primitive shall carry the number of the latest data block received from the server correctly.

The Block_Number parameter in an ACTION-RESPONSE-NEXT service primitive shall carry the number of the latest data block received from the client correctly.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

*Use*

Possible logical sequences of the ACTION service primitives is illustrated in Figure 35:

- for a successful confirmed ACTION, item a);
- for an unconfirmed ACTION, item d); and
- for an unsuccessful attempt due to a local error, item c).

In the first phase, the ACTION.request primitive is invoked by the client AP to invoke one or more methods of one or more COSEM interface objects of the server AP. If the complete list of COSEM method descriptors and method invocation parameters fits in a single APDU, the .request primitive is invoked with Request_Type == NORMAL or WITH-LIST as appropriate. Otherwise, it is invoked with Request_Type == FIRST-BLOCK or WITH-LIST-AND-FIRST-BLOCK as appropriate, then with Request_Type == ONE-BLOCK and finally with LAST-BLOCK. Upon reception of the .request primitive, the client AL builds the Action-Request APDU appropriate for the Request_Type and sends it to the server.

The ACTION.indication primitive is generated by the server AL upon reception of an Action-Request APDU.

During the block transfer of the method invocation parameters, the server AP invokes the ACTION.response primitive with Request_Type == NEXT until the last block is received.

The ACTION.confirm primitive is generated by the client AL upon reception of an Action-Response APDU.

Once all method invocation parameters are transferred, the server invokes the methods of COSEM interface objects referenced, and the second phase commences.

If the complete response fits in a single APDU, the ACTION.response primitive is invoked by the server AP with Response_Type == NORMAL or WITH-LIST, as appropriate. Otherwise, it is invoked with Response_Type == ONE-BLOCK and finally with LAST-BLOCK. Upon reception of the .response primitive, the server AL builds the Action-Response APDU appropriate for the Response_Type and sends it to the client.

The ACTION.confirm primitive is generated by the client AL to indicate the reception of an Action-Response APDU.

During the block transfer of the return parameters, the client AP invokes the .request primitive with Request_Type == NEXT until the last block is received.

The protocol for the ACTION service is specified in 7.3.5.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 6.9 The ACCESS service

### 6.9.1 Overview – Main features

#### 6.9.1.1 General

The ACCESS service is a unified service which can be used to access multiple COSEM object attributes and/or methods with a single .request / .response. The purpose of introducing it is to improve xDLMS messaging while maintaining co-existence with the existing xDLMS services.

#### 6.9.1.2 Unified WITH-LIST service to improve efficiency

The ACCESS service is a unified service using LN referencing that can be used to read or write multiple COSEM object attributes and/or to invoke multiple methods with a single .request / .response. Each request contains a list of requests and related data. Each response contains a list of return data and the result of the request.

NOTE SN referencing is currently not supported. It can be added by introducing new variants of the service.

Whereas GET- / SET- / ACTION-WITH-LIST service requests can include one request type – GET, SET and/or ACTION – only on the list, ACCESS service requests can include different request types. This allows reducing the number of exchanges and thereby improves efficiency.

The processing of the list of requests starts at the first request on the list and continues with processing the next one until the end is reached.

#### 6.9.1.3 Specific variants for selective access

The GET / SET .request service primitives shall always contain Access_Selection_Parameters even in the case when selective access is not available or not needed. In contrast, the ACCESS service provides specific variants to access attributes without or with selective access. This obviates the need to include Access_Selection_Parameters when selective access is not available or not needed thereby reducing overhead and improving efficiency.

#### 6.9.1.4 Long_Invoke_Id parameter

The ACCESS service uses the long_invoke_id parameter.

NOTE Description of the circumstances when long Invoke_Id-s are useful is beyond the Scope of this document.

#### 6.9.1.5 Self-descriptive responses

When requested by the client, the ACCESS.response service primitive carries not only the response to each request, i.e. the result of accessing each attribute / method and the return data but also the Access_Request_Specification service parameter – carrying the attribute / method references and where applicable, the Access_Selection parameters – rendering the .response service primitive self-descriptive. Such self-descriptive responses can be stored and processed on their own, without the need to pair responses and requests.

#### 6.9.1.6 Failure management

In the case of the GET- / SET- / ACTION-WITH-LIST services the client cannot control what should happen if one of the requests fails. In contrast, the ACCESS service allows the client to control if the requests that follow the failed one on the list should be processed or not.

#### 6.9.1.7 Time stamp as a service parameter

The xDLMS services specified earlier do not provide a service parameter in the .request or in the .response service primitive to carry a time stamp.

In contrast, ACCESS service primitives provide a service parameter to carry the time stamp holding the date and time of invoking the service primitive. This further reduces overhead.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 6.9.1.8 Presence of data in service primitives

There are important differences between the GET / SET / ACTION services and the ACCESS service as regards to data in the service primitives:

- GET service: data is not present in the request. In the response, either data or result (Data-Access-Result) is returned;
- SET service: data is present in the request. In the response only result (Data-Access-Result) is returned;
- ACTION service: method invocation parameters are optional in the request. In the response the result of invoking the method (Action-Result) and optionally the result of returning the return parameters (Data or Data-Access-Result) is returned;
- ACCESS service: data is associated with each attribute / method reference in the request. If data is not needed for a particular request, then null-data is included. In the response, both data and result are returned. If there is no data to return for a particular response, then null-data is included. In the case of accessing a method, Access-Response-Action (Action-Result) conveys both the result of invoking the method and the result of returning the return parameters.

### 6.9.2 Service specification

*Function*

The ACCESS service is a unified service using LN referencing that can be used to read or write multiple COSEM object attributes and/or to invoke multiple methods with a single .request / .response. Each request contains a list of requests and related data. Each response contains a list of return data and the result of the request. It can be invoked in a confirmed or unconfirmed manner. It can be used with the general block transfer and general ciphering mechanisms.

The use of the conformance block is the following:

- bit 17 *access* indicates the support of the ACCESS service;
- bit 1 *general-protection* indicates the availability of the general protection APDUs;
- bit 2 *general-block-transfer* indicates the availability of the GBT mechanism;
- bit 8 *attribute0-supported-with-set* and bit 10 *attribute0-supported-with-get* (10) are not relevant: attribute0 is always supported;
- bit 9 priority-mgmt-supported is relevant;
- bit 14 *multiple-references* is irrelevant: the ACCESS service always supports multiple references;
- bit 21 *selective-access* is relevant. The access selection parameters can be used only if the use of selective access has been successfully negotiated.

*Semantics of the service primitives*

The ACCESS service primitives shall provide parameters as shown in Table 48.

The Long_Invoke_Id, Self_Descriptive, Processing_Option, Service_Class and Priority parameters are mandatory. Their value in the .indication, .response and .confirm service primitives shall be the same as in the .request service primitive. They are carried by the bits of the long-invoke-id-and-priority field of the access-request / access-response APDU:

- long-invoke-id (bits 0-23) identifies the instance of the service invocation;
- self-descriptive (bit 28) indicates if the service response shall be not self-descriptive (FALSE) or self-descriptive (TRUE). When set to TRUE, the Access_Response_Body parameter shall contain the Access_Request_Specification parameter;

NOTE 1 The Access_Request_List_Of_Data parameter is not included in the .response service primitive.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- processing-option (bit 29) specifies what to do when processing a request on the list fails. When set to FALSE, processing continues. When set to TRUE, processing breaks i.e. the requests on the list that follow the failed one shall not be processed. As described in 6.9.1.2, processing of the list of requests shall start at the first request on the list and shall continue with processing the next one until the end of the list is reached;
- service-class (bit 30) indicates whether the service invocation is confirmed (TRUE) or unconfirmed (FALSE);

NOTE 2 The Service_Class parameter applies to the service invocation, not to the individual requests on the list.

NOTE 3 Depending on the communication profile, the Service_Class parameter may also determine the frame type to be used to carry the APDU.

- priority (bit 31) indicates the priority level associated to the instance of the service invocation. It may be normal (FALSE) or high (TRUE).

NOTE 4 The Priority parameter applies to the service invocation, not to the individual requests on the list.

The Date_Time service parameter is optional. When present, it shall contain the date and time of the invocation of the service .request / .response. It is carried by the date-time field – of type OCTET STRING – of the access-request / access-response APDU. When not present, then the OCTET STRING shall be of length 0. Unless otherwise specified in a project specific companion specification, the Date_Time parameter in the response shall be present if it was present in the request and shall not be present in the response if it was not present in the request.

**Table 48 – Service parameters of the ACCESS service**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Long_Invoke_Id | M | M (=) | M (=) | M (=) |
| Self_Descriptive | M | M (=) | M (=) | M (=) |
| Processing_Option | M | M (=) | M (=) | M (=) |
| Service_Class | M | M (=) | M (=) | M (=) |
| Priority | M | M (=) | M (=) | M (=) |
| Date_Time | U | U (=) | U | U (=) |
| Access_Request_Body | M | M (=) | – | – |
| Access_Request_Specification | M | M (=) | – | – |
| { Access_Request_Specification } | | | | |
| Access_Request_Get | U | U (=) | – | – |
| COSEM_Attribute_Descriptor | M | M (=) | – | – |
| Access_Request_Set | U | U (=) | – | – |
| COSEM_Attribute_Descriptor | M | M (=) | – | – |
| Access_Request_Action | U | U (=) | – | – |
| COSEM_Method_Descriptor | M | M (=) | – | – |
| Access_Request_Get_With_Selection | U | U (=) | – | – |
| COSEM_Attribute_Descriptor | M | M (=) | – | – |
| Access_Selection | M | M (=) | – | – |
| Access_Selector | M | M (=) | – | – |
| Access_Parameters | M | M (=) | – | – |
| Access_Request_Set_With_Selection | U | U (=) | – | – |
| COSEM_Attribute_Descriptor | M | M (=) | – | – |
| Access_Selection | M | M (=) | – | – |
| Access_Selector | M | M (=) | – | – |
| Access_Parameters | M | M (=) | – | – |
| Access_Request_List_Of_Data | M | M (=) | – | – |
| Data { Data } | | | – | – |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Access_Response_Body | – | – | M | M (=) |
| Access_Request_Specification | – | – | C (=) [1] | C (=) |
| { Access_Request_Specification } | | | | |
| Access_Response_List_Of_Data | – | – | M | M (=) |
| Data { Data } | | | | |
| Access_Response_Specification | – | – | M | M (=) |
| { Access_Response_Specification} | | | | |
| Access_Response_Get | – | – | C | C (=) |
| Result | – | – | M | M (=) |
| Access_Response_Set | – | – | C | C (=) |
| Result | – | – | M | M (=) |
| Access_Response_Action | – | – | C | C (=) |
| Result | – | – | M | M (=) |

[1] When the Access_Request_Specification service parameter is present in Access_Response_Body, then its value shall be the same as in the .request / .indication primitive.

The Access_Request_Body parameter contains the Access_Request_Specification and the Access_Request_List_Of_Data sub-parameters.

The Access_Request_Specification parameter carries a list of request specifications. The list may have 0 or more elements. Each request can be any of the following:

Without selective access:

- Access_Request_Get carries the COSEM_Attribute_Descriptor of an attribute to be read;
- Access_Request_Set carries the COSEM_Attribute_Descriptor of an attribute to be written;
- Access_Request_Action carries the COSEM_Method_Descriptor of a method to be invoked.

With selective access:

- Access_Request_Get_With_Selection carries the COSEM_Attribute_Descriptor of an attribute to be read with selective access, and the Access_Selection parameter that contains Access_Selector and Access_Parameters;
- Access_Request_Set_With_Selection carries the COSEM_Attribute_Descriptor of an attribute to be written with selective access, and the Access_Selection parameter that contains Access_Selector and Access_Parameters.

Access_Request_Get_ / Set_ With_Selection should not be used if selective access to the attribute is not available or if COSEM_Attribute_Descriptor identifies all attributes (Attribute_0).

The COSEM_Attribute_Descriptor parameter is a composite parameter:

- the (COSEM_Class_Id, COSEM_Object_Instance_Id) doublet non-ambiguously references one and only one COSEM object instance;
- the COSEM_Object_Attribute_Id element identifies the attribute of the object instance. COSEM_Object_Attribute_Id == 0 references all public attributes of the object.

The Access_Selection parameter carries the Access_Selector and the Access_Parameters sub-parameters. The possible values are defined in the relevant COSEM interface class definitions.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The Access_Request_List_Of_Data parameter carries the list of data related to the list of Access_Request_Specification parameters. The data depend on the kind of access request:

- Access_Request_Get referencing one or all attributes(Attribute_0): null-data;
- Access_Request_Set: the corresponding data carries the value to be written. In the case of referencing all attributes (Attribute_0), the data shall be a structure. The number of elements in the structure shall be the same as the number of attributes specified in the relevant COSEM IC specification. Each element in the structure contains the value to be written or null-data meaning that the given attribute need not be written;
- Access_Request_Action: the corresponding data carries the method invocation parameters, or if not required null-data.

The number and order of the elements on the two lists shall be the same.

The Access_Response_Body parameter contains the following sub-parameters:

- the Access_Request_Specification (optionally, only when the Self_Descriptive == TRUE);
- the Access_Response_List_Of_Data; and
- the Access_Response_Specification.

Notice that in the response Access_Request_List_Of_Data comes first followed by Access_Response_Specification. If data is provided but the result indicates a failure, then the client should discard the data.

The Access_Request_Specification parameter, when present, shall be the same as in the .request / .indication primitives.

The Access_Response_List_Of_Data parameter carries the data resulting from processing the requests. The number of elements on the list shall be the same as on the Access_Request_Specification list. The data depend on the kind of access request:

- Access_Request_Get referencing a single attribute: the value of the attribute requested or null-data when the value of the attribute cannot be returned;
- Access_Request_Get referencing all attributes (Attribute_0): data shall be a structure, containing the value of each attribute. The number of elements in the structure shall be the same as the number of attributes specified in the relevant COSEM IC specification. If the value of an attribute cannot be returned, then null-data shall be included for that attribute. If no attribute values can be returned then a single null-data shall be returned;
- Access_Request_Set referencing one or all attributes (Attribute_0): null-data;
- Access_Request_Action: the return parameters or when not provided, null-data.

The Access_Response_Specification parameter carries the result of each request. The number of elements on this list shall be the same as on the Access_Request_Specification list:

- Access_Request_Get referencing a single attribute: the result of reading the attribute: *success* or reason for the failure;
- Access_Request_Get referencing all attributes (Attribute_0): the result shall be *success* if the value of all attributes to which access right is granted could be returned. Otherwise, it shall be *Data-Access-Result* giving a reason for the failure;
- Access_Request_Set referencing a single attribute: the result of writing the attribute: *success* or a reason for the failure;
- Access_Request_Set referencing all attributes (Attribute_0): the result shall be *success* if the value of all attributes to which access right is granted could be written. Otherwise, it shall be *Data-Access-Result* giving a reason for the failure;

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- Access_Response_Action carries the result of invoking a method: *success* or a reason for the failure. The result shall be *success* if the method could be invoked successfully and – when the IC specification specifies return parameters – they could be successfully returned. Otherwise, it shall be *Action-Result* giving a reason for the failure.

If the Processing_Option parameter is set to TRUE and processing a request on the list fails, then for all requests following the failed one, Access_Response_List_Of_Data shall contain null-data and Access_Response_Specification shall carry the reason for the failure.

*Use*

Possible logical sequences of the ACCESS service primitives are illustrated in Figure 35:

- for a successful confirmed ACCESS, item a);
- for an unconfirmed ACCESS item d); and
- for an unsuccessful attempt due to a local error item c).

The ACCESS.request primitive is invoked by the client AP to read or write the value of a list of COSEM object attributes and/or to invoke a list of methods.

The ACCESS.indication primitive is generated by the server AL upon the reception of an Access-Request APDU.

The ACCESS.response primitive is invoked by the server AP – if Service_Class == Confirmed – to send a response to an .indication primitive received.

The ACCESS.confirm primitive is generated by the client AL to indicate the reception of an access-response APDU.

When the request or the response does not fit in a single APDU, then the general block transfer mechanism can be used. See 4.2.4.4.9.

If the response would be too long to fit in a single APDU but GBT is not supported, the response may be either a list of null-data and a list of results indicating the reason for the failure.

When cryptographic protection is required, the access-request / access-response APDUs can be transported in general-ded-ciphering, general-glo-ciphering, general-ciphering or general-signing APDUs depending on the kind of protection to be applied. See 6.5

The protocol of the ACCESS service is specified in 7.3.6.

### 6.10 The DataNotification service

*Function*

The DataNotification service is an unsolicited, unconfirmed or confirmed service. It is used by the server to push data to the client. It is an unconfirmed service. The push process is configured by "Push setup" objects; see IEC 62056-6-2:2021, 4.4.8.

*Semantics of the service primitives*

The DataNotification service primitives shall provide parameters as shown in Table 49.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 49 – Service parameters of the DataNotification service primitives**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Long_Invoke_Id | M | M (=) | M (=) | M (=) |
| Self_Descriptive | – | – | – | – |
| Processing_Option | – | – | – | – |
| Service_Class | M | M (=) | M (=) | M (=) |
| Priority | M | M (=) | – | – |
| Date_Time | U | U (=) | U | U (=) |
| Notification_Body | M | M (=) | – | – |
| Result | - | - | - | M |

The Long_Invoke_Id parameter identifies the instance of the service invocation.

The Self_Descriptive, Processing_Option and Service_Class parameters are not used in the case of this service.

The Service_Class parameter indicates whether the DataNotification.request service primitive is invoked in a confirmed or unconfirmed manner.

The Priority parameter indicates the priority level associated to the instance of the service invocation: normal (FALSE) or high (TRUE).

The optional Date_Time parameter indicates the time at which the DataNotification.request / DataNotification.response service primitive is invoked. When the DataNotification.confirm primitive is invoked as a result of a lower layer confirmation, no Date_Time is provided.

The Date_Time parameter indicates the time at which the DataNotification.request service primitive is invoked. It is octet-string, which may be of length zero when transmitting the Date_Time is not required.

The Notification_Body parameter contains the push data.

The Result parameter indicates the confirmation result and is CONFIRMED in case of Service_Class == Confirmed and received Data-Notification-Confirm APDU; or in case of Service_Class == Unconfirmed and received supporting protocol layer confirmation. The Result parameter is SUPPORTING_LAYER_ FAILED in case of supporting protocol layer failure.

The protocol for the DataNotification service is specified in 7.3.7.

*Use*

Possible logical sequence of the DataNotification service primitives are illustrated in Figure 35 a), b), and d).

The .request primitive is invoked by the server AP to push data to the remote client AP. Upon reception of the .request primitive, the server AL builds the DataNotification APDU.

The .indication primitive is generated by the client AL upon reception of a DataNotification APDU.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The DataNotification.response primitive is invoked by the client AP to confirm the receipt of a confirmable DataNotification. Upon the reception of the .response primitive, the client AL builds the Data-Notification-Confirm APDU and sends it to the server.

The DataNotification.confirm primitive is invoked by the server AL to convey the receipt of a Data-Notification-Confirm APDU or lower layer confirmation to the server AP.

The protocol for the DataNotification service is specified in 7.3.6.

### 6.11 The EventNotification service

*Function*

The EventNotification service is an unsolicited, non-client/server type service. It is requested by the server, upon occurrence of an event, in order to inform the client of the value of an attribute, as though it had been requested by the COSEM. It is an unconfirmed service.

*Semantics of the service primitives*

The EventNotification service primitives shall provide parameters as shown in Table 50.

**Table 50 – Service parameters of the EventNotification service primitives**

| | .request | .indication |
|---|---|---|
| Time | U | U (=) |
| Application_Addresses | U | U (=) |
| COSEM_Attribute_Descriptor | M | M (=) |
| COSEM_Class_Id | M | M (=) |
| COSEM_Object_Instance_Id | M | M (=) |
| COSEM_Object_Attribute_Id | M | M (=) |
| Attribute_Value | M | M (=) |

The optional Time parameter indicates the time at which the EventNotification.request service primitive was issued.

The Application_Addresses parameter is optional. It is present only when the EventNotification service is invoked outside of an established AA. In this case, it contains all protocol specific parameters required to identify the sender and destination APs.

When the .request primitive does not contain the optional Application_Addresses parameter, default addresses shall be used, those of the server management logical device and the client management AP. Both APs are always present and in any protocol profile, they are bound to known, pre-defined addresses.

The (COSEM_Class_Id, COSEM_Object_Instance_Id, COSEM_Object_Attribute_Id) triplet references non-ambiguously one and only one attribute of a COSEM interface object instance.

The Attribute_Value parameter carries the value of this attribute. More information about the notified event may be obtained by interrogating this COSEM interface object.

If the encoded form of the request does not fit in a single APDU, it can be transported in data blocks using the general block transfer mechanism.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

*Use*

A possible logical sequence of the EventNotification service primitives is illustrated in Figure 35 f) and g).

The .request primitive is invoked by the server AP to send the value of a COSEM interface object attribute to the remote client AP. Upon reception of the .request primitive, the Server AL builds the EventNotificationRequest APDU.

In some cases, the supporting lower layer protocol(s) do (does) not allow sending a protocol data unit in a real, unsolicited manner. In these cases, the client has to explicitly solicit sending an EventNotification frame, by invoking the Trigger_EventNotification_Sending service primitive.

The EventNotification.indication primitive is generated by the client AL upon reception of an EventNotificationRequest APDU.

The protocol for the EventNotification service is specified in 7.3.8.

### 6.12 The TriggerEventNotificationSending service

*Function*

The function of the TriggerEventNotificationSending service is to trigger the server by the client to send the frame carrying the EventNotification.request APDU.

This service is necessary in the case of protocols, when the server is not able to send a real non-solicited EventNotification.request APDU.

*Semantics of the service primitives*

The TriggerEventNotificationSending.request service primitive shall provide parameters as shown in Table 51.

**Table 51 – Service parameters of the TriggerEventNotificationSending.request service primitive**

| | .request |
|---|---|
| Protocol_Parameters | M |

The Protocol_Parameters parameter contains all lower protocol layer dependent information, which is required for triggering the server to send out an eventually pending frame containing an EventNotification.request APDU. This information includes the protocol identifier, and all the required lower layer parameters.

*Use*

Upon reception of a TriggerEventNotificationSending.request service invocation from the client AP, the client AL shall invoke the corresponding supporting layer service to send a trigger message to the server.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 6.13 Variable access specification

Variable_Access_Specification is a parameter of the xDLMS Read / Write / UnconfirmedWrite InformationReport .request / .indication service primitives. Its variants are shown in Table 52:

- Variable_Name identifies a DLMS® named variable;
- Parameterized_Access provides the capability to transport additional parameters;
- Block_Number_Access transports a block number;
- Read_Data_Block_Access transports block transfer control information and raw data;
- Write_Data_Block_Access transports block transfer control information.

The use of the different variants depends on the service and it is described in the respective SN service specifications.

**Table 52 – Variable Access Specification**

| Variable_Access_Specification | Read .request | Write .request | Unconfirmed Write request | Information Report |
|---|---|---|---|---|
| Kind_Of_Access | M | M | M | M |
| Variable_Name | S | S | S | M |
| Detailed_Access | Not used in DLMS®/COSEM | | | |
| Parameterized_Access | S | S | S | – |
| Variable_Name | M | M | M | |
| Selector | U | U | U | |
| Parameter | U | U | U | |
| **Block_Number_Access** | S | – | – | – |
| Block_Number | M | | | |
| **Read_Data_Block_Access** | S | – | – | – |
| Last_Block | M | | | |
| Block_Number | M | | | |
| Raw_Data | M | | | |
| **Write_Data_Block_Access** | – | S | – | – |
| Last_Block | | M | | |
| Block_Number | | M | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 6.14 The Read service

*Function*

The Read service is used with SN referencing. It is a confirmed service. Its functions are:

- to read the value of one or more COSEM interface object attributes. In this case, the encoded form of the .request service primitive shall fit in a single APDU. The result can be delivered in a single response, or – if it is too long to fit in a single response – in multiple responses, with block transfer;
- to invoke one or more COSEM interface object methods, when return parameters are expected. In this case, if either the .request (including the method references and the method invocation parameters) or the .response service primitive (including the results and return parameters) is too long to fit in a single APDU, then block transfer with multiple requests and/or responses can be used.

The Read service is specified in IEC 61334-4-41:1996, 10.4 and Annex A. For completeness and for consistency with the specification of services using LN referencing, the specification is reproduced here, together with the extensions made for DLMS®/COSEM.

*Semantics of the service primitives*

The Read service primitives shall provide service parameters as shown in Table 53.

**Table 53 – Service parameters of the Read service**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Variable_Access_Specification<br>{ Variable_Access_Specification } | M | M (=) | – | – |
| Variable_Name | S | S (=) | | |
| Parameterized_Access | S | S (=) | | |
| Variable_Name | M | M (=) | | |
| Selector | U | U (=) | | |
| Parameter | U | U (=) | | |
| **Read_Data_Block_Access** | S | S (=) | | |
| Last_Block | M | M (=) | | |
| Block_Number | M | M (=) | | |
| Raw_Data | M | M (=) | | |
| **Block_Number_Access** | S | S (=) | | |
| Block_Number | M | M (=) | | |
| Result (+) | | | S | S (=) |
| Read_Result { Read_Result } | – | – | M | M (=) |
| Data | | | S | S (=) |
| Data_Access_Error | | | S | S (=) |
| **Data_Block_Result** | | | S | S (=) |
| Last_Block | | | M | M (=) |
| Block_Number | | | M | M (=) |
| Raw_Data | | | M | M (=) |
| **Block_Number** | | | S | S (=) |
| Result (–) | – | – | S | S (=) |
| Error_Type | | | M | M (=) |
| NOTE For security parameters, see Table 40. | | | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The use of the different variants of the Variable-Access-Specification service parameter of the Read.request service primitive and the different choices of the Read.response primitive are shown in Table 54.

If the encoded form of the response does not fit in a single APDU, it can be transported in data blocks using the general block transfer mechanism.

**Table 54 – Use of the Variable_Access_Specification variants and the Read.response choices**

| Read.request Variable_Access_Specification | | Read.response CHOICE | |
|---|---|---|---|
| Variable_Name (Variable_Name} | References a list[1] of COSEM object attributes. | Data {Data} | Delivers the value of the attribute(s) referenced. |
| | | Data_Access_Error {Data_Access_Error} | Provides the reason for the read to fail. |
| | | Data_Block_Result | Delivers block transfer control information and one block of raw data. |
| Parameterized_Access {Parameterized_Access} | References a list[1] of COSEM object attributes to be read selectively. | Data {Data} | As above. |
| | | Data_Access_Error {Data_Access_Error} | |
| | | Data_Block_Result | |
| | References a list[1] of COSEM object methods, with method invocation parameters. | Data {Data} | Delivers the method invocation return parameters.<br>NOTE 1 If parameters are returned, this implies that the method invocation succeeded. |
| | | Data_Access_Error {Data_Access_Error} | Provides the reason for the method invocation to fail. |
| | | Data_Block_Result | As above. |
| Read_Data_Block_ Access | Carries block transfer control information and one part of encoded form of the COSEM method references and method invocation parameters. | Block_Number | Carries the number of the latest data block received. |
| Block_Number_Access | Carries the number of the latest data block received. | Data_Block_Result | As above. |

NOTE 2 The same Read.response choice can be present more than once, to show the possible responses to each request.

[1] A list may have one or more elements.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The Read.request service primitive may have one or more Variable_Access_Specification parameters.

- the Variable_Name variant is used to reference a complete COSEM object attribute to be read. The request may include one or more variable names;
- the Parameterized_Access variant is used either:
  - to reference a COSEM object attribute to be read selectively. In this case, the Variable_Name element references the COSEM object attribute, the Selector and the Parameter elements carry the access selector and the access parameters respectively as specified in the attribute specification; or
  - to reference a COSEM object method to be invoked. In this case, the Variable_Name element references the method, the Selector element is zero and the Parameter element carries the method invocation parameters (if any) or null data;
  - the request may include one or more parameterized access parameters;

NOTE 1 With this, the Read service can transport information in both directions, just like the ACTION service used with LN referencing: method invocation parameters from the client to the server and return parameters from the server to the client.

- the Read_Data_Block_Access variant is used when one or more COSEM object methods are invoked and the encoded form of the request does not fit in a single APDU. The request may include a single Read_Data_Block_Access parameter. It carries block transfer control information and raw data:
  - the Last_Block element indicates if the given block is the last one (TRUE) or not (FALSE);
  - the Block_Number element carries the number of the actual block sent;
  - the Raw_Data element carries a part of the encoded form of the list of Variable_Access_Specification parameters (as it would be used without block transfer) including the method references and the method invocation parameters. Here, only the variants Variable_Name and Parameterized_Access are allowed.
- the Block_Number_Access variant is used when the server uses block transfer to send a long response, to confirm the reception of a data block and to request the next data block. The request may include a single Block_Number_Access parameter. It carries the number of the latest data block received correctly.

The Result (+) parameter indicates that the requested service has succeeded.

Without block transfer, the .response / .confirm service primitives contain one or more Read_Result parameters. Their number and order shall be the same as that of the Variable_Name / Parameterized_Access parameters in the .request / .indication primitives.

If the Read service is used to read attribute(s), then:

- the Data choice is taken to carry the value of the attribute at the time of access;
- the Data_Access_Error is taken to carry the reason for the read to fail for this attribute.

If the Read service is used to invoke method(s), then:

- the Data choice is taken to carry the return parameters (if data are returned, this implies that the method invocation succeeded). If there are no return parameters, Data shall be null data;

  NOTE 2 However, if no return data are expected, the Write service is used to invoke methods.
- the Data_Access_Error choice is taken to carry the reason for the method invocation to fail for this method.

In the case of block transfer, the .response / .confirm primitive contains a single Read_Result parameter. The Data_Block_Result choice is taken to carry one block of the response:

- the Last_Block element indicates whether the given block is the last one (TRUE) or not (FALSE);
- the Block_Number element shall carry the number of the block sent;
- the Raw_Data element contains a part of the encoded form of the list of Read_Results.

If the data block cannot be provided, then the .response primitive shall carry a single Result parameter using the Data_Access_Error choice, carrying an appropriate error message, for example (14) data-block-unavailable.

If the block number in the request is not the one expected, or if the next block cannot be delivered, then the Read.response service primitive shall be returned with a single Read_Result parameter, with the choice Data_Access_Error, carrying an appropriate code, for example (19) data-block-number-invalid.

The Block_Number choice is taken when the Read service is used to invoke one or more methods and the request is sent in several blocks, to confirm the correct reception of a data block and to ask for the next block. It carries the number of the latest block received.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The Result (–) parameter indicates that the service previously requested failed. The Error_Type parameter provides the reason for failure. In this case, the server shall send back a ConfirmedServiceError APDU instead of a ReadResponse APDU.

*Use*

A possible logical sequence of the Read service primitives is illustrated in Figure 35 item a).

The Read.request primitive is invoked following the invocation of a GET or ACTION .request primitive by the client AP and mapping this to a Read.request primitive by the SN_MAPPER ASE. The client AL builds then the ReadRequest APDU and sends it to the server. For LN / SN service mapping, see 6.19.

The Read.indication primitive is generated by the server AL upon reception of a ReadRequest APDU.

The Read.response primitive is invoked by the server AP in order to send a response to a previously received Read.indication primitive. The server AL builds then the ReadResponse APDU and sends it to the client.

The Read.confirm primitive is generated by the client AL following the reception of a ReadResponse APDU. It is then mapped back to a GET or ACTION .confirm primitive by the SN_MAPPER ASE and the GET or ACTION .confirm primitive is generated.

The protocol of the Read service is specified in 7.3.9.

### 6.15 The Write service

*Function*

The Write service is used with SN referencing. It is a confirmed service. Its functions are:

- to write the value of one or more COSEM interface object attributes;
- to invoke one or more COSEM interface object methods when no return parameters are expected.

In both cases, if the encoded form of the .request service primitive does not fit in a single APDU, then it can be sent in several requests with block transfer. The .response service primitive shall always fit in a single APDU.

The Write service is specified in IEC 61334-4-41:1996, 10.5 and Annex A. For completeness and for consistency with the specification of services using LN referencing, the specification is reproduced here, together with the extensions made for DLMS®/COSEM.

*Semantics of the service primitives*

The Write service primitives shall provide service parameters as shown in Table 55.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 55 – Service parameters of the Write service**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Variable_Access_Specification<br>{ Variable_Access_Specification } | M | M(=) | – | – |
| Variable_Name | S | S (=) | | |
| Parameterized_Access | S | S (=) | | |
| Variable_Name | M | M (=) | | |
| Selector | M | M (=) | | |
| Parameter | M | M (=) | | |
| Write_Data_Block_Access | S | S (=) | – | – |
| Last_Block | M | M (=) | | |
| Block_Number | M | M (=) | | |
| Data { Data } | M | M (=) | – | – |
| Result (+) | – | – | S | S (=) |
| Write_Result { Write_Result } | - | - | S | S (=) |
| Success | | | S | S (=) |
| Data_Access_Error | | | S | S (=) |
| **Block_Number** | | | S | S (=) |
| Result (–) | | | S | S (=) |
| Error_Type | | | M | M (=) |
| NOTE For security parameters, see Table 40. | | | | |

The use of the different variants of the Variable-Access-Specification service parameter of the Write.request service primitive and the different choices of the Write.response primitive are shown in Table 56. The use of the Data service parameter is also explained.

If the encoded form of the request does not fit in a single APDU, it can be transported in data blocks using either the service-specific or the general block transfer mechanism.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 56 – Use of the Variable_Access_Specification variants and the Write.response choices**

| Write.request Variable_Access_Specification | | Write.response CHOICE | |
|---|---|---|---|
| Variable_Name {Variable_Name} | References a list[1] of COSEM object attributes.<br>The Data service parameter carries the data to be written or the method invocation parameter(s). | Success {Success} | Indicates that the attribute referenced could be successfully written. |
| | | Data_Access_Error {Data_Access_Error} | Provides the reason for the write to fail. |
| Parameterized_Access {Parameterized_Access} | References a list[1] of COSEM object attributes to be written selectively.<br>The Data service parameter carries the data to be written. | Success {Success} | As above. |
| | | Data_Access_Error {Data_Access_Error} | |
| Write_Data_Block_Access | Carries block transfer control information.<br>The Data service parameter carries raw-data, including the encoded form of the list[1] of COSEM object attribute or method references, and the list of data to be written or the list of method invocation parameters. | Block_Number | Carries the number of the latest data block received. |
| NOTE The same Write.response choice can be present more than once, to show the possible responses to each request.<br>[1] A list may have one or more elements. | | | |

The Write.request service primitive may have one or more Variable_Access_Specification parameters:

- the Variable_Name variant is used to reference a complete COSEM object attribute to be written or COSEM object method to be invoked. The request may include one or more variable names;
- the Parameterized_Access variant is used to reference a COSEM object attribute to be written selectively. In this case, the Variable_Name element references the COSEM object attribute, the Selector and the Parameter elements carry the access selector and the access parameters respectively as specified in the attribute specification. The request may include one or more Parameterized_Access parameters;

The Data service parameter carries the value(s) to be written to the attribute(s), or the method invocation parameter(s) of the method(s) to be invoked. The number and the order of the Data parameters shall be the same as that of the Variable_Access_Specification parameters.

If the Write.request service primitive does not fit into a single APDU, block transfer may be used. In this case:

- the Write_Data_Block_Access variant of the Variable_Access_Specification carries block transfer control information:
  - the Last_Block element indicates whether the given block is the last one (TRUE) or not (FALSE);
  - the Block_Number element carries the number of the actual block sent;
- the Data parameter carries one part of the list of the attribute references and the list of data to be written, or one part of the list of method references and the list of method invocation parameters.
- The request includes a single Write_Data_Block_Access and a single Data parameter.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The Result (+) parameter indicates that the service requested has succeeded.

The .response / .confirm service primitives contain a list of Write_Result parameters. Their number and order shall be the same as that of the Variable_Name / Parameterized_Access parameters in the .request / .indication service primitives.

Without block transfer, and with block transfer after receiving the last block:

- when the Write service is used to write attribute(s), each element carries either the success of the write access (Success) or a reason for the write to fail for this variable (Data_Access_Error);
- when the Write service is used to invoke method(s), each element carries either the success of the method invocation access (Success) or a reason for the method invocation to fail for this variable (Data_Access_Error).

The Block_Number choice is used during block transfer to confirm the correct reception of a data block and to ask for the next block. It carries the number of the latest block received.

If the block-number in the request is not the one expected, or if the block could not be received correctly, then the Write.response service primitive shall be returned with a single Write_Result parameter, with the choice Data_Access_Error, carrying an appropriate code, for example (19) data-block-number-invalid.

The Result (–) parameter indicates that the service requested has failed. The Error_Type parameter provides the reason for failure. In this case, the server shall send back a ConfirmedServiceError APDU instead of a WriteResponse APDU.

*Use*

A possible logical sequence of the Write service primitives is illustrated in Figure 35 item a).

The Write.request primitive is invoked following the invocation of a SET or ACTION .request primitive by the client AP and mapping this to a Write.request primitive by the SN_MAPPER ASE. The client AL builds then the WriteRequest APDU and sends it to the server. For LN / SN service mapping, see 6.19.

The Write.indication primitive is generated by the server AL upon reception of a WriteRequest APDU.

The Write.response primitive is invoked by the server AP in order to send a response to a previously received Write.indication primitive. The server AL builds then the WriteResponse APDU and sends it to the client.

The Write.confirm primitive is generated by the client AL following the reception of a WriteResponse APDU. It is mapped then back to a SET or ACTION .confirm primitive by the SN_MAPPER ASE and the SET or ACTION .confirm primitive is generated.

The protocol of the Write service is specified in 7.3.10.

### 6.16 The UnconfirmedWrite service

*Function*

The UnconfirmedWrite service is used with SN referencing. It is an unconfirmed service. Its functions are:

- to write the value of one or more COSEM object attributes;

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- to invoke one or more COSEM interface object method when no return parameters are expected.

The UnconfirmedWrite.request service primitive shall always fit in a single APDU.

The UnconfirmedWrite service is specified in IEC 61334-4-41:1996, 10.6 and Annex A. For completeness and for consistency with the specification of services using LN referencing, the specification is reproduced here, together with the extensions made for DLMS®/COSEM.

*Semantics of the service primitives*

The UnconfirmedWrite service primitives shall provide service parameters as shown in Table 57.

**Table 57 – Service parameters of the UnconfirmedWrite service**

| | .request | .indication |
|---|---|---|
| Variable_Access_Specification<br>{ Variable_Access_Specification } | M | M(=) |
| Variable_Name | S | S (=) |
| Parameterized_Access | S | S (=) |
| Variable_Name | M | M (=) |
| Selector | M | M (=) |
| Parameter | M | M (=) |
| Data { Data } | M | M (=) |
| NOTE For security parameters see Table 40. | | |

The use of the different variants of the Variable-Access-Specification service parameter of the UnconfirmedWrite.request service primitive is shown in Table 58. The use of the Data service parameter is also explained.

If the encoded form of the request does not fit in a single APDU, it can be transported in data blocks using the general block transfer mechanism.

**Table 58 – Use of the Variable_Access_Specification variants**

| UnconfirmedWrite.request Variable_Access_Specification | |
|---|---|
| Variable_Name<br>{Variable_Name} | References a COSEM object attribute.<br>The Data service parameter carries the data to be written or the method invocation parameter(s). |
| Parameterized_Access<br>{Parameterized_Access} | References a COSEM object attribute with selective access.<br>The Data service parameter carries the data to be written. |

The UnconfirmedWrite.request service primitive may have one or more Variable_Access_Specification parameters.

- the Variable_Name variant is used to reference a complete COSEM object attribute to be written or COSEM object method to be invoked;
- the Parameterized_Access variant is used to reference a COSEM object attribute to be written selectively. In this case, the Variable_Name element references the COSEM object attribute, the Selector and the Parameter elements carry the access selector and the access parameters respectively as specified in the attribute specification.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The Data service parameter carries the value(s) to be written to the attribute(s), or the method invocation parameter(s) of the method(s) to be invoked. The number and the order of the Data parameters shall be the same as that of the Variable_Access_Specification parameters.

*Use*

A possible logical sequence of the Write service primitives is illustrated in Figure 35 item d).

The UnconfirmedWrite.request primitive is invoked following the invocation of a SET or ACTION .request primitive with Service_Class == Unconfirmed by the client AP and mapping this to an UnconfirmedWrite.request primitive by the SN_MAPPER ASE. The client AL builds then the UnconfirmedWriteRequest APDU and sends it to the server.

The UnconfirmedWrite.indication primitive is generated by the server AL upon reception of a WriteRequest APDU.

The protocol of the UnconfirmedWrite service is specified in 7.3.11.

### 6.17 The InformationReport service

*Function*

The InformationReport service is an unsolicited, non-client/server type service. It is requested by a server using SN referencing, upon occurrence of an event, in order to inform the client of the value of one or more DLMS® named variables – mapped to COSEM interface object attributes – as though they had been requested by the client. It is an unconfirmed service.

The InformationReport service is specified in IEC 61334-4-41:1996, 10.7 and Annex A. For completeness and for consistency with the specification of services using LN referencing, the specification of the InformationReport service is reproduced here, together with the extensions made for DLMS®/COSEM.

*Semantics of the service primitives*

The InformationReport service primitives shall provide parameters as shown in Table 59.

**Table 59 – Service parameters of the InformationReport service**

| | .request | .indication |
|---|---|---|
| Current_Time | M | M (=) |
| Variable_Access_Specification<br>{ Variable_Access_Specification }<br>Variable_Name | M<br><br>M | M (=)<br><br>M (=) |
| Data { Data } | M | M(=) |

The Current_Time parameter indicates the time at which the InformationReport.request service primitive was issued.

The Variable_Access_Specification parameter of choice Variable_Name specifies one or more DLMS® named variables – mapped to COSEM interface object attributes – the value of which is sent by the server.

The Data parameter carries the value of the DLMS® named variable(s), in the same order as the order of the Variable_Access_Specification parameter(s).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The protocol for the InformationReport service is specified in 7.3.12.

### 6.18 Client side layer management services: the SetMapperTable.request

*Function*

The function of the SetMapperTable service is to manage the client SN_MAPPER ASE.

*Semantics of the service primitives*

There is only one primitive, the .request primitive. It shall provide parameters as follows, see Table 60.

**Table 60 – Service parameters of the SetMapperTable.request service primitives**

| | .request |
|---|---|
| Mapping_Table | M |

The Mapping_table parameter is mandatory. It contains the contents of the attribute *object_list* for the requested server and AA. The structure of the content is defined in IEC 62056-6-2:2021.

*Use*

The SetMapperTable.request service is invoked by the client AP to provide mapping information to the client SN_MAPPER ASE. This service does not cause any data transmission between the client and the server. The client AP uses this service primitive, in order to enhance the efficiency of the mapping process if SN referencing is used.

### 6.19 Summary of services and LN/SN data transfer service mapping

Table 61 and Table 62 provide a summary of the DLMS®/COSEM application layer services.

**Table 61 – Summary of ACSE services**

| Client side | Server side |
|---|---|
| COSEM-OPEN.request | COSEM-OPEN.indication |
| COSEM-OPEN.confirm | COSEM-OPEN.response |
| COSEM-RELEASE.request | COSEM-RELEASE.indication |
| COSEM-RELEASE.confirm | COSEM-RELEASE.response |
| COSEM-ABORT.indication | COSEM-ABORT.indication |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 62 – Summary of xDLMS services**

| Client side | Server side |
|---|---|
| **LN referencing** | |
| GET.request | GET.indication |
| GET.confirm | GET.response |
| SET.request | SET.indication |
| SET.confirm | SET.response |
| ACTION.request | ACTION.indication |
| ACTION.confirm | ACTION.response |
| ACCESS.request | ACCESS.indication |
| ACCESS.confirm | ACCESS.response |
| EventNotification.indication | EventNotification.request |
| TriggerEventNotificationSending.request | – |
| DataNotification.indication | DataNotification.request |
| DataNotification.response | DataNotification.confirm |
| **SN referencing** | |
| Read.request | Read.indication |
| Read.confirm | Read.response |
| Write.request | Write.indication |
| Write.confirm | Write.response |
| UnconfirmedWrite.request | UnconfirmedWrite.indication |
| InformationReport.indication | InformationReport.request |
| DataNotification.indication | DataNotification.request |
| DataNotification.response | DataNotification.confirm |
| NOTE The DataNotification service can be can be used in application contexts using either SN referencing or LN referencing. | |

When the server uses SN referencing, a mapping between the service primitives using LN referencing and SN referencing takes place on the client side. This mapping is specified in 7.3.9, 7.3.10, 7.3.11 and 7.3.12.

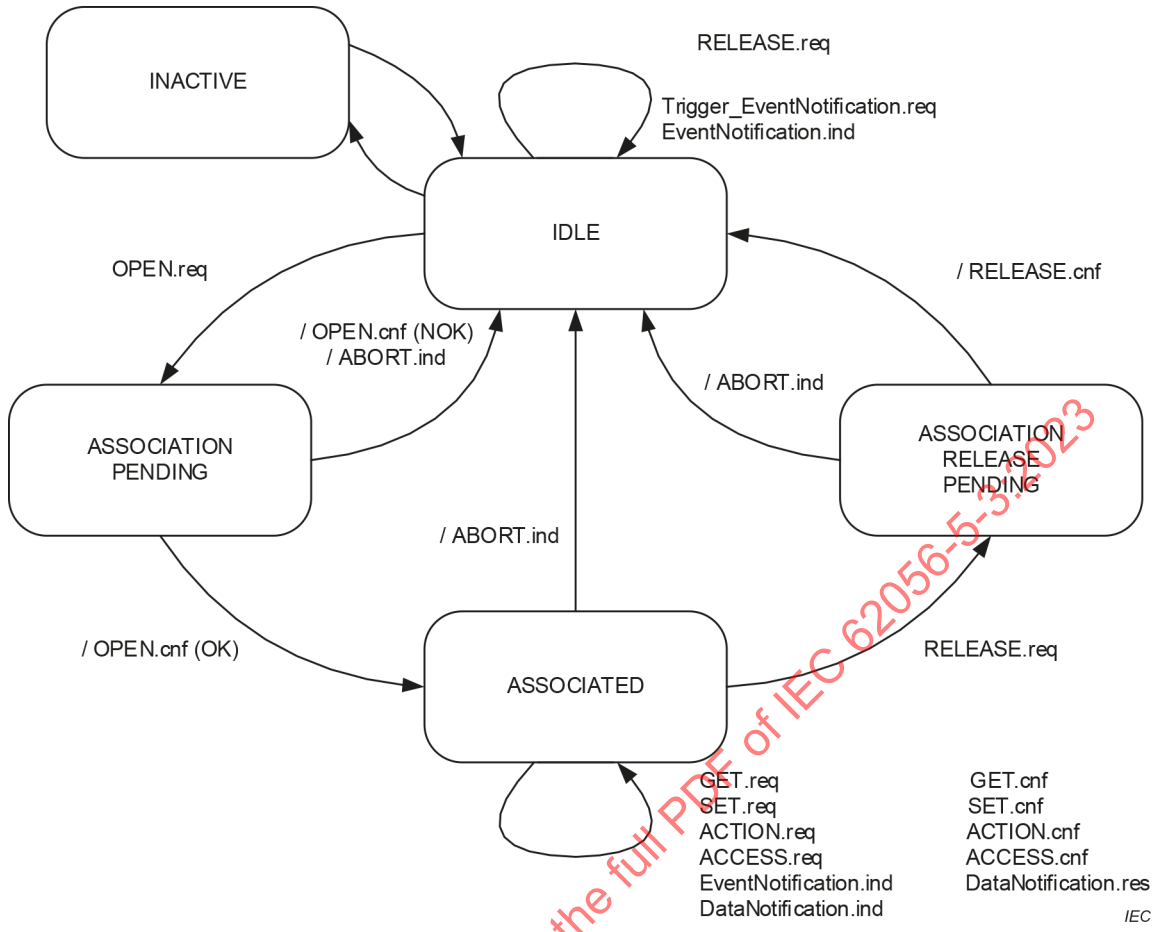
IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# 7 DLMS®/COSEM application layer protocol specification

## 7.1 The control function

### 7.1.1 State definitions of the client side control function

Figure 37 shows the state machine for the client side CF, see also Figure 9.




NOTE On the state diagrams of the client and server CF, the following conventions are used:

– service primitives with no "/" character as first character are "stimulants": the invocation of these primitives is the origin of the state transition;

– service primitives with an "/" character as first character are "outputs": the generation of these primitives is done on the state transition path.

**Figure 37 – Partial state machine for the client side control function**

The state definitions of the client CF – and of the AL including the CF – are as follows:

INACTIVE In this state, the CF has no activity at all: it neither provides services to the AP nor uses services of the supporting protocol layer.

IDLE This is the state of the CF when there is no AA existing, being released, or being established [5]. Nevertheless, some data exchange between the client and server – if the physical channel is already established – is possible. The CF can handle the EventNotification service.

NOTE State transitions between the INACTIVE and IDLE states are controlled outside of the protocol. For example, it can be considered that the CF makes the state transition from INACTIVE to IDLE by being instantiated and bound on the top of the supporting protocol layer. The opposite transition happens by deleting the given instance of the CF.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

5 Note that it is the state machine for the AL: lower layer connections, including the physical connection, are not taken into account. On the other hand, physical connection establishment is done outside of the protocol.

| | |
|---|---|
| ASSOCIATION PENDING | The CF leaves the IDLE state and enters this state when the AP requests the establishment of an AA by invoking the COSEM-OPEN.request primitive (OPEN.req). The CF may exit this state and enter either the ASSOCIATED state or return to the IDLE state, and generates the COSEM-OPEN.confirm primitive, (/OPEN.cnf(OK)) or (/OPEN.cnf(NOK)), depending on the result of the association request. The CF also exits this state and returns to the IDLE state with generating the COSEM-ABORT.indication primitive (/ABORT.ind). |
| ASSOCIATED | The CF enters this state when the AA has been successfully established. All xDLMS services and APDUs are available in this state. The CF remains in this state until the AP requests the release of the AA by invoking the COSEM-RELEASE.request primitive (RELEASE.req). The CF also exits this state and returns to the IDLE state with generating the COSEM-ABORT.indication primitive (/ABORT.ind). |
| ASSOCIATION RELEASE PENDING | The CF leaves the ASSOCIATED state and enters this state when the AP requests the release of the AA by invoking the COSEM-RELEASE.request primitive (RELEASE.req). The CF remains in this state, waiting for the response to this request from the server. As the server is not allowed to refuse a release request, after exiting this state, the CF always enters the IDLE state. The CF may exit this state by generating the COSEM-RELEASE.confirm primitive following the reception of a response from the server or by generating it locally (/RELEASE.cnf). The CF also exits this state and returns to the IDLE state with generating the COSEM-ABORT.indication primitive (/ABORT.ind). |

### 7.1.2 State definitions of the server side control function

Figure 38 shows the state machine for the server side CF, see Figure 9.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

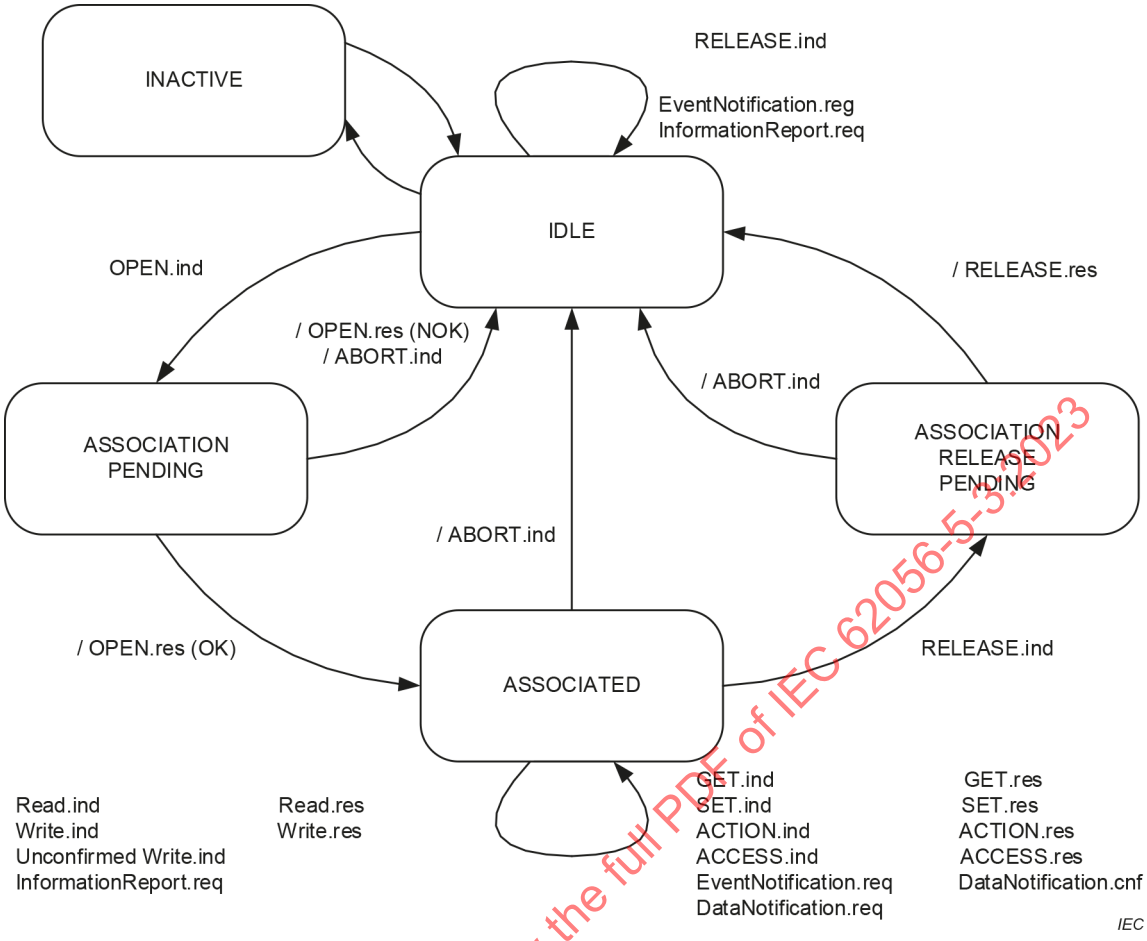



**Figure 38 – Partial state machine for the server side control function**

INACTIVE — In this state, the CF has no activity at all: it neither provides services to the AP nor uses services of the supporting protocol layer.

IDLE — This is the state of the CF when there is no AA existing, being released, or being established [6]. Nevertheless, some data exchange between the client and server – if the physical channel is already established – is possible. The CF can handle the EventNotification / InformationReport services.

ASSOCIATION PENDING — The CF leaves the IDLE state and enters this state when the client requests the establishment of an AA, and the server AL generates the COSEM-OPEN.indication primitive (/OPEN.ind). The CF may exit this state and enter either the ASSOCIATED state or return to the IDLE state, depending on the result of the association request, and invokes the COSEM-OPEN.response primitive, (/OPEN.res(OK)) or (/OPEN.res(NOK)). The CF also exits this state and returns to the IDLE state with generating the COSEM-ABORT.indication primitive (/ABORT.ind).

ASSOCIATED — The CF enters this state when the AA has been successfully established. All xDLMS services and APDUs are available in this state. The CF remains in this state until the client requests the release of the AA, and the server AL generates the COSEM-RELEASE.ind primitive (/RELEASE.ind). The CF also exits this state and returns to the IDLE state with generating the COSEM-ABORT.indication primitive (/ABORT.ind).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

6 Note that it is the state machine for the AL: lower layer connections, including the physical connection, are not taken into account. On the other hand, physical connection establishment is done outside of the protocol.

| | |
|---|---|
| ASSOCIATION RELEASE PENDING | The CF leaves the ASSOCIATED state and enters this state when the client requests the release of an AA, and the server AP receives the COSEM-RELEASE.indication primitive (/RELEASE.ind). The CF remains in this state, waiting that the AP accepts the release request. As the server is not allowed to refuse a release request, after exiting this state, the CF always enters the IDLE state. The CF may exit this state when the AP accepts the release of the AA, and invokes the COSEM-RELEASE.response primitive (RELEASE.res). The CF also exits this state and returns to the IDLE state with generating the COSEM-ABORT.indication primitive (/ABORT.ind). |

## 7.2 The ACSE services and APDUs

### 7.2.1 ACSE functional units, services and service parameters

The DLMS®/COSEM AL ACSE is based on the connection-oriented ACSE, as specified in ISO/IEC 15953 and ISO/IEC 15954.

Functional units are used to negotiate ACSE user requirements during association establishment. Five functional units are defined:

- Kernel functional unit;
- Authentication functional unit;
- ASO-context negotiation functional unit;

  NOTE 1 ISO/IEC 15953:1999 and ISO/IEC 15954:1999 use the term ‘ASO-context". In DLMS®/COSEM the term ‘Application context" is used as in ISO/IEC 8649 / ISO/IEC 8650-1.

- Higher Level Association functional unit; and
- Nested Association functional unit.

The DLMS®/COSEM AL uses only the Kernel and the Authentication functional unit.

The acse-requirements parameters of the AARQ and AARE APDUs are used to select the functional units for the association.

The Kernel functional unit is always available and includes the basic services A-ASSOCIATE, A-RELEASE.

The Authentication functional unit supports authentication during association establishment. The availability of this functional unit is negotiated during association establishment. This functional unit does not include additional services. It adds parameters to the A-ASSOCIATE service.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 63 – Functional Unit APDUs and their fields**

| Functional unit | Service | APDU | Field name | Presence |
|---|---|---|---|---|
| Kernel | A-ASSOCIATE | AARQ | protocol-version | O |
| | | | application-context-name | M |
| | | | called-AP-title | U |
| | | | called-AE-qualifier | U |
| | | | called-AP-invocation-identifier | U |
| | | | called-AE-invocation-identifier | U |
| | | | calling-AP-title | U |
| | | | calling-AE-qualifier | U |
| | | | calling-AP-invocation-identifier | U |
| | | | calling-AE-invocation-identifier | U |
| | | | implementation-information | O |
| | | | user-information 2) | M |
| | | | (carrying an xDLMS Initiate.request APDU) | |
| | | | dedicated-key | U |
| | | | response-allowed | U |
| | | | proposed-quality-of-service | U |
| | | | proposed-dlms-version-number | M |
| | | | proposed-conformance | M |
| | | | client-max-receive-pdu-size | M |
| | | AARE | protocol-version | O |
| | | | application-context-name | M |
| | | | result | M |
| | | | result-source-diagnostic | M |
| | | | responding-AP-title | U |
| | | | responding-AE-qualifier | U |
| | | | responding-AP-invocation-identifier | U |
| | | | responding-AE-invocation-identifier | U |
| | | | implementation-information | O |
| | | | user-information 3) | M |
| | | | (carrying an xDLMS initiateResponse APDU) | S |
| | | | negotiated-quality-of-service | U |
| | | | negotiated-dlms-version-number | M |
| | | | negotiated-conformance | M |
| | | | server-max-receive-pdu-size | M |
| | | | vaa-name | M |
| | | | (or carrying a confirmedServiceError APDU) | S |
| | A-RELEASE | RLRQ | reason | U |
| | | | user-information | U |
| | | RLRE | reason | U |
| | | | user-information | U |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Functional unit | Service | APDU | Field name | Presence |
|---|---|---|---|---|
| Authentication | A-ASSOCIATE | AARQ | sender-acse-requirements | U |
| | | | mechanism-name | U |
| | | | calling-authentication-value | U |
| | | AARE | responder-acse-requirements | U |
| | | | mechanism-name | U |
| | | | responding-authentication-value | U |
| NOTE 1 This table is based on ISO/IEC 15954:1999, Table 2 and 3. The fields are listed in the order as they are in the ACSE APDUs. | | | | |
| M Presence is mandatory<br>O Presence is ACPM option<br>U Presence is ACSE service-user option<br>S The parameter is selected among other S-parameters as internal response of the server ASE environment. | | | | |
| NOTE 2 According to ISO/IEC 15953:1999 the user-information parameter is optional. However, in the DLMS®/COSEM environment it is mandatory in the AARQ / AARE APDUs. | | | | |
| There are several changes in ISO/IEC 15953:1999 and ISO/IEC 15954:1999 compared to ISO/IEC 8649 / ISO/IEC 8650-1:<br>- In ISO/IEC 15954, protocol-version is mandatory in the AARQ and optional in the AARE. In DLMS®/COSEM it is kept as mandatory for backward compatibility;<br>- Instead of "application-context-name", "ASO-context-name" is used. In DLMS®/COSEM, "application-context-name" is kept. ISO/IEC 15954:1999, 7.1.5.2 specifies this: the ASO-context-name is optional. If backward compatibility with older implementations of ACSE is desired, it shall be present. Therefore, in DLMS®/COSEM it is mandatory;<br>- In ISO/IEC 15954, the result and result-source-diagnostic parameters are optional. ISO/IEC 15954:1999, 7.1.5.8 and 7.1.5.9 specifies this: The Result / Result-source-diagnostic are optional. If backward compatibility with older implementations of ACSE is desired, it shall be present. Therefore, in DLMS®/COSEM these parameters are mandatory. | | | | |

Table 63 shows the services, APDUs and APDU fields associated with the ACSE functional units, as used by the DLMS®/COSEM AL. The abstract syntax of the ACSE APDUs is specified in Clause 8.

In general, the value of each field of the AARQ APDU is determined by the parameters of the COSEM-OPEN.request service primitive. Similarly, the value if each field of the AARE is determined by the COSEM-OPEN.response primitive. The COSEM-OPEN service is specified in 6.2.

The fields of the AARQ and AARE APDU are specified below. Managing these fields is specified in 7.2.4.1.

- protocol-version: the DLMS®/COSEM AL uses the default value version 1. For details see ISO/IEC 15954:1999;
- application-context-name**:** COSEM application context names are specified in 7.2.2.2;

  NOTE 2 ISO/IEC 15953:1999 and ISO/IEC 15954:1999 use "ASO-context-name".
- called-, calling- and responding- titles, qualifiers and invocation-identifiers: these optional fields carry the value of the respective parameters of the COSEM-OPEN service. For details see ISO/IEC 15954:1999;
- implementation-information: this field is not used by the DLMS®/COSEM AL. For details see ISO/IEC 15954:1999;

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- user-information**:** in the AARQ APDU, it carries an xDLMS InitiateRequest APDU holding the elements of the Proposed_xDLMS_Context parameter of the COSEM-OPEN.request service primitive. In the AARE APDU, it carries an xDLMS InitiateResponse APDU, holding the elements of the Negotiated_xDLMS_Context parameter, or an xDLMS confirmedServiceError APDU, holding the elements of the xDLMS_Initiate_Error parameter of the COSEM-OPEN.response service primitive;
- sender- and responder-acse-requirements**:** this field is used to select the optional functional units of the AARQ / AARE. In COSEM, only the Authentication functional unit is used. When present, it carries the value of BIT STRING { authentication (0) }. Bit set: authentication functional unit selected;
- mechanism-name**:** COSEM authentication mechanism names are specified in 7.2.2.3;
- calling- and responding- authentication-value**:** see 5.2.2.2;
- result: the value of this field is determined by the COSEM AP (acceptor) or the DLMS®/COSEM AL (ACPM) as specified below. It is used to determine the value of the Result parameter of the COSEM-OPEN.confirm primitive:
  - if the AARQ APDU is rejected by the ACPM (i.e. the COSEM-OPEN.indication primitive is not issued by the DLMS®/COSEM AL), the value "rejected (permanent)" or "rejected (transient)" is assigned by the ACPM;
  - otherwise, the value is determined by the Result parameter of the COSEM-OPEN.response APDU;
- result-source-diagnostic: this field contains both the Result source value and the Diagnostic value. It is used to determine the value of the Failure_Type parameter of the COSEM-OPEN.confirm primitive:
  - Result-source value: if the AARQ is rejected by the ACPM, (i.e. the COSEM-OPEN.indication primitive is not issued by the DLMS®/COSEM AL) the ACPM assigns the value "ACSE service-provider". Otherwise, the ACPM assigns the value "ACSE service-user";
  - Diagnostic value: If the AARQ is rejected by the ACPM, the appropriate value is assigned by the ACPM. Otherwise, the value is determined by the Failure_Type parameter of the COSEM-OPEN.response primitive. If the Diagnostic parameter is not included in the .response primitive, the ACPM assigns the value "null".

The parameters of the RLRQ / RLRE APDUs – used when the COSEM-RELEASE service (see 6.3) is invoked with the parameter Use_RLRQ_RLRE == TRUE – are specified below.

- reason**:** carries the appropriate value as specified in 6.2;
- user-information: if present, it carries an xDLMS InitiateRequest / InitiateResponse APDU, holding the elements of the Proposed_xDLMS_Context / Negotiated_xDLMS_Context parameter of the COSEM-RELEASE.request / .response service primitive respectively. See 6.2.

### 7.2.2 Registered COSEM names

#### 7.2.2.1 General

Within an OSI environment, many different types of network objects shall be identified with globally unambiguous names. These network objects include abstract syntaxes, transfer syntaxes, application contexts, authentication mechanism names, etc. Names for these objects in most cases are assigned by the committee developing the particular basic ISO standard or by implementers' workshops, and should be registered. For DLMS®/COSEM, these object names are assigned by the DLMS® UA, and are specified below.

The decision no. 1999.01846 of OFCOM, Switzerland, attributes the following prefix for object identifiers specified by the DLMS® User Association.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| { joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS®-UA(8) } |
| --- |
| NOTE As specified in ITU-T X.660 A.2.4, for historical reasons, the secondary identifiers ccitt and joint-iso-ccitt are synonyms for itu-t and joint-iso-itu-t, respectively, and thus may appear in ASN.1 OBJECT IDENTIFIER values, and also identify the corresponding primary integer value. |

For DLMS®/COSEM, object identifiers are specified for naming the following items:

- COSEM application context names;
- COSEM authentication mechanism names;
- cryptographic algorithm ID-s.

#### 7.2.2.2 The COSEM application context

In order to effectively exchange information within an AA, the pair of AE-invocations shall be mutually aware of, and follow a common set of rules that govern the exchange. This common set of rules is called the application context of the AA. The application context that applies to an AA is determined during its establishment.

An AA has only one application context. However, the set of rules that make up the application context of an AA may contain rules for alteration of that set of rules during the lifetime of the AA.

The following methods may be used:

- identifying a pre-existing application context definition;
- transferring an actual description of the application context.

In the COSEM environment, it is intended that an application context pre-exists and it is referenced by its name during the establishment of an AA. The application context name is specified as OBJECT IDENTIFIER ASN.1 type. COSEM identifies the application context name by the following object identifier value:

| COSEM_Application_Context_Name::= |
| --- |
| {joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS®-UA(8) application-context(1) context_id(x)} |

The meaning of this general COSEM application context is:

- there are two ASEs present within the AE invocation, the ACSE and the xDLMS ASE;
- the xDLMS ASE is as it is specified in IEC 61334-4-41:1996

NOTE With the COSEM extensions to DLMS®, see 4.2.4.

- the transfer syntax is A-XDR.

The specific context_id-s and the use of ciphered and unciphered APDUs are shown in Table 64:

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 64 – COSEM application context names**

| Application context name | context_id | Unciphered APDUs | Ciphered APDUs |
|---|---|---|---|
| Logical_Name_Referencing_No_Ciphering::= | context_id(1) | Yes | No |
| Short_Name_Referencing_No_Ciphering::= | context_id(2) | Yes | No |
| Logical_Name_Referencing_With_Ciphering::= | context_id(3) | Yes | Yes |
| Short_Name_Referencing_With_Ciphering::= | context_id(4) | Yes | Yes |

In order to successfully establish an AA, the application-context-name parameter of the AARQ and AARE APDUs should carry one of the "valid" names. The client proposes an application context name using the Application_Context_Name parameter of the COSEM-OPEN.request primitive. The server may return any value; either the value proposed or the value it supports.

#### 7.2.2.3 The COSEM authentication mechanism name

Authentication of the client, the server or both is one of the security aspects addressed by the DLMS®/COSEM specification. Three authentication security levels are specified:

- no security (Lowest Level Security) authentication, see 5.2.2.2.2;
- Low Level Security (LLS) authentication, see 5.2.2.2.3;
- High Level Security (HLS) authentication, see 5.2.2.2.4.

DLMS®/COSEM identifies the authentication mechanisms by the following general object identifier value:

```
COSEM_Authentication_Mechanism_Name::=

{joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS®-UA(8)
authentication_mechanism_name(2) mechanism_id(x)}
```

The value of the mechanism_id element selects one of the security mechanisms specified. See Table 65.

**Table 65 – COSEM authentication mechanism names**

| | |
|---|---|
| COSEM_lowest_level_security_mechanism_name::= | mechanism_id(0) |
| COSEM_low_level_security_mechanism_name::= | mechanism_id(1) |
| COSEM_high_level_security_mechanism_name::= | mechanism_id(2) |
| COSEM_high_level_security_mechanism_name_using_MD5::= | mechanism_id(3) |
| COSEM_high_level_security_mechanism_name_using_SHA-1::= | mechanism_id(4) |
| COSEM_high_level_security_mechanism_name_using_GMAC::= | mechanism_id(5) |
| COSEM_high_level_security_mechanism_name_using_SHA-256::= | mechanism_id(6) |
| COSEM_high_level_security_mechanism_name_using_ECDSA::= | mechanism_id(7) |
| NOTE 1 With mechanism_id(2), the method of processing the challenge is secret.<br>NOTE 2 The use of authentication mechanisms 3 and 4 are not recommended for new implementations. | |

When the Authentication_Mechanism_Name is present in the COSEM-OPEN service, the authentication functional unit of the A-ASSOCIATE service shall be selected. The process of LLS and HLS authentication is described in 5.2.2.2 and in 5.7.3.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 7.2.2.4 Cryptographic algorithm ID-s

Cryptographic algorithm IDs identify the algorithm for which a derived secret symmetrical key will be used. See 5.3.4.6.5.

Cryptographic algorithms are identified by the following general object identifier value:

| |
|---|
| COSEM_Cryptographic_Algorithm_Id::=<br>{joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS®-UA(8) cryptographic-algorithms (3) algorithm_id(x)} |

The values of the algorithm_id-s are shown in Table 66. See also Table 8.

**Table 66 – Cryptographic algorithm ID-s**

| | |
|---|---|
| COSEM_cryptographic_algorithm_name_aes-gcm-128::= | algorithm_id(0) |
| COSEM_cryptographic_algorithm_name_aes-gcm-256::= | algorithm_id(1) |
| COSEM_cryptographic_algorithm_name_aes-wrap-128::= | algorithm_id(2) |
| COSEM_cryptographic_algorithm_name_aes-wrap-256::= | algorithm_id(3) |

### 7.2.3 APDU encoding rules

#### 7.2.3.1 Encoding of the ACSE APDUs

The ACSE APDUs shall be encoded in BER (ISO/IEC 8825-1). The user-information parameter of these APDUs shall carry the xDLMS InitiateRequest / InitiateResponse / confirmedServiceError APDU as appropriate, encoded in A-XDR, and then encoding the resulting OCTET STRING in BER.

Examples for AARQ/AARE APDU encoding are given in Annex D.

#### 7.2.3.2 Encoding of the xDLMS APDUs

The xDLMS APDUs shall be encoded in A-XDR, as specified in IEC 61334-6.

#### 7.2.3.3 XML

Depending on the parametrization of the "Push setup" object the DataNotification APDU can be encoded as an XML document using the XML schema specified in Clause 9.

NOTE The use of XML to encode the other APDUs is not in the scope of this document.

### 7.2.4 Protocol for application association establishment

#### 7.2.4.1 Protocol for the establishment of confirmed application associations

AA establishment using the A-Associate service of the ACSE is the key element of COSEM interoperability. The participants of an AA are:

- a client AP, proposing AAs; and
- a server AP, accepting them or not.

NOTE 1 To support multicast and broadcast services, an AA can also be established between a client AP and a group of server APs.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Figure 39 gives the MSC for the case, when:

- the COSEM-OPEN.request primitive requests a confirmed AA;
- the connection of the supporting lower layers is required for the establishment of this AA.

A client AP that desires to establish a confirmed AA, invokes the COSEM-OPEN.request primitive of the ASO with Service_Class == Confirmed. Note, that the PH layer has to be connected before the COSEM-OPEN service is invoked. The response-allowed parameter of the xDLMS InitiateRequest APDU is set to TRUE. The client AL waits for an AARE APDU, prior to generating the .confirm primitive, with a positive – or negative – result.

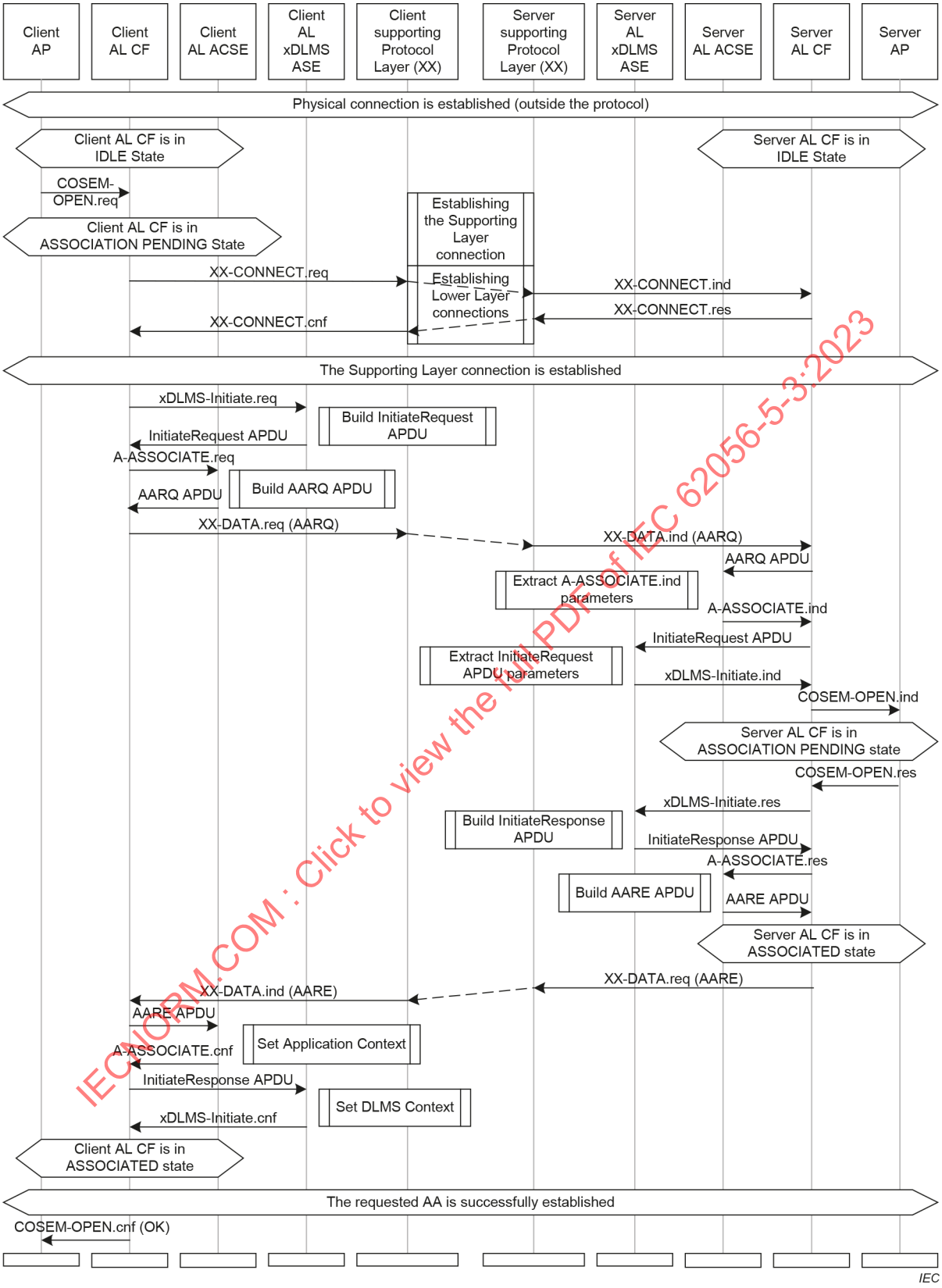
The client CF enters the ASSOCIATION PENDING state. It examines then the Protocol_Connection_Parameters parameter. If this indicates that the establishment of the supporting layer connection is required, it establishes the connection. The CF assembles then – with the help of the xDLMS ASE and the ACSE – the AARQ APDU containing the parameters of the COSEM-OPEN.request primitive received from the AP and sends it to the server.

The CF of the server AL gives the AARQ APDU received to the ACSE. It extracts the ACSE related parameters then gives back the control to the CF. The CF passes then the contents of the user-information parameter of the AARQ APDU – carrying an xDLMS InitiateRequest APDU – to the xDLMS_ASE. It retrieves the parameters of this APDU, then gives back the control to the CF. The CF generates the COSEM-OPEN.indication to the server AP with the parameters of the APDU received and enters the ASSOCIATION PENDING state.

NOTE 2 Some service parameters of the COSEM-OPEN.indication primitive (address information, User_Information) do not come from the AARQ APDU, but from the supporting layer frame carrying the AARQ APDU. In some communication profiles, the Service_Class parameter of the COSEM-OPEN service is linked to the frame type of the supporting layer. In some other communication profiles, it is linked to the response-allowed field of the xDLMS-Initiate.request APDU. See also Annex A.

NOTE 3 The ASEs only extract the parameters; their interpretation and the decision whether the proposed AA can be accepted or not is the job of the server AP.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure 39 – MSC for successful AA establishment preceded by a successful lower layer connection establishment**

The server AP parses the fields of the AARQ APDU as described below.

Fields of the Kernel functional unit:

- application-context-name: it carries the COSEM_Application_Context_Name the client proposes for the association;
- calling-AP-title: when the proposed application context uses ciphering, it shall carry the client system title. If a client system title has already been sent during a registration process, like in the S-FSK PLC profile, the calling-AP-title field shall carry the same system title. Otherwise, the AA shall be rejected and appropriate diagnostic information shall be sent;
- calling-AE-invocation-identifier: this field supports the client user identification process; see IEC 62056-6-2:2021, 4.4.2;
- calling-AE-qualifier: This field can be used to transport the public key certificate of the digital signature key of the client.

Fields of the authentication functional unit (when present):

- sender-acse-requirements:
  a) if not present or present but bit 0 = 0, then the authentication functional unit is not selected. Any following fields of the authentication functional unit may be ignored;
  b) if present and bit 0 = 1 then the authentication functional unit is selected;
- mechanism-name: it carries the COSEM_Authentication_Mechanism_Name the client proposes for the association;
- calling-authentication-value: it carries the authentication value generated by the client.

If the value of the mechanism-name or the calling-authentication-value fields are not acceptable then the proposed AA shall be refused.

When the parsing of the fields of the Kernel and the authentication functional unit is completed, the server continues with parsing the parameters of the xDLMS InitiateRequest APDU, carried by the user-information field of the AARQ:

- dedicated-key: it carries the dedicated key to be used in the AA being established;
- response-allowed: If the proposed AA is confirmed, the value of this parameter is TRUE (default), and the server shall send back an AARE APDU. Otherwise, the server shall not respond. See also Annex A.
- proposed-dlms-version-number, see 4.2.4;
- proposed-conformance, see 7.3.1;
- client-max-receive-pdu-sizes, see 4.2.4.

If all elements of the proposed AA are acceptable, the server AP invokes the COSEM-OPEN.response service primitive with the following parameters:

- Application_Context_Name: the same as the one proposed;
- Result: accepted;
- Failure_Type: Result-source: acse-service-user; Diagnostic: null;
- Responding_AP_title: if the negotiated application context uses ciphering, it shall carry the server system title. If a server system title has already been sent during a registration process, like in the case of the S-FSK PLC profile, the Responding_AP_Title parameter shall carry the same system title. Otherwise, the AA shall be aborted by the client;
- Responding_AE_Qualifier: This field can be used to transport the public key certificate of the digital signature key of the server;
- Fields of the AARE authentication functional unit:
  – (Responder_)ACSE_Requirements:

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

i) when no security (Lowest Level Security) authentication or Low Level Security (LLS) authentication is used, this field shall not be present, or if present, bit 0 (authentication) shall be set to 0. Any following fields of the authentication functional unit may be ignored;

ii) when High Level Security (HLS) authentication is used, this field shall be present and bit 0 (authentication) shall be set to 1;

– Security_Mechanism_Name: it shall carry the COSEM_Authentication_Mechanism_Name negotiated;

– Responding_Authentication_Value: it carries the authentication value generated by the server (StoC).

- Negotiated_xDLMS_context.

The CF assembles the AARE APDU – with the help of the xDLMS ASE and the ACSE – and sends it to the client AL via the supporting lower layer protocols, and enters the ASSOCIATED state. The proposed AA is established now, the server is able to receive xDLMS data transfer service request(s) – both confirmed and unconfirmed – and to send responses to confirmed service requests within this AA.

At the client side, the parameters of the AARE APDU received are extracted with the help of the ACSE and the xDLMS ASE, and passed to the client AP via the COSEM-OPEN.confirm service primitive. At the same time, the client AL enters the ASSOCIATED state. The AA is established now with the application context and xDLMS context negotiated.

If the application context proposed by the client is not acceptable or the authentication of the client is not successful, the COSEM-OPEN.response primitive is invoked with the following parameters:

- Application_Context_Name: the same as the one proposed, or the one supported by the server;
- Result: rejected-permanent or rejected-transient;
- Failure_Type: Result-source: acse-service-user; Diagnostic: an appropriate value;
- User_Information: an xDLMS InitiateResponse APDU with the parameters of the xDLMS context supported by the server.

If the application context proposed by the client is acceptable and the authentication of the client is successful but the xDLMS context cannot be accepted, the COSEM-OPEN.response primitive shall be invoked with the following parameters:

- Application_Context_Name: the same as the one proposed;
- Result: rejected-permanent or rejected-transient;
- Failure_Type: Result-source: acse-service-user; Diagnostic: no-reason-given;
- xDLMS_Initiate_Error, indicating the reason for not accepting the proposed xDLMS context.

In these two cases, upon invocation of the .response primitive, the CF assembles and sends the AARE APDU to the client via the supporting lower layer protocols. The proposed AA is not established, the server CF returns to the IDLE state.

At the client side, the parameters of the AARE APDU received are extracted with the help of the ACSE and the xDLMS_ASE, and passed to the client AP via the COSEM-OPEN.confirm primitive. The proposed AA is not established, the client CF returns to the IDLE state.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The server ACSE may not be capable of supporting the requested association, for example if the AARQ syntax or the ACSE protocol version are not acceptable. In this situation, it returns a COSEM-OPEN.response primitive to the client with an appropriate Result parameter. The Result Source parameter is appropriately assigned the symbolic value of "ACSE service-provider". The COSEM-OPEN.indication primitive is not issued. The association is not established.

#### 7.2.4.2 Repeated COSEM-OPEN service invocations

If a COSEM-OPEN.request primitive is invoked by the client AP referring to an already established AA, then the AL locally and negatively confirms this request with the reason that the requested AA already exists. Note, that this is always the case for pre-established AAs; see 7.2.4.4.

If, nevertheless, a server AL receives an AARQ APDU referencing to an already existing AA, it simply discards this AARQ, or, if it is implemented, it may also respond with the optional ExceptionResponse APDU.

#### 7.2.4.3 Establishment of unconfirmed application associations

A client AP that desires to establish an unconfirmed AA, invokes the COSEM-OPEN.request primitive of the ASO with Service_Class == Unconfirmed. The response-allowed parameter of the xDLMS InitiateRequest APDU, carried by the user-information parameter of the AARQ is set to FALSE. The client AL does not wait any response from the server: the .confirm primitive is locally generated. Otherwise the procedure is the same as in the case of the establishment of confirmed AAs.

As the establishment of unconfirmed AAs does not require the server AP to respond to the association request coming from the client, in some cases – for example in the case of one-way communications or broadcasting – the establishment of unconfirmed AA is the only possibility.

After the establishment of an unconfirmed AA, xDLMS data transfer services using LN referencing can be invoked only in an unconfirmed manner, until the association is released. With SN referencing, only the UnconfirmedWrite service can be used.

#### 7.2.4.4 Pre-established application associations

The purpose of pre-established AAs is to simplify data exchange. The AA establishment and release phases (phases 1 and 3 on Figure 4), using the COSEM-OPEN and COSEM-RELEASE services are eliminated and only data transfer services are used.

This International Standard does not specify how to establish such AAs: it can be considered, that this has already been done. Pre-established AAs can be considered to exist from the moment the lower layers are able to transmit APDUs between the client and the server.

As for all AAs, the logical devices contain an Association LN / SN interface object for the pre-established associations, too.

A pre-established AA can be either confirmed or unconfirmed (depending on the way it has been pre-established).

A pre-established AA cannot be released.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 7.2.5 Protocol for application association release

#### 7.2.5.1 Overview

An existing AA can be released gracefully or non-gracefully. Graceful release is initiated by the client AP. Non-graceful release takes place when an event unexpected by the AP occurs, for example a physical disconnection is detected.

#### 7.2.5.2 Graceful release of an application association

DLMS®/COSEM provides two mechanisms to release AAs:

- by disconnecting the supporting layer of the AL;
- by using the ACSE A-Release service.

The first mechanism shall be supported in all profiles where the supporting layer of the AL is connection oriented.

EXAMPLE The 3-layer, HDLC based, connection oriented profile.

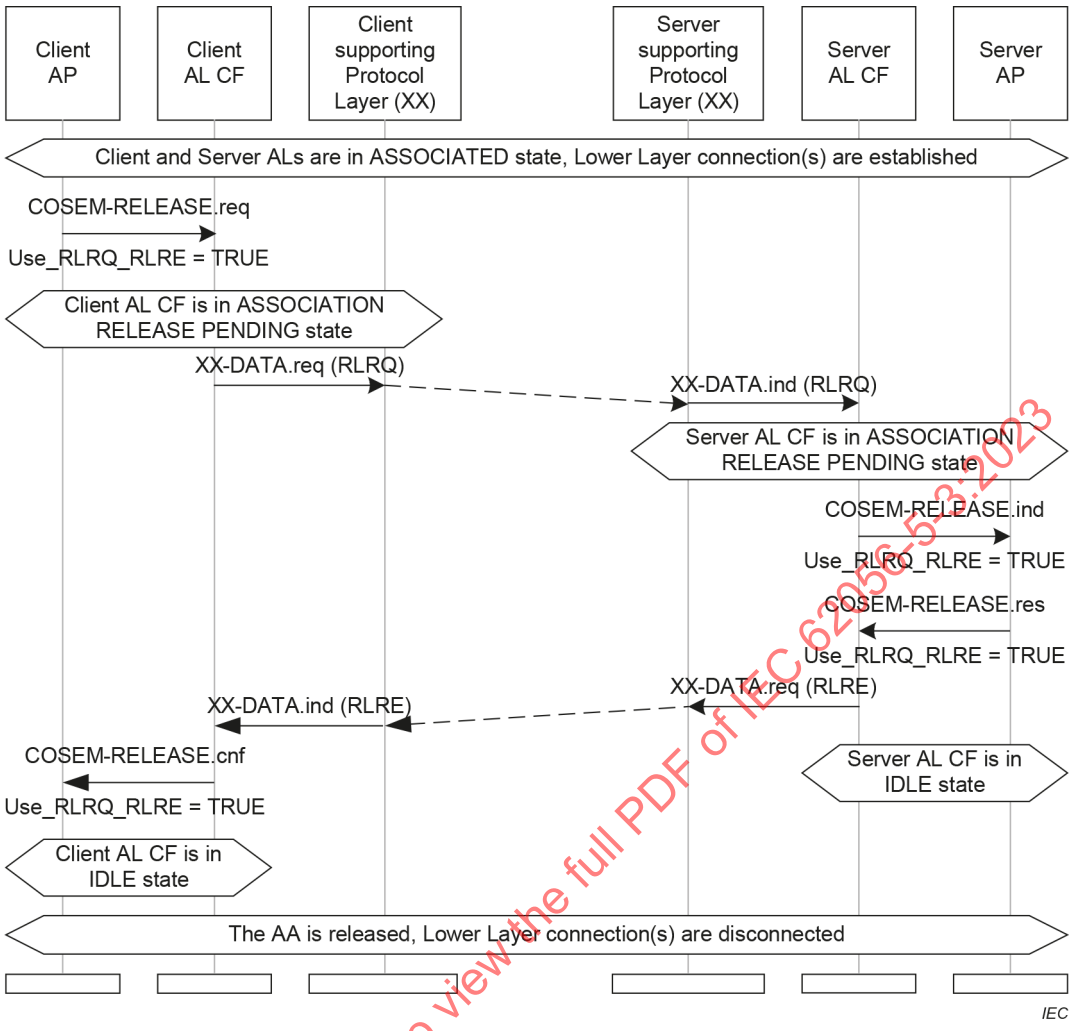
To release an AA this way, the COSEM-RELEASE service shall be invoked with the Use_RLRQ_RLRE parameter not present or FALSE. Disconnecting the supporting layer shall release all AAs built on that supporting layer connection.

The second mechanism can be used to release an AA without disconnecting the supporting layer. It shall be supported in all profiles when the supporting layer is connectionless. It may be also used when the supporting layer is connection-oriented but the connection is not managed by the AL, or disconnection of the supporting layer is not practical because other applications may use it, or when there is a need to secure the COSEM-RELEASE service. It is the only way to release unconfirmed AAs.

To release an AA in this way, the COSEM-RELEASE service shall be invoked with the Use_RLRQ_RLRE parameter = TRUE. As specified in 6.3, the COSEM-RELEASE service can be secured by including a ciphered xDLMS InitiateRequest / InitiateResponse in the user-information field of the RLRQ / RLRE APDUs respectively, thus preventing a potential denial of service attack.

An example for releasing an AA using the ACSE A-RELEASE service is shown in Figure 40.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




NOTE The release of an AA may require internal communication between the ASEs of the CF. These are not shown in Figure 40 to Figure 42.

**Figure 40 – Graceful AA release using the A-RELEASE service**

A client AP that desires to release an AA using the A-RELEASE service, shall invoke the COSEM-RELEASE.request service primitive with Use_RLRQ_RLRE == TRUE. The client CF enters the ASSOCIATION RELEASE PENDING state. It constructs then an RLRQ APDU and sends it to the server. If the AA to be released has been established with a ciphered context, then the user information field of the RLRQ APDU may contain a ciphered xDLMS InitiateRequest APDU, see 6.3.

When the server AL CF receives the RLRQ APDU, it checks first if the user-information field contains a ciphered xDLMS InitiateRequest APDU. If so, it tries to decipher it. If this is successful, it enters the ASSOCIATION RELEASE PENDING state and generates a COSEM-RELEASE.indication primitive with Use_RLRQ_RLRE == TRUE. Otherwise, it silently discards the RLRQ APDU and stays in the ASSOCIATED state.

The .response primitive is invoked by the server AP to indicate if the release of the AA is accepted or not, but only if the AA to be released is confirmed. Note, that the server AP cannot refuse a release request. Upon reception of the .response primitive the server AL CF constructs a RLRE APDU and sends it to the client. If the RLRQ APDU contained a ciphered xDLMS InititateRequest APDU then the RLRE APDU shall contain a ciphered xDLMS InitiateResponse APDU. The server AL CF returns to the IDLE state.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The .confirm primitive is generated by the client AL CF when the RLRE APDU is received. The supporting layer is not disconnected. The client AL CF returns to the IDLE state.

If the RLRE APDU received contains a ciphered xDLMS InitiateResponse APDU but it cannot be deciphered, then the RLRE APDU shall be discarded. It is left to the client to cope with the situation.

Figure 41 gives an example of graceful releasing a confirmed AA by disconnecting the corresponding lower layer connection.

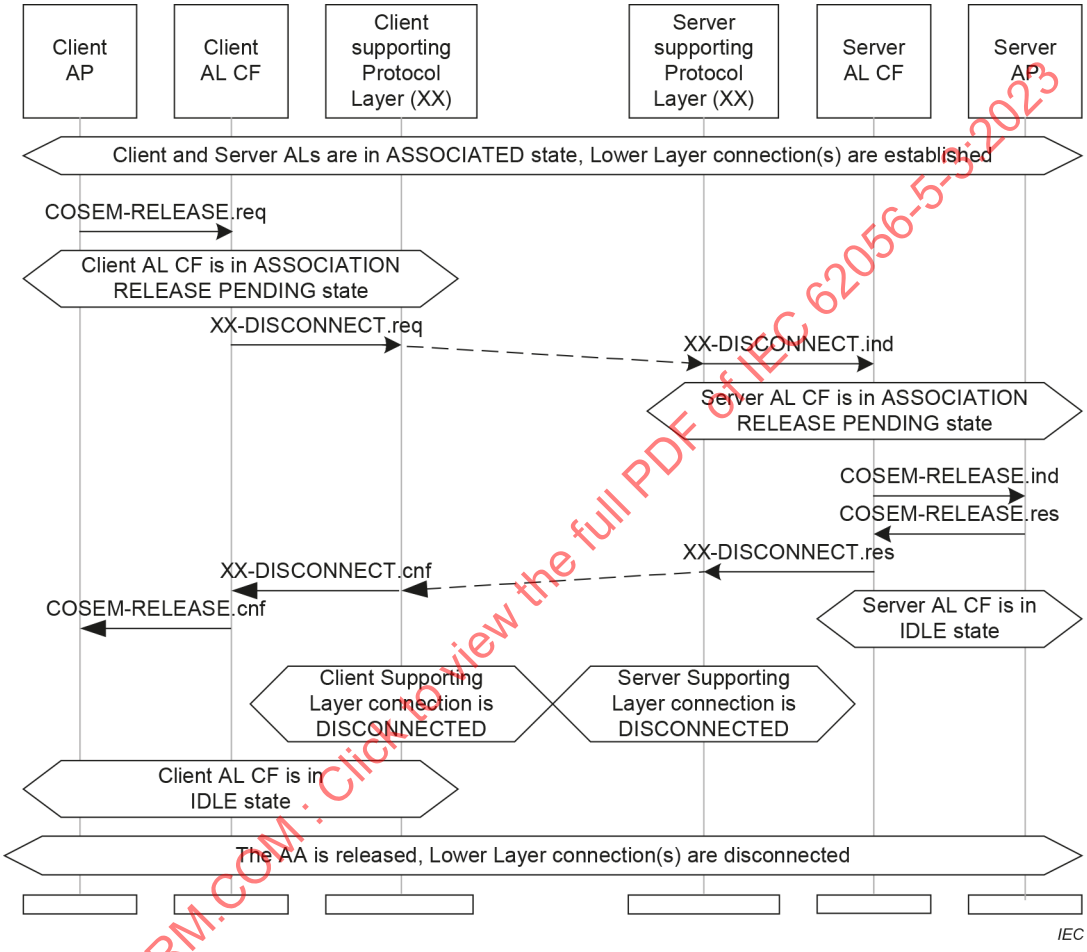



**Figure 41 – Graceful AA release by disconnecting the supporting layer**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

A client AP that desires to release an AA not using the A-RELEASE service, invokes the COSEM-RELEASE.request primitive with Use_RLRQ_RLRE == FALSE or not present. The client AL CF enters the ASSOCIATION RELEASE PENDING state.

In communication profiles where the RLRQ service is mandatory, invoking the .request primitive with no Use_RLRQ_RLRE or with Use_RLRQ_RLRE == FALSE may lead to an error: the .request shall be locally and negatively confirmed. The client AL CF returns to the IDLE state.

When the client AL CF receives the .request primitive, it sends an XX-DISCONNECT.request primitive to the server.

When the server AL CF receives the XX-DISCONNECT.request primitive, the CF enters the ASSOCIATION RELEASE PENDING state. The COSEM-RELEASE.indication primitive is generated by the server AL CF with Use_RLRQ_RLRE == FALSE or not present.

The COSEM-RELEASE.response primitive is invoked by the server AP to indicate if the release of the AA is accepted or not. Note, that the server AP cannot refuse a release request. Upon reception of this primitive, the server AL CF sends an XX-DISCONNECT.response primitive to the client and returns to the IDLE state.

The COSEM-RELEASE.confirm primitive is generated by the client AL when the XX-DISCONNECT.confirm primitive is received. The supporting layer is disconnected. The client AL CF returns to the IDLE state.

#### 7.2.5.3 Non-graceful release of an application association

Various events may result in a non-graceful release of an AA: detection of the disconnection of any lower layer connection (including the physical connection), detecting a local error, etc.

Non-graceful release – abort – of an AA is indicated to the COSEM AP with the help of the COSEM-ABORT service. The Diagnostics parameter of this service indicates the reason for the non-graceful AA release. The non-graceful release of AA is not selective: if it happens, all the existing association(s) – except the pre-established ones – shall be aborted.

Figure 42 shows the message sequence chart for aborting the AA, due to the detection of a physical disconnection.

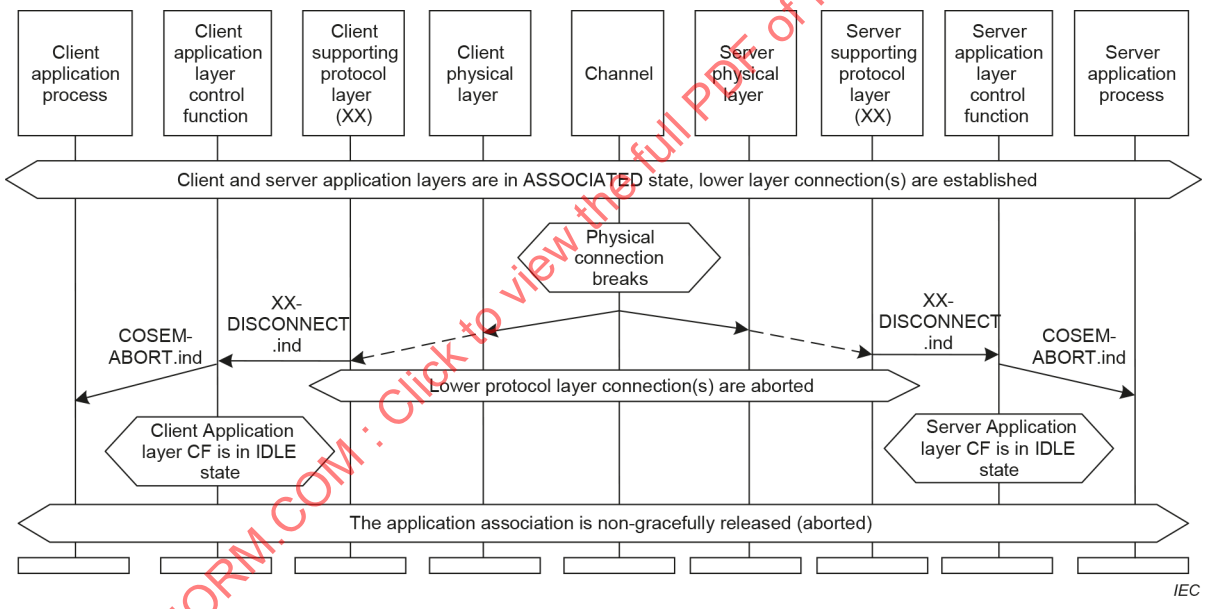



**Figure 42 – Aborting an AA following a PH-ABORT.indication**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 7.3 Protocol for the data transfer services

#### 7.3.1 Negotiation of services and options – the conformance block

The conformance block allows clients and servers using the same DLMS®/COSEM protocol, but supporting different capabilities to negotiate a compatible set of capabilities so that they can communicate. It is carried by the DLMS_Conformance parameter of the COSEM-OPEN service.

In DLMS®/COSEM none of the services or options are mandatory: the ones to be used are negotiated via the COSEM-OPEN service (the proposed-conformance parameter of the xDLMS InitiateRequest APDU and the negotiated-conformance parameter of the xDLMS InitiateResponse APDU). An implemented service shall be fully conforming to its specification. If a service or option is not present in the negotiated conformance block, it may not be requested by the client.

The xDLMS conformance block can be distinguished from the DLMS® conformance block specified in IEC 61334-4-41:1996 by its tag: APPLICATION 31. It is shown in Table 67.

**Table 67 – xDLMS Conformance block**

| Conformance block bit | Reserved | LN referencing | SN referencing |
|---|---|---|---|
| 0 | x | | |
| 1 | | general-protection [1] | general-protection [1] |
| 2 | | general-block-transfer | general-block-transfer |
| 3 | | | read |
| 4 | | | write |
| 5 | | | unconfirmed-write |
| 6 | | delta-value-encoding | delta-value-encoding |
| 7 | x | | |
| 8 | | attribute0-supported-with-set | |
| 9 | | priority-mgmt-supported | |
| 10 | | attribute0-supported-with-get | |
| 11 | | block-transfer-with-get-or-read | block-transfer-with-get-or-read |
| 12 | | block-transfer-with-set-or-write | block-transfer-with-set-or-write |
| 13 | | block-transfer-with-action | |
| 14 | | multiple-references | multiple-references |
| 15 | | | information-report |
| 16 | | data-notification | data-notification |
| 17 | | access | |
| 18 | | | parameterized-access |
| 19 | | get | |
| 20 | | set | |
| 21 | | selective-access | |
| 22 | | event-notification | |
| 23 | | action | |

[1] general-protection includes general-glo-ciphering, general-ded-ciphering, general-ciphering and general-signing

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 7.3.2 Confirmed and unconfirmed service invocations

#### 7.3.2.1 Service invocations by the client

In general, data transfer services may be invoked in a confirmed or an unconfirmed manner. The time sequence of the service primitives corresponds to:

- Figure 35 item a) in case of confirmed service invocations; and
- Figure 35 item d) in case of unconfirmed service invocations.

A client AP that desires to access an attribute or a method of a COSEM interface object invokes the appropriate .request service primitive. The client AL constructs the APDU corresponding to the .request primitive and sends it to the server.

Upon the receipt of the .indication primitive, the server AP checks whether the service can be provided or not (validity, client access rights, availability, etc.). If it can, it locally applies the service required on the corresponding "real" object. In the case of confirmed services, the server AP invokes the appropriate .response primitive. The server AL constructs the APDU corresponding to the .response primitive and sends it to the client. The client AL generates the .confirm primitive.

If a confirmed service request cannot be processed by the server AL – for example the request has been received without establishing an AA first, or the request is otherwise erroneous – it is either discarded, or when possible, the server AL responds with a ConfirmedServiceError APDU, or, when implemented, with an ExceptionResponse APDU. These APDUs may contain diagnostic information about the reason of not being able to process the request. They are defined in Clause 8.

Within confirmed AAs, client/server type data transfer services can be invoked in a confirmed or unconfirmed manner.

Within unconfirmed AAs, client/server type data transfer services may be invoked in an unconfirmed manner only. With this, collisions due to potential multiple responses in the case of multicasting and/or broadcasting can be avoided.

In the case of unconfirmed services, three different kinds of destination addresses are possible: individual, group or broadcast. Depending on the destination address type, the receiving station shall handle incoming APDUs differently, as follows:

- XX-APDUs with an individual address of a COSEM logical device. If they are received within an established AA they shall be sent to the COSEM logical device addressed, otherwise they shall be discarded;
- XX-APDUs with a group address of a group of COSEM logical devices. These shall be sent to the group of COSEM logical devices addressed. However, the message received shall be discarded if there is no AA established between a client and the group of COSEM logical devices addressed;
- XX-APDUs with the broadcast address shall be sent to all COSEM logical devices addressed. However, the message received shall be discarded if there is no AA established between a client and the All-station address.

NOTE Unconfirmed AA-s between a client and a group of logical devices are established with a COSEM-OPEN service with Service_Class = Unconfirmed and a group of logical device addresses (for example broadcast address).

#### 7.3.2.2 Service invocations by the server (unsolicited services)

The unsolicited services that may be invoked by the server are:

- InformationReport;
- EventNotification;
- DataNotification.

The InformationReport and the EventNotification services may be invoked only in an unconfirmed manner. The corresponding time sequence diagram is Figure 35 item d).

The DataNotification service may be invoked in three different ways:

a) unconfirmed, retry on supporting protocol layer failure;

b) unconfirmed, retry on missing supporting protocol layer confirmation;

c) confirmed, retry on missing confirmation.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The time sequence of the service primitives corresponds to:

– in the case a) Figure 35 item d);
– in the case b) Figure 35 item b);
– in the case c) Figure 35 item a).

The protocol of the DataNotification service is described in 6.10.

### 7.3.3 Protocol for the GET service

When the client AP desires to read the value of one or more COSEM interface object attributes, it uses the GET service.

As explained in 6.6, the encoded form of the request shall always fit in a single APDU.

On the other hand, the result may be too long to fit in a single APDU. In this case, either the service-specific or the general block transfer mechanism may be used. It is negotiated via bit 2 or bit 11 of the conformance block; see 7.3.1.

NOTE In some DLMS®/COSEM communication profiles segmentation is available to transfer long APDUs.

The GET service primitive types and the corresponding APDUs are shown in Table 68.

**Table 68 – GET service types and APDUs**

| GET .req / .ind | Request APDU | Response APDU | GET .res / .cnf |
|---|---|---|---|
| NORMAL | Get-Request-Normal | Get-Response-Normal | NORMAL |
| | | Get-Response-With-Datablock with Last-Block = FALSE | ONE-BLOCK |
| NEXT | Get-Request-Next | Get-Response-With-Datablock with Last-Block = FALSE | ONE-BLOCK |
| | | Get-Response-With-Datablock with Last-Block = TRUE | LAST-BLOCK |
| WITH-LIST | Get-Request-With-List | Get-Response-With-List | WITH-LIST |
| | | Get-Response-With-Datablock with Last-Block = FALSE | ONE-BLOCK |

Figure 43 shows the MSC for a confirmed GET service in the case of success, without block transfer.

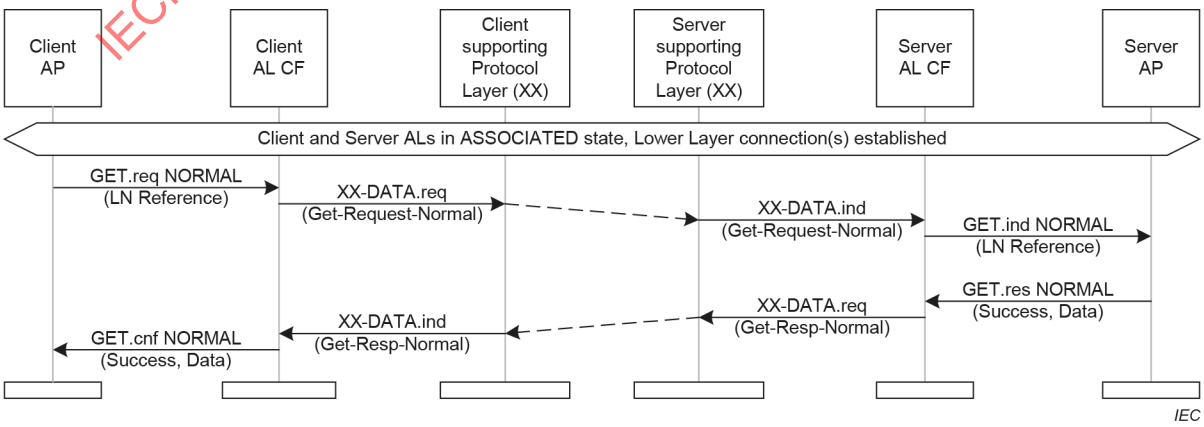



**Figure 43 – MSC of the GET service**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Figure 44 shows the MSC of a confirmed GET service in the case of success, with the result returned in three blocks.

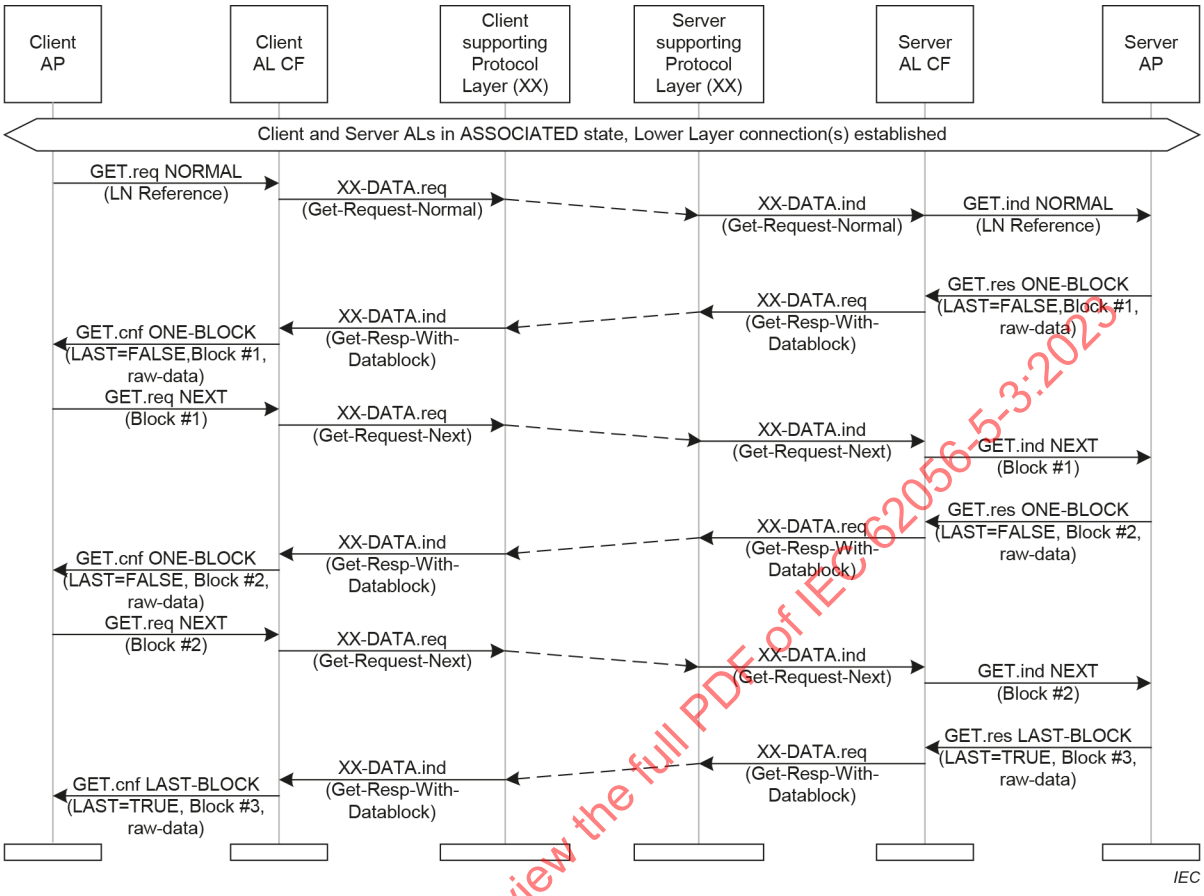



**Figure 44 – MSC of the GET service with block transfer**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The GET.request primitive is invoked with Request_Type == NORMAL or WITH-LIST as appropriate. As in this case the data to be returned is too long to fit in a single APDU, the server AP sends it in blocks. First, the data is encoded, as if it would fit into a single APDU:

- if the value of a single attribute was requested, only the type and value shall be encoded (Data). If the Data cannot be delivered, the response shall be of type GET-NORMAL with Data_Access_Result;
- if the value of a list of attributes was requested, then the list of results shall be encoded: for each attribute, either Data or Data_Access_Result.

The result is a series of bytes, $B_1$, $B_2$, $B_3$,...., $B_N$. The server may generate the complete response upon receipt of the first GET.indication primitive or dynamically (on the fly).

The server AP assembles then a DataBlock_G structure:

- Last_Block == FALSE;
- Block_Number == 1;
- Result (Raw_Data) == the first K bytes of the encoded data: $B_1$, $B_2$, $B_3$,...., $B_K$.

It is recommended to start the numbering of the blocks from 1.

The server AP invokes then the GET-RESPONSE-ONE-BLOCK service primitive with Response_Type == ONE-BLOCK carrying this DataBlock-G structure.

Upon reception of the .response primitive, the server AL builds a Get-Response-With-Datablock APDU carrying the parameters of the .response primitive and sends it to the client.

Upon reception of this APDU, the client AL generates a .confirm primitive with Response_Type == ONE-BLOCK. The client AP is informed now that the response is provided in several blocks. It stores the data block received ($B_1$, $B_2$, $B_3$,…., $B_K$), then acknowledges its reception and asks for the next one by invoking a GET-REQUEST-NEXT service primitive. The block number shall be the same as the block number of the data block received. The client AL builds a Get-Request-Next APDU and sends it to the server.

When the server AL invokes the GET.indication primitive with Request_Type == NEXT, the server AP prepares and sends the next data block, including $B_{K+1}$, $B_{K+2}$, $B_{K+3}$,…., $B_L$, with block-number == 2. This exchange of sending data blocks and acknowledgements continues until the last data block, carrying $B_M$, $B_{M+1}$, $B_{M+2}$,…., $B_N$ is sent. The last GET.response primitive is invoked with Response_Type == LAST-BLOCK: in the DataBlock-G structure Last_Block == TRUE. This last data block is not acknowledged by the client.

Throughout the whole procedure, the Invoke_Id and the Priority parameters shall be the same in each primitive.

If during a long data transfer the server receives another service request, it is served according to the priority rules and the priority management settings (Conformance block bit 9).

If any error occurs during the long data transfer, the transfer is aborted. The error cases are:

a) The server is not able to provide the next block of data for any reason. In this case, the server AP shall invoke a GET-RESPONSE-LAST-BLOCK service primitive. The Result parameter shall contain a DataBlock_G structure with:
   - Last_Block == TRUE;
   - Block_Number == number of the block confirmed by the client +1;
   - Result == Data_Access_Result, indicating the reason of the failure.

b) The Block_Number parameter in a GET-REQUEST-NEXT service primitive is not equal to the number of the previous block sent by the server. The server interprets this, as if the client would like to abort the ongoing transfer. In this case, the server AP shall invoke a GET-RESPONSE-LAST-BLOCK service primitive. The Result parameter shall contain a DataBlock_G structure with:
   - Last_Block == TRUE;
   - Block_Number == equal to the block number received from the client;
   - Result == Data-Access-Result, long-get-aborted.

c) The server may receive a Get-Request-Next APDU when no long data transfer is in progress. In this case, the server AP shall invoke a GET-RESPONSE-LAST-BLOCK service primitive. The Result parameter shall contain a DataBlock_G structure with:
   - Last-block == TRUE;
   - Block-Number == equal to the block number received from the client;
   - Result == Data-Access-Result, no-long-get-in-progress.

d) The block number sent by the server is not equal to the next in sequence. In this case, the client shall abort the block transfer (see case b).

If, in the error cases above, the server is not able to invoke a GET-RESPONSE-LAST-BLOCK service primitive for any reason, it shall invoke a GET-RESPONSE-NORMAL service primitive, with the Data-Access-Result parameter indicating the reason of the failure. The server shall send a Get-Response-Normal APDU.

The MSC for error case b), long get aborted, is shown in Figure 45:

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

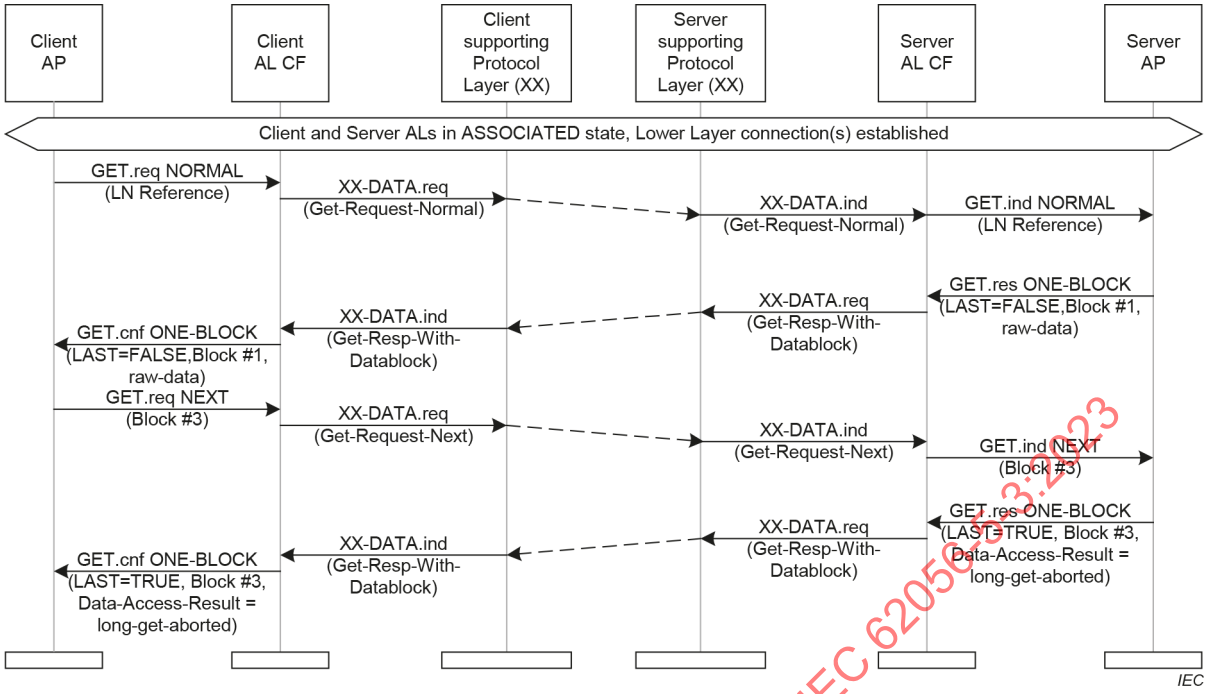



**Figure 45 – MSC of the GET service with block transfer, long GET aborted**

### 7.3.4 Protocol for the SET service

When the client AP desires to write the value of one or more COSEM interface object attributes, it uses the SET service.

As explained in 6.7, the encoded form of the request may fit in a single request or not. In this latter case, either the service-specific or the general block transfer mechanism may be used. It is negotiated via bit 2 or bit 12 of the conformance block; see 7.3.1.

NOTE In some DLMS®/COSEM communication profiles segmentation is available to transfer long APDUs.

The SET service primitive types and the corresponding APDUs are shown in Table 69.

**Table 69 – SET service types and APDUs**

| SET .req / .ind | Request APDU | Response APDU | SET .res / .cnf |
|---|---|---|---|
| NORMAL | Set-Request-Normal | Set-Response-Normal | NORMAL |
| FIRST-BLOCK | Set-Request-With-First-Datablock<br>Last-Block = FALSE | Set-Response-Datablock | ACK-BLOCK |
| ONE-BLOCK | Set-Request-With-Datablock<br>Last-Block = FALSE | | |
| LAST-BLOCK | Set-Request-With-Datablock<br>Last-Block = TRUE | Set-Response-Last-Datablock | LAST-BLOCK |
| | | | LAST-BLOCK-WITH-LIST |
| WITH-LIST | Set-Request-With-List | Set-Response-With-List | WITH-LIST |
| FIRST-BLOCK-WITH-LIST | Set-Request-With-List-And-With-First-Datablock | Set-Response-Datablock | ACK-BLOCK |

Figure 46 shows the MSC of a confirmed SET service, in the case of success, without block transfer.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

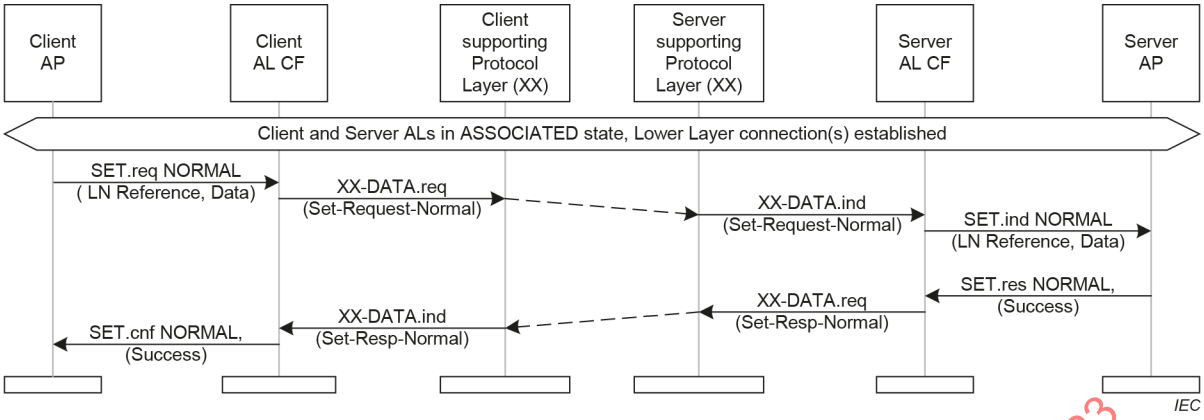



**Figure 46 – MSC of the SET service**

Figure 47 shows the MSC of a confirmed SET service in the case of success, with the request sent in three blocks.

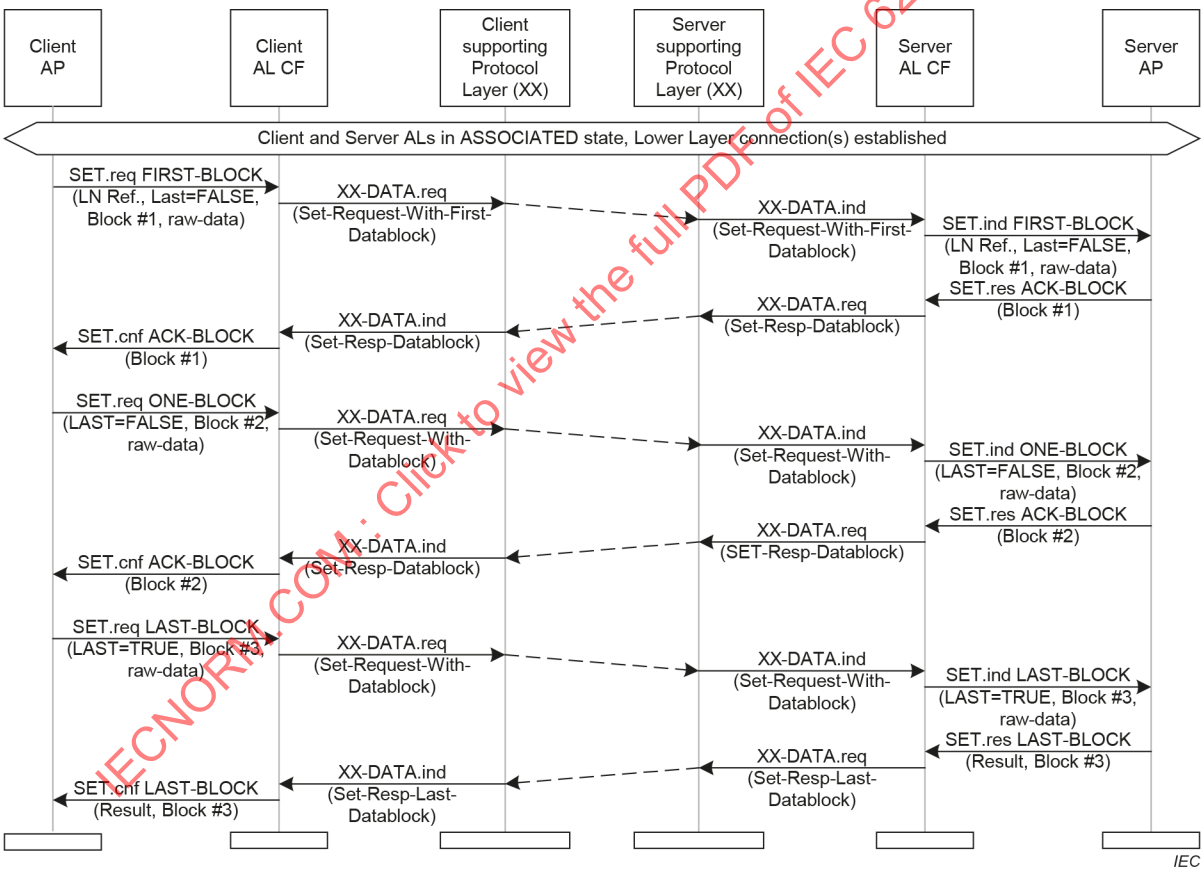



**Figure 47 – MSC of the SET service with block transfer**

As in this case the data to be sent is too long to fit in a single APDU in this case, the client AP sends it in blocks. First, the data is encoded as if it would fit in a single APDU. The result is a series of bytes, $B_1$, $B_2$, $B_3$,...., $B_N$. The client may generate the complete request ($B_1$, $B_2$, $B_3$,....$B_N$) in one step or dynamically (on the fly).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The client AP assembles then a DataBlock_SA structure:

- Last_Block == FALSE;
- Block_Number == 1;
- Raw_Data == the first K bytes of the encoded data: $B_1$, $B_2$, $B_3$,...., $B_K$.

The client AP invokes then the SET-REQUEST-FIRST-BLOCK or SET-REQUEST-FIRST-BLOCK-WITH-LIST service primitive as appropriate, carrying the attribute reference(s) and this DataBlock-SA structure.

Upon reception of the .request primitive, the client AL builds the appropriate Set-Request APDU carrying the parameters of the .request primitive and sends it to the server.

The server stores the data block received, then acknowledges its reception and asks for the next one by invoking a SET.response primitive with Response_Type == ACK-BLOCK and with the same block number as the number of the block received.

To send the next data block carrying $B_{K+1}$, $B_{K+2}$, $B_{K+3}$,...., $B_{L,}$ the client AP invokes a SET-REQUEST-ONE-BLOCK service primitive. This exchange of sending data blocks and acknowledgements continues until the last data block carrying $B_M$, $B_{M+1}$, $B_{M+2}$,...., $B_N$ is sent, by invoking a SET-REQUEST-LAST-BLOCK service primitive with Last_Block == TRUE.

When these primitives are invoked, the client AL builds a Set-Request-With-Datablock APDU, carrying a DataBlock_SA structure and sends these APDUs to the server.

When the server AP receives the last datablock, it invokes a SET-RESPONSE-LAST-BLOCK or SET-RESPONSE-LAST-BLOCK-WITH-LIST service primitive as appropriate. The Result parameter carries the result of the complete SET service invocation. The Block_Number parameter confirms the reception of the last block.

Throughout the whole procedure, the Invoke_Id and the Priority parameters shall be the same in each primitive.

If during a long data transfer the server receives another service request, it is served according to the priority rules and the priority management settings (Conformance block bit 9).

If any error occurs during the long data transfer, the transfer is aborted. The error cases are:

a) The server is not able to handle the data block received, for any reason. In this case, the server AP shall invoke a SET-RESPONSE-LAST-BLOCK or SET-RESPONSE-LAST-BLOCK-WITH-LIST service primitive as appropriate. The Result parameter indicates the reason for aborting the transfer;

b) The Block_Number parameter in a SET-REQUEST-ONE-BLOCK service primitive is not equal to the block number expected by the server (last received + 1). The server interprets this as if the client would like to abort the ongoing transfer. In this case, the server AP shall invoke a SET-RESPONSE-LAST-BLOCK or SET-RESPONSE-LAST-BLOCK-WITH-LIST service primitive as appropriate with the Result parameter Data_Access_Result == long-set-aborted;

c) The server may receive a Set-Request-With-Datablock APDU when no long data transfer is in progress. In this case, the server AP shall invoke a SET-RESPONSE-LAST-BLOCK service primitive with the Result parameter Data_Access_Result == no-long-set-in-progress.

If, in the error cases above, for any reason the server is not able to invoke a SET-RESPONSE-LAST-BLOCK service primitive, it invokes a SET-RESPONSE-NORMAL service primitive with the Data-Access-Result parameter indicating the reason of the failure.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 7.3.5 Protocol for the ACTION service

When the client AP desires to invoke one or more COSEM interface objects methods, it uses the ACTION service. As explained in 6.8, the ACTION service comprises two phases.

If the method references and method invocation parameters or the return parameters do not fit in a single APDU, either the service-specific or the general block transfer mechanism may be used. It is negotiated via bit 2 or bit 13 of the conformance block; see 7.3.1.

NOTE In some DLMS®/COSEM communication profiles segmentation is available to transfer long APDUs.

The ACTION service primitive types and the corresponding APDUs are shown in Table 70.

**Table 70 – ACTION service types and APDUs**

| ACTION .req / .ind | Request APDU | Response APDU | SET .res / .cnf |
|---|---|---|---|
| NORMAL | Action-Request-Normal | Action-Response-Normal | NORMAL |
| | | Action-Response-With-Pblock | ONE-BLOCK |
| NEXT | Action-Request-Next-Pblock | Action-Response-With-Pblock | ONE-BLOCK |
| | | Action-Response-With-Pblock | LAST-BLOCK |
| FIRST-BLOCK | Action-Request-With-First-Pblock | Action-Response-Next-Pblock | NEXT |
| ONE-BLOCK | Action-Request-With-Pblock | | |
| LAST-BLOCK | Action-Request-With-Pblock | Action-Response-Normal | NORMAL |
| | | Action-Response-With-Pblock | ONE-BLOCK |
| WITH-LIST | Action-Request-With-List | Action-Response-With-List | WITH-LIST |
| | | Action-Response-With-Pblock | ONE-BLOCK |
| WITH-LIST-AND-FIRST-BLOCK | Action-Request-With-List-And-With-First-Pblock | Action-Response-Next-Pblock | NEXT |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Figure 48 illustrates the MSC of a confirmed ACTION service in the case of success, without block transfer.

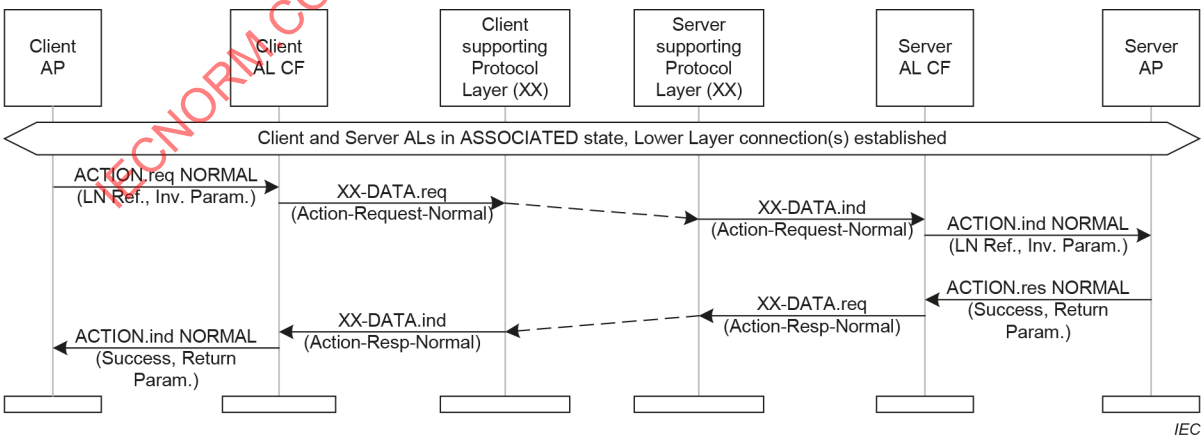



**Figure 48 – MSC of the ACTION service**

The ACTION service can transport data in both directions:

- in the first phase, the client sends the ACTION.request with the method invocation parameters for the method(s) referenced and the server acknowledges them. The process is essentially the same as in the case of the SET service;
- in the second phase, the server sends the ACTION.response with the result of invoking the method(s) and the return parameters. The process is essentially the same as in the case of the GET service.

Throughout the whole procedure, the Invoke_Id and the Priority parameters shall be the same in each primitive.

If during a long data transfer the server receives another service request, it is served according to the priority rules and the priority management settings (Conformance block bit 9).

Figure 49 illustrates the MSC in the case when block transfer takes place in both directions.

If any error occurs during the long data transfer, the transfer shall be aborted. Error cases are the same as in the case of the GET and SET services.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

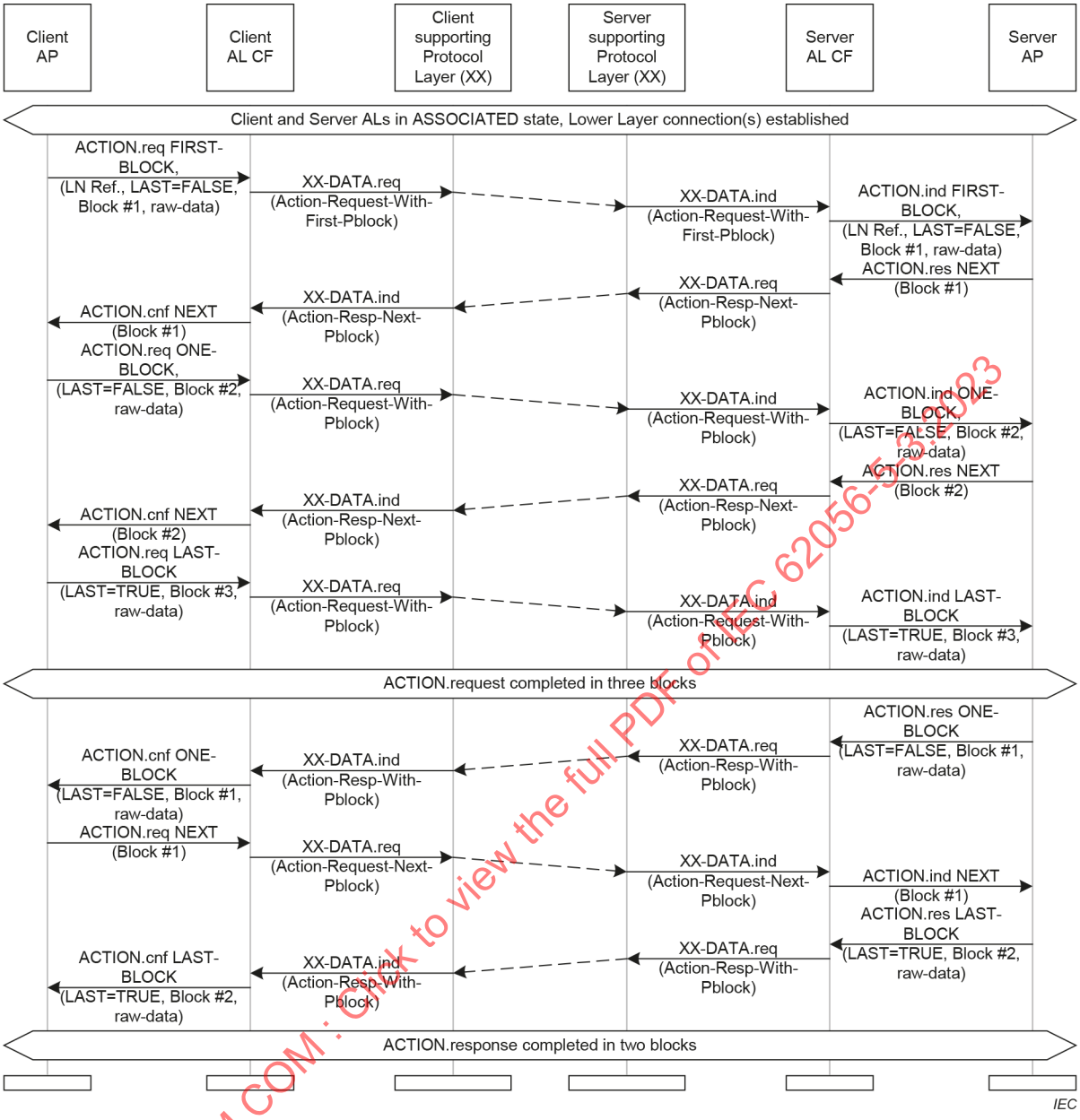



**Figure 49 – MSC of the ACTION service with block transfer**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 7.3.6 Protocol for the ACCESS service

The client can use the ACCESS service to read or write the value of one or more COSEM object attributes or to invoke one or more methods.

The protocol of the ACCESS service is specified by way of message sequence charts, including cases where it is used together with general block transfer and general message protection.

NOTE See also 4.2.4.4.7, 6.5 and 7.3.13.

Figure 50 shows the MSC of an ACCESS service used to get one COSEM object attribute values. The request fits in a single APDU. The response is long therefore the server sends back the response using the GBT mechanism: portions of the access-response APDU are carried by the block-data field of general-block-transfer APDUs.

When the client receives the first general-block-transfer APDU, it switches to GBT announcing that streaming with window size 3 is supported.

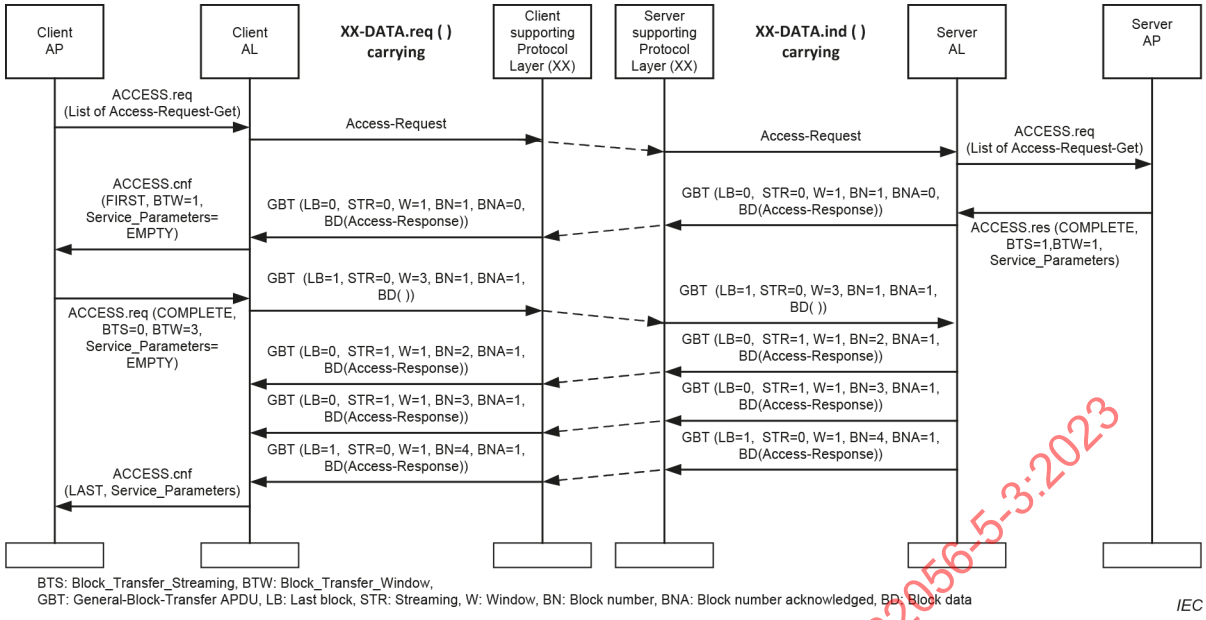



**Figure 50 – Access Service with long response**

Figure 51 shows the MSC of an ACCESS service used to carry a list of requests, which does not fit in a single APDU, therefore GBT is used. The response is also long therefore the server also uses GBT. Both parties know a priori that the other party supports streaming with window size = 3.

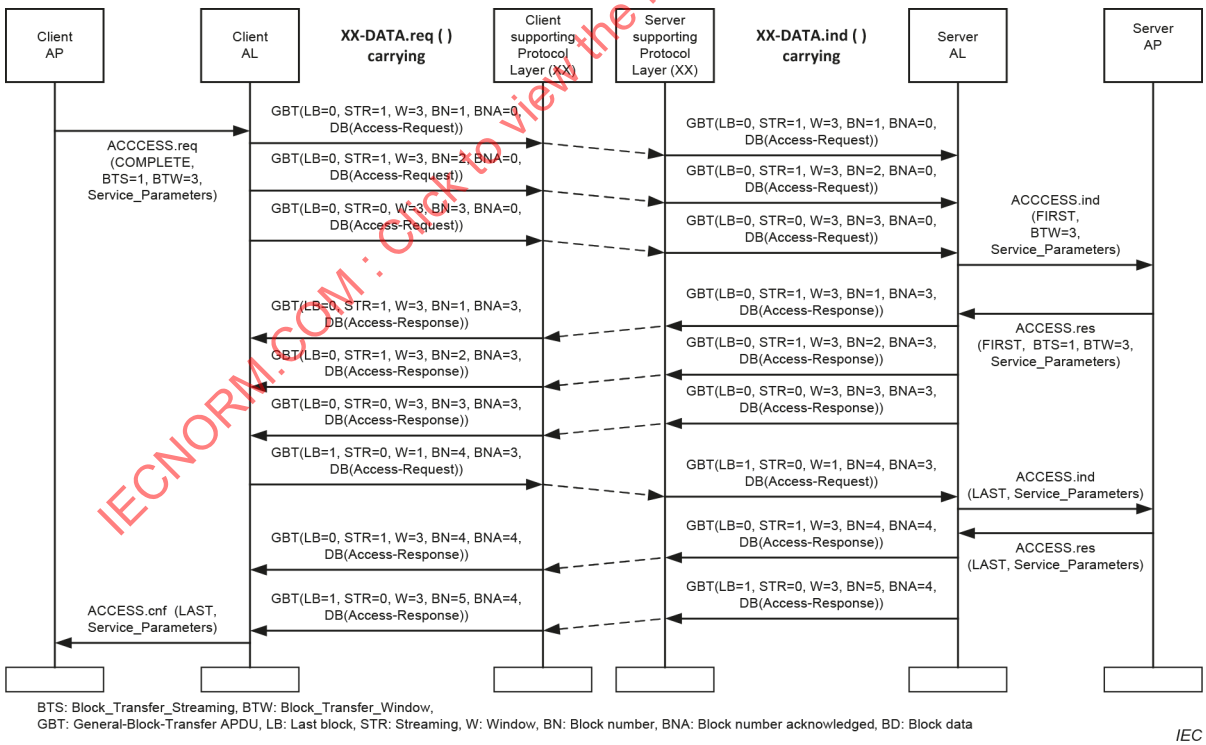



**Figure 51 – Access Service with long request and response**

### 7.3.7 Protocol of the DataNotification service

When the server AP invokes a DataNotification.request service primitive, the server AL builds the DataNotification APDU and passes it to the supporting protocol layer.

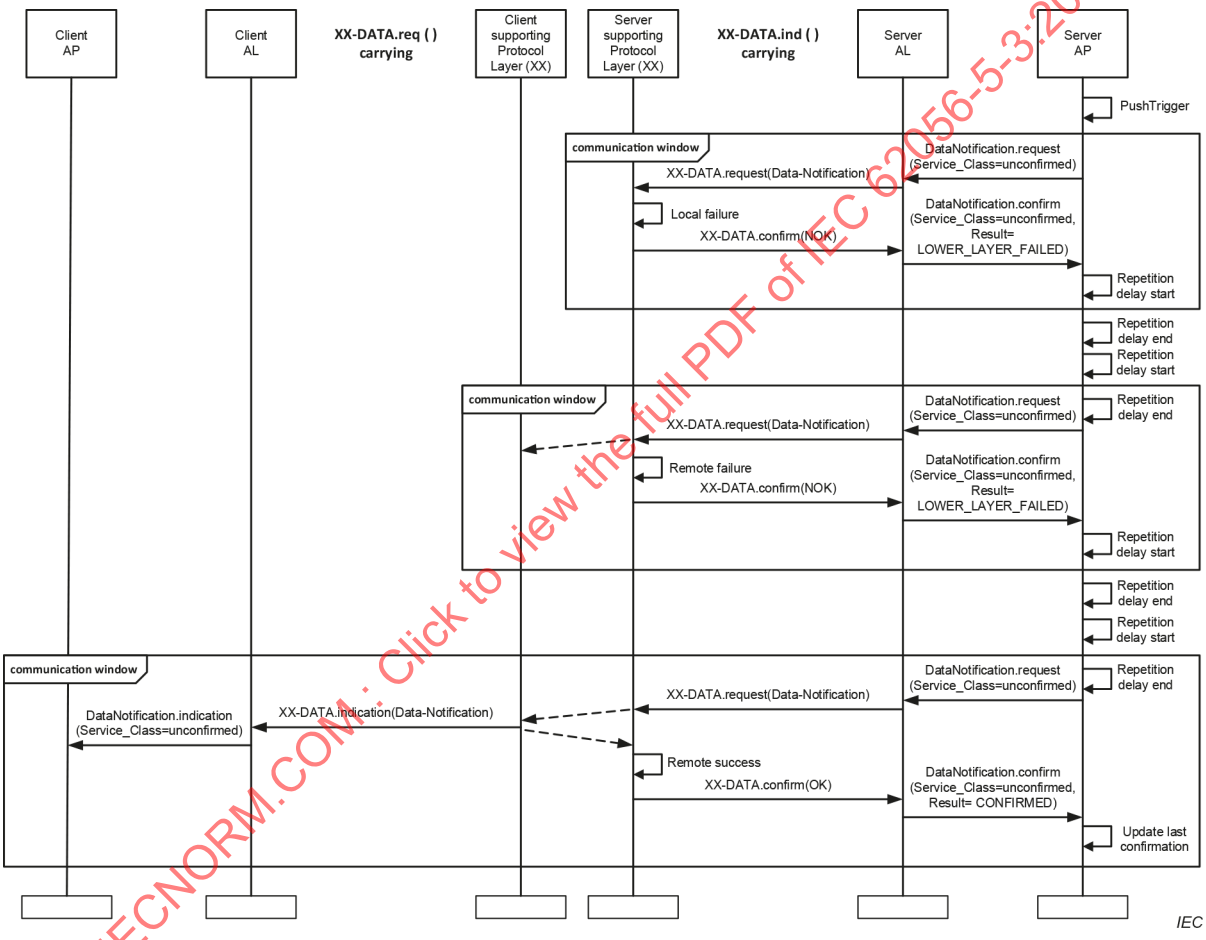
IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

When the client AL receives this APDU, it invokes the DataNotification.indication service primitive.

In case a), unconfirmed, retry on supporting protocol layer failure (see 7.3.2.2):

- the push operation is deemed as failed when the server AP receives a DataNotification.cnf service primitive with Service_Class == Unconfirmed and Result == LOWER_LAYER_FAILED. In this case, the server AP may attempt a retry after a repetition delay;
- the push operation is deemed as successful when the server AP receives a DataNotification.cnf with Service_Class == Unconfirmed and Result == CONFIRMED;

Figure 52 shows the MSC of the DataNotification service for this case.




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure 52 – MSC for the DataNotification service, case a)**

In case b), unconfirmed, retry on missing supporting protocol layer confirmation (see 7.3.2.2):

- the push operation is deemed as failed if the repetition delay expires before a confirmation is received. In this case, the server may attempt a retry;
- the push operation is deemed as successful when the server supporting protocol layer informs the AL that the DataNotification APDU has been successfully received by the remote supporting protocol layer. In this case, the server AL invokes a DataNotification.cnf service primitive with Service_Class == Unconfirmed and Result == CONFIRMED.

Figure 53 shows the MSC of the DataNotification service for this case.

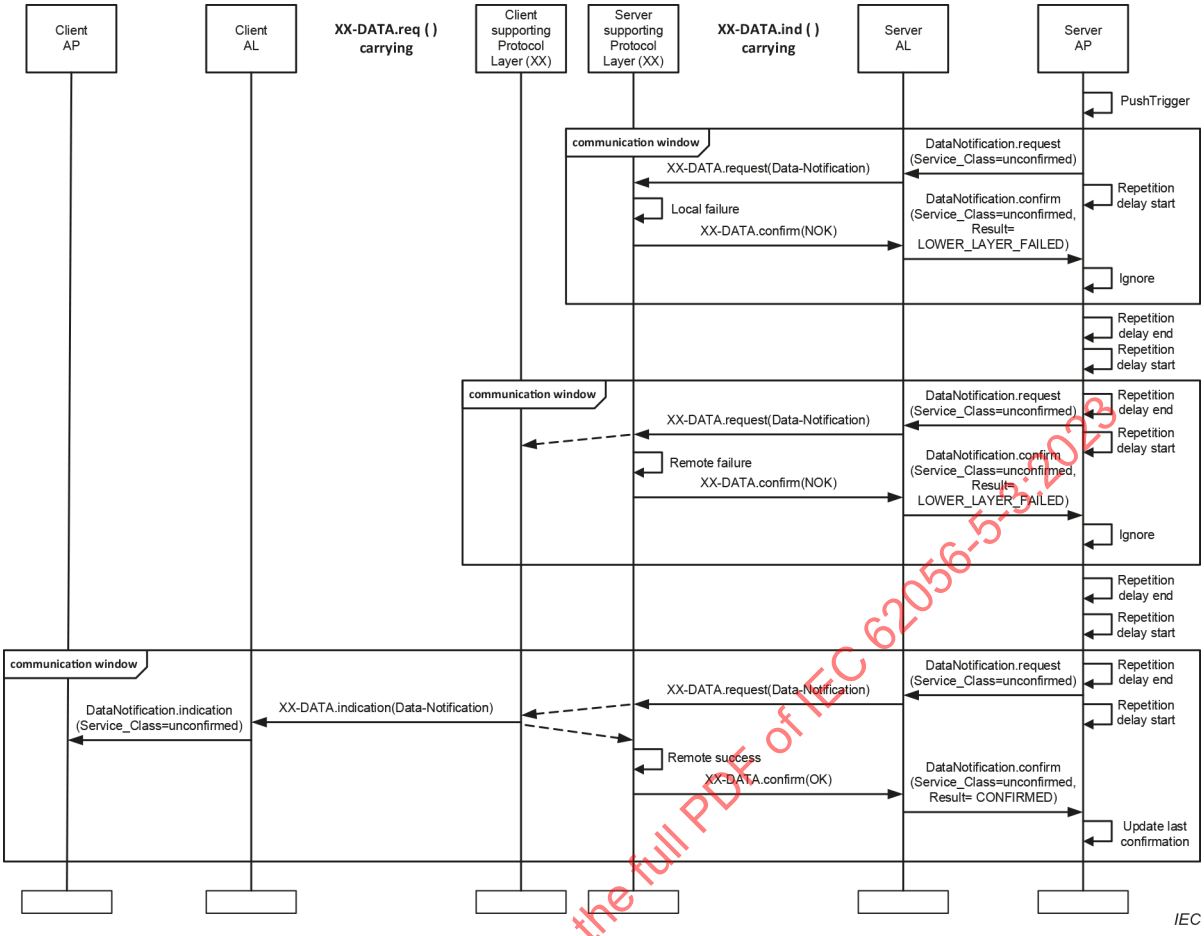



**Figure 53 – MSC for the DataNotification service, case b)**

In case c), confirmed, retry on missing confirmation (see 7.3.2.2):

- when the client AL receives a correct DataNotification APDU, it invokes the DataNotification.ind service primitive. If the Data-Notification APDU cannot be processed for any reason, it is discarded;
- the client AP invokes the DataNotification.response service primitive with Result == Confirmed. The client AL builds the Data-Notification-Confirm APDU and sends it to the server;
- the push operation is deemed as failed if the repetition delay expires before a confirmation is received. In this case, the server may attempt a retry;
- the push operation is deemed as successful when the AL layer receives the Data-Notification-Confirm APDU and invokes the DataNotification.cnf service primitive with Service_Class == Confirmed and Result == CONFIRMED.

Figure 54 shows the MSC of the DataNotification service for this case.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

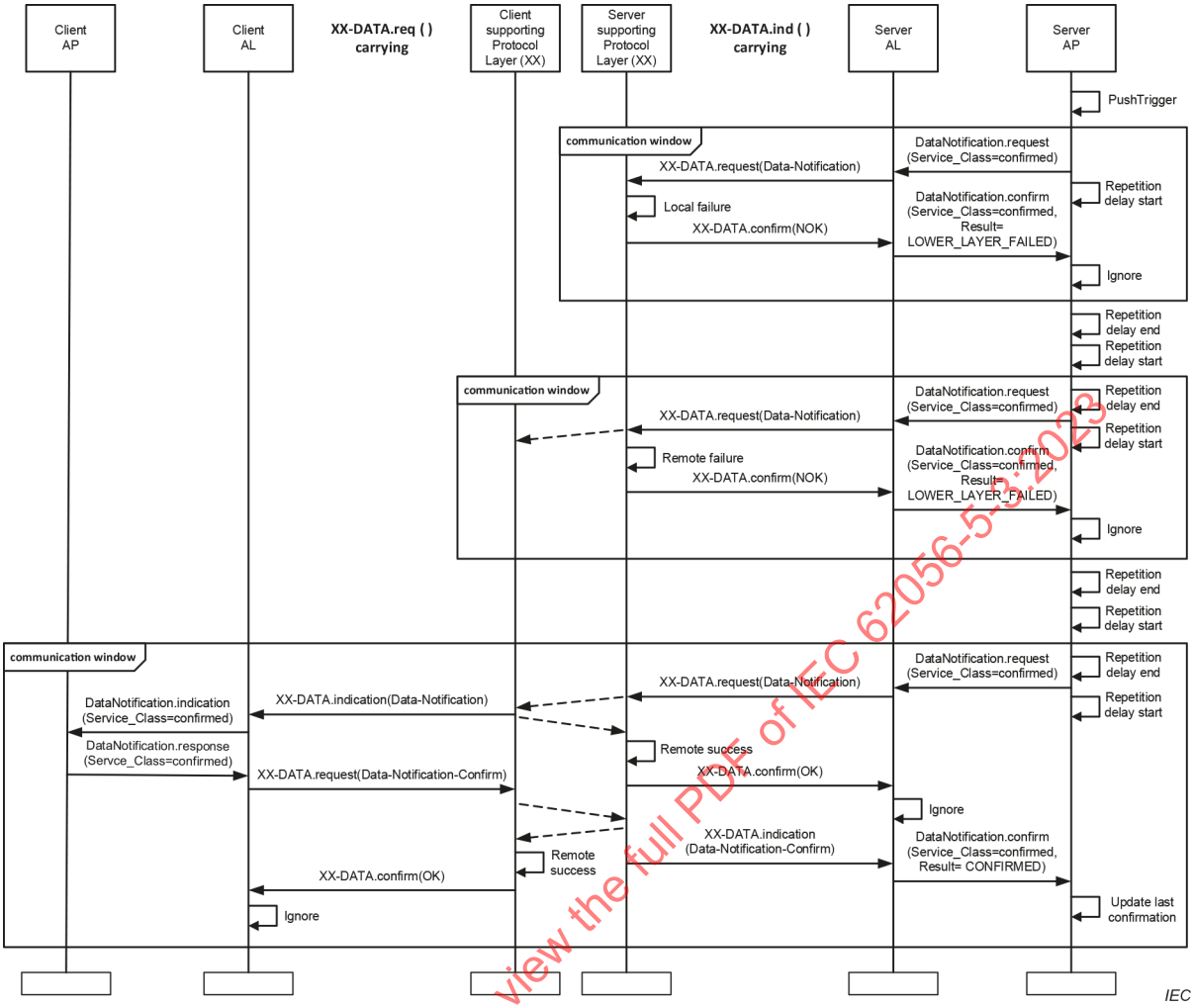



**Figure 54 – MSC for the DataNotification service, case c)**

### 7.3.8 Protocol for the EventNotification service

Upon invocation of the EventNotification.request service, the Server AL builds an EventNotificationRequest APDU. The possibilities to send out this APDU depend on the communication profile and the connection status of the lower layers. Therefore, the protocol of the EventNotification service is further discussed in the parts of IEC 62056 describing the communication profiles;

In any case, in order to send the value(s) of attribute(s) to the client, without the client requesting it:

- the server uses the EventNotification.request service primitive;
- upon invocation of this primitive, the server AL builds an EventNotificationRequest APDU;
- this APDU is carried by the supporting layer service at the first opportunity to the client. The service type and the availability of this first opportunity depends on the communications profile used;
- upon reception of the EventNotificationRequest APDU, the client AL generates an EventNotification.indication primitive to the COSEM client AP;

NOTE At the client side, it is always EventNotification.indication, independently of the referencing scheme (LN or SN) used by the server.

- by default, event notifications are sent from the management logical device (server) to the management AP (client).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 7.3.9 Protocol for the Read service

As explained in 6.14, the Read service is used when the server uses SN referencing, either to read (a) COSEM interface object attribute(s), or to invoke (a) method(s) when return parameters are expected:

- in the first case, the GET.request service primitives are mapped to Read.request primitives and the Read.confirm primitives are mapped to GET.confirm primitives. The mapping and the corresponding SN APDUs are shown in Table 71;
- in the second case, the ACTION.request service primitives are mapped to Read.request primitives and the Read.response primitives are mapped to ACTION.response primitives. The mapping and the corresponding SN APDUs is shown in Table 72.

NOTE In the mapping tables below, the following notation is used:

- for LN services, only the request and response types are shown without service parameters;
- for SN services, the name of the service primitive is followed by the service parameters in brackets. Service parameter name elements are capitalized and joined with an underscore to signify a single entity. Parameters that may be repeated are shown in curly brackets. The choices that can be taken for the Variable_Access_Specification parameter are listed following the symbol "=". Alternatives are separated by the vertical bar "|";
- for SN APDUs, the name of the APDU is followed by the symbol "::=" and the fields in brackets. The field name elements are not capitalized and are joined with a dash to signify a single entity. Fields that may be repeated are shown in curly brackets. Alternatives are separated by the vertical bar "|".

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 71 – Mapping between the GET and the Read services**

| From GET.request of type | To Read.request | SN APDU |
|---|---|---|
| NORMAL | Read.request (Variable_Access_Specification)<br>Variable_Access_Specification = Variable_Name \| Parameterized_Access; | ReadRequest::= (variable-name \| parameterized-access) |
| NEXT | Read.request (Variable_Access_Specification)<br>Variable_Access_Specification = Block_Number_Access; | ReadRequest::= (block-number-access) |
| WITH-LIST | Read.request ({Variable_Access_Specification})<br>Variable_Access_Specification = Variable_Name \| Parameterized_Access; | ReadRequest::= ({variable-name \| parameterized-access}) |
| **To GET.confirm of type** | **SN APDU** | **From Read.response** |
| NORMAL | ReadResponse::= (data \| data-access-error) | Read.response (Data \| Data_Access_Error) |
| ONE-BLOCK | ReadResponse::= (data-block-result) | Read.response (Data_Block_Result) with Last_Block = FALSE |
| LAST-BLOCK | ReadResponse::= (data-block-result) | Read.response (Data_Block_Result) with Last_Block = TRUE |
| WITH-LIST | ReadResponse::= ({data \| data-access-error}) | Read.response ({Data \| Data_Access_Error}) |

**Table 72 – Mapping between the ACTION and the Read services**

| From ACTION.request of type | To Read.request | SN APDU |
|---|---|---|
| NORMAL | Read.request (Variable_Access_Specification)<br>Variable_Access_Specification = Parameterized_Access;<br>with<br>Variable_Name = method reference,<br>Selector = 0,<br>Parameter = method invocation parameter or null-data | ReadRequest::= (parameterized-access) |
| NEXT | Read.request (Variable_Access_Specification)<br>Variable_Access_Specification = Block_Number_Access; | ReadRequest:: = (block-number-access) |
| FIRST-BLOCK | Read.request (Variable_Access_Specification)<br>Variable_Access_Specification = Read_Data_Block_Access;<br>with<br>Last_Block = FALSE,<br>Block_Number = 1,<br>Raw_Data = one part of the method reference(s) and method invocation parameter | ReadRequest::= (read-data-block-access) |
| ONE-BLOCK | Read.request (Variable_Access_Specification)<br>Variable_Access_Specification = Read_Data_Block_Access;<br>with<br>Last_Block = FALSE,<br>Block_Number = next number,<br>Raw_Data = as above | ReadRequest::= (read-data-block-access) |
| LAST-BLOCK | Read.request (Variable_Access_Specification)<br>Variable_Access_Specification = Read_Data_Block_Access;<br>with<br>Last_Block = TRUE,<br>Block_Number = next number,<br>Raw_Data = as above | ReadRequest::= (read-data-block-access) |
| WITH-LIST | Read.request ({Variable_Access_Specification})<br>Variable_Access_Specification = Parameterized_Access;<br>with<br>Variable_Name = method reference,<br>Selector = 0,<br>Parameter = method invocation parameter or null-data | ReadRequest::= ({parameterized-access}) |
| WITH-LIST-AND-FIRST-BLOCK | Read.request (Variable_Access_Specification)<br>Variable_Access_Specification = Read_Data_Block_Access;<br>with<br>Last_Block = FALSE,<br>Block_Number = 1,<br>Raw_Data = as above | ReadRequest::= (read-data-block-access) |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| To ACTION.confirm | SN APDU | From Read.response |
|---|---|---|
| NORMAL | ReadResponse::=<br>(data I data-access-error) | Read.response (Read_Result)<br>Read_Result =<br>Data I Data_Access_Error; |
| ONE-BLOCK | ReadResponse::=<br>(data-block-result) | Read.response (Read_Result)<br>Read_Result = Data_Block_Result;<br>with Last_Block = FALSE |
| LAST-BLOCK | ReadResponse::=<br>(data-block-result) | Read.response (Read_Result)<br>Read_result = Data_Block_Result;<br>with Last_Block = TRUE |
| NEXT | ReadResponse::=<br>(block-number) | Read.confirm (Read_Result)<br>Read_Result = Block_Number; |
| WITH-LIST | ReadResponse::=<br>({data I data-access-error}) | Read.response ({Read_Result})<br>Read_Result =<br>Data I Data_Access_Error; |

Figure 55 shows the MSC of a Read service used to read the value of a single attribute.

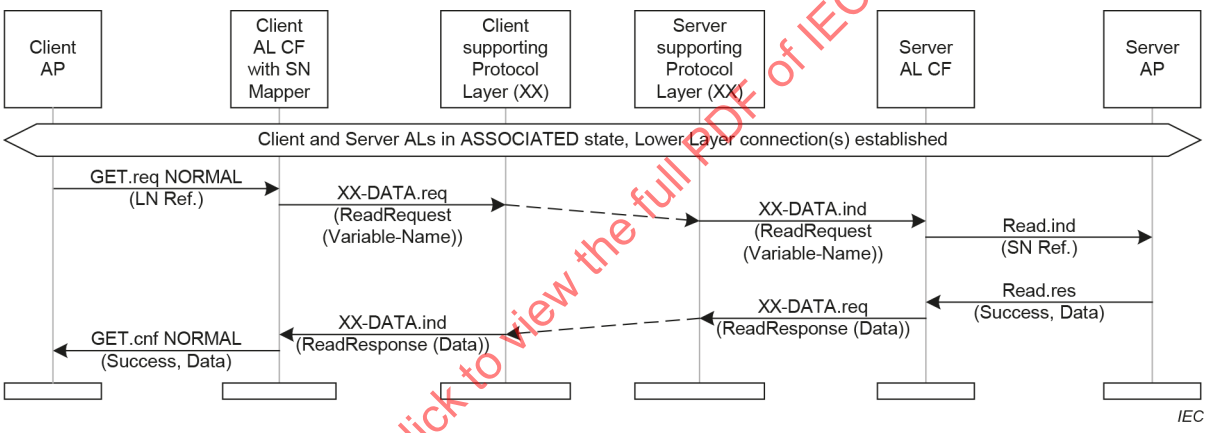



**Figure 55 – MSC of the Read service used for reading an attribute**

Figure 56 shows the MSC of a Read service used to invoke a single method.

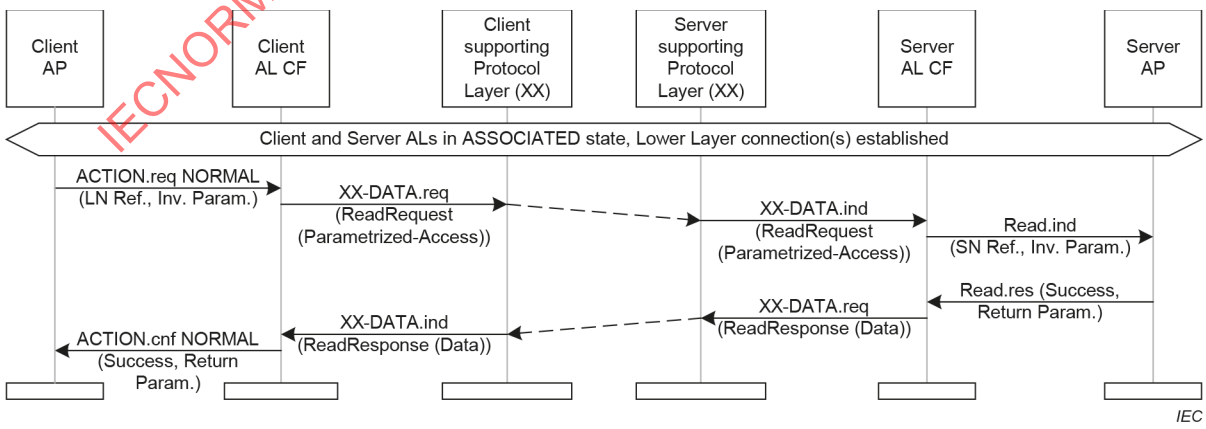



**Figure 56 – MSC of the Read service used for invoking a method**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Figure 57 shows the MSC of a Read service for reading a single attribute, with the result returned in three blocks using the service-specific block transfer mechanism.

Alternatively, the general block transfer mechanism can be used.

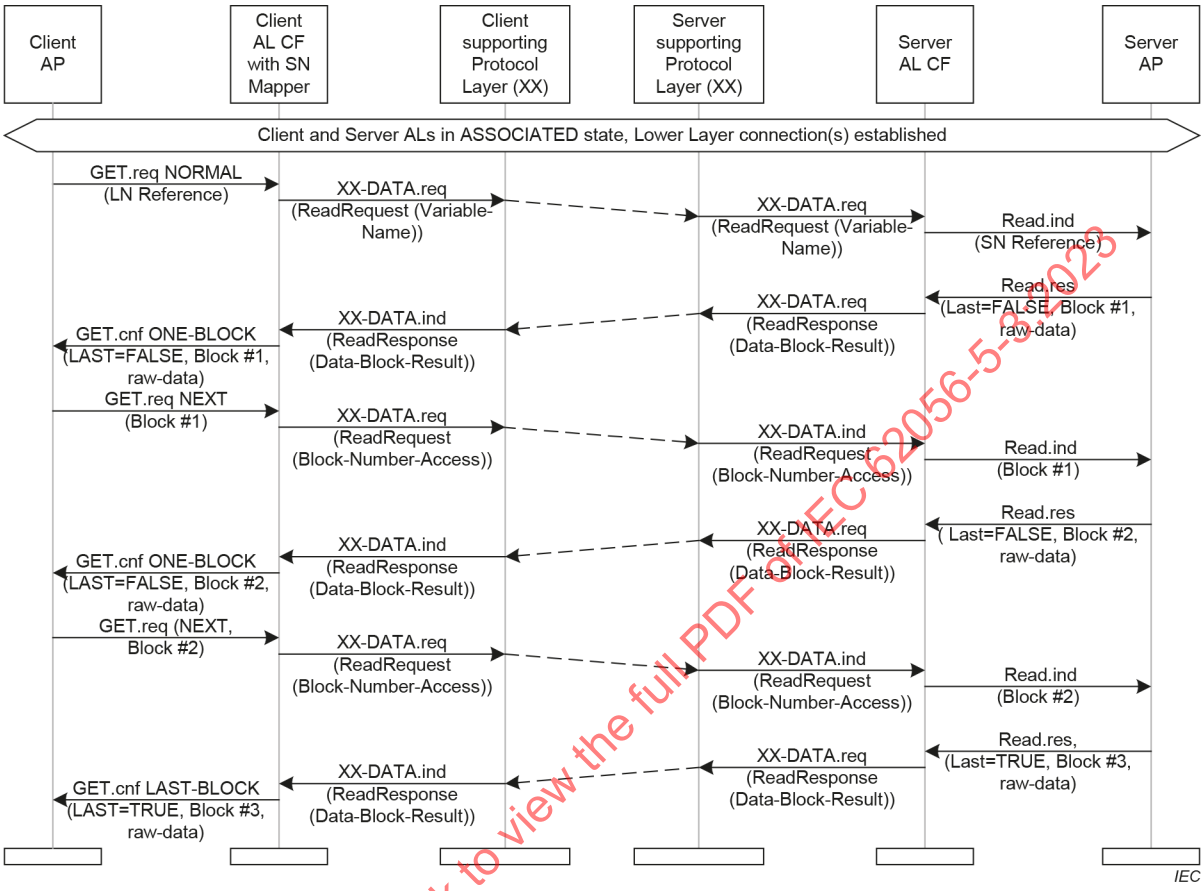



**Figure 57 – MSC of the Read Service used for reading an attribute, with block transfer**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The process of preparing and transporting the long data is essentially the same as in the case of the GET and ACTION services.

- if the Read service is used to read the value of (a) COSEM object attribute(s), the Raw_Data element of the Data_Block_Result construct carries one part of the list of Read_Result(s);
- if the Read service is used to invoke (a) COSEM object method(s) and long method invocation parameters have to be sent, the Raw_Data element of the Read_Data_Block_Access construct carries one part of the method reference(s) and method invocation parameter(s). If long method invocation responses are returned, the Raw_Data element of the Data_Block_Result construct carries one part of the method invocation response(s).

If an error occurs, the server should return a Read.response service primitive with Data_Access_Error carrying appropriate diagnostic information; for example data-block-number-invalid.

### 7.3.10 Protocol for the Write service

As explained in 6.15, the Write service is used when the server uses SN referencing, either to write (a) COSEM object attribute(s), or to invoke (a) method(s) when no return parameters are expected:

- in the first case, the SET.request service primitives are mapped to Write.request primitives and the Write.confirm primitives to SET.confirm primitives. The mapping and the corresponding SN APDUs are shown in Table 73;
- in the second case, the ACTION.request service primitives are mapped to Write.request primitives and the Write.response primitives to ACTION.confirm primitives. The mapping and the corresponding SN APDUs are shown in Table 74.

**Table 73 – Mapping between the SET and the Write services**

| From SET.request of type | To Write.request | SN APDU |
|---|---|---|
| NORMAL | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Variable_Name I Parameterized_Access; | WriteRequest::= (variable-name I parameterized-access, data) |
| FIRST-BLOCK | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Write_Data_Block_Access;<br>with<br>Last_Block = FALSE,<br>Block_Number = 1,<br>Data = raw-data, carrying the encoded form of the attribute reference(s) and write data. | WriteRequest::= (write-data-block-access, data) |
| ONE-BLOCK | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Write_Data_Block_Access;<br>with<br>Last_Block = FALSE,<br>Block_Number = next number,<br>Data = as above | WriteRequest::= (write-data-block-access, data) |
| LAST-BLOCK | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Write_Data_Block_Access;<br>with<br>Last_Block = TRUE,<br>Block_Number = next number,<br>Data = as above | WriteRequest::= (write-data-block-access, data) |
| WITH-LIST | Write.request ({Variable_Access_Specification}, {Data})<br>Variable_Access_Specification = Variable_Name I Parameterized_Access; | WriteRequest::= ({variable-name I parameterized-access}, {data}) |
| FIRST-BLOCK-WITH-LIST | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Write_Data_Block_Access;<br>with<br>Last_Block = FALSE,<br>Block_Number = 1<br>Data =as above | WriteRequest::= (write-data-block-access, data) |
| **To SET.confirm of type** | **SN APDU** | **From Write.response** |
| NORMAL | WriteResponse::= (success I data-access-error) | Write.response (Write_Result)<br>Write_Result = Success I Data_Access_Error; |
| ACK-BLOCK | WriteResponse::= (block-number) | Write.response (Write_Result)<br>Write_Result = Block_Number; |
| LAST-BLOCK | WriteResponse::= (success I data-access-error) | Write.response (Write_Result)<br>Write_Result = Success I Data_Access_Error; |
| WITH-LIST | WriteResponse::= ({success I data-access-error}) | Write.response ({Write_Result})<br>Write_Result = Success I Data_Access_Error; |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| From SET.request of type | To Write.request | SN APDU |
|---|---|---|
| LAST-BLOCK-WITH-LIST | WriteResponse::=<br>({success I data-access-error}) | Write.response ({Write_Result})<br>Write_Result =<br>Success I Data_Access_Error; |

**Table 74 – Mapping between the ACTION and the Write service**

| From ACTION.request of type | To Write.request | SN APDU |
|---|---|---|
| NORMAL | Write.request<br>(Variable_Access_Specification, Data)<br>Variable_Access_Specification =<br>Variable_Name;<br>Data = method invocation parameters or null-data | WriteRequest::=<br>(variable-name, data) |
| FIRST-BLOCK | Write.request<br>(Variable_Access_Specification, Data)<br>Variable_Access_Specification =<br>Write_Data_Block_Access;<br>with<br>Last_Block = FALSE,<br>Block_Number = 1,<br>Data = raw-data, carrying the encoded form of the method reference(s) and method invocation parameters; | WriteRequest::=<br>(write-data-block-access, data) |
| ONE-BLOCK | Write.request<br>(Variable_Access_Specification, Data)<br>Variable_Access_Specification =<br>Write_Data_Block_Access;<br>with<br>Last_Block = FALSE,<br>Block_Number = next number,<br>Data = as above | WriteRequest::=<br>(write-data-block-access, data) |
| LAST-BLOCK | Write.request<br>(Variable_Access_Specification, Data)<br>Variable_Access_Specification =<br>Write_Data_Block_Access;<br>with<br>Last_Block = TRUE,<br>Block_Number = next number,<br>Data = as above | WriteRequest::=<br>(write-data-block-access, data) |
| WITH-LIST | Write.request<br>({Variable_Access_Specification},<br>{Data})<br>Variable_Acces_Specification =<br>Variable_Name;<br>Data = method invocation parameters or null data | WriteRequest::=<br>({variable-name}, {data}) |
| WITH-LIST-AND-FIRST-BLOCK | Write.request<br>(Variable_Access_Specification, Data)<br>Variable_Access_Specification =<br>Write_Data_Block_Access;<br>with<br>Last_Block = FALSE,<br>Block_Number = 1,<br>Data = as with first block | WriteRequest::=<br>(write-data-block-access, data) |
| **To ACTION.confirm** | **SN APDU** | **From Write.response** |
| NORMAL | WriteResponse::=<br>(success I data-access-error) | Write.response (Write_Result)<br>Write_Result =<br>Success I Data_Access_Error |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| From ACTION.request of type | To Write.request | SN APDU |
|---|---|---|
| NEXT | WriteResponse::= (block-number) | Write.response (Block_Number) |
| **To ACTION.confirm** | **SN APDU** | **From Write.response** |
| WITH-LIST | WriteResponse::= ({success I data-access-error}) | Write.response ({Write_Result}) Write_Result = Success I Data_Access_Error |

Figure 58 shows the MSC of a Write service used to write the value of a single attribute, in the case of success.

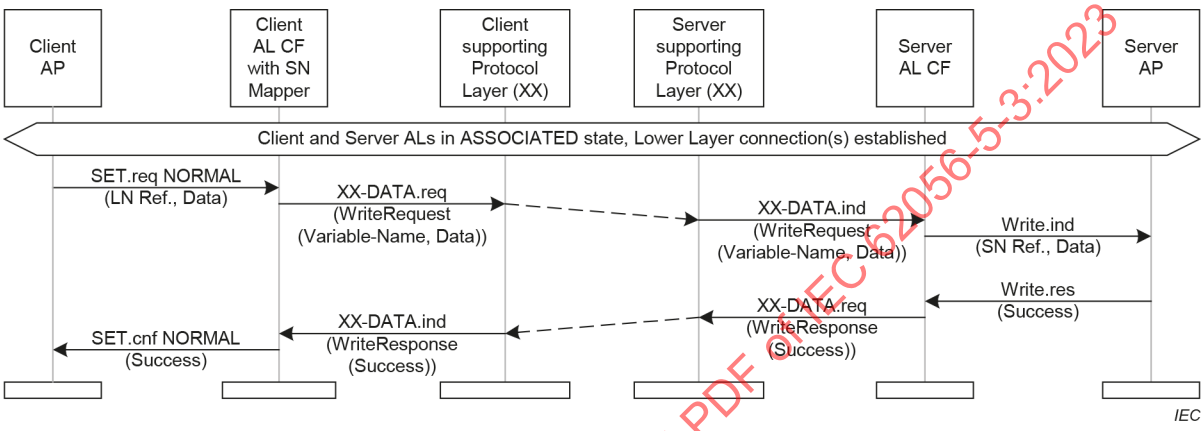



**Figure 58 – MSC of the Write service used for writing an attribute**

Figure 59 shows the MSC of a Write service used to invoke a single method, in the case of success.

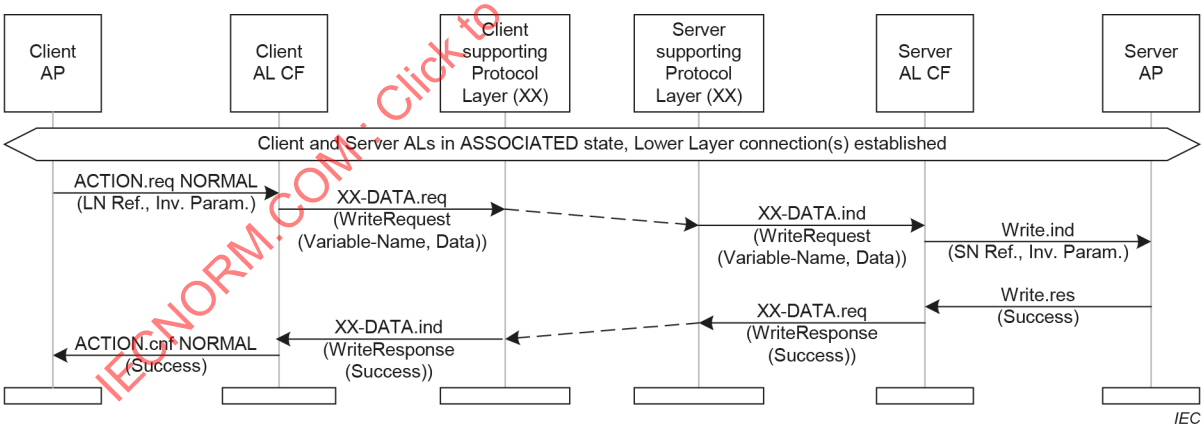



**Figure 59 – MSC of the Write service used for invoking a method**

Figure 60 shows the MSC of a Write service for writing a single attribute with the result returned in three blocks using the service-specific block transfer mechanism.

Alternatively, the general block transfer mechanism can be used.

The process of preparing and transporting the long data is essentially the same as in the case of the SET and ACTION services:

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

If an error occurs, the server should return a Write.response service primitive with the Data_Access_Error carrying appropriate diagnostic information; for example data-block-number-invalid.

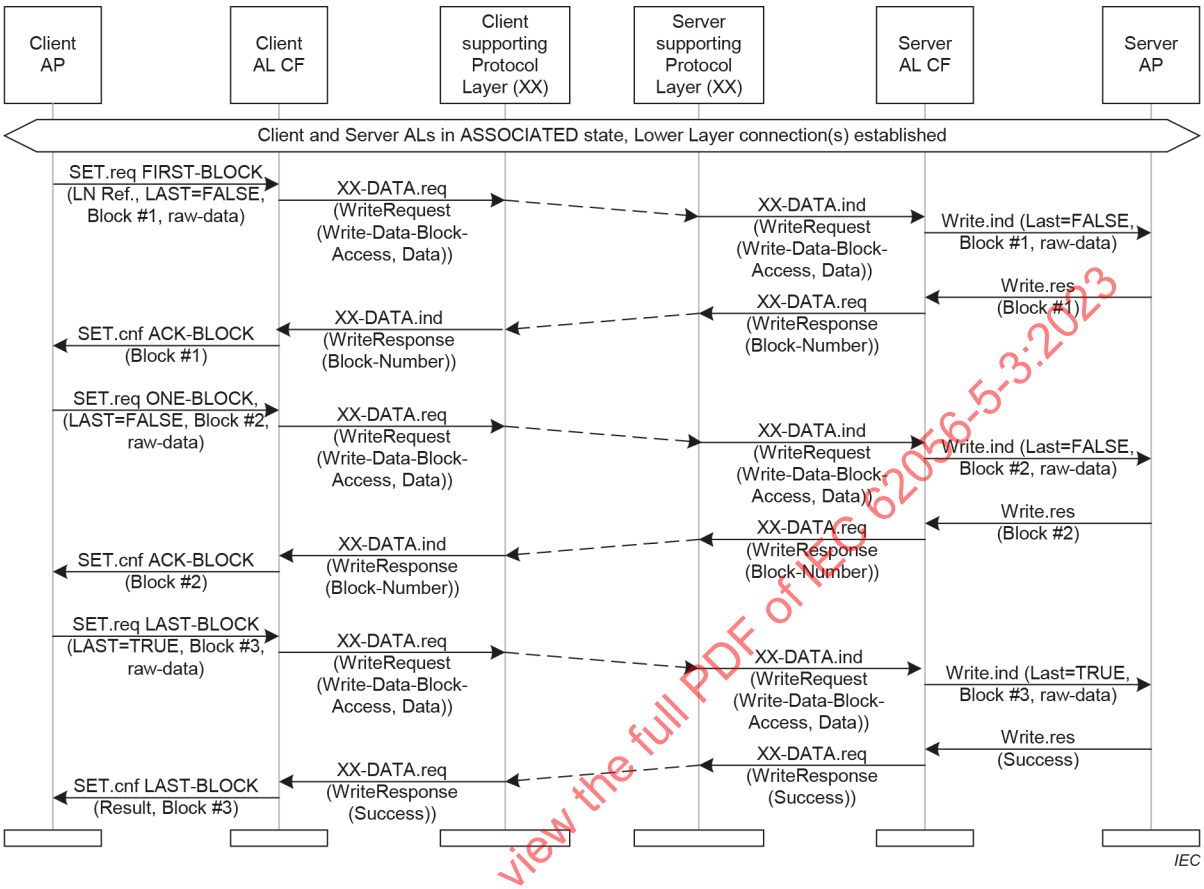



**Figure 60 – MSC of the Write Service used for writing an attribute, with block transfer**

### 7.3.11 Protocol for the UnconfirmedWrite service

This service may be invoked only when an AA has already been established. Depending on the communication profile, the APDU corresponding to the request may be transported using the connection-oriented (CO) or connectionless data (CL) services of the supporting protocol layer.

As explained in 6.16, the UnconfirmedWrite service may be used either to write (a) COSEM object attribute(s), or to invoke (a) method(s) when no return parameters are expected:

- in the first case, the SET.request service primitives are mapped to UnconfirmedWrite.request primitives. The mapping and the corresponding SN APDUs are shown in Table 75;
- in the second case, the ACTION.request service primitives are mapped to UnconfirmedWrite.request primitives. The mapping and the corresponding SN APDUs are shown in Table 76.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table 75 – Mapping between the SET and the UnconfirmedWrite services**

| From SET.request of type | To UnconfirmedWrite.request | SN APDU |
|---|---|---|
| NORMAL | UnconfirmedWrite.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Variable_Name I Parameterized_Access; | UnconfirmedWriteRequest::= (variable-name I parameterized-access, data) |
| WITH-LIST | UnconfirmedWrite.request ({Variable_Access_Specification}, {Data})<br>Variable_Acces_Specification = Variable_Name I Parameterized_Access; | UnconfirmedWriteRequest::= ({variable-name I parameterized-access}, {data}) |

**Table 76 – Mapping between the ACTION and the UnconfirmedWrite services**

| From ACTION.request of type | To UnconfirmedWrite.request | SN APDU |
|---|---|---|
| NORMAL | UnconfirmedWrite.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Variable_Name;<br>Data = method invocation parameters or null data | UnconfirmedWriteRequest::= (variable-name, data) |
| WITH-LIST | UnconfirmedWrite.request ({Variable_Access_Specification}, {Data})<br>Variable_Acces_Specification = Variable_Name;<br>Data = as above | UnconfirmedWriteRequest::= ({variable-name}, {data}) |

Figure 61 shows the MSC of a Write service used to write the value of a single attribute, in the case of success.

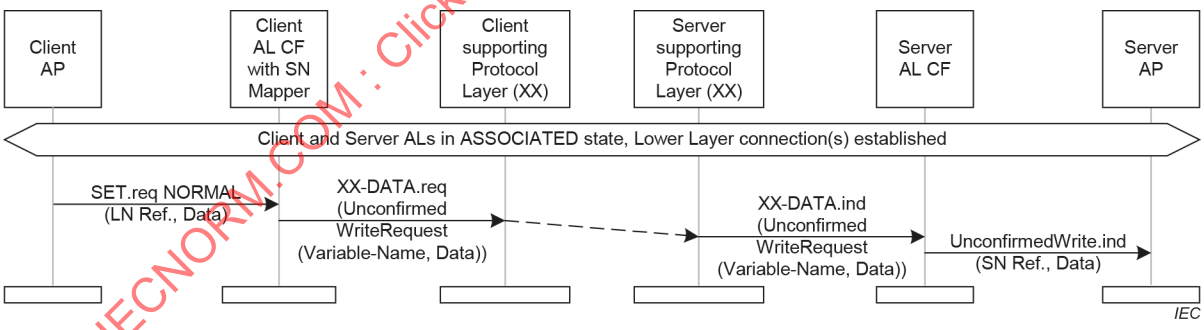



**Figure 61 – MSC of the Unconfirmed Write service used for writing an attribute**

When the service parameters are long, the general block transfer mechanism can be used.

### 7.3.12 Protocol for the InformationReport service

The protocol for the InformationReport service, specified in 6.17, is essentially the same as that of the EventNotification service, see 7.3.8.

As, unlike the EventNotification service, the InformationReport service does not contain the optional Application_Addresses parameter, the information report is always sent by the Server Management Logical Device to the Client Management AP.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Upon invocation of the InformationReport.request service, the server AP builds an InformationReportRequest APDU. This APDU is sent from the SAP of the management logical device to the SAP of the client management device, using data services of the lower layers, in a non-solicited manner, at the first available opportunity.

The possibilities to send out this APDU depend on the communication profile and the connection status of the lower layers. Therefore, the protocol of the InformationReport service is further discussed in Annex A.

The InformationReport service may carry several attribute names and their contents. On the other hand, the EventNotification service specified in 6.11 contains only one attribute reference. Therefore, when the InformationReportRequest APDU contains more than one attribute, it shall be mapped to several EventNotification.ind services, as shown in Table 77.

**Table 77 – Mapping between the EventNotification and InformationReport services**

| EventNotification.ind (one or more) | InformationReport.ind |
|---|---|
| Time (optional) | Current-time (optional) |
| COSEM_Class_Id,<br>COSEM_Object_Instance_Id,<br>COSEM_Object_Attribute_Id | Variable_Name {Variable_Name} |
| Attribute_Value | Data {Data} |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 7.3.13 Protocol of general block transfer mechanism

#### 7.3.13.1 General

The general block transfer (GBT) mechanism can be used to carry any xDLMS APDU when the service parameters are long i.e. their encoded form with the protection overhead, if any, is longer than the Max Receive PDU Size of the peer negotiated. In this case, the AL uses one or more General-Block-Transfer (GBT) xDLMS APDUs to transport such long APDUs

The service primitive invocations may be complete including all the service parameters or partial including only one part of the service parameters. Using complete or partial service invocations is left to the implementation.

Following the reception of a service .request / .response service primitive from the AP, the AL:

- builds the APDU that carries the service primitive;
- when ciphering is required it applies the protection as required by the Security_Options and builds the appropriate ciphered APDU;
- when the resulting APDU is longer than the negotiated max APDU size, then the AL uses the GBT mechanism to send the complete message in several GBT APDUs.

However, there is no direct relationship between partial invocations and the GBT APDUs sent. The AL may apply the protection using complete or partial service invocations.

Following the reception of GBT APDUs from a remote party, the AL:

- assembles the block-data fields of the GBT APDUs received together;
- when the resulting complete APDU is ciphered, it checks and removes the protection;
- it invokes the appropriate service primitive, passing the additional Security_Status, the General_Block_Transfer_Parameters and the Protection_Element.

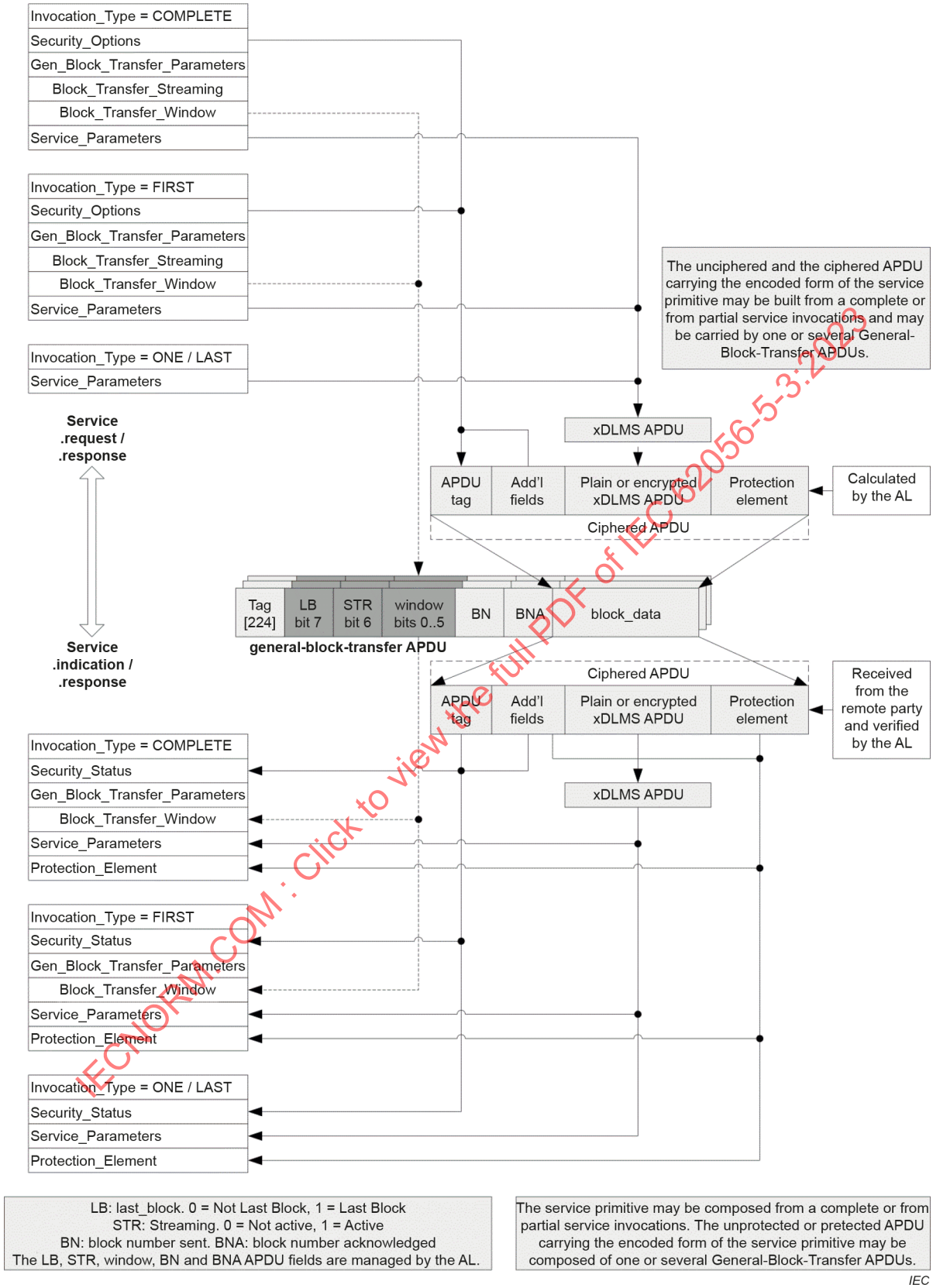
However, there is no direct relationship between the GBT APDUs received and the partial service invocations. The AL may verify and remove the protection processing the GBT APDUs or processing the complete, assembled APDU.

See also Figure 36.

A confirmed message exchange may be started without or with using GBT. However, if one party sends a request or a response using GBT, the other party shall follow. The parties continue then using GBT until the end, i.e. until the complete response will have been received.

Streaming of blocks is managed by the AL taking into account the GBT parameters passed from the local AP to the AL – see 6.5 – and the fields of GBT APDUs – see Figure 62 – received from the remote AL.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

NOTE Applying and checking/removing cryptographic protection on APDUs is independent from the GBT process. It is included here for completeness.

**Figure 62 – Partial service invocations and GBT APDUs**

The various service invocation types – COMPLETE, FIRST-PART, ONE-PART or LAST-PART – and the relationship between these invocations, the service parameters and the fields of the ciphered APDUs and the General-Block-Transfer (GBT) APDUs are shown in Figure 62.

The Block_Transfer_Streaming (BTS) parameter is passed by the AP to the AL to indicate that the AL can send blocks in streams, i.e. without waiting for a confirmation of each block received by the remote party. This parameter is not included in the APDU.

The Block_Transfer_Window (BTW) parameter indicates the size of the streaming window supported, i.e. the maximum number of blocks that can be received. The Block_Transfer_Window parameter of the other party may be known *a priori* by the parties. However, the window size is managed by the AL: it can use a lower value, for example during lost block recovery.

NOTE 1 This relationship is indicated using a dotted line in Figure 62 between the Block_Transfer_Window parameter and the window field of the APDU.

In the case of unconfirmed services the Block_Transfer_Streaming parameter shall be set to FALSE and Block_Transfer_Window shall be set to 0. This indicates to the AL that it shall send the encoded form of the whole service primitive in as many GBT APDUs as needed without waiting for confirmation of the blocks sent.

The use of the fields of the GBT APDU is specified below:

- the last-block (LB) bit indicates if the block is the last one (LB = 1) or not (LB = 0) For the management of the last-block field see 7.3.13.4.3.2;
- the streaming bit indicates if streaming is in progress (STR = 1) or finished (STR = 0). When streaming is finished, the remote party shall confirm the blocks received. However, the client does not confirm the reception of the blocks received in the last stream. Blocks received in an unconfirmed GBT are not confirmed either. When the Block_Transfer_Streaming parameter has been set to FALSE, the streaming bit shall be also set to 0;
- the window field indicates the number of blocks that can be received by the party sending the APDU. Its maximum value is equal to the Block_Transfer_Window parameter passed by the AP to the AL. Note, that the AL may use a lower value during lost block recovery. In the case when the GBT APDUs carry an unconfirmed service (BTS = FALSE, BTW = 0; see above), the value of the window shall be 0 indicating that no GBT APDUs received shall be confirmed (and hence no lost blocks can be recovered);
- the block-number (BN) field indicates the number of the block sent. The first block sent shall have block-number = 1. Block-number shall be increased with each GBT APDU sent, even if block-data (BD) is empty. However, during lost block recovery a block number may be repeated;
- the block-number-acknowledged (BNA) field indicates the number of the block acknowledged. If no blocks have been lost, it shall be equal to the number of the last block received. However, if one or more blocks are lost, it shall be equal to the number of the block up to which no blocks are missing;
- the block-data (BD) field carries one part of the xDLMS APDU that is sent using the GBT mechanism.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 7.3.13.2 The GBT procedure

#### 7.3.13.2.1 Overview

The GBT procedure is shown in Figure 63. It can be:

- confirmed, to carry APDUs corresponding to the service primitives of any confirmed xDLMS service;
- unconfirmed, to carry an APDU corresponding to an unconfirmed service primitive from the client to the server or to carry an APDU corresponding to an unsolicited service request from the server to the client.

NOTE GBT is often used with the DataNotification and Access services.

The GBT procedure is essentially the same on the client and on the server side. It is called by the AL when:

a) a service primitive – .request or .response – is invoked by the AP and either:
   - Invocation_Type = COMPLETE or FIRST-PART with the GBT parameters BTS and BTW present; or
   - the encoded and cryptographically protected APDU to be sent is too long to fit into the maximum APDU size negotiated;
b) a GBT APDU is received from the peer.

The GBT procedure comprises three sub-procedures:

1) *Send GBT APDU stream,* see 7.3.13.4;
2) *Process GBT APDU,* see 7.3.13.5;
3) *Check RQ and fill gaps,* see 7.3.13.6.

These sub-procedures are called by the AL as described in 7.3.13.2.2 and 7.3.13.2.3.

The model of the GBT procedure uses a Send Queue SQ and a Receive Queue RQ. Each block B in the queues has a block number BN, a flag LastBlock LB and a payload BlockData BD (that may be empty). The queues are ordered by BN.

The SQ is filled by the AL after processing the service primitives invoked by the AP i.e.:

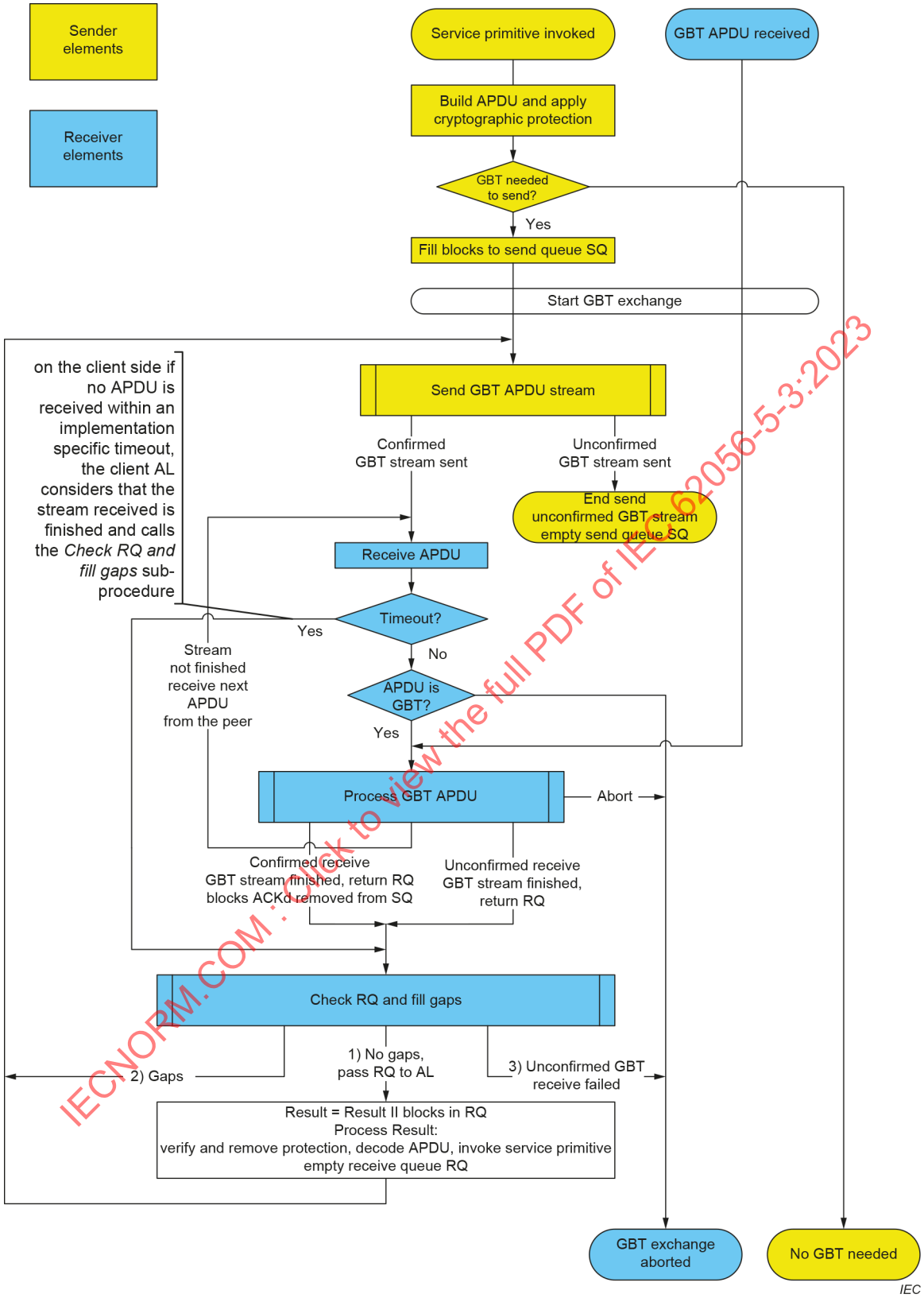
- building the APDU;
- applying cryptographic protection as stipulated by the protection parameters;
- splitting the protected APDU into blocks.

For all service primitives, partial service invocations may be used as specified in 6.5, and 7.3.13. However, this has no consequence for the GBT protocol. The service Invocation_Type may be COMPLETE, FIRST-PART, ONE-PART or LAST-PART. Each service invocation adds one or more blocks to the SQ. The block number BN is incremented every time a new block is added.

The blocks in the SQ are sent by the *Send GBT APDU stream* sub-procedure.

The RQ is filled by the *Process GBT APDU* sub-procedure. Gaps in the RQ – in the case of a confirmed GBT procedure – are filled by the *Check RQ and fill gaps* sub-procedure.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure 63 – The GBT procedure**

#### 7.3.13.2.2 The confirmed GBT procedure

##### 7.3.13.2.2.1 General

The confirmed GBT procedure may be triggered by a local service invocation or by the reception of a GBT APDU from the peer.

##### 7.3.13.2.2.2 Confirmed GBT procedure triggered by a service invocation

When the AP invokes a service primitive and GBT is needed – see 7.3.13.2.1 a) – the AL fills the blocks to the SQ and invokes the *Send GBT APDU stream* sub-procedure; see 7.3.13.4. When the last block of a GBT stream has been sent the control is given back to the AL.

The AL listens then for an APDU from the peer.

On the client side if no APDU is received within an implementation specific timeout, the client AL considers that the stream received is finished and calls the *Check RQ and fill gaps* sub-procedure.

When an APDU is received the AL checks if it is a GBT APDU.

NOTE 1 This is determined from the tag of the APDU received.

If yes, the AL calls the *Process GBT APDU* sub-procedure; see 7.3.13.5. Otherwise, the GBT exchange is aborted. See also 7.3.13.8.

Once the GBT APDU received is processed, the control is given back to the AL:

- if the stream is not yet finished, the control is given back to the AL that waits for the next APDU from the peer;
- if the stream is finished, the AL calls the *Check RQ and fill gaps* sub-procedure; see 7.3.13.6.

If gaps are found the control is given back to the AL that calls again the *Send GBT APDU* stream sub-procedure.

If no gaps are found the control is given back to the AL and the blocks in the RQ are passed for processing:

- the blocks in the RQ are concatenated with result of the previous streams received;
- cryptographic protection is verified and removed;
- the APDU is decoded;
- the appropriate service primitive is invoked.

NOTE 2 Processing the result may lead to service invocations of Invocation_Type COMPLETE, FIRST-PART, ONE-PART or LAST-PART.

The *Send GBT APDU* and the *Process GBT APDU / Check RQ and fill gaps* sub-procedures are called alternately both by the sender and the recipient until the GBT exchange is complete.

Whereas the server acknowledges the blocks in the last GBT stream received, the client does not. Consequently:

- on the client side a confirmed GBT exchange ends when the client LB has been ACKd and the server LB has been passed to the AL (i.e. all blocks have been received without gaps);
- on the server side a confirmed GBT exchange ends when the client LB has been ACKd, the server LB has been sent, and the client has not initiated lost block recovery.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 7.3.13.2.2.3 Confirmed GBT procedure triggered by reception of a GBT APDU from the peer

If a GBT procedure has not yet been started and the AL receives a GBT APDU, it calls the *Process GBT APDU sub-procedure*; see 7.3.13.5. From this point, the GBT procedure is started and continues as described above.

#### 7.3.13.2.3 The unconfirmed GBT procedure

In this case, the sender sends a single stream using the *Send GBT APDU stream* sub-procedure and the recipient receives a single stream using the *Process GBT APDU* sub-procedure.

If a timeout occurs while listening for the next GBT APDU, the AL considers that the stream received from the server is finished and calls the *Check RQ and fill gaps* sub-procedure.

When the complete stream is received the AL calls *the Check RQ and fill gaps sub-procedure*; see 7.3.13.6.

If any blocks are missing, the blocks received are not processed and the unconfirmed GBT procedure ends.

An unconfirmed GBT procedure ends:

- on the sender side, when the last block has been sent;
- on the recipient side, when the last block has been received;
- when one or more gaps are found in the RQ.

### 7.3.13.3 GBT procedure state variables

The AL and the GBT procedure use several state variables, specified in Table 78.

**Table 78 – GBT procedure state variables**

| Name | Use | Initial value |
|---|---|---|
| BNApeer | *Send GBT stream* sub-procedure: – | – |
| | *Process GBT APDU* sub-procedure: The number of the block acknowledged by the peer. The value is taken from Gr.BNA. | – |
| BNAself | *Send GBT stream* sub-procedure: The number of the block acknowledged. The value is determined by the *Check RQ and fill gaps* sub-procedure and it is put in Gs.BNA.<br>For an unconfirmed GBT stream the value is 0. | 0 |
| | *Process GBT APDU* sub-procedure: – | – |
| | *Check RQ and fill gaps* sub-procedure: Set to the B.BN of the block before the first gap found. | – |
| NextBN | The block number of the next block in the SQ. It is incremented every time a new block is added.<br>Note that a new block may be added by the AL or – when the SQ is empty – by the *Send GBT stream* sub-procedure. | 1 |
| STRpeer | *Send GBT stream* sub-procedure: – | |
| | *Process GBT APDU* sub-procedure: Indicates if the peer is streaming or not. The value is taken from Gr.STR. | – |
| STRself | *Send GBT stream* sub-procedure: The value to be put in Gs.STR.<br>For an unconfirmed stream the value is FALSE. | BTS parameter of the service primitive invocation.<br>(COMPLETE or FIRST-PART) (Default FALSE). |
| | *Process GBT APDU* sub-procedure: – | – |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

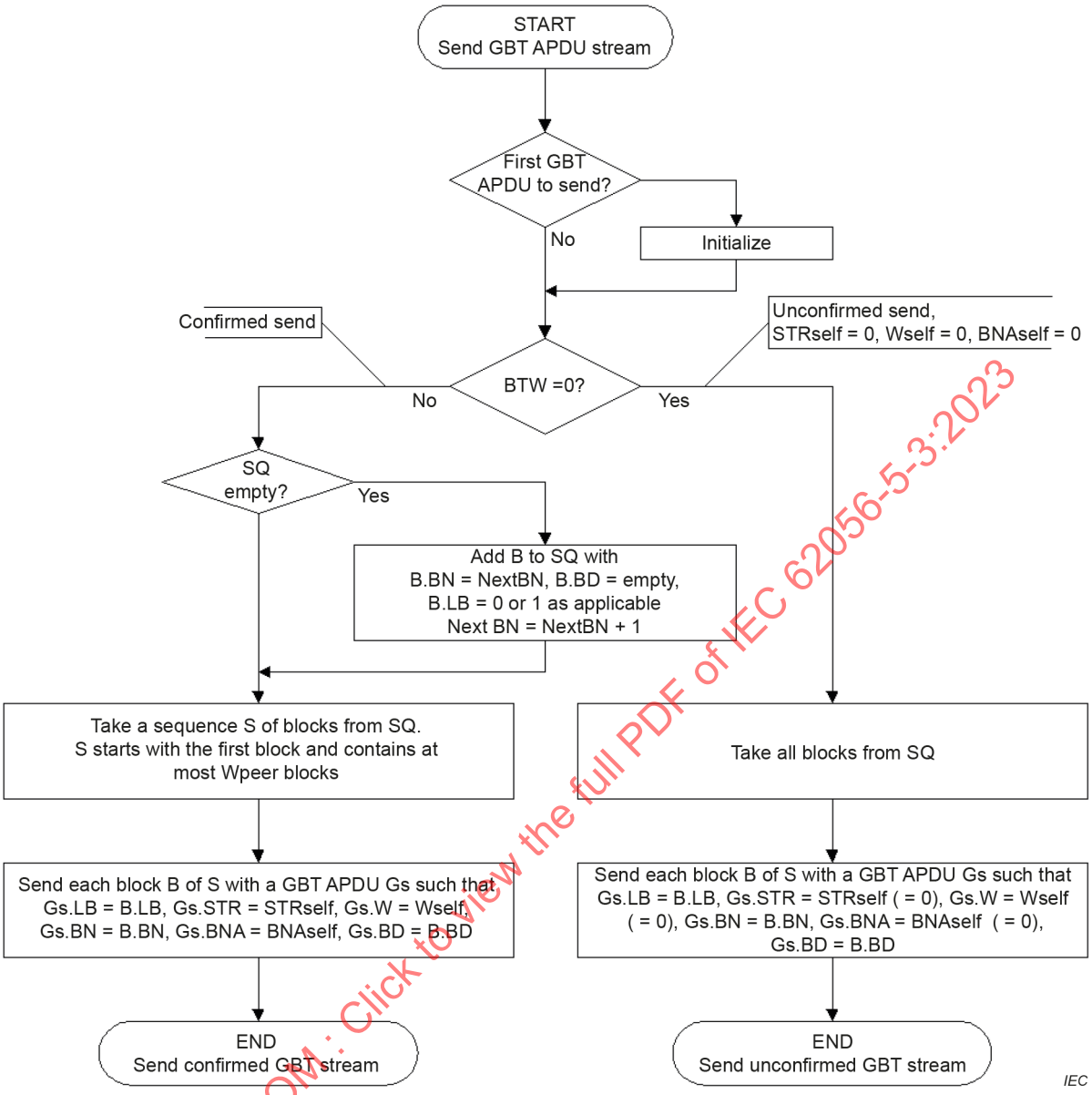
| Name | Use | Initial value |
|---|---|---|
| Wpeer | *Send GBT stream* sub-procedure: The size of the streaming window of the peer, i.e. the maximum number of GBT APDUs that can be sent in a GBT stream. It is taken by the *Process GBT APDU* sub-procedure from Gr.W. | A value known a priori, or if not known, 1. It determines the size of the first GBT stream to be sent. |
| | *Process GBT APDU* sub-procedure: The size of the streaming window of the peer. It is taken from Gr.W and it is used by the *Send GBT stream* sub-procedure for the next GBT stream to be sent. | – |
| Wself | *Send GBT stream* sub-procedure: The size of the streaming window i.e. the maximum number of GBT APDUs the sender can receive from the peer in one stream. It is put in Gs.W.<br>For an unconfirmed stream the value is 0. | BTW parameter of the service primitive invocation.<br>(COMPLETE or FIRST-PART)<br>Default = 1. |
| | *Process GBT APDU* sub-procedure: The number of GBT APDUs the recipient can receive in one GBT stream. | For a confirmed procedure, BTW parameter of the recipient (a value shared by the AP and the AL).<br>For an unconfirmed procedure it is an implementation specific value. |
| | *Check RQ and fill gaps* sub-procedure: This is set to the number of missing blocks to be recovered. This can be any value from 1 up to the number of blocks to be recovered. | – |

#### 7.3.13.4 Send GBT APDU stream sub-procedure

##### 7.3.13.4.1 General

The *Send GBT stream* sub-procedure – shown in Figure 64 – is used both for confirmed and unconfirmed GBT procedures.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




See also 7.3.13.4.3.2 to determine the management of B.LB.

**Figure 64 – Send GBT APDU stream sub-procedure**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

It sends a sequence S of blocks B to the peer. It is invoked by the AL when there are blocks in the SQ, or by the GBT procedure itself to acknowledge the blocks received in a stream and to send further blocks from the SQ.

### 7.3.13.4.2 Initialization

Before the first GBT APDU is sent, the initial values shall be set according to Table 78.

During a confirmed GBT exchange the value of the BNApeer state variable is updated by the *Process GBT APDU* sub-procedure. The value of the Wself and BNAself state variables are updated by the *Check RQ and fill gaps* sub-procedure.

For an unconfirmed GBT procedure the number of blocks the receiver is able to receive should be known a priori. The value of the STRself, Wself and BNAself state variables shall be set to 0.

#### 7.3.13.4.3 Confirmed GBT stream send

##### 7.3.13.4.3.1 General

For a confirmed GBT exchange, the AP invokes the service primitive with BTW > 0.

First, it is checked if the SQ is empty. On the client side this occurs when all blocks have been sent and acknowledged. In the server side this occurs if blocks are not yet available or all blocks have been sent and acknowledged. If the SQ is empty, an empty block is added to the SQ and NextBN is incremented.

##### 7.3.13.4.3.2 Last block management

The Last Block bit LB is controlled as follows:

- the client sends each GBT APDU with LB = 0 unless it has no further blocks to send. When the last block is reached LB is set to 1. Note that, according to the model, a block is removed from the SQ when it is acknowledged by the peer, and therefore, if the server needs to recover blocks earlier than the last block, those blocks will be sent with LB = 0. Once all blocks *including* the last block to be re-sent have been acknowledged by the server then any following GBT APDUs required for the purpose of acknowledging GBT APDUs sent by the server will be sent with an empty block and LB = 1;
- the server sends each GBT ADPU with LB = 0 unless the last block of the client is received and the block is the last block for the server to send.

##### 7.3.13.4.3.3 Process

The sub-procedure takes a sequence S of blocks from the SQ. The sequence S starts with the first block and contains at most Wpeer blocks. Each block B of S is sent with a GBT APDU Gs such that Gs.LB = B.LB, Gs.STR = STRself, Gs.W = Wself, Gs.BN = B.BN, Gs.BNA = BNAself and Gs.BD = B.BD.

The sub-procedure is finished once all blocks in S have been sent. The last GBT APDU Gs of the stream shall be sent with Gs.STR = FALSE. Gs.STR shall also be set to FALSE if B.LB = TRUE.

The blocks acknowledged by the peer are removed from the SQ by the *Process GBT APDU* sub-procedure. Therefore when all blocks have been sent and acknowledged, the SQ is empty except that the blocks sent with the last stream of the server are not acknowledged by the client, thus they remain in SQ and have to be removed when the end of the GBT exchange is detected.

#### 7.3.13.4.4 Unconfirmed GBT send

For an unconfirmed GBT send, the AP invokes the service primitive with BTW = 0.

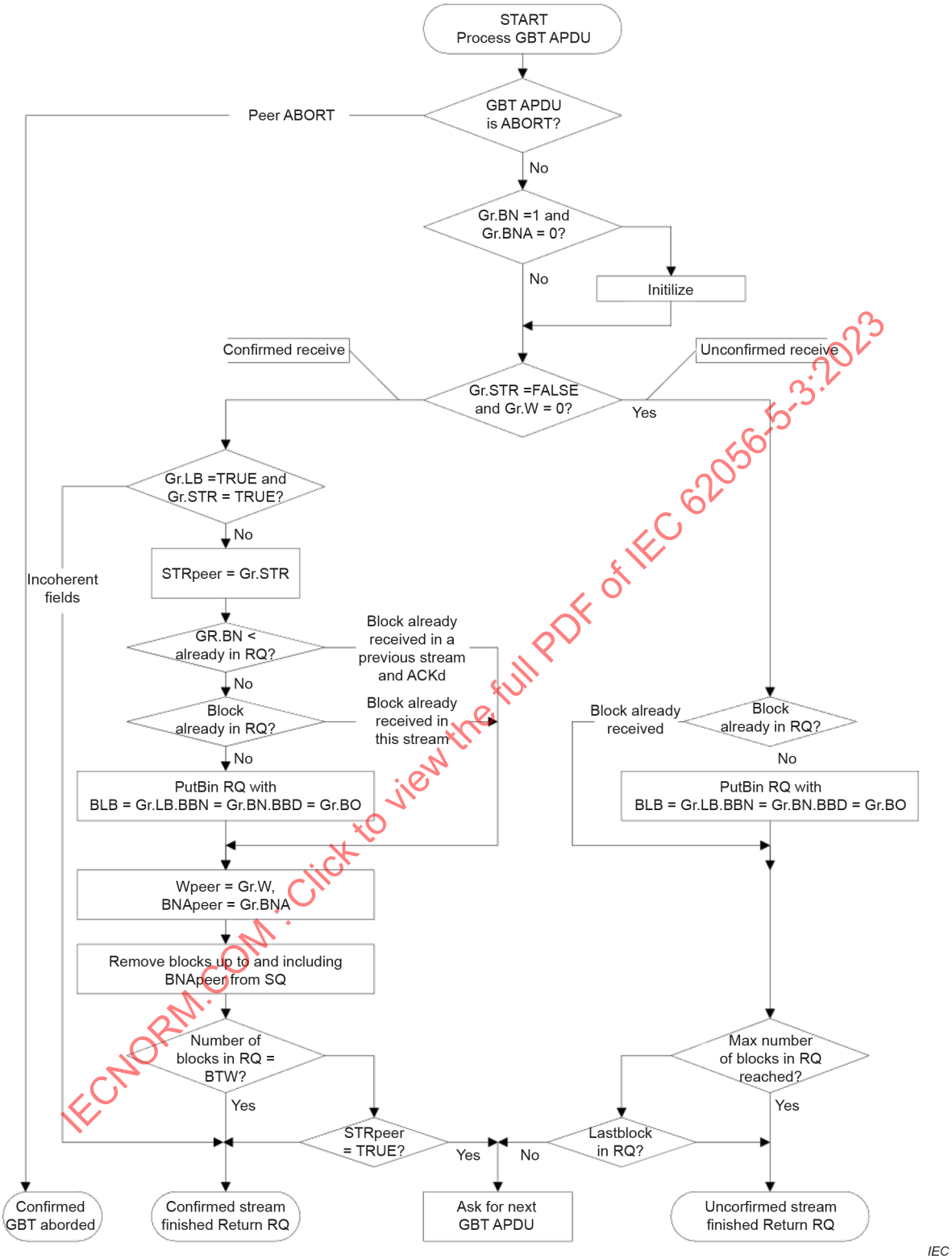
The sub-procedure sends the blocks in the SQ in a single stream. The SQ is emptied when the last block has been sent.

### 7.3.13.5 Process GBT APDU sub-procedure

#### 7.3.13.5.1 General

The *Process GBT APDU* sub-procedure is shown in Figure 65.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure 65 – Process GBT APDU sub-procedure**

First, the value of Gr.STR and Gr.BN is checked. If the GBT APDU is an ABORT GBT APDU – see 7.3.13.8 – this indicates that the peer wants to abort the exchange.

Then, the value of Gr.BN and Gr.BNA is checked. If Gr.BN = 1 and Gr.BNA = 0, this indicates the start of a new GBT exchange and the initial values shall be set according to Table 78.

Then, the value of Gr.STR and Gr.W is checked. If Gr.STR = FALSE and Gr.W = 0, this indicates that the GBT procedure is unconfirmed; otherwise it is confirmed.

#### 7.3.13.5.2 Processing GBT APDUs in a confirmed GBT procedure

First, the values of Gr.LB and Gr.STR are checked. If Gr.LB = TRUE and Gr.STR = TRUE the values are incoherent: streaming should be finished when the last block is sent.

Then STRpeer is set to Gr.STR.

Then a check is made to determine if the block has already been received in a previous stream (Gr.BN ≤ BNAself) or in the current stream (block is already in RQ; this is determined if a value of B.BN in the RQ is equivalent to Gr.BN). In either cases, the duplicate block is not put to RQ.

Otherwise, the block B it is put to the RQ with B.LB = Gr.LB, B.BN = Gr. BN B.BD = Gr.BD.

Wpeer is set to Gr.W and BNApeer is set to Gr.BNA.

The blocks up to and including BNApeer are removed from the SQ.

If the number of blocks in the RQ reaches Wself (BTW parameter of the recipient, a value shared by the AP and the AL), then the stream is considered to be finished.

Otherwise, the value of STRpeer is checked:

- if it is TRUE, then the control is given back to the AL that waits for the next APDU;
- if it is FALSE, the stream is considered to be finished.

When the stream is finished no more GBT APDUs are listened for. Any GBT APDUs received are considered as overflow and shall not be acknowledged.

The blocks in the RQ are passed to the AL that calls the *Check RQ and fill gaps* sub-procedure.

#### 7.3.13.5.3 Processing GBT APDUs in an unconfirmed GBT procedure

First, the value of Gr.BN is checked. If the block is already in the RQ it is discarded. Otherwise, it is put to the RQ with B.LB = Gr.LB, B.BN = Gr.BN B.BD = Gr.BD.

If the number of the blocks in the RQ reaches the maximum number of blocks that can be accommodated, then the stream is considered to be finished.

Otherwise, it is checked if the last block (with B.LB = TRUE) is in the RQ:

- if no, the control is given back to the AL that waits for the next APDU;
- if yes, the stream is considered to be finished.

When the stream is finished no more GBT APDUs are listened for. Any GBT APDUs received are considered as overflow and shall not be acknowledged. The blocks in the RQ are passed to the AL, the RQ is emptied and the unconfirmed receive GBT procedure is ended.

### 7.3.13.6 Check RQ and fill gaps sub-procedure

#### 7.3.13.6.1 General

The *Check RQ and fill gaps* sub-procedure is shown in Figure 66.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

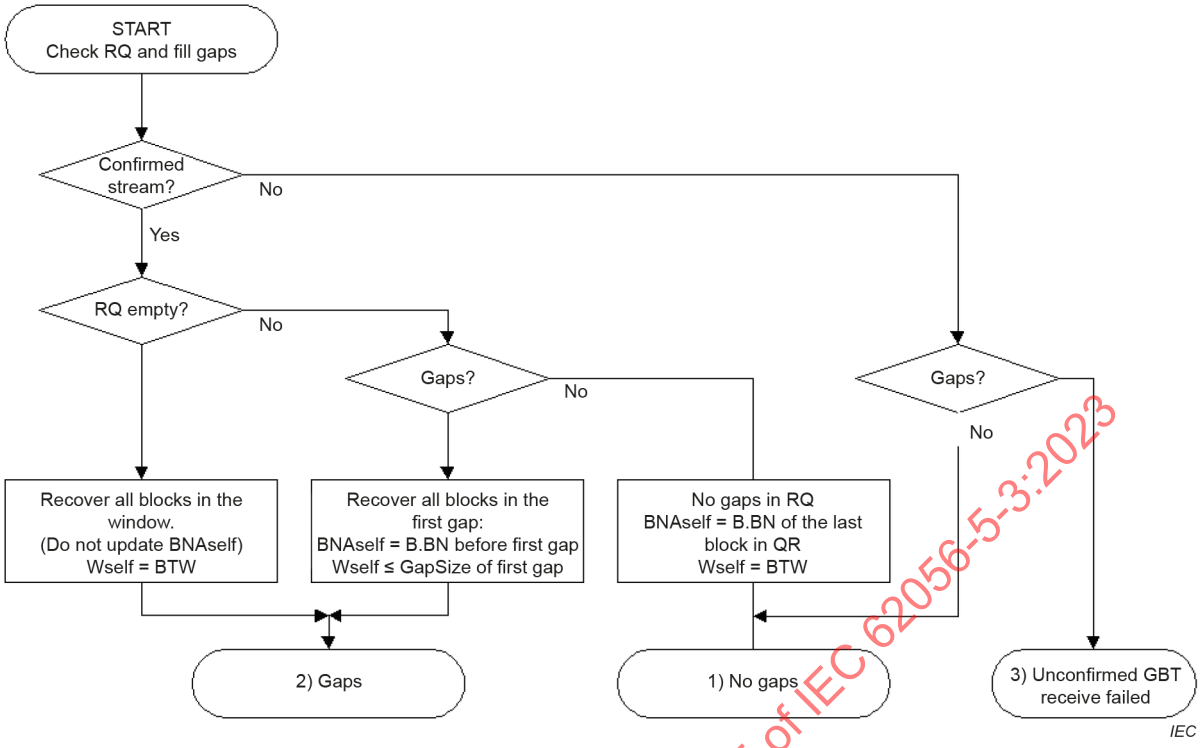



**Figure 66 – Check RQ and fill gaps sub-procedure**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 7.3.13.6.2 Confirmed GBT procedure

This subclause describes one possible recovery strategy. Other strategies that result in the recovery of the missing blocks may also be used.

If the GBT stream is confirmed, first it is checked if the RQ is empty. This occurs if no GBT APDUs have been received or all GBT APDUs in the stream have been discarded. Notice that the stream received may contain only one GBT APDU. In this case, BNAself is not updated, Wself is set to BTW and the control is given back to the AL.

If the RQ is not empty, it is checked for gaps:

- if gaps are found then the GBT procedure attempts to recover the blocks in the first gap found: BNAself is set to B.BN before the first gap, Wself is set to the number of blocks to be recovered and the control is given back to the AL that calls the *Send GBT APDU stream* sub-procedure;
- if no gaps are found, then BNAself is set to B.BN of the last block in the RQ, Wself is set to BTW and the control is given back to the AL.

#### 7.3.13.6.3 Unconfirmed GBT procedure

If the GBT procedure is unconfirmed, then the RQ is checked for gaps. If no gaps are found, then the control is given back to the AL and the blocks in the RQ are passed to the AL. Otherwise all blocks are discarded.

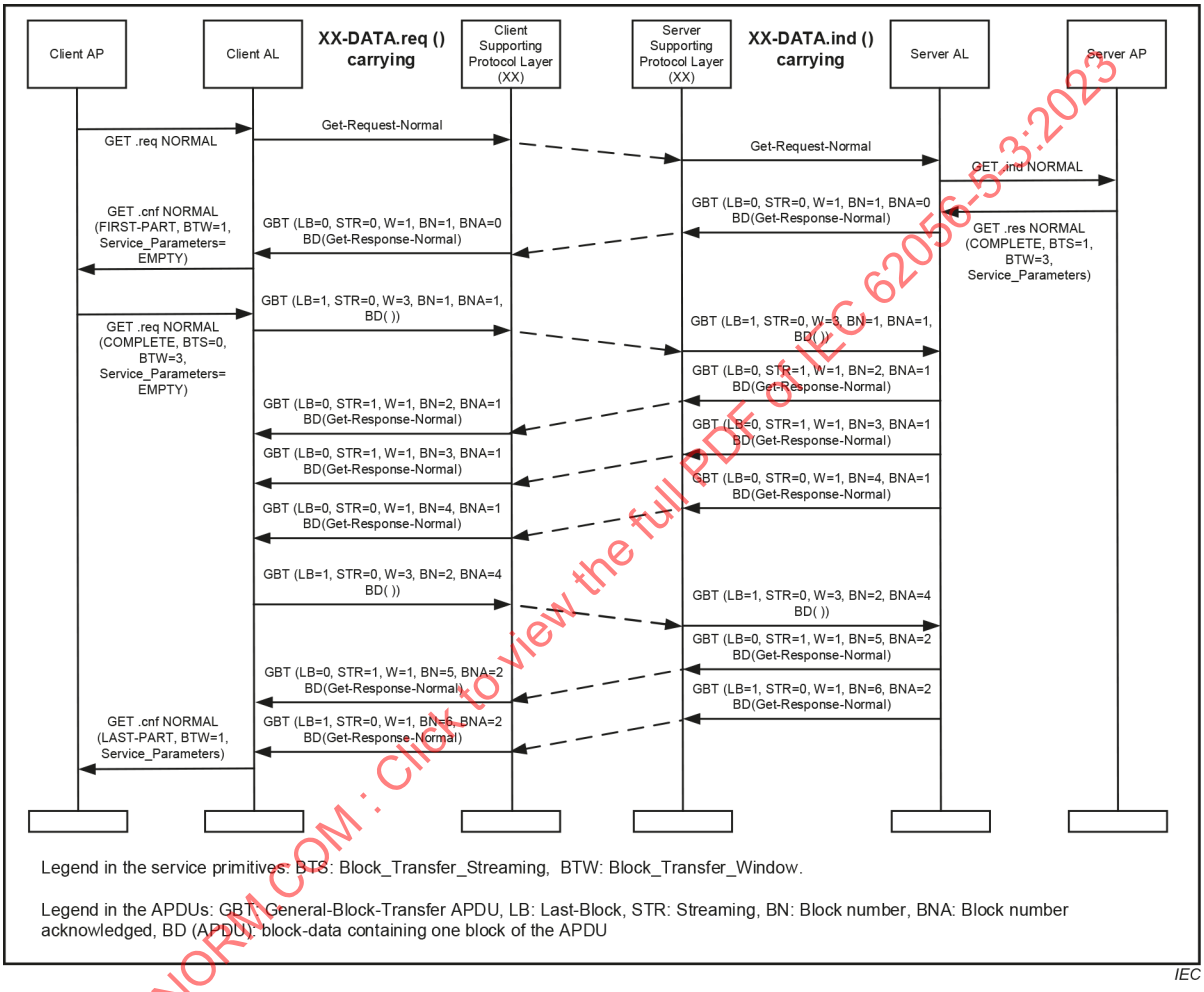
### 7.3.13.7 GBT protocol examples

The protocol of the GBT mechanism is further explained with the help of Figure 67 through Figure 73. In these examples, it is assumed that both parties support GBT and six blocks are required to transfer the complete response or request (except in the DataNotification example, where four blocks are required).

NOTE 1 In these examples the service specific block transfer mechanism is not used.

Abbreviations used in the Figures:

- BTS: Block_Transfer_Streaming, BTW: Block_Transfer_Window;
- GBT: general-block-transfer APDU;
- LB: last-block;
- STR: Streaming;
- BN: block-number, BNA: block-number-acknowledged;
- BD (APDU): block-data containing one block of the APDU.




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure 67 – GET service with GBT, switching to streaming**

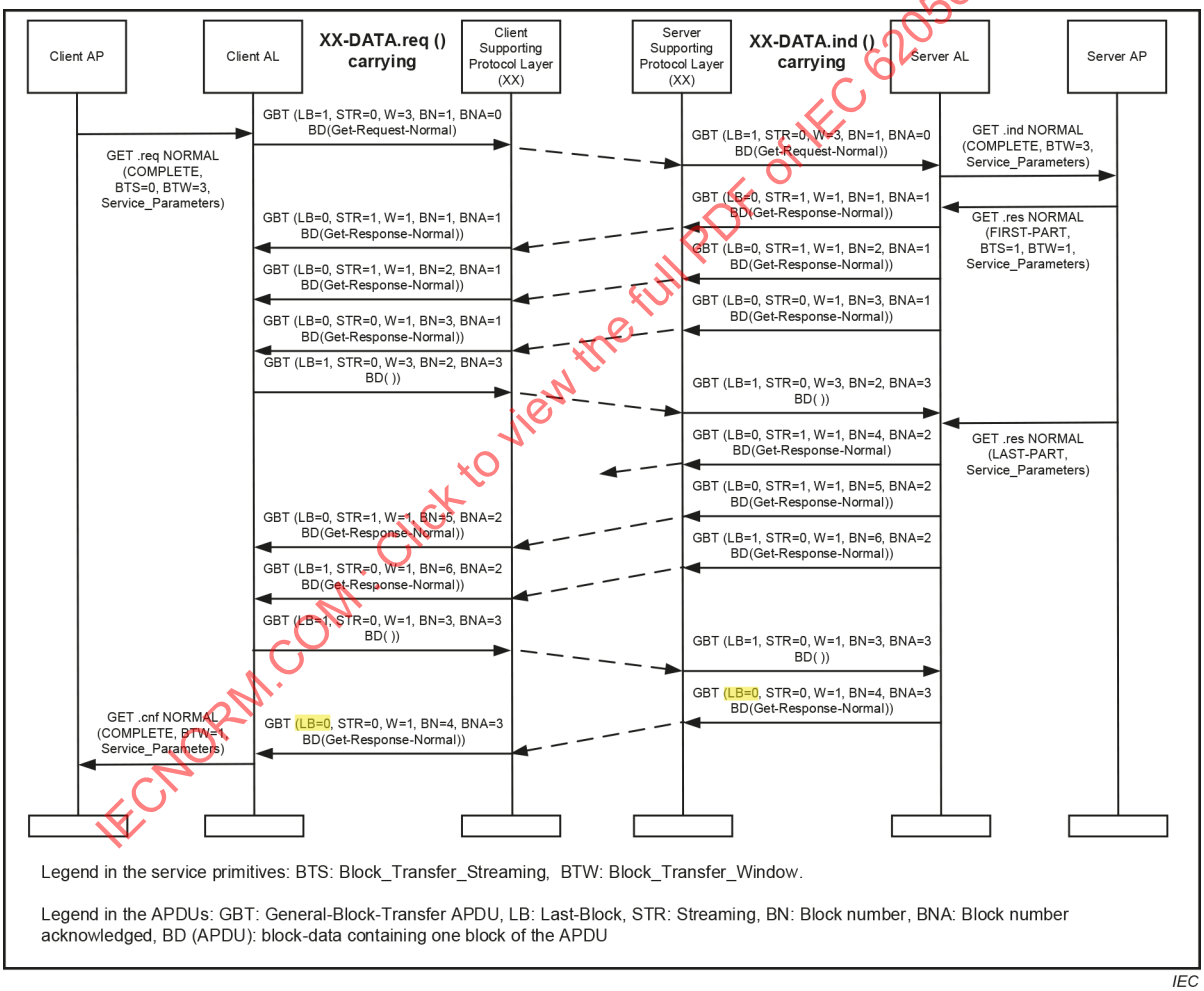
Figure 67 shows a GET service using GBT. After receiving the first GBT APDU, the client informs the server that it supports streaming. The server switches then to streaming. The process is as follows:

- the client AP invokes a GET.request NORMAL service primitive, without additional service parameters. The client AL sends the request in a Get-Request-Normal APDU;
- the GET.response service parameters are long so the server AP invokes a GET.response NORMAL service primitive with additional service parameters: Invocation_Type = COMPLETE, BTS = 1, BTW = 1 meaning that the server allows sending block streams, but it does not accept block streams from the client. The server AL sends a GBT APDU, containing the first block of the response;
- the client AL invokes a GET.confirm NORMAL service primitive, Invocation_Type = FIRST-PART, BTW = 1. The Service_Parameters are empty. This informs the AP that the response from the server is going to be long;

- the client AP invokes a GET.request NORMAL service primitive, with Invocation_Type = COMPLETE, BTS = 0, BTW = 3, to advertise its capabilities to receive block streams. Note that the Service_Parameters are empty, as these have already been passed in the first GET.request NORMAL service invocation. The client AL sends a GBT APDU. The last-block bit in the APDU is set to 1 and the streaming bit is set to 0 since the client has no blocks to send;
- the server sends then the 2nd (STR = 1, BN = 2), 3rd (STR = 1, BN = 3) and 4th (STR = 0, BN = 4) blocks;
- the client AL sends a GBT APDU to confirm the reception of the 2nd, 3rd and 4th block (LB = 1, STR = 0, W = 3, BN = 2, BNA = 4);
- the server AL sends the 5th (STR =1, BN = 5) and the 6th, last block (LB = 1, STR = 0, BN = 6);

NOTE 2 The last block sent by the server is not confirmed. However, if it is lost, it can be recovered. See Figure 66.

- the client AL invokes a GET.confirm NORMAL service primitive with Invocation_Type = LAST-PART. The service parameters include the complete response to the GET.request.




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure 68 – GET service with partial invocations, GBT and streaming, recovery of 4th block sent in the 2nd stream**

Figure 68 shows an example of a GET service using GBT, with partial service invocations on the server side and streaming. The client advertises its streaming capabilities in the first request (BTW = 3; BTW > 1 means that block streams can be received). In this example, the 4th block, sent in the second stream by the server is lost but it is recovered using the following process:

- the client AP invokes a GET.request NORMAL service primitive, with Invocation_Type = COMPLETE, BTS = 0, BTW = 3. The client AL sends a GBT APDU with STR = 0, Window = 3. The server AL invokes the GET.indication NORMAL service primitive with Invocation_Type = COMPLETE, BTW = 3;
- the server AP invokes a GET.response NORMAL service primitive with Invocation_Type = FIRST-PART, BTS = 1, BTW = 1. Service_Parameters include the first part of the response. The server AL sends the 1st, 2nd and 3rd block;
- the client AL sends a GBT APDU to confirm the reception of the three blocks. The server AP invokes a GET.response NORMAL service primitive with Invocation_Type = LAST-PART. Service_Parameters include the second, last part of the response (which in this case is the last part). The server AL sends the 4th, 5th and 6th block (LB = 1, STR = 0, BN = 6). However, the 4th block is lost;
- the client AL indicates that the 4th block has not been received by sending a GBT APDU confirming the reception of the 3rd block (STR = 0, window = 1, BNA = 3). Notice that the client AL reduces the window size to 1 to indicate that only one block has to be re-sent;
- the server sends the 4th block again (LB = 0, STR = 0, BN = 4, BNA = 3);
- now that the client has received now all of the blocks, it invokes a GET.confirm NORMAL service primitive with Invocation_Type = COMPLETE, BTW = 1. The Service_Parameters of this invocation contain the complete response to the GET.request.

Figure 69 shows a scenario which is essentially the same as in Figure 68 except that the 4th and 5th blocks are lost and recovered.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

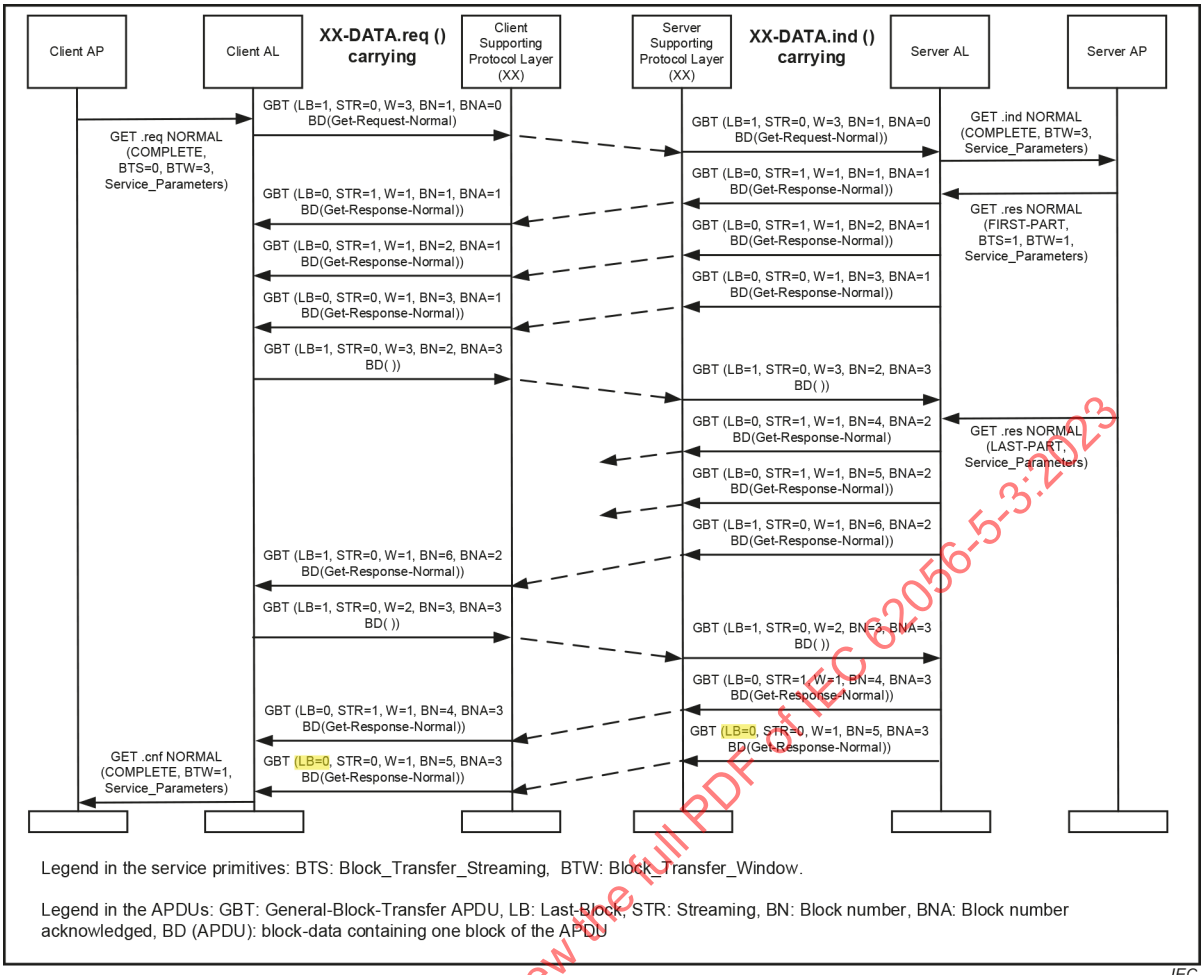



**Figure 69 – GET service with partial invocations, GBT and streaming, recovery of 4th and 5th block**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The process is as follows:

- the client receives the 6th block (LB = 1, STR = 0, BN = 6, BNA = 2);
- the client indicates that the 4th and the 5th blocks have been lost, by sending a GBT APDU with W = 2, BNA = 3, which means that no blocks are missing up to the 3rd block but two blocks have been lost and that the server can send these two using streaming;
- the lost (ie not acknowledged) blocks are then sent by the server. These are the 4th (LB = 0, STR = 1, BN = 4) and 5th (LB = 0, STR = 0, BN = 5). (Note that LB = 0 in the 5th block.) Although it is the last block of the two re-sent blocks, it is not the last block of the original whole message, that was block 6. The original value of LB is preserved during the recovery process.);
- now that the client has received all of the blocks, it invokes a GET.confirm NORMAL service primitive with Invocation_Type = COMPLETE, BTW = 1. The Service_Parameters include the parameters of the complete response to the GET.request.

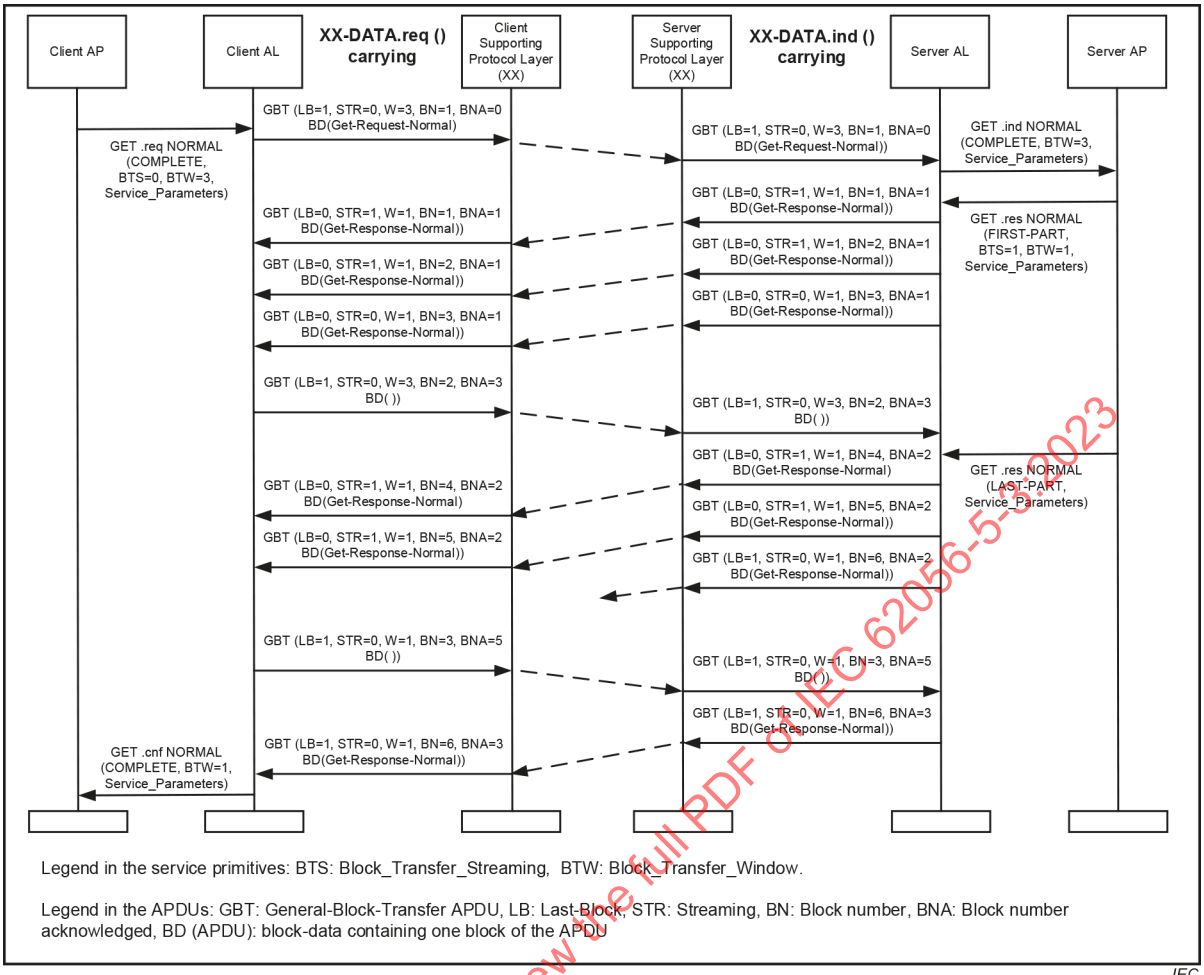
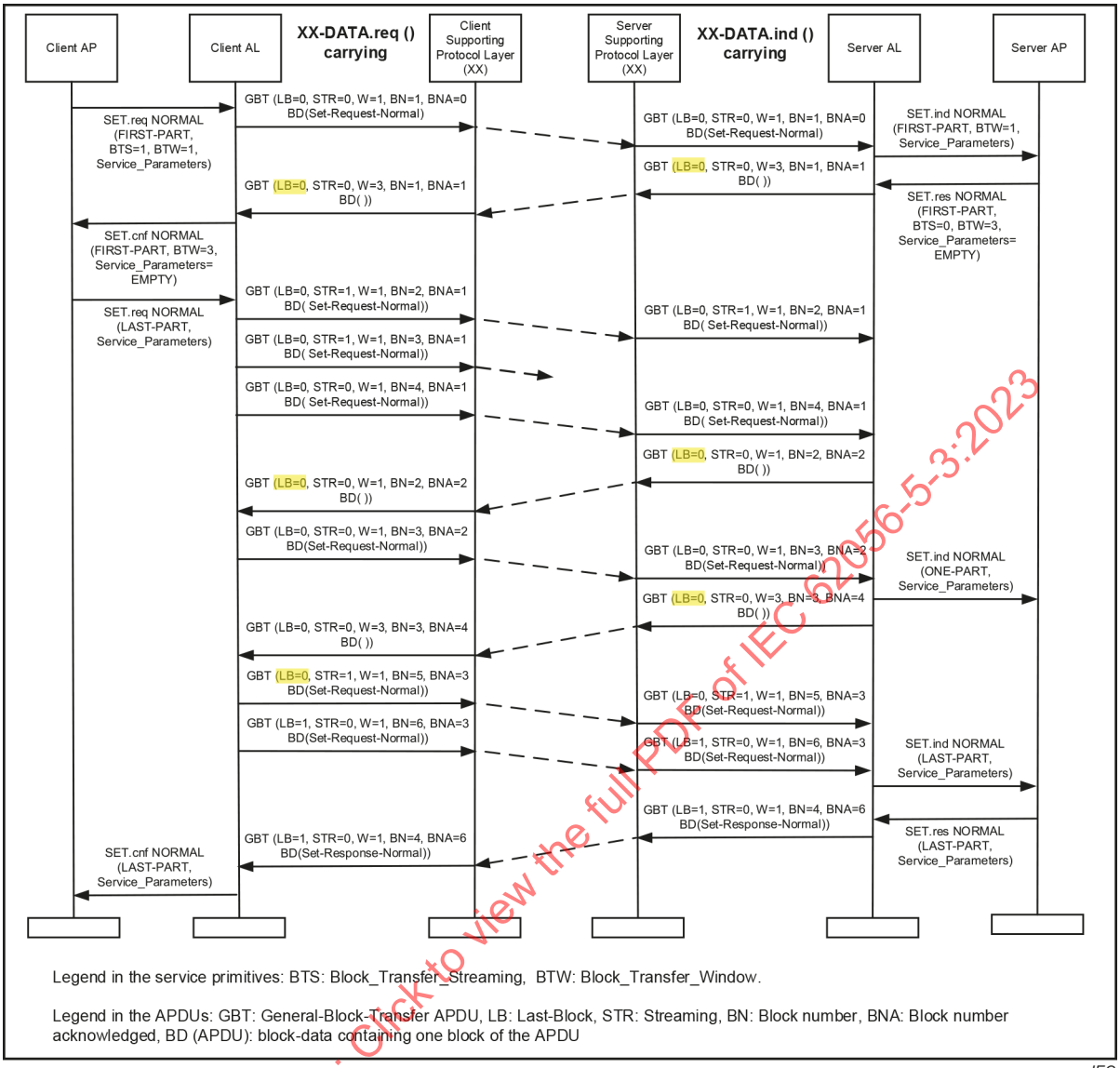



**Figure 70 – GET service with partial invocations, GBT and streaming, recovery of last block**

Figure 70 shows a scenario when the last block sent in the second stream is lost and is recovered. The process is as follows:

- the client receives the 5th block carried by a GBT APDU (LB = 0, STR = 1, BN = 5);
- as this is not the last block, client waits for a period of time which is implementation specific. If the last block has not been received after this time, it sends a GBT APDU (LB = 1, STR = 0, BN = 3, BNA = 5);
- the server then sends the lost (not confirmed) 6th block carried by a GBT APDU (LB = 1, STR = 0, W = 1, BN = 6 and BNA = 3);
- when the client receives this APDU, it invokes a GET.confirm NORMAL service primitive with Invocation_Type = COMPLETE, BTW = 1. The Service_Parameters include the parameters of the complete response to the GET.request.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023



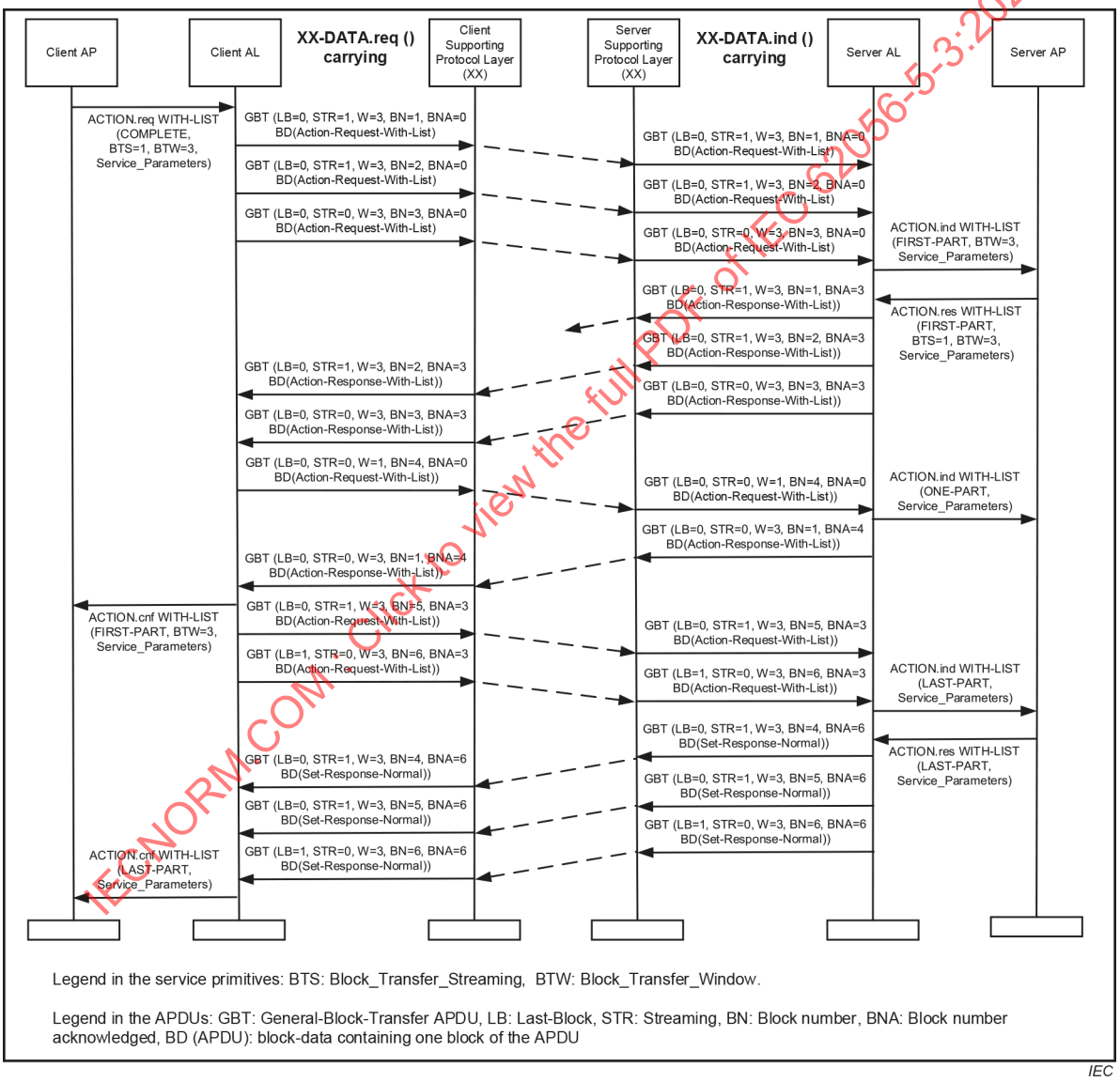

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure 71 – SET service with GBT, with server not supporting streaming, recovery of 3rd block**

Figure 71 shows a SET service with GBT and streaming. In this example, the 3rd block sent by the client is lost and recovered. The process is as follows:

- the client AP invokes a SET.request NORMAL service primitive with Invocation_Type = FIRST-PART, BTS = 1, BTW = 1. The Service_Parameters include the first part of the SET.request. The client AL only sends the first block because it does not know the size of the streaming window supported by the server;
- the server AL invokes a SET.indication NORMAL service primitive with Invocation_Type = FIRST-PART, BTW = 1. The Service_Parameters include the first part of the parameters of the SET.request;
- the server AP responds with a SET.response NORMAL service primitive with Invocation_Type = FIRST-PART, BTS = 0, BTW = 3. The Service_Parameters are empty. The server AL sends a GBT APDU with LB = 0, STR = 0, Window = 3; block-data is empty. This informs the client that the server can receive block streams and the window size = 3. It therefore sends the 2nd, 3rd and 4th block in a stream. However, the 3rd block is lost;
- the server indicates that block 3 is lost by confirming the reception of the 2nd block and reduces the window size to 1 (LB = 0, STR = 0, W = 1, BN = 2, BNA = 2). The client sends then the 3rd block again (LB = 0, STR = 0, BN = 3, BNA = 2);

- the server AL invokes a SET.indication NORMAL service primitive with Invocation_Type = ONE-PART. The server AL confirms the reception of the blocks up to the 4th block (LB = 0, STR = 0, W = 3, BNA = 4). Notice that the window size has been increased again to 3;
- the client then sends the 5th and the 6th block using streaming;
- when the server AL receives the 6th block which is the last block, it invokes a SET.indication NORMAL service primitive with Invocation_Type = LAST-PART. Service_Parameters include the last part of the parameters of the SET.request;
- the server AP invokes a SET.response NORMAL service primitive with Invocation_Type = LAST-PART, with the Service_Parameters containing the result of the set operation(s). This is sent by the server AL in a GBT APDU (LB = 1, BN = 4, BNA = 6). The client AL invokes the SET.confirm NORMAL service primitive with Invocation_Type = LAST-PART. Service_Parameters include the result of the set operations.




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure 72 – ACTION-WITH-LIST service with bi-directional GBT and block recovery**

Figure 72 shows an ACTION-WITH-LIST service with partial service invocations, bidirectional block transfer and streaming. Each party already knows that the other party supports streaming with window size = 3. In this example, the first block sent by the server is lost and recovered. The process is as follows:

- the client invokes and ACTION.request of type WITH-LIST service primitive with Invocation_Type = COMPLETE, BTS = 1, BTW = 3. The client AL sends the first three blocks that carry a part of this request to the server. The server AL invokes and ACTION.indication of type WITH-LIST service primitive with Invocation_Type = FIRST-PART, BTW = 3. Service parameters contain the first part of the request;
- the server AP processes this request and has the first part of the response available. It invokes an ACTION.response of type WITH-LIST service primitive with Invocation_Type = FIRST-PART, BTS = 1, BTW = 3. The server AL sends this in three blocks using streaming. However, the 1st block is lost;
- the client AL asks the server to send the lost 1st block again by not confirming any blocks received. It also sends its 4th block (LB = 0, STR = 0, W = 1, BN = 4, BNA = 0). Notice that the client AL has reduced the window size to 1;
- the server AL invokes an ACTION.indication of type WITH-LIST service primitive with INVOCATION_Type = ONE-PART. Service_Parameters contain one part of the request;
- the server sends the lost 1st block and confirms the 4th block received from the client (BN = 1, BNA = 4);
- the client AL invokes an ACTION.confirm of type WITH-LIST service primitive with additional parameters: Invocation_Type = FIRST-PART, BTW = 3. The Service_Parameters include one part of the response from the server;
- the client AL sends the 5th and the 6th,(last) block. The window size is increased again to 3;
- the server AL invokes an ACTION.indication of type WITH-LIST service primitive with Invocation_Type = LAST-PART and with Service_Parameters containing the last part of the ACTION.request;
- the server AP processes this and invokes an ACTION.response of type WITH-LIST service primitive with Invocation_Type = LAST-PART; the Service_Parameters contain the remaining part of the response. This is sent to client in three blocks using streaming;
- the client AL invokes an ACTION.confirm of type WITH-LIST service primitive with Invocation_Type = LAST-PART. The Service_Parameters include the last part of the response from the server.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

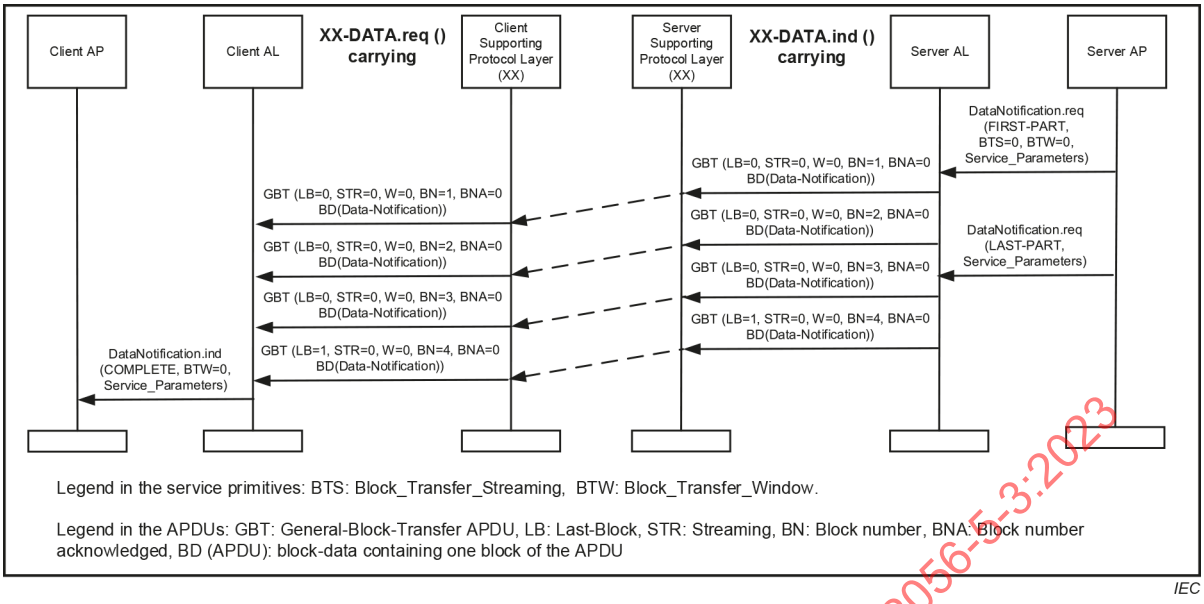



**Figure 73 – DataNotification service with GBT with partial invocation**

Figure 73 shows a DataNotification service with GBT, with partial service invocations on the server side. The process is the following:

- the server AP invokes a DataNotification.request service primitive with Invocation_Type = FIRST-PART, BTS = 0, BTW = 0. The Service_Parameters include one part of the DataNotification.request;
- the server AL sends the GBT APDUs to the client. The reception of the blocks is not confirmed, block recovery is not available;
- when the client AL receives the last block, it assembles the block-data together and invokes a DataNotification.indication service primitive with Invocation_Type = COMPLETE, BTW = 0. The Service_Parameters include the complete DataNotification service parameters.

#### 7.3.13.8 Aborting the GBT procedure

A GBT procedure can be aborted any time by:

- an ABORT GBT APDU with LB = 1, STR = FALSE, BN = 0 and BNA = 0;
- starting a new GBT procedure. This is detected by receiving a GBT APDU with BN = 1 and BNA = 0;
- an APDU other than GBT.

A confirmed GBT procedure shall also be aborted if a party confirms the reception of a block not yet sent by the other party.

It is not possible to abort GBT with unconfirmed DataNotification

### 7.3.14 Protocol of exception mechanism

xDLMS service requests may result in an exception when the request cannot be processed and therefore, the response corresponding to the request cannot be sent back. In such cases, the server optionally uses a confirmedServiceError APDU in case of SN referencing and exception-response APDU in case of LN referencing.

- confirmedServiceError APDU: this is an APDU that can be sent by the server to provide service-specific diagnostic information if a confirmed service .request invocation fails and the corresponding .response primitive cannot be sent back. It can be used with the InitiateRequest, Read and Write services;

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- the exception-response APDU: this is an APDU that can be sent by the server when a .response primitive corresponding to a .request using LN referencing cannot be sent back. It can be used with the GET, SET, ACTION and ACCESS services;
- Conditions to use the xDLMS exception mechanism are specified in Table 79.

**Table 79 – xDLMS exception mechanism**

| Request processing condition | LN referencing response APDU | SN referencing response APDU |
|---|---|---|
| Association is not established | exception-response<br>(state-error = service-not-allowed, service-error = operation-not-possible) | confirmedServiceError<br>(read or write, vde-state-error = no-dlms-context) |
| Request APDU size > server-max-receive-pdu-size | exception-response<br>(state-error = service-not-allowed, service-error = pdu-too-long) | confirmedServiceError<br>(read or write, service = pdu-size) |
| Protection is not verified or removed due to invocation counter error | exception-response<br>(state-error = service-not-allowed, service-error = invocation-counter-error)<br>value = lowest acceptable value of invocation counter | confirmedServiceError<br>(read or write, application-reference = deciphering-error) |
| Protection is not verified or removed due to deciphering error | exception-response<br>(state-error = service-not-allowed, service-error = deciphering-error) | confirmedServiceError<br>(read or write, application-reference = deciphering-error) |
| Request is not well formed | exception-response<br>(state-error = service-not-known, service-error = operation-not-possible) | confirmedServiceError<br>(read or write, definition = other) |
| Request not supported with negotiated conformance | exception-response<br>(state-error = service-not-allowed, service-error = service-not-supported) | confirmedServiceError<br>(read or write, service = service-unsupported) |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 8 Abstract syntax of ACSE and COSEM APDUs

The abstract syntax of COSEM APDUs is specified in this clause using ASN.1. See ISO/IEC 8824-1:2008.

```
COSEMpdu DEFINITIONS::= BEGIN

ACSE-APDU::= CHOICE
{
    aarq                                AARQ-apdu,
    aare                                AARE-apdu,
    rlrq                                RLRQ-apdu,          -- OPTIONAL
    rlre                                RLRE-apdu           -- OPTIONAL
}

XDLMS-APDU::= CHOICE
{
-- standardised xDLMS pdus used in DLMS/COSEM

-- with no ciphering

    initiateRequest                 [1]  IMPLICIT       InitiateRequest,
    readRequest                     [5]  IMPLICIT       ReadRequest,
    writeRequest                    [6]  IMPLICIT       WriteRequest,

    initiateResponse                [8]  IMPLICIT       InitiateResponse,
```

```
    readResponse                    [12] IMPLICIT       ReadResponse,
    writeResponse                   [13] IMPLICIT       WriteResponse,

    confirmedServiceError           [14]                ConfirmedServiceError,

-- data-notification

    data-notification               [15] IMPLICIT       Data-Notification,
    data-notification-confirm       [16] IMPLICIT       Data-Notification-Confirm,
    unconfirmedWriteRequest         [22] IMPLICIT       UnconfirmedWriteRequest,
    informationReportRequest        [24] IMPLICIT       InformationReportRequest,

-- The APDU tag of each ciphered xDLMS APDU indicates the type of the unciphered APDU
and whether
-- global or dedicated key is used. The type of the key is carried by the security
header, and after
-- removing the encryption and/or verifying the authentication tag, the original APDU
with its APDU
-- TAG is restored. Therefore, the APDU tags of the ciphered APDUs carry redundant
information, but
-- they are retained for consistency.

-- with global ciphering

    glo-initiateRequest             [33]  IMPLICIT      OCTET STRING,
    glo-readRequest                 [37]  IMPLICIT      OCTET STRING,
    glo-writeRequest                [38]  IMPLICIT      OCTET STRING,

    glo-initiateResponse            [40]  IMPLICIT      OCTET STRING,
    glo-readResponse                [44]  IMPLICIT      OCTET STRING,
    glo-writeResponse               [45]  IMPLICIT      OCTET STRING,

    glo-confirmedServiceError       [46]  IMPLICIT      OCTET STRING,

    glo-unconfirmedWriteRequest     [54]  IMPLICIT      OCTET STRING,
    glo-informationReportRequest    [56]  IMPLICIT      OCTET STRING,

-- with dedicated ciphering

-- not used in DLMS/COSEM
    ded-initiateRequest             [65]  IMPLICIT      OCTET STRING,

    ded-readRequest                 [69]  IMPLICIT      OCTET STRING,
    ded-writeRequest                [70]  IMPLICIT      OCTET STRING,

-- not used in DLMS/COSEM
    ded-initiateResponse            [72]  IMPLICIT      OCTET STRING,

    ded-readResponse                [76]  IMPLICIT      OCTET STRING,
    ded-writeResponse               [77]  IMPLICIT      OCTET STRING,

    ded-confirmedServiceError       [78]  IMPLICIT      OCTET STRING,

    ded-unconfirmedWriteRequest     [86]  IMPLICIT      OCTET STRING,
    ded-informationReportRequest    [88]  IMPLICIT      OCTET STRING,

-- xDLMS APDUs used with LN referencing
-- with no ciphering

    get-request                     [192] IMPLICIT      Get-Request,
    set-request                     [193] IMPLICIT      Set-Request,
    event-notification-request      [194] IMPLICIT      EventNotificationRequest,
    action-request                  [195] IMPLICIT      Action-Request,

    get-response                    [196] IMPLICIT      Get-Response,
    set-response                    [197] IMPLICIT      Set-Response,
    action-response                 [199] IMPLICIT      Action-Response,

-- with global ciphering
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
    glo-get-request                    [200] IMPLICIT      OCTET STRING,
    glo-set-request                    [201] IMPLICIT      OCTET STRING,
    glo-event-notification-request     [202] IMPLICIT      OCTET STRING,
    glo-action-request                 [203] IMPLICIT      OCTET STRING,

    glo-get-response                   [204] IMPLICIT      OCTET STRING,
    glo-set-response                   [205] IMPLICIT      OCTET STRING,
    glo-action-response                [207] IMPLICIT      OCTET STRING,

-- with dedicated ciphering

    ded-get-request                    [208] IMPLICIT      OCTET STRING,
    ded-set-request                    [209] IMPLICIT      OCTET STRING,
    ded-event-notification-request     [210] IMPLICIT      OCTET STRING,
    ded-actionRequest                  [211] IMPLICIT      OCTET STRING,

    ded-get-response                   [212] IMPLICIT      OCTET STRING,
    ded-set-response                   [213] IMPLICIT      OCTET STRING,
    ded-action-response                [215] IMPLICIT      OCTET STRING,

-- the exception response pdu

    exception-response                 [216] IMPLICIT      ExceptionResponse,


-- access

    access-request                     [217] IMPLICIT      Access-Request,
    access-response                    [218] IMPLICIT      Access-Response,

-- general APDUs
    general-glo-ciphering              [219] IMPLICIT      General-Glo-Ciphering,
    general-ded-ciphering              [220] IMPLICIT      General-Ded-Ciphering,
    general-ciphering                  [221] IMPLICIT      General-Ciphering,
    general-signing                    [223] IMPLICIT      General-Signing,
    general-block-transfer             [224] IMPLICIT      General-Block-Transfer

-- The tags 230 and 231 are reserved for DLMS Gateway
-- reserved                           [230]
-- reserved                           [231]
}

AARQ::= [APPLICATION 0] IMPLICIT SEQUENCE
{
-- [APPLICATION 0] == [ 60H ] = [ 96 ]

    protocol-version                   [0] IMPLICIT        BIT STRING {version1 (0)}
DEFAULT {version1},
    application-context-name           [1]                 Application-context-name,
    called-AP-title                    [2]                 AP-title OPTIONAL,
    called-AE-qualifier                [3]                 AE-qualifier OPTIONAL,
    called-AP-invocation-id            [4]                 AP-invocation-identifier
OPTIONAL,
    called-AE-invocation-id            [5]                 AE-invocation-identifier
OPTIONAL,
    calling-AP-title                   [6]                 AP-title OPTIONAL,
    calling-AE-qualifier               [7]                 AE-qualifier OPTIONAL,
    calling-AP-invocation-id           [8]                 AP-invocation-identifier
OPTIONAL,
    calling-AE-invocation-id           [9]                 AE-invocation-identifier
OPTIONAL,

-- The following field shall not be present if only the kernel is used.
    sender-acse-requirements           [10] IMPLICIT      ACSE-requirements OPTIONAL,

-- The following field shall only be present if the authentication functional unit is
selected.
    mechanism-name                     [11] IMPLICIT      Mechanism-name OPTIONAL,
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
-- The following field shall only be present if the authentication functional unit is
selected.

    calling-authentication-value       [12] EXPLICIT      Authentication-value
OPTIONAL,
    implementation-information         [29] IMPLICIT      Implementation-data
OPTIONAL,
    user-information                   [30] EXPLICIT      Association-information
OPTIONAL
}

-- The user-information field shall carry an InitiateRequest APDU encoded in A-XDR,
and then
-- encoding the resulting OCTET STRING in BER.

AARE-apdu::= [APPLICATION 1] IMPLICIT SEQUENCE
{
-- [APPLICATION 1] == [ 61H ] = [ 97 ]

    protocol-version                   [0] IMPLICIT       BIT STRING {version1 (0)}
DEFAULT {version1},
    application-context-name           [1]                Application-context-name,
    result                             [2]                Association-result,
    result-source-diagnostic           [3]                Associate-source-
diagnostic,
    responding-AP-title                [4]                AP-title OPTIONAL,
    responding-AE-qualifier            [5]                AE-qualifier OPTIONAL,
    responding-AP-invocation-id        [6]                AP-invocation-identifier
OPTIONAL,
    responding-AE-invocation-id        [7]                AE-invocation-identifier
OPTIONAL,

-- The following field shall not be present if only the kernel is used.
    responder-acse-requirements        [8] IMPLICIT       ACSE-requirements
OPTIONAL,

-- The following field shall only be present if the authentication functional unit is
selected.
    mechanism-name                     [9] IMPLICIT       Mechanism-name OPTIONAL,

-- The following field shall only be present if the authentication functional unit is
selected.
    responding-authentication-value    [10] EXPLICIT      Authentication-value
OPTIONAL,
    implementation-information         [29] IMPLICIT      Implementation-data
OPTIONAL,
    user-information                   [30] EXPLICIT      Association-information
OPTIONAL
}

-- The user-information field shall carry either an InitiateResponse (or, when the
proposed xDLMS
-- context is not accepted by the server, a confirmedServiceError) APDU encoded in A-
XDR, and then
-- encoding the resulting OCTET STRING in BER.

RLRQ-apdu::= [APPLICATION 2] IMPLICIT SEQUENCE
{
-- [APPLICATION 2] == [ 62H ] = [ 98 ]

    reason                             [0] IMPLICIT       Release-request-reason
OPTIONAL,
    user-information                   [30] EXPLICIT      Association-information
OPTIONAL
}

RLRE-apdu::= [APPLICATION 3] IMPLICIT SEQUENCE
{
-- [APPLICATION 3] == [ 63H ] = [ 99 ]
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
    reason                              [0] IMPLICIT        Release-response-reason
OPTIONAL,
    user-information                    [30] EXPLICIT       Association-information
OPTIONAL
}

-- The user-information field of the RLRQ / RLRE APDU may carry an InitiateRequest
APDU encoded in
-- A-XDR, and then encoding the resulting OCTET STRING in BER, when the AA to be
released uses
-- ciphering.

-- types used in the fields of the ACSE APDUs, in the order of their occurrence

Application-context-name::=           OBJECT IDENTIFIER

AP-title::=                           OCTET STRING

AE-qualifier::=                       OCTET STRING

AP-invocation-identifier::=           INTEGER

AE-invocation-identifier::=           INTEGER

ACSE-requirements::=                  BIT STRING {authentication(0)}

Mechanism-name::=                     OBJECT IDENTIFIER

Authentication-value::= CHOICE
{
    charstring                         [0] IMPLICIT    GraphicString,
    bitstring                          [1] IMPLICIT    BIT STRING
}

Implementation-data::=                GraphicString

Association-information::=            OCTET STRING

Association-result::=                 INTEGER
{
    accepted                           (0),
    rejected-permanent                 (1),
    rejected-transient                 (2)
}

Associate-source-diagnostic::= CHOICE
{
    acse-service-user                  [1] INTEGER
    {
        null                                                    (0),
        no-reason-given                                         (1),
        application-context-name-not-supported                  (2),
        calling-AP-title-not-recognized                         (3),
        calling-AP-invocation-identifier-not-recognized         (4),
        calling-AE-qualifier-not-recognized                     (5),
        calling-AE-invocation-identifier-not-recognized         (6),
        called-AP-title-not-recognized                          (7),
        called-AP-invocation-identifier-not-recognized          (8),
        called-AE-qualifier-not-recognized                      (9),
        called-AE-invocation-identifier-not-recognized          (10),
        authentication-mechanism-name-not-recognised            (11),
        authentication-mechanism-name-required                  (12),
        authentication-failure                                  (13),
        authentication-required                                 (14)
    },
    acse-service-provider              [2] INTEGER
    {
        null                                (0),
        no-reason-given                     (1),
        no-common-acse-version              (2)
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
    }
}

Release-request-reason::= INTEGER
{
    normal                              (0),
    urgent                              (1),
    user-defined                        (30)
}

Release-response-reason::= INTEGER
{
    normal                              (0),
    not-finished                        (1),
    user-defined                        (30)
}


-- Useful types

Integer8::=                             INTEGER(-128..127)
Integer16::=                            INTEGER(-32768..32767)
Integer32::=                            INTEGER(-2147483648..2147483647)
Integer64::=                            INTEGER(-
9223372036854775808..9223372036854775807)
Unsigned8::=                            INTEGER(0..255)
Unsigned16::=                           INTEGER(0..65535)
Unsigned32::=                           INTEGER(0..4294967295)
Unsigned64::=                           INTEGER(0..18446744073709551615)


-- xDLMS APDU-s used during Association establishment

InitiateRequest::= SEQUENCE
{
--  shall not be encoded in DLMS without ciphering
    dedicated-key                       OCTET STRING OPTIONAL,
    response-allowed                    BOOLEAN DEFAULT TRUE,
    proposed-quality-of-service         [0] IMPLICIT Integer8 OPTIONAL,
    proposed-dlms-version-number        Unsigned8,
    proposed-conformance                Conformance, -- Shall be encoded in BER
    client-max-receive-pdu-size         Unsigned16
}

-- In DLMS/COSEM, the quality-of-service parameter is not used. Any value shall be
accepted.

-- The Conformance field shall be encoded in BER. See IEC 61334-6 Example 1.

InitiateResponse::= SEQUENCE
{
    negotiated-quality-of-service       [0] IMPLICIT Integer8 OPTIONAL,
    negotiated-dlms-version-number      Unsigned8,
    negotiated-conformance              Conformance, -- Shall be encoded in BER
    server-max-receive-pdu-size         Unsigned16,
    vaa-name                            ObjectName
}

-- In the case of LN referencing, the value of the vaa-name is 0x0007
-- In the case of SN referencing, the value of the vaa-name is the base name of the
-- Current Association object, 0xFA00

-- Conformance Block

-- SIZE constrained BIT STRING is extension of ASN.1 notation

Conformance::= [APPLICATION 31] IMPLICIT BIT STRING
{
    -- the bit is set when the corresponding service or functionality is available
    reserved-zero                       (0),
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
    -- The actual list of general protection services depends on the security suite
    general-protection                  (1),
    general-block-transfer              (2),
    read                                (3),
    write                               (4),
    unconfirmed-write                   (5),
    delta-value-encoding                (6),
    reserved-seven                      (7),
    attribute0-supported-with-set       (8),
    priority-mgmt-supported             (9),
    attribute0-supported-with-get       (10),
    block-transfer-with-get-or-read     (11),
    block-transfer-with-set-or-write    (12),
    block-transfer-with-action          (13),
    multiple-references                 (14),
    information-report                  (15),
    data-notification                   (16),
    access                              (17),
    parameterized-access                (18),
    get                                 (19),
    set                                 (20),
    selective-access                    (21),
    event-notification                  (22),
    action                              (23)
}

ObjectName::=                               Integer16
-- for named variable objects (short names), the last three bits shall be set to 000;
-- for vaa-name objects, the last three bits shall be set to 111.

-- The Confirmed ServiceError APDU is used only with the InitiateRequest, ReadRequest
and
-- WriteRequest APDUs when the request fails, to provide diagnostic information.

ConfirmedServiceError::= CHOICE
{
-- tag 0 is reserved
-- In DLMS/COSEM only initiateError, read and write are relevant

    initiateError                   [1] ServiceError,
    getStatus                       [2] ServiceError,
    getNameList                     [3] ServiceError,
    getVariableAttribute            [4] ServiceError,
    read                            [5] ServiceError,
    write                           [6] ServiceError,
    getDataSetAttribute             [7] ServiceError,
    getTIAttribute                  [8] ServiceError,
    changeScope                     [9] ServiceError,
    start                           [10] ServiceError,
    stop                            [11] ServiceError,
    resume                          [12] ServiceError,
    makeUsable                      [13] ServiceError,
    initiateLoad                    [14] ServiceError,
    loadSegment                     [15] ServiceError,
    terminateLoad                   [16] ServiceError,
    initiateUpLoad                  [17] ServiceError,
    upLoadSegment                   [18] ServiceError,
    terminateUpLoad                 [19] ServiceError
}

ServiceError::= CHOICE
{
    application-reference           [0] IMPLICIT ENUMERATED
    {
    -- DLMS provider only
        other                           (0),
        time-elapsed                    (1),  -- time out since request sent
        application-unreachable         (2),  -- peer AEi not reachable
        application-reference-invalid   (3),  -- addressing trouble
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
        application-context-unsupported     (4),  -- application-context
incompatibility
        provider-communication-error        (5),  -- error at the local or distant
equipment
        deciphering-error                   (6)   -- error detected by the
deciphering function
    },

    hardware-resource                   [1] IMPLICIT ENUMERATED
    {
    -- VDE hardware troubles
        other                                (0),
        memory-unavailable                   (1),
        processor-resource-unavailable       (2),
        mass-storage-unavailable             (3),
        other-resource-unavailable           (4)
    },

    vde-state-error                     [2] IMPLICIT ENUMERATED
    {
    -- Error source description
        other                                (0),
        no-dlms-context                      (1),
        loading-data-set                     (2),
        status-nochange                      (3),
        status-inoperable                    (4)
    },

    service                             [3] IMPLICIT ENUMERATED
    {
    -- service handling troubles
        other                                (0),
        pdu-size                             (1),  -- pdu too long
        service-unsupported                  (2)   -- as defined in the conformance
block
    },

    definition                          [4] IMPLICIT ENUMERATED
    {
    -- object bound troubles in a service
        other                                (0),
        object-undefined                     (1),  -- object not defined at the VDE
        object-class-inconsistent            (2),  -- class of object incompatible
with asked service
        object-attribute-inconsistent        (3)   -- object attributes are
inconsistent
    },

    access                              [5] IMPLICIT ENUMERATED
    {
    -- object access error
        other                                (0),
        scope-of-access-violated             (1),  -- access denied through
authorisation reason
        object-access-violated               (2),  -- access incompatible with object
attribute
        hardware-fault                       (3),  -- access fail for hardware reason
        object-unavailable                   (4)   -- VDE hands object for unavailable
    },

    initiate                            [6] IMPLICIT ENUMERATED
    {
    -- initiate service error
        other                                (0),
        dlms-version-too-low                 (1),  -- proposed DLMS version too low
        incompatible-conformance             (2),  -- proposed service not sufficient
        pdu-size-too-short                   (3),  -- proposed PDU size too short
        refused-by-the-VDE-Handler           (4)   -- vaa creation impossible or not
allowed
    },
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
    load-data-set                       [7] IMPLICIT ENUMERATED
    {
    -- data set load services error
        other                                (0),
        primitive-out-of-sequence            (1),  -- according to the DataSet loading
state transitions
        not-loadable                         (2),  -- loadable attribute set to FALSE
        dataset-size-too-large               (3),  -- evaluated Data Set size too
large
        not-awaited-segment                  (4),  -- proposed segment not awaited
        interpretation-failure               (5),  -- segment interpretation error
        storage-failure                      (6),  -- segment storage error
        data-set-not-ready                   (7)   -- Data Set not in correct state
for uploading
    },

    -- change-scope                     [8] IMPLICIT ENUMERATED

    task                                [9] IMPLICIT ENUMERATED
    {
    -- TI services error
        other                                (0),
        no-remote-control                    (1),  -- Remote Control parameter set to
FALSE
        ti-stopped                           (2),  -- TI in stopped state
        ti-running                           (3),  -- TI in running state
        ti-unusable                          (4)   -- TI in unusable state
    }

    -- other                            [10] IMPLICIT ENUMERATED
}


-- COSEM APDUs using short name referencing

ReadRequest::= SEQUENCE OF Variable-Access-Specification

ReadResponse::= SEQUENCE OF CHOICE
{
    data                                [0] Data,
    data-access-error                   [1] IMPLICIT Data-Access-Result,
    data-block-result                   [2] IMPLICIT Data-Block-Result,
    block-number                        [3] IMPLICIT Unsigned16
}

WriteRequest::= SEQUENCE
{
    variable-access-specification       SEQUENCE OF Variable-Access-Specification,
    list-of-data                        SEQUENCE OF Data
}

WriteResponse::= SEQUENCE OF CHOICE
{
    success                             [0] IMPLICIT NULL,
    data-access-error                   [1] IMPLICIT Data-Access-Result,
    block-number                        [2] Unsigned16
}

UnconfirmedWriteRequest::= SEQUENCE
{
    variable-access-specification       SEQUENCE OF Variable-Access-Specification,
    list-of-data                        SEQUENCE OF Data
}

InformationReportRequest::= SEQUENCE
{
    current-time                        GeneralizedTime OPTIONAL,
    variable-access-specification       SEQUENCE OF Variable-Access-Specification,
    list-of-data                        SEQUENCE OF Data
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
}


-- COSEM APDUs using logical name referencing

Get-Request::= CHOICE
{
    get-request-normal                [1] IMPLICIT    Get-Request-Normal,
    get-request-next                  [2] IMPLICIT    Get-Request-Next,
    get-request-with-list             [3] IMPLICIT    Get-Request-With-List
}

Get-Request-Normal::= SEQUENCE
{
    invoke-id-and-priority            Invoke-Id-And-Priority,
    cosem-attribute-descriptor        Cosem-Attribute-Descriptor,
    access-selection                  Selective-Access-Descriptor OPTIONAL
}

Get-Request-Next::= SEQUENCE
{
    invoke-id-and-priority            Invoke-Id-And-Priority,
    block-number                      Unsigned32
}

Get-Request-With-List::= SEQUENCE
{
    invoke-id-and-priority            Invoke-Id-And-Priority,
    attribute-descriptor-list         SEQUENCE OF Cosem-Attribute-Descriptor-With-
Selection
}

Get-Response::= CHOICE
{
    get-response-normal               [1] IMPLICIT    Get-Response-Normal,
    get-response-with-datablock       [2] IMPLICIT    Get-Response-With-Datablock,
    get-response-with-list            [3] IMPLICIT    Get-Response-With-List
}

Get-Response-Normal::= SEQUENCE
{
    invoke-id-and-priority            Invoke-Id-And-Priority,
    result                            Get-Data-Result
}
Get-Response-With-Datablock::= SEQUENCE
{
    invoke-id-and-priority            Invoke-Id-And-Priority,
    result                            DataBlock-G
}

Get-Response-With-List::= SEQUENCE
{
    invoke-id-and-priority            Invoke-Id-And-Priority,
    result                            SEQUENCE OF Get-Data-Result
}

Set-Request::= CHOICE
{
  set-request-normal                        [1] IMPLICIT  Set-Request-Normal,
  set-request-with-first-datablock          [2] IMPLICIT  Set-Request-With-First-
Datablock,
  set-request-with-datablock                [3] IMPLICIT  Set-Request-With-
Datablock,
  set-request-with-list                     [4] IMPLICIT  Set-Request-With-List,
  set-request-with-list-and-first-datablock [5] IMPLICIT  Set-Request-With-List-And-
First-Datablock
}

Set-Request-Normal::= SEQUENCE
{
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
    invoke-id-and-priority              Invoke-Id-And-Priority,
    cosem-attribute-descriptor          Cosem-Attribute-Descriptor,
    access-selection                    Selective-Access-Descriptor OPTIONAL,
    value                               Data
}

Set-Request-With-First-Datablock::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    cosem-attribute-descriptor          Cosem-Attribute-Descriptor,
    access-selection                    [0] IMPLICIT Selective-Access-Descriptor
OPTIONAL,
    datablock                           DataBlock-SA
}

Set-Request-With-Datablock::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    datablock                           DataBlock-SA
}

Set-Request-With-List::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    attribute-descriptor-list           SEQUENCE OF Cosem-Attribute-Descriptor-With-
Selection,
    value-list                          SEQUENCE OF Data
}

Set-Request-With-List-And-First-Datablock::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    attribute-descriptor-list           SEQUENCE OF Cosem-Attribute-Descriptor-With-
Selection,
    datablock                           DataBlock-SA
}

Set-Response::= CHOICE
{
    set-response-normal                     [1] IMPLICIT   Set-Response-Normal,
    set-response-datablock                  [2] IMPLICIT   Set-Response-Datablock,
    set-response-last-datablock             [3] IMPLICIT   Set-Response-Last-
Datablock,
    set-response-last-datablock-with-list   [4] IMPLICIT   Set-Response-Last-
Datablock-With-List,
    set-response-with-list                  [5] IMPLICIT   Set-Response-With-List
}

Set-Response-Normal::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    result                              Data-Access-Result
}

Set-Response-Datablock::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    block-number                        Unsigned32
}

Set-Response-Last-Datablock::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    result                              Data-Access-Result,
    block-number                        Unsigned32
}

Set-Response-Last-Datablock-With-List::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
    result                              SEQUENCE OF Data-Access-Result,
    block-number                        Unsigned32
}

Set-Response-With-List::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    result                              SEQUENCE OF Data-Access-Result
}

Action-Request::= CHOICE
{
  action-request-normal                       [1] IMPLICIT Action-Request-Normal,
  action-request-next-pblock                  [2] IMPLICIT Action-Request-Next-Pblock,
  action-request-with-list                    [3] IMPLICIT Action-Request-With-List,
  action-request-with-first-pblock            [4] IMPLICIT Action-Request-With-First-
Pblock,
  action-request-with-list-and-first-pblock   [5] IMPLICIT Action-Request-With-List-
And-First-Pblock,
  action-request-with-pblock                  [6] IMPLICIT Action-Request-With-Pblock
}

Action-Request-Normal::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    cosem-method-descriptor             Cosem-Method-Descriptor,
    method-invocation-parameters        Data OPTIONAL
}

Action-Request-Next-Pblock::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    block-number                        Unsigned32
}

Action-Request-With-List::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    cosem-method-descriptor-list        SEQUENCE OF Cosem-Method-Descriptor,
    method-invocation-parameters        SEQUENCE OF Data
}

Action-Request-With-First-Pblock::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    cosem-method-descriptor             Cosem-Method-Descriptor,
    pblock                              DataBlock-SA
}

Action-Request-With-List-And-First-Pblock::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    cosem-method-descriptor-list        SEQUENCE OF Cosem-Method-Descriptor,
    pblock                              DataBlock-SA
}

Action-Request-With-Pblock::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    pblock                              DataBlock-SA
}

Action-Response::= CHOICE
{
    action-response-normal              [1] IMPLICIT    Action-Response-Normal,
    action-response-with-pblock         [2] IMPLICIT    Action-Response-With-Pblock,
    action-response-with-list           [3] IMPLICIT    Action-Response-With-List,
    action-response-next-pblock         [4] IMPLICIT    Action-Response-Next-Pblock
}
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
Action-Response-Normal::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    single-response                     Action-Response-With-Optional-Data
}

Action-Response-With-Pblock::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    pblock                              DataBlock-SA
}

Action-Response-With-List::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    list-of-responses                   SEQUENCE OF Action-Response-With-Optional-Data
}

Action-Response-Next-Pblock::= SEQUENCE
{
    invoke-id-and-priority              Invoke-Id-And-Priority,
    block-number                        Unsigned32
}

EventNotificationRequest::= SEQUENCE
{
    time                                OCTET STRING  OPTIONAL,
    cosem-attribute-descriptor          Cosem-Attribute-Descriptor,
    attribute-value                     Data
}

ExceptionResponse::= SEQUENCE
{
    state-error                         [0] IMPLICIT ENUMERATED
    {
        service-not-allowed                 (1),
        service-unknown                     (2)
    },
    service-error                       [1] IMPLICIT ENUMERATED
    {
        operation-not-possible              (1),
        service-not-supported               (2),
        other-reason                        (3)
    }
}


--      Access

Access-Request::= SEQUENCE
{
   long-invoke-id-and-priority          Long-Invoke-Id-And-Priority,
   date-time                            OCTET STRING,
   access-request-body                  Access-Request-Body
}

Access-Response::= SEQUENCE
{
       long-invoke-id-and-priority        Long-Invoke-Id-And-Priority,
       date-time                          OCTET STRING,
       access-response-body               Access-Response-Body
}


--      Data-Notification

Data-Notification::= SEQUENCE
{
   long-invoke-id-and-priority          Long-Invoke-Id-And-Priority,
   date-time                            OCTET STRING,
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
   notification-body                    Notification-Body
}
Data-Notification-Confirm::= SEQUENCE
{
   long-invoke-id-and-priority          Long-Invoke-Id-And-Priority,
   date-time                            OCTET STRING
}
-- General APDUs

General-Ded-Ciphering::= SEQUENCE
{
   system-title                         OCTET STRING,
   ciphered-content                     OCTET STRING
}

General-Glo-Ciphering::= SEQUENCE
{
   system-title                         OCTET STRING,
   ciphered-content                     OCTET STRING
}

General-Ciphering::= SEQUENCE
{
   transaction-id                       OCTET STRING,
   originator-system-title              OCTET STRING,
   recipient-system-title               OCTET STRING,
   date-time                            OCTET STRING,
   other-information                    OCTET STRING,
   key-info                             Key-Info OPTIONAL,
   ciphered-content                     OCTET STRING
}

General-Signing::= SEQUENCE
{
   transaction-id                       OCTET STRING,
   originator-system-title              OCTET STRING,
   recipient-system-title               OCTET STRING,
   date-time                            OCTET STRING,
   other-information                    OCTET STRING,
   content                              OCTET STRING,
   signature                            OCTET STRING
}

General-Block-Transfer::= SEQUENCE
{
   block-control                        Block-Control,
   block-number                         Unsigned16,
   block-number-ack                     Unsigned16,
   block-data                           OCTET STRING
}


-- Types used in the xDLMS data transfer services

Variable-Access-Specification::= CHOICE
{
    variable-name                       [2] IMPLICIT ObjectName,
-- detailed-access [3] is not used in DLMS/COSEM
    parameterized-access                [4] IMPLICIT Parameterized-Access,
    block-number-access                 [5] IMPLICIT Block-Number-Access,
    read-data-block-access              [6] IMPLICIT Read-Data-Block-Access,
    write-data-block-access             [7] IMPLICIT Write-Data-Block-Access
}

Parameterized-Access::= SEQUENCE
{
    variable-name                       ObjectName,
    selector                            Unsigned8,
    parameter                           Data
}
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
Block-Number-Access::= SEQUENCE
{
    block-number                        Unsigned16
}

Read-Data-Block-Access::= SEQUENCE
{
    last-block                          BOOLEAN,
    block-number                        Unsigned16,
    raw-data                            OCTET STRING
}


Write-Data-Block-Access::= SEQUENCE
{
   last-block                           BOOLEAN,
   block-number                         Unsigned16
}

Data::= CHOICE
{
    null-data                           [0]   IMPLICIT   NULL,
    array                               [1]   IMPLICIT   SEQUENCE OF Data,
    structure                           [2]   IMPLICIT   SEQUENCE OF Data,
    boolean                             [3]   IMPLICIT   BOOLEAN,
    bit-string                          [4]   IMPLICIT   BIT STRING,
    double-long                         [5]   IMPLICIT   Integer32,
    double-long-unsigned                [6]   IMPLICIT   Unsigned32,
    octet-string                        [9]   IMPLICIT   OCTET STRING,
    visible-string                      [10]  IMPLICIT   VisibleString,
    utf8-string                         [12]  IMPLICIT   UTF8String,
    bcd                                 [13]  IMPLICIT   Integer8,
    integer                             [15]  IMPLICIT   Integer8,
    long                                [16]  IMPLICIT   Integer16,
    unsigned                            [17]  IMPLICIT   Unsigned8,
    long-unsigned                       [18]  IMPLICIT   Unsigned16,
    compact-array                       [19]  IMPLICIT   SEQUENCE
    {
        contents-description                  [0]              TypeDescription,
        array-contents                        [1]   IMPLICIT   OCTET STRING
    },
    long64                              [20]  IMPLICIT   Integer64,
    long64-unsigned                     [21]  IMPLICIT   Unsigned64,
    enum                                [22]  IMPLICIT   Unsigned8,
    float32                             [23]  IMPLICIT   OCTET STRING (SIZE(4)),
    float64                             [24]  IMPLICIT   OCTET STRING (SIZE(8)),
    date-time                           [25]  IMPLICIT   OCTET STRING (SIZE(12)),
    date                                [26]  IMPLICIT   OCTET STRING (SIZE(5)),
    time                                [27]  IMPLICIT   OCTET STRING (SIZE(4)),
    delta-integer                       [28]  IMPLICIT   Integer8,
    delta-long                          [29]  IMPLICIT   Integer16,
    delta-double-long                   [30]  IMPLICIT   Integer32,
    delta-unsigned                      [31]  IMPLICIT   Unsigned8,
    delta-long-unsigned                 [32]  IMPLICIT   Unsigned16,
    delta-double-long-unsigned          [33]  IMPLICIT   Unsigned32,

    dont-care                           [255] IMPLICIT   NULL
}

-- The following TypeDescription relates to the compact-array data Type

TypeDescription::= CHOICE
{
    null-data                           [0]   IMPLICIT  NULL,
    array                               [1]   IMPLICIT  SEQUENCE
    {
        number-of-elements          Unsigned16,
        type-description            TypeDescription
    },
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
    structure                       [2]   IMPLICIT  SEQUENCE OF TypeDescription,
    boolean                         [3]   IMPLICIT  NULL,
    bit-string                      [4]   IMPLICIT  NULL,
    double-long                     [5]   IMPLICIT  NULL,
    double-long-unsigned            [6]   IMPLICIT  NULL,
    octet-string                    [9]   IMPLICIT  NULL,
    visible-string                  [10]  IMPLICIT  NULL,
    utf8-string                     [12]  IMPLICIT  NULL,
    bcd                             [13]  IMPLICIT  NULL,
    integer                         [15]  IMPLICIT  NULL,
    long                            [16]  IMPLICIT  NULL,
    unsigned                        [17]  IMPLICIT  NULL,
    long-unsigned                   [18]  IMPLICIT  NULL,
    long64                          [20]  IMPLICIT  NULL,
    long64-unsigned                 [21]  IMPLICIT  NULL,
    enum                            [22]  IMPLICIT  NULL,
    float32                         [23]  IMPLICIT  NULL,
    float64                         [24]  IMPLICIT  NULL,
    date-time                       [25]  IMPLICIT  NULL,
    date                            [26]  IMPLICIT  NULL,
    time                            [27]  IMPLICIT  NULL,
    dont-care                       [255] IMPLICIT  NULL
}

Data-Access-Result::= ENUMERATED
{
    success                         (0),
    hardware-fault                  (1),
    temporary-failure               (2),
    read-write-denied               (3),
    object-undefined                (4),
    object-class-inconsistent       (9),
    object-unavailable              (11),
    type-unmatched                  (12),
    scope-of-access-violated        (13),
    data-block-unavailable          (14),
    long-get-aborted                (15),
    no-long-get-in-progress         (16),
    long-set-aborted                (17),
    no-long-set-in-progress         (18),
    data-block-number-invalid       (19),
    other-reason                    (250)
}

Action-Result::= ENUMERATED
{
    success                         (0),
    hardware-fault                  (1),
    temporary-failure               (2),
    read-write-denied               (3),
    object-undefined                (4),
    object-class-inconsistent       (9),
    object-unavailable              (11),
    type-unmatched                  (12),
    scope-of-access-violated        (13),
    data-block-unavailable          (14),
    long-action-aborted             (15),
    no-long-action-in-progress      (16),
    other-reason                    (250)
}

-- IEC 61334-6:2000 Clause 5 specifies that bits of any byte are numbered from 1 to
8,
-- where bit 8 is the most significant.
-- In IEC 62056-5-3, bits are numbered from 0 to 7.
-- Use of Invoke-Id-And-Priority
--    invoke-id               bits 0-3
--    reserved                bits 4-5
--    service-class           bit  6        0 = Unconfirmed, 1 = Confirmed
--    priority                bit  7        0 = Normal, 1 = High
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
Invoke-Id-And-Priority::=             Unsigned8

-- Use of Long-Invoke-Id-And-Priority
--    long-invoke-id           bits 0-23
--    reserved                 bits 24-27
--    self-descriptive         bit  28       0 = Not-Self-Descriptive, 1 = Self-
Descriptive
--    processing-option        bit  29       0 = Continue on Error, 1 = Break on
Error
--    service-class            bit  30       0 = Unconfirmed, 1 = Confirmed
--    priority                 bit  31       0 = Normal, 1 = High
Long-Invoke-Id-And-Priority::=        Unsigned32

Cosem-Attribute-Descriptor::= SEQUENCE
{
    class-id                          Cosem-Class-Id,
    instance-id                       Cosem-Object-Instance-Id,
    attribute-id                      Cosem-Object-Attribute-Id
}

Cosem-Method-Descriptor::= SEQUENCE
{
    class-id                          Cosem-Class-Id,
    instance-id                       Cosem-Object-Instance-Id,
    method-id                         Cosem-Object-Method-Id
}

Cosem-Class-Id::=                     Unsigned16

Cosem-Object-Instance-Id::=           OCTET STRING (SIZE(6))

Cosem-Object-Attribute-Id::=          Integer8

Cosem-Object-Method-Id::=             Integer8

Selective-Access-Descriptor::= SEQUENCE
{
    access-selector                   Unsigned8,
    access-parameters                 Data
}

Cosem-Attribute-Descriptor-With-Selection::= SEQUENCE
{
    cosem-attribute-descriptor        Cosem-Attribute-Descriptor,
    access-selection                  Selective-Access-Descriptor OPTIONAL
}

Get-Data-Result::= CHOICE
{
   data                               [0] Data,
   data-access-result                 [1] IMPLICIT Data-Access-Result
}

Data-Block-Result::= SEQUENCE  -- Used in ReadResponse with block transfer
{
    last-block                        BOOLEAN,
    block-number                      Unsigned16,
    raw-data                          OCTET STRING
}

DataBlock-G::= SEQUENCE     -- G == DataBlock for the GET-response
{
    last-block                        BOOLEAN,
    block-number                      Unsigned32,
    result  CHOICE
    {
        raw-data                      [0] IMPLICIT OCTET STRING,
        data-access-result            [1] IMPLICIT Data-Access-Result
    }
}
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
DataBlock-SA::= SEQUENCE    -- SA == DataBlock for the SET-request, ACTION-request
and ACTION-response
{
    last-block                          BOOLEAN,
    block-number                        Unsigned32,
    raw-data                            OCTET STRING
}

Action-Response-With-Optional-Data::= SEQUENCE
{
    result                              Action-Result,
    return-parameters                   Get-Data-Result  OPTIONAL
}

Notification-Body::= SEQUENCE
{
   data-value                           Data
}

List-Of-Data::= SEQUENCE OF Data

Access-Request-Get::= SEQUENCE
{
   cosem-attribute-descriptor           Cosem-Attribute-Descriptor
}

Access-Request-Get-With-Selection::= SEQUENCE
{
   cosem-attribute-descriptor           Cosem-Attribute-Descriptor,
   access-selection                     Selective-Access-Descriptor
}

Access-Request-Set::= SEQUENCE
{
   cosem-attribute-descriptor           Cosem-Attribute-Descriptor
}

Access-Request-Set-With-Selection::= SEQUENCE
{
   cosem-attribute-descriptor           Cosem-Attribute-Descriptor,
   access-selection                     Selective-Access-Descriptor
}

Access-Request-Action::= SEQUENCE
{
   cosem-method-descriptor              Cosem-Method-Descriptor
}

Access-Request-Specification::= CHOICE
{
   access-request-get                   [1] Access-Request-Get,
   access-request-set                   [2] Access-Request-Set,
   access-request-action                [3] Access-Request-Action,
   access-request-get-with-selection    [4] Access-Request-Get-With-Selection,
   access-request-set-with-selection    [5] Access-Request-Set-With-Selection
}

List-Of-Access-Request-Specification::= SEQUENCE OF Access-Request-Specification

Access-Request-Body::= SEQUENCE
{
   access-request-specification         List-Of-Access-Request-Specification,
   access-request-list-of-data          List-Of-Data
}

Access-Response-Get::= SEQUENCE
{
   result                               Data-Access-Result
}
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
Access-Response-Set::= SEQUENCE
{
   result                              Data-Access-Result
}

Access-Response-Action::= SEQUENCE
{
   result                              Action-Result
}

Access-Response-Specification::= CHOICE
{
   access-response-get                 [1] Access-Response-Get,
   access-response-set                 [2] Access-Response-Set,
   access-response-action              [3] Access-Response-Action
}

List-Of-Access-Response-Specification::= SEQUENCE OF Access-Response-Specification

Access-Response-Body::= SEQUENCE
{
   access-request-specification        [0] List-Of-Access-Request-Specification
OPTIONAL,
   access-response-list-of-data        List-Of-Data,
   access-response-specification       List-Of-Access-Response-Specification
}

-- Key-info

Key-Id::= ENUMERATED
{
       global-unicast-encryption-key       (0),
       global-broadcast-encryption-key     (1)
}

Kek-Id::= ENUMERATED
{
       master-key                          (0)
}


Identified-Key::= SEQUENCE
{
       key-id                              Key-Id
}

Wrapped-Key::= SEQUENCE
{
    kek-id                                 Kek-Id,
    key-ciphered-data                      OCTET STRING
}

Agreed-Key::= SEQUENCE
{
    key-parameters                         OCTET STRING,
    key-ciphered-data                      OCTET STRING
}

Key-Info::= CHOICE
{
    identified-key                         [0] Identified-Key,
    wrapped-key                            [1] Wrapped-Key,
    agreed-key                             [2] Agreed-Key,
}

-- Use of Block-Control
--    window                    bits 0-5      window advertise
--    streaming                 bit  6        0 = No Streaming active, 1 = Streaming
active
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
--    last-block              bit  7        0 = Not Last Block, 1 = Last Block
Block-Control::=                      Unsigned8


END
```

# 9 COSEM APDU XML schema

## 9.1 General

ITU-T recommendations X.693 and X.694 provide XML encoding rules to Abstract Syntax Notation 1 (ASN.1) and XML Schema Definitions Language (XSD) mapping to ASN.1. No recommendation is provided to map ASN.1 to XSD. In this Clause 9 COSEMpdu ASN.1 definition is provided and mapped to COSEMpdu XSD definition.

XML has gained wide acceptance in the IT industry. The purpose of such encoding is to enable transfer of COSEM model content with various means in the form of XML encoded content. It can be a XML document exchanged between applications, content included in Web services SOAP messages or content encapsulated in e-mail messages, to name just few of the applications. Interoperability and mapping between ASN.1 encoded APDUs and XML encoded content is important in both directions. On one side ASN.1 has enabled creation of XML encoded content with support for XML Encoding Rules (See ITU-T X.693 and ITU-T X.694). On the other hand IT industry is searching for solutions for optimal transfer of XML content with XML optimized packaging. For that purposes conversion between W3C XML Schema and ASN.1 definition is crucial. Conversion in both directions enables proper conversion of XML content to ASN.1 encoded content and vice versa. Subclause 9.2 contains mapping of COSEMPdu ASN.1 definition into COSEMPdu XML Schema (XSD).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 9.2 XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns="http://www.dlms.com/COSEMpdu"
            targetNamespace="http://www.dlms.com/COSEMpdu"
            elementFormDefault="qualified">

   <!-- ASN.1 definitions -->
   <xsd:complexType name="NULL" final="#all" />

   <xsd:simpleType name="BitString">
       <xsd:restriction base="xsd:string">
           <xsd:pattern value="[0-1]{0,}" />
       </xsd:restriction>
   </xsd:simpleType>

   <xsd:simpleType name="ObjectIdentifier">
       <xsd:restriction base="xsd:token">
           <xsd:pattern value="[0-2](\.[1-3]?[0-9]?(\.\d+)*)?" />
       </xsd:restriction>
   </xsd:simpleType>


   <!-- ACSE-APDU definition -->
   <xsd:element name="aCSE-APDU" type="ACSE-APDU"/>
   <xsd:complexType name="ACSE-APDU">
      <xsd:choice>
         <xsd:element name="aarq" type="AARQ-apdu"/>
         <xsd:element name="aare" type="AARE-apdu"/>
         <xsd:element name="rlrq" type="RLRQ-apdu"/>
         <xsd:element name="rlre" type="RLRE-apdu"/>
      </xsd:choice>
   </xsd:complexType>

   <!-- xDLMS-APDU definition -->
   <xsd:element name="xDLMS-APDU" type="XDLMS-APDU"/>
   <xsd:complexType name="XDLMS-APDU">
      <xsd:choice>
         <xsd:element name="initiateRequest" type="InitiateRequest"/>
         <xsd:element name="readRequest" type="ReadRequest"/>
```

```
        <xsd:element name="writeRequest" type="WriteRequest"/>
        <xsd:element name="initiateResponse" type="InitiateResponse"/>
        <xsd:element name="readResponse" type="ReadResponse"/>
        <xsd:element name="writeResponse" type="WriteResponse"/>
        <xsd:element name="confirmedServiceError" type="ConfirmedServiceError"/>
        <xsd:element name="data-notification" type="Data-Notification"/>
        <xsd:element name="data-notification-confirm" type="Data-Notification-Confirm"/>
        <xsd:element name="unconfirmedWriteRequest" type="UnconfirmedWriteRequest"/>
        <xsd:element name="informationReportRequest" type="InformationReportRequest"/>
        <xsd:element name="glo-initiateRequest" type="xsd:hexBinary"/>
        <xsd:element name="glo-readRequest" type="xsd:hexBinary"/>
        <xsd:element name="glo-writeRequest" type="xsd:hexBinary"/>
        <xsd:element name="glo-initiateResponse" type="xsd:hexBinary"/>
        <xsd:element name="glo-readResponse" type="xsd:hexBinary"/>
        <xsd:element name="glo-writeResponse" type="xsd:hexBinary"/>
        <xsd:element name="glo-confirmedServiceError" type="xsd:hexBinary"/>
        <xsd:element name="glo-unconfirmedWriteRequest" type="xsd:hexBinary"/>
        <xsd:element name="glo-informationReportRequest" type="xsd:hexBinary"/>
        <xsd:element name="ded-initiateRequest" type="xsd:hexBinary"/>
        <xsd:element name="ded-readRequest" type="xsd:hexBinary"/>
        <xsd:element name="ded-writeRequest" type="xsd:hexBinary"/>
        <xsd:element name="ded-initiateResponse" type="xsd:hexBinary"/>
        <xsd:element name="ded-readResponse" type="xsd:hexBinary"/>
        <xsd:element name="ded-writeResponse" type="xsd:hexBinary"/>
        <xsd:element name="ded-confirmedServiceError" type="xsd:hexBinary"/>
        <xsd:element name="ded-unconfirmedWriteRequest" type="xsd:hexBinary"/>
        <xsd:element name="ded-informationReportRequest" type="xsd:hexBinary"/>
        <xsd:element name="get-request" type="Get-Request"/>
        <xsd:element name="set-request" type="Set-Request"/>
        <xsd:element name="event-notification-request" type="EventNotificationRequest"/>
        <xsd:element name="action-request" type="Action-Request"/>
        <xsd:element name="get-response" type="Get-Response"/>
        <xsd:element name="set-response" type="Set-Response"/>
        <xsd:element name="action-response" type="Action-Response"/>
        <xsd:element name="glo-get-request" type="xsd:hexBinary"/>
        <xsd:element name="glo-set-request" type="xsd:hexBinary"/>
        <xsd:element name="glo-event-notification-request" type="xsd:hexBinary"/>
        <xsd:element name="glo-action-request" type="xsd:hexBinary"/>
        <xsd:element name="glo-get-response" type="xsd:hexBinary"/>
        <xsd:element name="glo-set-response" type="xsd:hexBinary"/>
        <xsd:element name="glo-action-response" type="xsd:hexBinary"/>
        <xsd:element name="ded-get-request" type="xsd:hexBinary"/>
        <xsd:element name="ded-set-request" type="xsd:hexBinary"/>
        <xsd:element name="ded-event-notification-request" type="xsd:hexBinary"/>
        <xsd:element name="ded-actionRequest" type="xsd:hexBinary"/>
        <xsd:element name="ded-get-response" type="xsd:hexBinary"/>
        <xsd:element name="ded-set-response" type="xsd:hexBinary"/>
        <xsd:element name="ded-action-response" type="xsd:hexBinary"/>
        <xsd:element name="exception-response" type="ExceptionResponse"/>
        <xsd:element name="access-request" type="Access-Request"/>
        <xsd:element name="access-response" type="Access-Response"/>
        <xsd:element name="general-glo-ciphering" type="General-Glo-Ciphering"/>
        <xsd:element name="general-ded-ciphering" type="General-Ded-Ciphering"/>
        <xsd:element name="general-ciphering" type="General-Ciphering"/>
        <xsd:element name="general-signing" type="General-Signing"/>
        <xsd:element name="general-block-transfer" type="General-Block-Transfer"/>
      </xsd:choice>
   </xsd:complexType>
   <xsd:complexType name="Data-Notification-Confirm">
       <xsd:sequence>
           <xsd:element name="long-invoke-id-and-priority" type="Long-Invoke-Id-And-Priority"/>
           <xsd:element name="date-time" type="xsd:hexBinary"/>
       </xsd:sequence>
    </xsd:complexType>

   <xsd:simpleType name="Application-context-name">
      <xsd:restriction base="ObjectIdentifier"/>
   </xsd:simpleType>

   <xsd:simpleType name="AP-title">
      <xsd:restriction base="xsd:hexBinary"/>
   </xsd:simpleType>

   <xsd:simpleType name="AE-qualifier">
      <xsd:restriction base="xsd:hexBinary"/>
   </xsd:simpleType>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
<xsd:simpleType name="AP-invocation-identifier">
   <xsd:restriction base="xsd:integer"/>
</xsd:simpleType>

<xsd:simpleType name="AE-invocation-identifier">
   <xsd:restriction base="xsd:integer"/>
</xsd:simpleType>

<xsd:simpleType name="ACSE-requirements">
   <xsd:union memberTypes="BitString">
   <xsd:simpleType>
      <xsd:list>
         <xsd:simpleType>
            <xsd:restriction base="xsd:token">
               <xsd:enumeration value="authentication"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:list>
   </xsd:simpleType>
   </xsd:union>
</xsd:simpleType>

<xsd:simpleType name="Mechanism-name">
   <xsd:restriction base="ObjectIdentifier"/>
</xsd:simpleType>

<xsd:simpleType name="Implementation-data">
   <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="Association-information">
   <xsd:restriction base="xsd:hexBinary"/>
</xsd:simpleType>

<xsd:simpleType name="Association-result">
   <xsd:union>
      <xsd:simpleType>
         <xsd:restriction base="xsd:token">
            <xsd:enumeration value="accepted"/>
            <xsd:enumeration value="rejected-permanent"/>
            <xsd:enumeration value="rejected-transient"/>
         </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
         <xsd:restriction base="xsd:integer"/>
      </xsd:simpleType>
   </xsd:union>
</xsd:simpleType>

<xsd:simpleType name="Release-request-reason">
   <xsd:union>
      <xsd:simpleType>
         <xsd:restriction base="xsd:token">
            <xsd:enumeration value="normal"/>
            <xsd:enumeration value="urgent"/>
            <xsd:enumeration value="user-defined"/>
         </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
         <xsd:restriction base="xsd:integer"/>
      </xsd:simpleType>
   </xsd:union>
</xsd:simpleType>

<xsd:simpleType name="Release-response-reason">
   <xsd:union>
      <xsd:simpleType>
         <xsd:restriction base="xsd:token">
            <xsd:enumeration value="normal"/>
            <xsd:enumeration value="not-finished"/>
            <xsd:enumeration value="user-defined"/>
         </xsd:restriction>
      </xsd:simpleType>
      <xsd:simpleType>
         <xsd:restriction base="xsd:integer"/>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
      </xsd:simpleType>
   </xsd:union>
</xsd:simpleType>

<xsd:simpleType name="Integer8">
   <xsd:restriction base="xsd:byte"/>
</xsd:simpleType>

<xsd:simpleType name="Integer16">
   <xsd:restriction base="xsd:short"/>
</xsd:simpleType>

<xsd:simpleType name="Integer32">
   <xsd:restriction base="xsd:int"/>
</xsd:simpleType>

<xsd:simpleType name="Integer64">
   <xsd:restriction base="xsd:long"/>
</xsd:simpleType>

<xsd:simpleType name="Unsigned8">
   <xsd:restriction base="xsd:unsignedByte"/>
</xsd:simpleType>

<xsd:simpleType name="Unsigned16">
   <xsd:restriction base="xsd:unsignedShort"/>
</xsd:simpleType>

<xsd:simpleType name="Unsigned32">
   <xsd:restriction base="xsd:unsignedInt"/>
</xsd:simpleType>

<xsd:simpleType name="Unsigned64">
   <xsd:restriction base="xsd:unsignedLong"/>
</xsd:simpleType>

<xsd:simpleType name="Conformance">
   <xsd:union memberTypes="BitString">
   <xsd:simpleType>
      <xsd:list>
         <xsd:simpleType>
            <xsd:restriction base="xsd:token">
               <xsd:enumeration value="reserved-zero"/>
               <xsd:enumeration value="general-protection"/>
               <xsd:enumeration value="general-block-transfer"/>
               <xsd:enumeration value="read"/>
               <xsd:enumeration value="write"/>
               <xsd:enumeration value="unconfirmed-write"/>
               <xsd:enumeration value="delta-value-encoding"/>
               <xsd:enumeration value="reserved-seven"/>
               <xsd:enumeration value="attribute0-supported-with-set"/>
               <xsd:enumeration value="priority-mgmt-supported"/>
               <xsd:enumeration value="attribute0-supported-with-get"/>
               <xsd:enumeration value="block-transfer-with-get-or-read"/>
               <xsd:enumeration value="block-transfer-with-set-or-write"/>
               <xsd:enumeration value="block-transfer-with-action"/>
               <xsd:enumeration value="multiple-references"/>
               <xsd:enumeration value="information-report"/>
               <xsd:enumeration value="data-notification"/>
               <xsd:enumeration value="access"/>
               <xsd:enumeration value="parameterized-access"/>
               <xsd:enumeration value="get"/>
               <xsd:enumeration value="set"/>
               <xsd:enumeration value="selective-access"/>
               <xsd:enumeration value="event-notification"/>
               <xsd:enumeration value="action"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:list>
   </xsd:simpleType>
   </xsd:union>
</xsd:simpleType>

<xsd:simpleType name="ObjectName">
   <xsd:restriction base="Integer16"/>
</xsd:simpleType>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
<xsd:simpleType name="Data-Access-Result">
   <xsd:restriction base="xsd:token">
      <xsd:enumeration value="success"/>
      <xsd:enumeration value="hardware-fault"/>
      <xsd:enumeration value="temporary-failure"/>
      <xsd:enumeration value="read-write-denied"/>
      <xsd:enumeration value="object-undefined"/>
      <xsd:enumeration value="object-class-inconsistent"/>
      <xsd:enumeration value="object-unavailable"/>
      <xsd:enumeration value="type-unmatched"/>
      <xsd:enumeration value="scope-of-access-violated"/>
      <xsd:enumeration value="data-block-unavailable"/>
      <xsd:enumeration value="long-get-aborted"/>
      <xsd:enumeration value="no-long-get-in-progress"/>
      <xsd:enumeration value="long-set-aborted"/>
      <xsd:enumeration value="no-long-set-in-progress"/>
      <xsd:enumeration value="data-block-number-invalid"/>
      <xsd:enumeration value="other-reason"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="Action-Result">
   <xsd:restriction base="xsd:token">
      <xsd:enumeration value="success"/>
      <xsd:enumeration value="hardware-fault"/>
      <xsd:enumeration value="temporary-failure"/>
      <xsd:enumeration value="read-write-denied"/>
      <xsd:enumeration value="object-undefined"/>
      <xsd:enumeration value="object-class-inconsistent"/>
      <xsd:enumeration value="object-unavailable"/>
      <xsd:enumeration value="type-unmatched"/>
      <xsd:enumeration value="scope-of-access-violated"/>
      <xsd:enumeration value="data-block-unavailable"/>
      <xsd:enumeration value="long-action-aborted"/>
      <xsd:enumeration value="no-long-action-in-progress"/>
      <xsd:enumeration value="other-reason"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="Invoke-Id-And-Priority">
   <xsd:restriction base="Unsigned8"/>
</xsd:simpleType>

<xsd:simpleType name="Long-Invoke-Id-And-Priority">
   <xsd:restriction base="Unsigned32"/>
</xsd:simpleType>

<xsd:simpleType name="Cosem-Class-Id">
   <xsd:restriction base="Unsigned16"/>
</xsd:simpleType>

<xsd:simpleType name="Cosem-Object-Instance-Id">
   <xsd:restriction base="xsd:hexBinary">
      <xsd:length value="6"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="Cosem-Object-Attribute-Id">
   <xsd:restriction base="Integer8"/>
</xsd:simpleType>

<xsd:simpleType name="Cosem-Object-Method-Id">
   <xsd:restriction base="Integer8"/>
</xsd:simpleType>

<xsd:simpleType name="Key-Id">
   <xsd:restriction base="xsd:token">
      <xsd:enumeration value="global-unicast-encryption-key"/>
      <xsd:enumeration value="global-broadcast-encryption-key"/>
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="Kek-Id">
   <xsd:restriction base="xsd:token">
      <xsd:enumeration value="master-key"/>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
   </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="Block-Control">
   <xsd:restriction base="Unsigned8"/>
</xsd:simpleType>

<xsd:complexType name="Authentication-value">
   <xsd:choice>
      <xsd:element name="charstring" type="xsd:string"/>
      <xsd:element name="bitstring" type="BitString"/>
   </xsd:choice>
</xsd:complexType>

<xsd:complexType name="AARQ-apdu">
   <xsd:sequence>
      <xsd:element name="protocol-version" minOccurs="0">
         <xsd:simpleType>
            <xsd:union memberTypes="BitString">
            <xsd:simpleType>
               <xsd:list>
                  <xsd:simpleType>
                     <xsd:restriction base="xsd:token">
                        <xsd:enumeration value="version1"/>
                     </xsd:restriction>
                  </xsd:simpleType>
               </xsd:list>
            </xsd:simpleType>
            </xsd:union>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="application-context-name" type="Application-context-name"/>
      <xsd:element name="called-AP-title" minOccurs="0" type="AP-title"/>
      <xsd:element name="called-AE-qualifier" minOccurs="0" type="AE-qualifier"/>
      <xsd:element name="called-AP-invocation-id" minOccurs="0" type="AP-invocation-identifier"/>
      <xsd:element name="called-AE-invocation-id" minOccurs="0" type="AE-invocation-identifier"/>
      <xsd:element name="calling-AP-title" minOccurs="0" type="AP-title"/>
      <xsd:element name="calling-AE-qualifier" minOccurs="0" type="AE-qualifier"/>
      <xsd:element name="calling-AP-invocation-id" minOccurs="0" type="AP-invocation-identifier"/>
      <xsd:element name="calling-AE-invocation-id" minOccurs="0" type="AE-invocation-identifier"/>
      <xsd:element name="sender-acse-requirements" minOccurs="0" type="ACSE-requirements"/>
      <xsd:element name="mechanism-name" minOccurs="0" type="Mechanism-name"/>
      <xsd:element name="calling-authentication-value" minOccurs="0" type="Authentication-value"/>
      <xsd:element name="implementation-information" minOccurs="0" type="Implementation-data"/>
      <xsd:element name="user-information" minOccurs="0" type="Association-information"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Associate-source-diagnostic">
   <xsd:choice>
      <xsd:element name="acse-service-user">
         <xsd:simpleType>
            <xsd:union>
               <xsd:simpleType>
                  <xsd:restriction base="xsd:token">
                     <xsd:enumeration value="null"/>
                     <xsd:enumeration value="no-reason-given"/>
                     <xsd:enumeration value="application-context-name-not-supported"/>
                     <xsd:enumeration value="calling-AP-title-not-recognized"/>
                     <xsd:enumeration value="calling-AP-invocation-identifier-not-recognized"/>
                     <xsd:enumeration value="calling-AE-qualifier-not-recognized"/>
                     <xsd:enumeration value="calling-AE-invocation-identifier-not-recognized"/>
                     <xsd:enumeration value="called-AP-title-not-recognized"/>
                     <xsd:enumeration value="called-AP-invocation-identifier-not-recognized"/>
                     <xsd:enumeration value="called-AE-qualifier-not-recognized"/>
                     <xsd:enumeration value="called-AE-invocation-identifier-not-recognized"/>
                     <xsd:enumeration value="authentication-mechanism-name-not-recognised"/>
                     <xsd:enumeration value="authentication-mechanism-name-required"/>
                     <xsd:enumeration value="authentication-failure"/>
                     <xsd:enumeration value="authentication-required"/>
                  </xsd:restriction>
               </xsd:simpleType>
               <xsd:simpleType>
                  <xsd:restriction base="xsd:integer"/>
               </xsd:simpleType>
            </xsd:union>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="acse-service-provider">
            <xsd:simpleType>
               <xsd:union>
                  <xsd:simpleType>
                     <xsd:restriction base="xsd:token">
                        <xsd:enumeration value="null"/>
                        <xsd:enumeration value="no-reason-given"/>
                        <xsd:enumeration value="no-common-acse-version"/>
                     </xsd:restriction>
                  </xsd:simpleType>
                  <xsd:simpleType>
                     <xsd:restriction base="xsd:integer"/>
                  </xsd:simpleType>
               </xsd:union>
            </xsd:simpleType>
         </xsd:element>
      </xsd:choice>
   </xsd:complexType>

   <xsd:complexType name="AARE-apdu">
      <xsd:sequence>
         <xsd:element name="protocol-version" minOccurs="0">
            <xsd:simpleType>
               <xsd:union memberTypes="BitString">
               <xsd:simpleType>
                  <xsd:list>
                     <xsd:simpleType>
                        <xsd:restriction base="xsd:token">
                           <xsd:enumeration value="version1"/>
                        </xsd:restriction>
                     </xsd:simpleType>
                  </xsd:list>
               </xsd:simpleType>
               </xsd:union>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="application-context-name" type="Application-context-name"/>
         <xsd:element name="result" type="Association-result"/>
         <xsd:element name="result-source-diagnostic" type="Associate-source-diagnostic"/>
         <xsd:element name="responding-AP-title" minOccurs="0" type="AP-title"/>
         <xsd:element name="responding-AE-qualifier" minOccurs="0" type="AE-qualifier"/>
         <xsd:element name="responding-AP-invocation-id" minOccurs="0" type="AP-invocation-
identifier"/>
         <xsd:element name="responding-AE-invocation-id" minOccurs="0" type="AE-invocation-
identifier"/>
         <xsd:element name="responder-acse-requirements" minOccurs="0" type="ACSE-requirements"/>
         <xsd:element name="mechanism-name" minOccurs="0" type="Mechanism-name"/>
         <xsd:element name="responding-authentication-value" minOccurs="0" type="Authentication-
value"/>
         <xsd:element name="implementation-information" minOccurs="0" type="Implementation-data"/>
         <xsd:element name="user-information" minOccurs="0" type="Association-information"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="RLRQ-apdu">
      <xsd:sequence>
         <xsd:element name="reason" minOccurs="0" type="Release-request-reason"/>
         <xsd:element name="user-information" minOccurs="0" type="Association-information"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="RLRE-apdu">
      <xsd:sequence>
         <xsd:element name="reason" minOccurs="0" type="Release-response-reason"/>
         <xsd:element name="user-information" minOccurs="0" type="Association-information"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="InitiateRequest">
      <xsd:sequence>
         <xsd:element name="dedicated-key" minOccurs="0" type="xsd:hexBinary"/>
         <xsd:element name="response-allowed" default="true" type="xsd:boolean"/>
         <xsd:element name="proposed-quality-of-service" minOccurs="0" type="Integer8"/>
         <xsd:element name="proposed-dlms-version-number" type="Unsigned8"/>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
      <xsd:element name="proposed-conformance" type="Conformance"/>
      <xsd:element name="client-max-receive-pdu-size" type="Unsigned16"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="TypeDescription">
   <xsd:choice>
      <xsd:element name="null-data" type="NULL"/>
      <xsd:element name="array">
         <xsd:complexType>
            <xsd:sequence>
               <xsd:element name="number-of-elements" type="Unsigned16"/>
               <xsd:element name="type-description" type="TypeDescription"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="structure">
         <xsd:complexType>
            <xsd:sequence minOccurs="0" maxOccurs="unbounded">
               <xsd:element name="TypeDescription" type="TypeDescription"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="boolean" type="NULL"/>
      <xsd:element name="bit-string" type="NULL"/>
      <xsd:element name="double-long" type="NULL"/>
      <xsd:element name="double-long-unsigned" type="NULL"/>
      <xsd:element name="octet-string" type="NULL"/>
      <xsd:element name="visible-string" type="NULL"/>
      <xsd:element name="utf8-string" type="NULL"/>
      <xsd:element name="bcd" type="NULL"/>
      <xsd:element name="integer" type="NULL"/>
      <xsd:element name="long" type="NULL"/>
      <xsd:element name="unsigned" type="NULL"/>
      <xsd:element name="long-unsigned" type="NULL"/>
      <xsd:element name="long64" type="NULL"/>
      <xsd:element name="long64-unsigned" type="NULL"/>
      <xsd:element name="enum" type="NULL"/>
      <xsd:element name="float32" type="NULL"/>
      <xsd:element name="float64" type="NULL"/>
      <xsd:element name="date-time" type="NULL"/>
      <xsd:element name="date" type="NULL"/>
      <xsd:element name="time" type="NULL"/>
      <xsd:element name="dont-care" type="NULL"/>
   </xsd:choice>
</xsd:complexType>

 <xsd:complexType name="SequenceOfData">
     <xsd:choice minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="null-data" type="NULL"/>
         <xsd:element name="array" type="SequenceOfData"/>
         <xsd:element name="structure" type="SequenceOfData"/>
         <xsd:element name="boolean" type="xsd:boolean"/>
         <xsd:element name="bit-string" type="BitString"/>
         <xsd:element name="double-long" type="Integer32"/>
         <xsd:element name="double-long-unsigned" type="Unsigned32"/>
         <xsd:element name="octet-string" type="xsd:hexBinary"/>
         <xsd:element name="visible-string" type="xsd:string"/>
         <xsd:element name="utf8-string" type="xsd:string"/>
         <xsd:element name="bcd" type="Integer8"/>
         <xsd:element name="integer" type="Integer8"/>
         <xsd:element name="long" type="Integer16"/>
         <xsd:element name="unsigned" type="Unsigned8"/>
         <xsd:element name="long-unsigned" type="Unsigned16"/>
         <xsd:element name="compact-array">
             <xsd:complexType>
                 <xsd:sequence>
                     <xsd:element name="contents-description" type="TypeDescription"/>
                     <xsd:element name="array-contents" type="xsd:hexBinary"/>
                 </xsd:sequence>
             </xsd:complexType>
         </xsd:element>
         <xsd:element name="long64" type="Integer64"/>
         <xsd:element name="long64-unsigned" type="Unsigned64"/>
         <xsd:element name="enum" type="Unsigned8"/>
         <xsd:element name="float32" type="xsd:float"/>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
            <xsd:element name="float64" type="xsd:double"/>
            <xsd:element name="date-time">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:hexBinary">
                        <xsd:length value="12"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="date">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:hexBinary">
                        <xsd:length value="5"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="time">
                <xsd:simpleType>
                    <xsd:restriction base="xsd:hexBinary">
                        <xsd:length value="4"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="delta-integer" type="Integer8"/>
            <xsd:element name="delta-long" type="Integer16"/>
            <xsd:element name="delta-double-long" type="Integer32"/>
            <xsd:element name="delta-unsigned" type="Unsigned8"/>
            <xsd:element name="delta-long-unsigned" type="Unsigned16"/>
            <xsd:element name="delta-double-long-unsigned" type="Unsigned32"/>
            <xsd:element name="dont-care" type="NULL"/>
        </xsd:choice>
    </xsd:complexType>

    <xsd:complexType name="Data">
      <xsd:choice>
        <xsd:element name="null-data" type="NULL"/>
        <xsd:element name="array" type="SequenceOfData"/>
        <xsd:element name="structure" type="SequenceOfData"/>
        <xsd:element name="boolean" type="xsd:boolean"/>
        <xsd:element name="bit-string" type="BitString"/>
        <xsd:element name="double-long" type="Integer32"/>
        <xsd:element name="double-long-unsigned" type="Unsigned32"/>
        <xsd:element name="octet-string" type="xsd:hexBinary"/>
        <xsd:element name="visible-string" type="xsd:string"/>
        <xsd:element name="utf8-string" type="xsd:string"/>
        <xsd:element name="bcd" type="Integer8"/>
        <xsd:element name="integer" type="Integer8"/>
        <xsd:element name="long" type="Integer16"/>
        <xsd:element name="unsigned" type="Unsigned8"/>
        <xsd:element name="long-unsigned" type="Unsigned16"/>
        <xsd:element name="compact-array">
           <xsd:complexType>
              <xsd:sequence>
                 <xsd:element name="contents-description" type="TypeDescription"/>
                 <xsd:element name="array-contents" type="xsd:hexBinary"/>
              </xsd:sequence>
           </xsd:complexType>
        </xsd:element>
        <xsd:element name="long64" type="Integer64"/>
        <xsd:element name="long64-unsigned" type="Unsigned64"/>
        <xsd:element name="enum" type="Unsigned8"/>
        <xsd:element name="float32" type="xsd:float"/>
        <xsd:element name="float64" type="xsd:double"/>
        <xsd:element name="date-time">
           <xsd:simpleType>
              <xsd:restriction base="xsd:hexBinary">
                 <xsd:length value="12"/>
              </xsd:restriction>
           </xsd:simpleType>
        </xsd:element>
        <xsd:element name="date">
           <xsd:simpleType>
              <xsd:restriction base="xsd:hexBinary">
                 <xsd:length value="5"/>
              </xsd:restriction>
           </xsd:simpleType>
        </xsd:element>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
            <xsd:element name="time">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:hexBinary">
                     <xsd:length value="4"/>
                  </xsd:restriction>
               </xsd:simpleType>
            </xsd:element>
            <xsd:element name="delta-integer" type="Integer8"/>
            <xsd:element name="delta-long" type="Integer16"/>
            <xsd:element name="delta-double-long" type="Integer32"/>
            <xsd:element name="delta-unsigned" type="Unsigned8"/>
            <xsd:element name="delta-long-unsigned" type="Unsigned16"/>
            <xsd:element name="delta-double-long-unsigned" type="Unsigned32"/>
            <xsd:element name="dont-care" type="NULL"/>
         </xsd:choice>
      </xsd:complexType>

      <xsd:complexType name="Parameterized-Access">
         <xsd:sequence>
            <xsd:element name="variable-name" type="ObjectName"/>
            <xsd:element name="selector" type="Unsigned8"/>
            <xsd:element name="parameter" type="Data"/>
         </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="Block-Number-Access">
         <xsd:sequence>
            <xsd:element name="block-number" type="Unsigned16"/>
         </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="Read-Data-Block-Access">
         <xsd:sequence>
            <xsd:element name="last-block" type="xsd:boolean"/>
            <xsd:element name="block-number" type="Unsigned16"/>
            <xsd:element name="raw-data" type="xsd:hexBinary"/>
         </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="Write-Data-Block-Access">
         <xsd:sequence>
            <xsd:element name="last-block" type="xsd:boolean"/>
            <xsd:element name="block-number" type="Unsigned16"/>
         </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="Variable-Access-Specification">
         <xsd:choice>
            <xsd:element name="variable-name" type="ObjectName"/>
            <xsd:element name="parameterized-access" type="Parameterized-Access"/>
            <xsd:element name="block-number-access" type="Block-Number-Access"/>
            <xsd:element name="read-data-block-access" type="Read-Data-Block-Access"/>
            <xsd:element name="write-data-block-access" type="Write-Data-Block-Access"/>
         </xsd:choice>
      </xsd:complexType>

      <xsd:complexType name="ReadRequest">
         <xsd:sequence minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="Variable-Access-Specification" type="Variable-Access-Specification"/>
         </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="WriteRequest">
         <xsd:sequence>
            <xsd:element name="variable-access-specification">
               <xsd:complexType>
                  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                     <xsd:element name="Variable-Access-Specification" type="Variable-Access-
Specification"/>
                  </xsd:sequence>
               </xsd:complexType>
            </xsd:element>
            <xsd:element name="list-of-data">
               <xsd:complexType>
                  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                     <xsd:element name="Data" type="Data"/>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="InitiateResponse">
   <xsd:sequence>
      <xsd:element name="negotiated-quality-of-service" minOccurs="0" type="Integer8"/>
      <xsd:element name="negotiated-dlms-version-number" type="Unsigned8"/>
      <xsd:element name="negotiated-conformance" type="Conformance"/>
      <xsd:element name="server-max-receive-pdu-size" type="Unsigned16"/>
      <xsd:element name="vaa-name" type="ObjectName"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Data-Block-Result">
   <xsd:sequence>
      <xsd:element name="last-block" type="xsd:boolean"/>
      <xsd:element name="block-number" type="Unsigned16"/>
      <xsd:element name="raw-data" type="xsd:hexBinary"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ReadResponse">
   <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="CHOICE">
         <xsd:complexType>
            <xsd:choice>
               <xsd:element name="data" type="Data"/>
               <xsd:element name="data-access-error" type="Data-Access-Result"/>
               <xsd:element name="data-block-result" type="Data-Block-Result"/>
               <xsd:element name="block-number" type="Unsigned16"/>
            </xsd:choice>
         </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="WriteResponse">
   <xsd:sequence minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="CHOICE">
         <xsd:complexType>
            <xsd:choice>
               <xsd:element name="success" type="NULL"/>
               <xsd:element name="data-access-error" type="Data-Access-Result"/>
               <xsd:element name="block-number" type="Unsigned16"/>
            </xsd:choice>
         </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="ServiceError">
   <xsd:choice>
      <xsd:element name="application-reference">
         <xsd:simpleType>
            <xsd:restriction base="xsd:token">
               <xsd:enumeration value="other"/>
               <xsd:enumeration value="time-elapsed"/>
               <xsd:enumeration value="application-unreachable"/>
               <xsd:enumeration value="application-reference-invalid"/>
               <xsd:enumeration value="application-context-unsupported"/>
               <xsd:enumeration value="provider-communication-error"/>
               <xsd:enumeration value="deciphering-error"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="hardware-resource">
         <xsd:simpleType>
            <xsd:restriction base="xsd:token">
               <xsd:enumeration value="other"/>
               <xsd:enumeration value="memory-unavailable"/>
               <xsd:enumeration value="processor-resource-unavailable"/>
               <xsd:enumeration value="mass-storage-unavailable"/>
               <xsd:enumeration value="other-resource-unavailable"/>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
<xsd:element name="vde-state-error">
   <xsd:simpleType>
      <xsd:restriction base="xsd:token">
         <xsd:enumeration value="other"/>
         <xsd:enumeration value="no-dlms-context"/>
         <xsd:enumeration value="loading-data-set"/>
         <xsd:enumeration value="status-nochange"/>
         <xsd:enumeration value="status-inoperable"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
<xsd:element name="service">
   <xsd:simpleType>
      <xsd:restriction base="xsd:token">
         <xsd:enumeration value="other"/>
         <xsd:enumeration value="pdu-size"/>
         <xsd:enumeration value="service-unsupported"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
<xsd:element name="definition">
   <xsd:simpleType>
      <xsd:restriction base="xsd:token">
         <xsd:enumeration value="other"/>
         <xsd:enumeration value="object-undefined"/>
         <xsd:enumeration value="object-class-inconsistent"/>
         <xsd:enumeration value="object-attribute-inconsistent"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
<xsd:element name="access">
   <xsd:simpleType>
      <xsd:restriction base="xsd:token">
         <xsd:enumeration value="other"/>
         <xsd:enumeration value="scope-of-access-violated"/>
         <xsd:enumeration value="object-access-violated"/>
         <xsd:enumeration value="hardware-fault"/>
         <xsd:enumeration value="object-unavailable"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
<xsd:element name="initiate">
   <xsd:simpleType>
      <xsd:restriction base="xsd:token">
         <xsd:enumeration value="other"/>
         <xsd:enumeration value="dlms-version-too-low"/>
         <xsd:enumeration value="incompatible-conformance"/>
         <xsd:enumeration value="pdu-size-too-short"/>
         <xsd:enumeration value="refused-by-the-VDE-Handler"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
<xsd:element name="load-data-set">
   <xsd:simpleType>
      <xsd:restriction base="xsd:token">
         <xsd:enumeration value="other"/>
         <xsd:enumeration value="primitive-out-of-sequence"/>
         <xsd:enumeration value="not-loadable"/>
         <xsd:enumeration value="dataset-size-too-large"/>
         <xsd:enumeration value="not-awaited-segment"/>
         <xsd:enumeration value="interpretation-failure"/>
         <xsd:enumeration value="storage-failure"/>
         <xsd:enumeration value="data-set-not-ready"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
<xsd:element name="task">
   <xsd:simpleType>
      <xsd:restriction base="xsd:token">
         <xsd:enumeration value="other"/>
         <xsd:enumeration value="no-remote-control"/>
         <xsd:enumeration value="ti-stopped"/>
         <xsd:enumeration value="ti-running"/>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
                <xsd:enumeration value="ti-unusable"/>
             </xsd:restriction>
          </xsd:simpleType>
       </xsd:element>
    </xsd:choice>
 </xsd:complexType>

 <xsd:complexType name="ConfirmedServiceError">
    <xsd:choice>
       <xsd:element name="initiateError" type="ServiceError"/>
       <xsd:element name="getStatus" type="ServiceError"/>
       <xsd:element name="getNameList" type="ServiceError"/>
       <xsd:element name="getVariableAttribute" type="ServiceError"/>
       <xsd:element name="read" type="ServiceError"/>
       <xsd:element name="write" type="ServiceError"/>
       <xsd:element name="getDataSetAttribute" type="ServiceError"/>
       <xsd:element name="getTIAttribute" type="ServiceError"/>
       <xsd:element name="changeScope" type="ServiceError"/>
       <xsd:element name="start" type="ServiceError"/>
       <xsd:element name="stop" type="ServiceError"/>
       <xsd:element name="resume" type="ServiceError"/>
       <xsd:element name="makeUsable" type="ServiceError"/>
       <xsd:element name="initiateLoad" type="ServiceError"/>
       <xsd:element name="loadSegment" type="ServiceError"/>
       <xsd:element name="terminateLoad" type="ServiceError"/>
       <xsd:element name="initiateUpLoad" type="ServiceError"/>
       <xsd:element name="upLoadSegment" type="ServiceError"/>
       <xsd:element name="terminateUpLoad" type="ServiceError"/>
    </xsd:choice>
 </xsd:complexType>

 <xsd:complexType name="Notification-Body">
    <xsd:sequence>
       <xsd:element name="data-value" type="Data"/>
    </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="Data-Notification">
    <xsd:sequence>
       <xsd:element name="long-invoke-id-and-priority" type="Long-Invoke-Id-And-Priority"/>
       <xsd:element name="date-time" type="xsd:hexBinary"/>
       <xsd:element name="notification-body" type="Notification-Body"/>
    </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="UnconfirmedWriteRequest">
    <xsd:sequence>
       <xsd:element name="variable-access-specification">
          <xsd:complexType>
             <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="Variable-Access-Specification" type="Variable-Access-
Specification"/>
             </xsd:sequence>
          </xsd:complexType>
       </xsd:element>
       <xsd:element name="list-of-data">
          <xsd:complexType>
             <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="Data" type="Data"/>
             </xsd:sequence>
          </xsd:complexType>
       </xsd:element>
    </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="InformationReportRequest">
    <xsd:sequence>
       <xsd:element name="current-time" minOccurs="0" type="xsd:dateTime"/>
       <xsd:element name="variable-access-specification">
          <xsd:complexType>
             <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                <xsd:element name="Variable-Access-Specification" type="Variable-Access-
Specification"/>
             </xsd:sequence>
          </xsd:complexType>
       </xsd:element>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
        <xsd:element name="list-of-data">
           <xsd:complexType>
              <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                 <xsd:element name="Data" type="Data"/>
              </xsd:sequence>
           </xsd:complexType>
        </xsd:element>
     </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Cosem-Attribute-Descriptor">
     <xsd:sequence>
        <xsd:element name="class-id" type="Cosem-Class-Id"/>
        <xsd:element name="instance-id" type="Cosem-Object-Instance-Id"/>
        <xsd:element name="attribute-id" type="Cosem-Object-Attribute-Id"/>
     </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Selective-Access-Descriptor">
     <xsd:sequence>
        <xsd:element name="access-selector" type="Unsigned8"/>
        <xsd:element name="access-parameters" type="Data"/>
     </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Get-Request-Normal">
     <xsd:sequence>
        <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
        <xsd:element name="cosem-attribute-descriptor" type="Cosem-Attribute-Descriptor"/>
        <xsd:element name="access-selection" minOccurs="0" type="Selective-Access-Descriptor"/>
     </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Get-Request-Next">
     <xsd:sequence>
        <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
        <xsd:element name="block-number" type="Unsigned32"/>
     </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Cosem-Attribute-Descriptor-With-Selection">
     <xsd:sequence>
        <xsd:element name="cosem-attribute-descriptor" type="Cosem-Attribute-Descriptor"/>
        <xsd:element name="access-selection" minOccurs="0" type="Selective-Access-Descriptor"/>
     </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Get-Request-With-List">
     <xsd:sequence>
        <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
        <xsd:element name="attribute-descriptor-list">
           <xsd:complexType>
              <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                 <xsd:element name="Cosem-Attribute-Descriptor-With-Selection" type="Cosem-Attribute-
Descriptor-With-Selection"/>
              </xsd:sequence>
           </xsd:complexType>
        </xsd:element>
     </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Get-Request">
     <xsd:choice>
        <xsd:element name="get-request-normal" type="Get-Request-Normal"/>
        <xsd:element name="get-request-next" type="Get-Request-Next"/>
        <xsd:element name="get-request-with-list" type="Get-Request-With-List"/>
     </xsd:choice>
  </xsd:complexType>

  <xsd:complexType name="Set-Request-Normal">
     <xsd:sequence>
        <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
        <xsd:element name="cosem-attribute-descriptor" type="Cosem-Attribute-Descriptor"/>
        <xsd:element name="access-selection" minOccurs="0" type="Selective-Access-Descriptor"/>
        <xsd:element name="value" type="Data"/>
     </xsd:sequence>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
   </xsd:complexType>

   <xsd:complexType name="DataBlock-SA">
      <xsd:sequence>
         <xsd:element name="last-block" type="xsd:boolean"/>
         <xsd:element name="block-number" type="Unsigned32"/>
         <xsd:element name="raw-data" type="xsd:hexBinary"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Set-Request-With-First-Datablock">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="cosem-attribute-descriptor" type="Cosem-Attribute-Descriptor"/>
         <xsd:element name="access-selection" minOccurs="0" type="Selective-Access-Descriptor"/>
         <xsd:element name="datablock" type="DataBlock-SA"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Set-Request-With-Datablock">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="datablock" type="DataBlock-SA"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Set-Request-With-List">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="attribute-descriptor-list">
            <xsd:complexType>
               <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                  <xsd:element name="Cosem-Attribute-Descriptor-With-Selection" type="Cosem-Attribute-
Descriptor-With-Selection"/>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
         <xsd:element name="value-list">
            <xsd:complexType>
               <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                  <xsd:element name="Data" type="Data"/>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Set-Request-With-List-And-First-Datablock">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="attribute-descriptor-list">
            <xsd:complexType>
               <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                  <xsd:element name="Cosem-Attribute-Descriptor-With-Selection" type="Cosem-Attribute-
Descriptor-With-Selection"/>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
         <xsd:element name="datablock" type="DataBlock-SA"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Set-Request">
      <xsd:choice>
         <xsd:element name="set-request-normal" type="Set-Request-Normal"/>
         <xsd:element name="set-request-with-first-datablock" type="Set-Request-With-First-Datablock"/>
         <xsd:element name="set-request-with-datablock" type="Set-Request-With-Datablock"/>
         <xsd:element name="set-request-with-list" type="Set-Request-With-List"/>
         <xsd:element name="set-request-with-list-and-first-datablock" type="Set-Request-With-List-And-
First-Datablock"/>
      </xsd:choice>
   </xsd:complexType>

   <xsd:complexType name="EventNotificationRequest">
      <xsd:sequence>
         <xsd:element name="time" minOccurs="0" type="xsd:hexBinary"/>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
      <xsd:element name="cosem-attribute-descriptor" type="Cosem-Attribute-Descriptor"/>
      <xsd:element name="attribute-value" type="Data"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Cosem-Method-Descriptor">
   <xsd:sequence>
      <xsd:element name="class-id" type="Cosem-Class-Id"/>
      <xsd:element name="instance-id" type="Cosem-Object-Instance-Id"/>
      <xsd:element name="method-id" type="Cosem-Object-Method-Id"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Action-Request-Normal">
   <xsd:sequence>
      <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
      <xsd:element name="cosem-method-descriptor" type="Cosem-Method-Descriptor"/>
      <xsd:element name="method-invocation-parameters" minOccurs="0" type="Data"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Action-Request-Next-Pblock">
   <xsd:sequence>
      <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
      <xsd:element name="block-number" type="Unsigned32"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Action-Request-With-List">
   <xsd:sequence>
      <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
      <xsd:element name="cosem-method-descriptor-list">
         <xsd:complexType>
            <xsd:sequence minOccurs="0" maxOccurs="unbounded">
               <xsd:element name="Cosem-Method-Descriptor" type="Cosem-Method-Descriptor"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="method-invocation-parameters">
         <xsd:complexType>
            <xsd:sequence minOccurs="0" maxOccurs="unbounded">
               <xsd:element name="Data" type="Data"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Action-Request-With-First-Pblock">
   <xsd:sequence>
      <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
      <xsd:element name="cosem-method-descriptor" type="Cosem-Method-Descriptor"/>
      <xsd:element name="pblock" type="DataBlock-SA"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Action-Request-With-List-And-First-Pblock">
   <xsd:sequence>
      <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
      <xsd:element name="cosem-method-descriptor-list">
         <xsd:complexType>
            <xsd:sequence minOccurs="0" maxOccurs="unbounded">
               <xsd:element name="Cosem-Method-Descriptor" type="Cosem-Method-Descriptor"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
      <xsd:element name="pblock" type="DataBlock-SA"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Action-Request-With-Pblock">
   <xsd:sequence>
      <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
      <xsd:element name="pblock" type="DataBlock-SA"/>
   </xsd:sequence>
</xsd:complexType>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
   <xsd:complexType name="Action-Request">
      <xsd:choice>
         <xsd:element name="action-request-normal" type="Action-Request-Normal"/>
         <xsd:element name="action-request-next-pblock" type="Action-Request-Next-Pblock"/>
         <xsd:element name="action-request-with-list" type="Action-Request-With-List"/>
         <xsd:element name="action-request-with-first-pblock" type="Action-Request-With-First-Pblock"/>
         <xsd:element name="action-request-with-list-and-first-pblock" type="Action-Request-With-List-
And-First-Pblock"/>
         <xsd:element name="action-request-with-pblock" type="Action-Request-With-Pblock"/>
      </xsd:choice>
   </xsd:complexType>

   <xsd:complexType name="Get-Data-Result">
      <xsd:choice>
         <xsd:element name="data" type="Data"/>
         <xsd:element name="data-access-result" type="Data-Access-Result"/>
      </xsd:choice>
   </xsd:complexType>

   <xsd:complexType name="Get-Response-Normal">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="result" type="Get-Data-Result"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="DataBlock-G">
      <xsd:sequence>
         <xsd:element name="last-block" type="xsd:boolean"/>
         <xsd:element name="block-number" type="Unsigned32"/>
         <xsd:element name="result">
            <xsd:complexType>
               <xsd:choice>
                  <xsd:element name="raw-data" type="xsd:hexBinary"/>
                  <xsd:element name="data-access-result" type="Data-Access-Result"/>
               </xsd:choice>
            </xsd:complexType>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Get-Response-With-Datablock">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="result" type="DataBlock-G"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Get-Response-With-List">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="result">
            <xsd:complexType>
               <xsd:sequence minOccurs="0" maxOccurs="unbounded">
                  <xsd:element name="Get-Data-Result" type="Get-Data-Result"/>
               </xsd:sequence>
            </xsd:complexType>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Get-Response">
      <xsd:choice>
         <xsd:element name="get-response-normal" type="Get-Response-Normal"/>
         <xsd:element name="get-response-with-datablock" type="Get-Response-With-Datablock"/>
         <xsd:element name="get-response-with-list" type="Get-Response-With-List"/>
      </xsd:choice>
   </xsd:complexType>

   <xsd:complexType name="Set-Response-Normal">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="result" type="Data-Access-Result"/>
      </xsd:sequence>
   </xsd:complexType>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
   <xsd:complexType name="Set-Response-Datablock">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="block-number" type="Unsigned32"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Set-Response-Last-Datablock">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="result" type="Data-Access-Result"/>
         <xsd:element name="block-number" type="Unsigned32"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Set-Response-Last-Datablock-With-List">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="result">
            <xsd:simpleType>
               <xsd:list itemType="Data-Access-Result"/>
            </xsd:simpleType>
         </xsd:element>
         <xsd:element name="block-number" type="Unsigned32"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Set-Response-With-List">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="result">
            <xsd:simpleType>
               <xsd:list itemType="Data-Access-Result"/>
            </xsd:simpleType>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Set-Response">
      <xsd:choice>
         <xsd:element name="set-response-normal" type="Set-Response-Normal"/>
         <xsd:element name="set-response-datablock" type="Set-Response-Datablock"/>
         <xsd:element name="set-response-last-datablock" type="Set-Response-Last-Datablock"/>
         <xsd:element name="set-response-last-datablock-with-list" type="Set-Response-Last-Datablock-
With-List"/>
         <xsd:element name="set-response-with-list" type="Set-Response-With-List"/>
      </xsd:choice>
   </xsd:complexType>

   <xsd:complexType name="Action-Response-With-Optional-Data">
      <xsd:sequence>
         <xsd:element name="result" type="Action-Result"/>
         <xsd:element name="return-parameters" minOccurs="0" type="Get-Data-Result"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Action-Response-Normal">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="single-response" type="Action-Response-With-Optional-Data"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Action-Response-With-Pblock">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="pblock" type="DataBlock-SA"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Action-Response-With-List">
      <xsd:sequence>
         <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
         <xsd:element name="list-of-responses">
            <xsd:complexType>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
            <xsd:sequence minOccurs="0" maxOccurs="unbounded">
               <xsd:element name="Action-Response-With-Optional-Data" type="Action-Response-With-
Optional-Data"/>
            </xsd:sequence>
         </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Action-Response-Next-Pblock">
   <xsd:sequence>
      <xsd:element name="invoke-id-and-priority" type="Invoke-Id-And-Priority"/>
      <xsd:element name="block-number" type="Unsigned32"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Action-Response">
   <xsd:choice>
      <xsd:element name="action-response-normal" type="Action-Response-Normal"/>
      <xsd:element name="action-response-with-pblock" type="Action-Response-With-Pblock"/>
      <xsd:element name="action-response-with-list" type="Action-Response-With-List"/>
      <xsd:element name="action-response-next-pblock" type="Action-Response-Next-Pblock"/>
   </xsd:choice>
</xsd:complexType>

<xsd:complexType name="ExceptionResponse">
   <xsd:sequence>
      <xsd:element name="state-error">
         <xsd:simpleType>
            <xsd:restriction base="xsd:token">
               <xsd:enumeration value="service-not-allowed"/>
               <xsd:enumeration value="service-unknown"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
      <xsd:element name="service-error">
         <xsd:simpleType>
            <xsd:restriction base="xsd:token">
               <xsd:enumeration value="operation-not-possible"/>
               <xsd:enumeration value="service-not-supported"/>
               <xsd:enumeration value="other-reason"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Access-Request-Get">
   <xsd:sequence>
      <xsd:element name="cosem-attribute-descriptor" type="Cosem-Attribute-Descriptor"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Access-Request-Set">
   <xsd:sequence>
      <xsd:element name="cosem-attribute-descriptor" type="Cosem-Attribute-Descriptor"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Access-Request-Action">
   <xsd:sequence>
      <xsd:element name="cosem-method-descriptor" type="Cosem-Method-Descriptor"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Access-Request-Get-With-Selection">
   <xsd:sequence>
      <xsd:element name="cosem-attribute-descriptor" type="Cosem-Attribute-Descriptor"/>
      <xsd:element name="access-selection" type="Selective-Access-Descriptor"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="Access-Request-Set-With-Selection">
   <xsd:sequence>
      <xsd:element name="cosem-attribute-descriptor" type="Cosem-Attribute-Descriptor"/>
      <xsd:element name="access-selection" type="Selective-Access-Descriptor"/>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Access-Request-Specification">
      <xsd:choice>
         <xsd:element name="access-request-get" type="Access-Request-Get"/>
         <xsd:element name="access-request-set" type="Access-Request-Set"/>
         <xsd:element name="access-request-action" type="Access-Request-Action"/>
         <xsd:element name="access-request-get-with-selection" type="Access-Request-Get-With-
Selection"/>
         <xsd:element name="access-request-set-with-selection" type="Access-Request-Set-With-
Selection"/>
      </xsd:choice>
   </xsd:complexType>

   <xsd:complexType name="List-Of-Access-Request-Specification">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="Access-Request-Specification" type="Access-Request-Specification"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="List-Of-Data">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="Data" type="Data"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Access-Request-Body">
      <xsd:sequence>
         <xsd:element name="access-request-specification" type="List-Of-Access-Request-Specification"/>
         <xsd:element name="access-request-list-of-data" type="List-Of-Data"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Access-Request">
      <xsd:sequence>
         <xsd:element name="long-invoke-id-and-priority" type="Long-Invoke-Id-And-Priority"/>
         <xsd:element name="date-time" type="xsd:hexBinary"/>
         <xsd:element name="access-request-body" type="Access-Request-Body"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Access-Response-Get">
      <xsd:sequence>
         <xsd:element name="result" type="Data-Access-Result"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Access-Response-Set">
      <xsd:sequence>
         <xsd:element name="result" type="Data-Access-Result"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Access-Response-Action">
      <xsd:sequence>
         <xsd:element name="result" type="Action-Result"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Access-Response-Specification">
      <xsd:choice>
         <xsd:element name="access-response-get" type="Access-Response-Get"/>
         <xsd:element name="access-response-set" type="Access-Response-Set"/>
         <xsd:element name="access-response-action" type="Access-Response-Action"/>
      </xsd:choice>
   </xsd:complexType>

   <xsd:complexType name="List-Of-Access-Response-Specification">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="Access-Response-Specification" type="Access-Response-Specification"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Access-Response-Body">
      <xsd:sequence>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
        <xsd:element name="access-request-specification" minOccurs="0" type="List-Of-Access-Request-
Specification"/>
        <xsd:element name="access-response-list-of-data" type="List-Of-Data"/>
        <xsd:element name="access-response-specification" type="List-Of-Access-Response-
Specification"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Access-Response">
      <xsd:sequence>
         <xsd:element name="long-invoke-id-and-priority" type="Long-Invoke-Id-And-Priority"/>
         <xsd:element name="date-time" type="xsd:hexBinary"/>
         <xsd:element name="access-response-body" type="Access-Response-Body"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="General-Glo-Ciphering">
      <xsd:sequence>
         <xsd:element name="system-title" type="xsd:hexBinary"/>
         <xsd:element name="ciphered-content" type="xsd:hexBinary"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="General-Ded-Ciphering">
      <xsd:sequence>
         <xsd:element name="system-title" type="xsd:hexBinary"/>
         <xsd:element name="ciphered-content" type="xsd:hexBinary"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Identified-Key">
      <xsd:sequence>
         <xsd:element name="key-id" type="Key-Id"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Wrapped-Key">
      <xsd:sequence>
         <xsd:element name="kek-id" type="Kek-Id"/>
         <xsd:element name="key-ciphered-data" type="xsd:hexBinary"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Agreed-Key">
      <xsd:sequence>
         <xsd:element name="key-parameters" type="xsd:hexBinary"/>
         <xsd:element name="key-ciphered-data" type="xsd:hexBinary"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="Key-Info">
      <xsd:choice>
         <xsd:element name="identified-key" type="Identified-Key"/>
         <xsd:element name="wrapped-key" type="Wrapped-Key"/>
         <xsd:element name="agreed-key" type="Agreed-Key"/>
      </xsd:choice>
   </xsd:complexType>

   <xsd:complexType name="General-Ciphering">
      <xsd:sequence>
         <xsd:element name="transaction-id" type="xsd:hexBinary"/>
         <xsd:element name="originator-system-title" type="xsd:hexBinary"/>
         <xsd:element name="recipient-system-title" type="xsd:hexBinary"/>
         <xsd:element name="date-time" type="xsd:hexBinary"/>
         <xsd:element name="other-information" type="xsd:hexBinary"/>
         <xsd:element name="key-info" minOccurs="0" type="Key-Info"/>
         <xsd:element name="ciphered-content" type="xsd:hexBinary"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:complexType name="General-Signing">
      <xsd:sequence>
         <xsd:element name="transaction-id" type="xsd:hexBinary"/>
         <xsd:element name="originator-system-title" type="xsd:hexBinary"/>
         <xsd:element name="recipient-system-title" type="xsd:hexBinary"/>
         <xsd:element name="date-time" type="xsd:hexBinary"/>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
            <xsd:element name="other-information" type="xsd:hexBinary"/>
            <xsd:element name="content" type="xsd:hexBinary"/>
            <xsd:element name="signature" type="xsd:hexBinary"/>
         </xsd:sequence>
      </xsd:complexType>

      <xsd:complexType name="General-Block-Transfer">
         <xsd:sequence>
            <xsd:element name="block-control" type="Block-Control"/>
            <xsd:element name="block-number" type="Unsigned16"/>
            <xsd:element name="block-number-ack" type="Unsigned16"/>
            <xsd:element name="block-data" type="xsd:hexBinary"/>
         </xsd:sequence>
      </xsd:complexType>

</xsd:schema>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# Annex A
(normative)

# Using the DLMS®/COSEM application layer in various communications profiles

## A.1 General

The COSEM interface model for energy metering equipment, specified in IEC 62056-6-2:2021 has been designed for use with a variety of communication profiles for exchanging data over various communication media. In each such profile, the application layer is the COSEM AL, providing the xDLMS services to access attributes and methods of COSEM objects. For each communication profile, the following elements shall be specified:

- the targeted communication environments;
- the structure of the profile (the set of protocol layers);
- the identification / addressing scheme;
- mapping of the DLMS®/COSEM AL services to the service set provided and used by the supporting layer;
- communication profile specific parameters of the DLMS®/COSEM AL services;
- other specific considerations / constraints for using certain services within a given profile.

## A.2 Targeted communication environments

This part identifies the communication environments, for which the given communication profile is specified.

## A.3 The structure of the profile

This part specifies the protocol layers included in the given profile.

## A.4 Identification and addressing schemes

This part describes the identification and addressing schemes specific for the profile.

As described in IEC 62056-6-2:2021, 4.1.7, metering equipment are modelled in COSEM as physical devices, containing one or more logical devices. In the COSEM client/server type model, data exchange takes place within application associations, between a COSEM client AP and a COSEM Logical Device, playing the role of a server AP.

To be able to establish the required AA and then to exchange data with the help of the supporting lower layer protocols, the client- and server APs shall be identified and addressed, according to the rules of a communication profile. At least the following elements need to be identified / addressed:

- physical devices hosting clients and servers;
- client- and server APs;

The client- and server APs also identify the AAs.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## A.5 Supporting layer services and service mapping

This part specifies the service mapping between the services requested by the COSEM AL and the services provided by its supporting layer.

In each communication profile, the COSEM AL provides the same set of services to the client- and server APs. However, the supporting protocol layer in the various profiles provides a different set of services to the service user AL.

The service mapping specifies how the AL is using the services of its supporting layer to provide ACSE and xDLMS services to its service user. For this purpose, MSCs are generally used, showing the sequence of the events following a service invocation by the COSEM AP.

## A.6 Communication profile specific parameters of the COSEM AL services

In COSEM, only the COSEM-OPEN service has communication profile specific parameters (the Protocol_Connection_Parameters). Their values and use are defined as part of the communication profile specification.

## A.7 Specific considerations / constraints using certain services within a given profile

The availability and the protocol of some of the services may depend on the communication profile. These elements are specified as part of the communication profile specification.

## A.8 The 3-layer, connection-oriented, HDLC based communication profile

This profile is specified in IEC 62056-7-6.

## A.9 The TCP-UDP/IP based communication profiles (COSEM_on_IP)

This profile is specified in IEC 62056-9-7.

## A.10 The wired and wireless M-Bus communication profiles

This profiles are specified in IEC 62056-7-3.

## A.11 The S-FSK PLC profile

This profile is specified in IEC 62056-8-3.

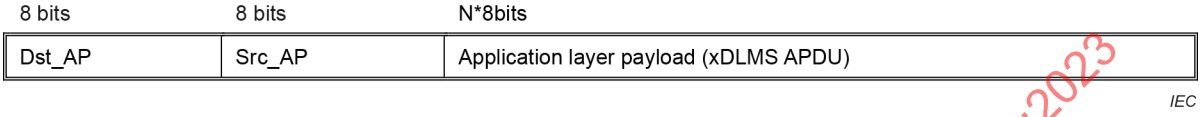
IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# Annex B
(normative)

## SMS short wrapper

This Annex specifies the transport of xDLMS APDUs in an SMS.

The payload of an SMS message is the xDLMS APDU prepended with the identifier of the Destination_AP and the Source_AP as shown in Figure B.1:




- Dst_AP = Destination AP identifies the destination Application Process;
- Src_AP = Source AP identifies the source Application Process.

**Figure B.1 – Short wrapper**

Table B.1 specifies the identifiers of reserved Application Processes:

**Table B.1 – Reserved Application Processes**

| **Client side reserved SAPs** | |
|---|---|
| No-station | 0x00 |
| Client Management Process | 0x01 |
| Public Client | 0x10 |
| *Open for client AP assignment* | 0x02 …0x0F |
| | 0x11 and up |
| **Server side reserved SAPs** | |
| No-station | 0x00 |
| Management Logical Device | 0x01 |
| Reserved | 0x02..0xF |
| *Open for server SAP assignment* | 0x10…0x7E |
| All-station (Broadcast) | 0x7F |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# Annex C
(normative)

# Gateway protocol

## C.1 General

This Annex specifies a method for exchanging data between DLMS®/COSEM clients and servers via a gateway, when this gateway is connected to a Wide Area Network (WAN) or to a Neighbourhood Network (NN) on the one hand, and to a Local Network (LAN) on the other hand, with DLMS®/COSEM servers connected to this LAN.

The gateway acts bidirectional, i.e. it is also possible for a server in the LAN to send messages to a client in the WAN/NN using the gateway (push application).

The gateway function itself may be implemented in a DLMS®/COSEM meter or in a stand-alone device.

The DLMS®/COSEM specification for meter data exchange is based on the client/server model, where the head end system (HES) acts as a client requesting services, and the end devices (e.g. meters) act as servers providing the services requested. In many cases, the client can reach each meter directly, using unicast, multicast or broadcast messages.

There are cases however, when it is practical to connect several end devices to a LAN, and reach those devices via a gateway. The protocol stack used on the LAN may be the same as the one used on the WAN/NN or it may be different.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

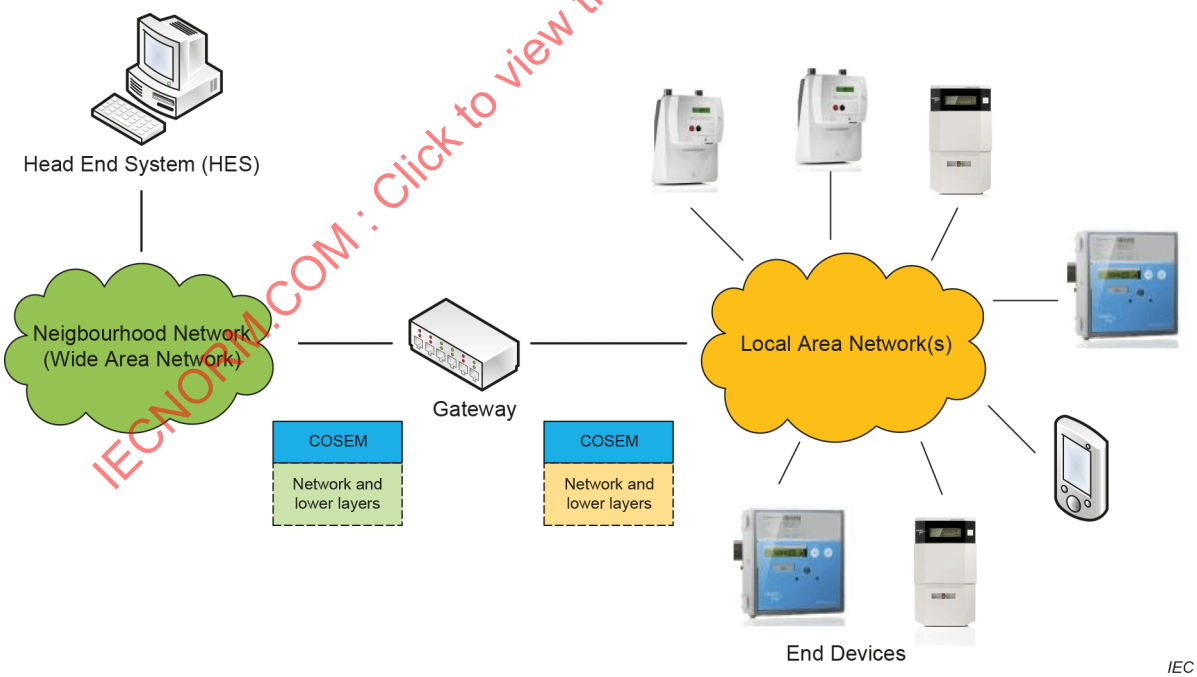



**Figure C.1 – General architecture with gateway**

The DLMS®/COSEM client (HES) reaches the gateway via a WAN or via NNAP; see Figure C.1. The gateway itself may be a stand-alone device or a DLMS®/COSEM meter capable of acting as a gateway. If configured accordingly, it passes the COSEM APDUs transparently between the HES or NNAP and the COSEM servers (end devices).

The gateway may act bidirectional, i.e. it is also possible for a COSEM server (end device) in the LAN to send COSEM APDUs to the COSEM client (HES) in the WAN/NN using the gateway (push application).

## C.2 The gateway protocol

The top layer of any DLMS®/COSEM communication profile is the DLMS®/COSEM application layer.

In order to leave both, the suite of lower layers and the DLMS®/COSEM AL unaffected, the task of routing each message from the client to the end device on the LAN is solved by pre-fixing the COSEM APDUs with a few bytes specifying the network to be used and the address of the device on the LAN and vice versa.

The gateway extracts the payload, a COSEM APDU – together with the application addresses – from the WAN/NN protocol and puts it as a payload to the LAN protocol, and the other way round.

The structure of the prefix with four fields is shown in Figure C.2.

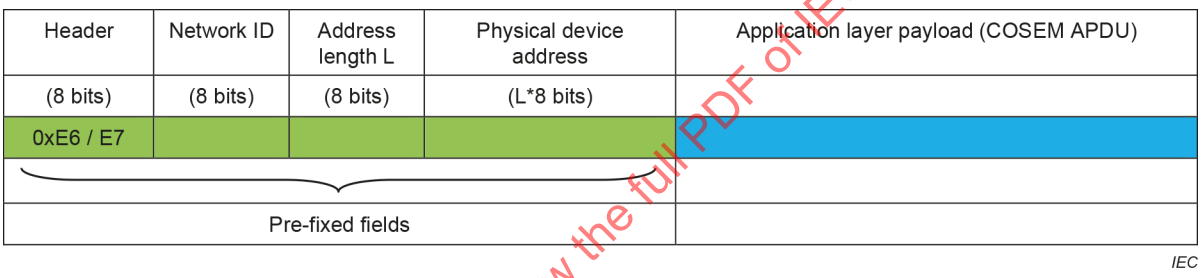



**Figure C.2 – The fields used for pre-fixing the COSEM APDUs**

- the value of the first byte (header) is either 0xE6 or 0xE7. It indicates that the following bytes don't contain a plain COSEM APDU but contain a COSEM APDU with a prefix:
  - 0xE6 indicates a request message from a DLMS®/COSEM client to a DLMS®/COSEM server or a request message from a DLMS®/COSEM server to a DLMS®/COSEM client (data notification);
  - 0xE7 indicates a response from the DLMS®/COSEM server to the DLMS®/COSEM client;
- the second byte carries an identifier of the destination network where the messages are transferred to. This allows accessing several networks using the same or different communication protocols through the same gateway. The network ID is not linked to the communication protocol and can be set to any value. If only one network exists, 0x00 shall be used.
- the third byte defines the length of the physical device address given in the next L bytes. It depends on the communication protocol used.
- bytes 4 to 4+(L-1) carry the physical device address of the end device or the HES as requested by the communication protocol.

When a telegram with pre-fixed fields reaches a device, which is not a gateway or which doesn't support pre-fixed fields, it shall be simply discarded.

When the client exchanges data directly with the master meter, the pre-fixed fields are not present.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## C.3 HES in the WAN/NN acting as Initiator (Pull operation)

In the sequence diagram shown in Figure C.3 the traditional pull data exchange between the DLMS®/COSEM client and server via a gateway is further elaborated:

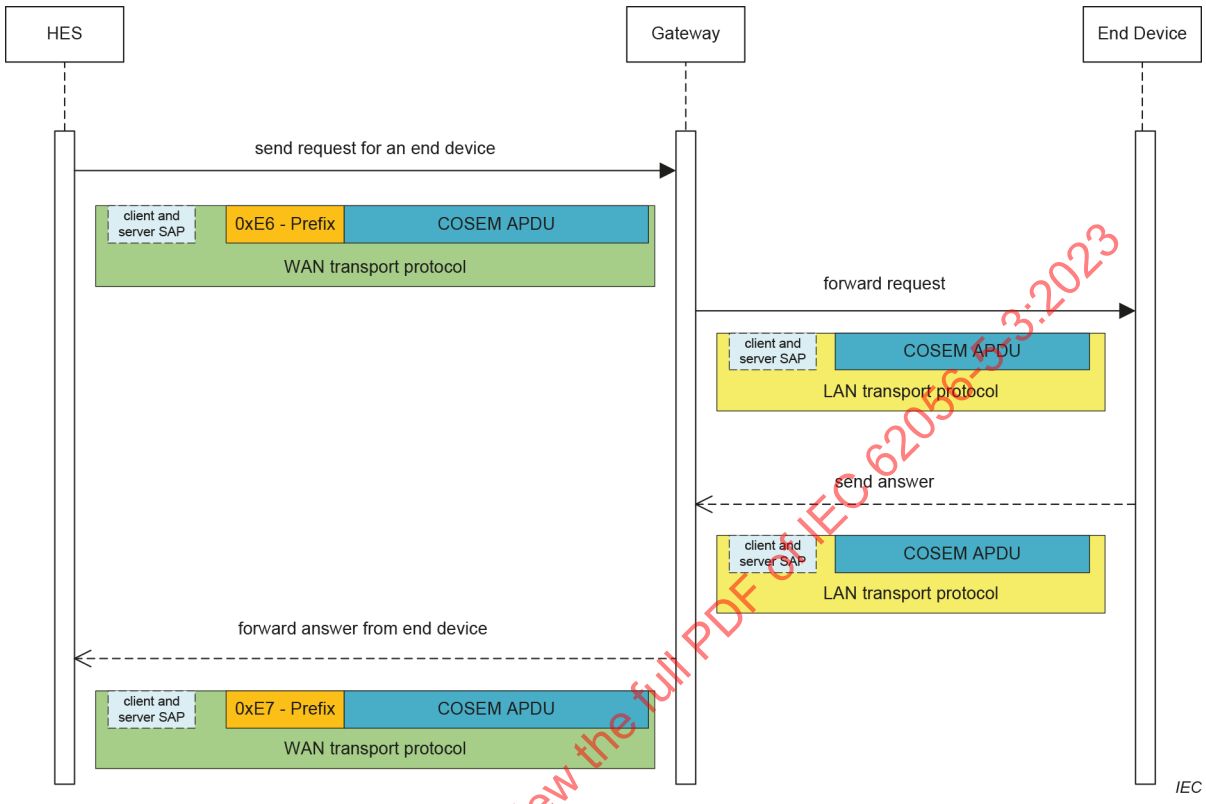



**Figure C.3 – Pull message sequence chart**

Prerequisite: The client in the WAN/NN has to know the network ID, the protocol and the physical device address of the server it wants to reach in the LAN.

The DLMS®/COSEM client (HES) sends every request, carried by a COSEM APDU, prefixed with four fields as shown in Figure C.2 using the protocol layer supporting the DLMS®/COSEM AL on the WAN/NN.

The gateway forwards each COSEM APDU carrying a .request service primitive to the appropriate network using the network ID and the physical device address contained in the pre-fixed fields. The client and server SAPs are extracted from the protocol layer supporting the DLMS®/COSEM AL on the WAN/NN and inserted into the supporting layer of the DLMS®/COSEM AL on the LAN.

The APDUs carrying the requests do not have any prefix when they arrive in the end devices in the LAN. Every end device processes the request and provides the answer the same way as if it's connected directly to the client.

When the device responds to a request, it is done as if it's connected to the client directly: The APDU does not need to be pre-fixed.

When the gateway receives a COSEM APDU carrying a .response service primitive on the LAN, it extracts the client and server SAPs from the protocol layer supporting the DLMS®/COSEM AL on the LAN. Afterwards it inserts them into the supporting layer of the DLMS®/COSEM AL on the WAN/NN and sends the message with pre-fixed fields to the client using the WAN/NN protocol.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## C.4 End devices in the LAN acting as Initiators (Push operation)

### C.4.1 General

It is also possible for a server (end device) in the LAN to send messages to a client (HES) in the WAN / NN using the gateway without having received a request service before (push application). Depending on the capabilities of the gateway two scenarios are supported.

### C.4.2 End device with WAN/NN knowledge

Prerequisite: The server in the LAN has to know the network ID, the protocol and the address of the client it wants to reach in the WAN/NN.

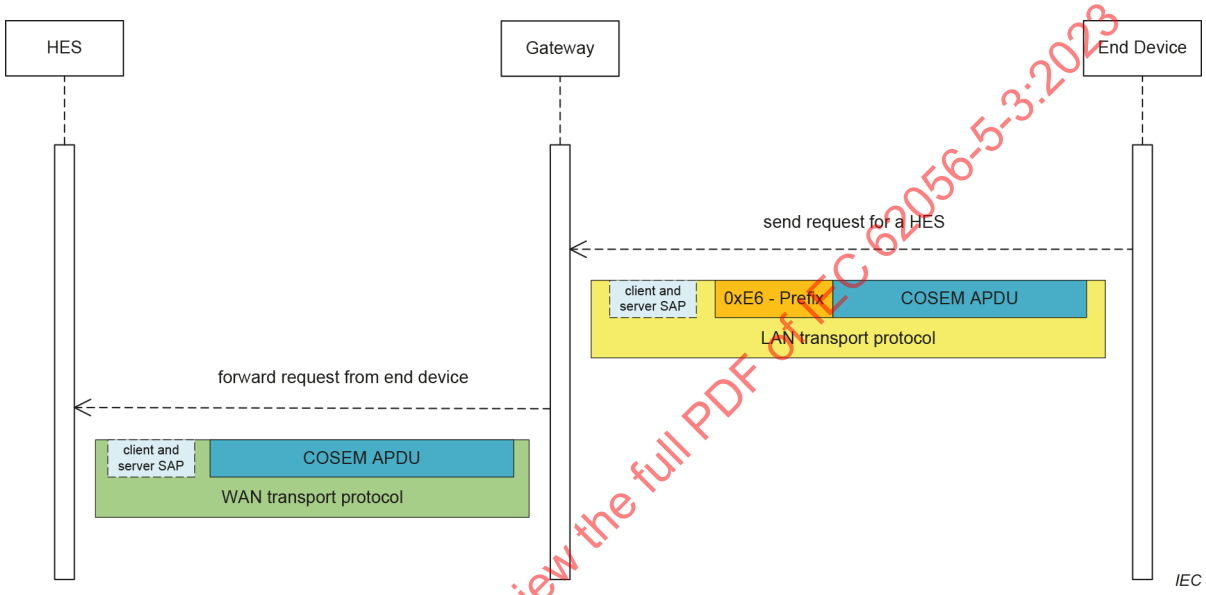



**Figure C.4 – Push message sequence chart**

The server (end device) sends every request (e.g. data notification request), carried by the COSEM APDU, pre-fixed with 4 fields as defined before to the gateway using the protocol layer supporting the DLMS®/COSEM AL on the LAN, as shown in Figure C.4.

The gateway forwards each COSEM APDU carrying a .request service primitive using the network ID, the protocol and the client address (e.g. WAN/NN MAC address) contained in the pre-fixed fields.

The client and server SAPs are extracted from the protocol layer supporting the DLMS®/COSEM AL on the LAN and inserted into the supporting layer of the DLMS®/COSEM AL on the WAN/NN.

### C.4.3 End devices without WAN/NN knowledge

If the end device has no knowledge on the WAN/NN network or if it has no knowledge if it is connected to a gateway at all, it can send standard (not pre-fixed) data notification requests to the gateway. It is then the duty of the gateway to further deal with such messages.

Since this does not require a protocol extension, this use case is not described any further.

## C.5 Security

The DLMS®/COSEM AL security mechanisms ensure end-to-end security through the gateway.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# Annex D
(informative)

# AARQ and AARE encoding examples

## D.1 General

This annex contains examples of encoding of the AARQ and AARE APDUs, in cases of using various levels of authentication and in cases of success and failure.

The AARQ, AARE, RLRQ and RLRE APDUs – see 7.2 – shall be encoded in BER (ISO/IEC 8825-1). The user-information field of the AARQ and AARE APDUs contains the xDLMS InitiateRequest / InitiateResponse or ConfirmedServiceError APDUs respectively, encoded in A-XDR as OCTET STRING.

## D.2 Encoding of the xDLMS InitiateRequest / InitiateResponse APDU

The xDLMS InitiateRequest / InitiateResponse APDUs are specified as follows:

```
InitiateRequest::= SEQUENCE
{
--  shall not be encoded in DLMS without ciphering
    dedicated-key                       OCTET STRING OPTIONAL,
    response-allowed                    BOOLEAN DEFAULT TRUE,
    proposed-quality-of-service         IMPLICIT Integer8 OPTIONAL,
    proposed-dlms-version-number        Unsigned8,
    proposed-conformance                Conformance,
    client-max-receive-pdu-size         Unsigned16
}

InitiateResponse::= SEQUENCE
{
    negotiated-quality-of-service       IMPLICIT Integer8 OPTIONAL,
    negotiated-dlms-version-number      Unsigned8,
    negotiated-conformance              Conformance,
    server-max-receive-pdu-size         Unsigned16,
    vaa-name                            ObjectName
}
```

The xDLMS InitiateRequest and InitiateResponse APDUs are encoded in A-XDR and they are inserted in the user-information field of the AARQ / AARE APDU respectively.

In the examples below, the following values are used:

- dedicated key: not present; no ciphering is used;
- response-allowed: TRUE (default value);
- proposed-quality-of-service and negotiated-quality-of-service: not present (not used in DLMS®/COSEM);
- proposed-conformance and negotiated-conformance: see below;
- proposed-dlms-version-number and negotiated-dlms-version-number = 6;
- client-max-receive-pdu-size: $1200_D$ = 0x04B0;
- server-max-receive-pdu-size: $500_D$ = 0x01F4;
- vaa-name in the case of LN referencing: the dummy value 0x0007;
- vaa-name in the case of SN referencing: the base_name of the current "Association SN" object, 0xFA00.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- the proposed-conformance and the negotiated-conformance elements carry the proposed conformance block and the negotiated conformance block respectively. The values of these examples, for LN referencing and SN referencing respectively, are shown in Table D.1.

**Table D.1 – Conformance block**

| Conformance::= **[APPLICATION 31] IMPLICIT BIT STRING** (SIZE(24)) | | **LN referencing** | | **SN referencing** | |
|---|---|---|---|---|---|
| -- the bit is set when the corresponding service or functionality is available | **Used with** | **Proposed/ Negotiated** | | **Proposed / Negotiated** | |
| reserved-zero (0), | | 0 | 0 | 0 | 0 |
| reserved-one (1), | | 0 | 0 | 0 | 0 |
| reserved-two (2), | | 0 | 0 | 0 | 0 |
| read (3), | SN | 0 | 0 | 1 | 1 |
| write (4), | SN | 0 | 0 | 1 | 1 |
| unconfirmed-write (5), | SN | 0 | 0 | 1 | 1 |
| reserved-six (6), | | 0 | 0 | 0 | 0 |
| reserved-seven (7), | | 0 | 0 | 0 | 0 |
| attribute0-supported-with-set (8), | LN | 0 | 0 | 0 | 0 |
| priority-mgmt-supported (9), | LN | 1 | 1 | 0 | 0 |
| attribute0-supported-with-get (10), | LN | 1 | 0 | 0 | 0 |
| block-transfer-with-get-or-read (11), | LN | 1 | 1 | 0 | 0 |
| block-transfer-with-set-or-write (12), | LN | 1 | 0 | 0 | 0 |
| block-transfer-with-action (13), | LN | 1 | 0 | 0 | 0 |
| multiple-references (14), | LN/SN | 1 | 0 | 1 | 1 |
| information-report (15), | SN | 0 | 0 | 1 | 1 |
| reserved-sixteen (16), | | 0 | 0 | 0 | 0 |
| reserved-seventeen (17), | | 0 | 0 | 0 | 0 |
| parameterized-access (18), | SN | 0 | 0 | 1 | 1 |
| get (19), | LN | 1 | 1 | 0 | 0 |
| set (20), | LN | 1 | 1 | 0 | 0 |
| selective-access (21), | LN | 1 | 1 | 0 | 0 |
| event-notification (22), | LN | 1 | 1 | 0 | 0 |
| action (23) | LN | 1 | 1 | 0 | 0 |
| Value of the bit string | | 00 7E 1F | 00 50 1F | 1C 03 20 | 1C 03 20 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

With these parameters, the A-XDR encoding of the xDLMS InitiateRequest APDU is as shown in Table D.2.

**Table D.2 – A-XDR encoding of the xDLMS InitiateRequest APDU**

| ***-- A-XDR encoding of the xDLMS InitiateRequest APDU*** | **LN referencing** | **SN referencing** |
|---|---|---|
| *// encoding of the tag of the DLMS APDU CHOICE (InitiateRequest)* | 01 | 01 |
| *-- encoding of the dedicated-key component (**OCTET STRING OPTIONAL**)* | | |
| *// usage flag(**FALSE**, not present)* | 00 | 00 |
| *-- encoding of the response-allowed component (**BOOLEAN DEFAULT TRUE**)* | | |
| *// usage flag(**FALSE**, default value **TRUE** conveyed)* | 00 | 00 |
| *-- encoding of the proposed-quality-of-service component ([0] **IMPLICIT** Integer8 **OPTIONAL**)* | | |
| *// usage flag(**FALSE**, not present)* | 00 | 00 |
| *-- encoding of the proposed-dlms-version-number component (Unsigned8)* | | |
| *// value= 6, the encoding of an Unsigned8 is its value* | 06 | 06 |
| *-- encoding of the proposed-conformance component (Conformance, [APPLICATION 31] **IMPLICIT BIT STRING** (SIZE(24))* [1] | | |
| *// encoding of the [APPLICATION 31] tag (ASN.1 explicit tag)* [2] | 5F 1F | 5F 1F |
| *// encoding of the length of the 'contents' field in octet (4)* | 04 | 04 |
| *// encoding of the number of unused bits in the final octet of the BIT STRING (0)* | 00 | 00 |
| *// encoding of the fixed length BIT STRING value* | 00 7E 1F | 1C 03 20 |
| *-- encoding of the client-max-receive-pdu-size component (Unsigned16)* | | |
| *// value = 0x04B0, the encoding of an Unsigned16 is its value* | 04 B0 | 04 B0 |
| *-- resulting octet-string, to be inserted in the user-information field of the AARQ APDU* | 01 00 00 00 06 5F 1F 04 00 00 7E 1F 04 B0 | 01 00 00 00 06 5F 1F 04 00 1C 03 20 04 B0 |

[1] As specified in IEC 61334-6:2000, Annex C, Examples 1 and 2, the proposed-conformance element of the xDLMS InitiateRequest APDU and the negotiated-conformance element of the xDLMS InitiateResponse APDU are encoded in BER. That's why the length of the bit-string and the number of the unused bits are encoded.

[2] For encoding of identifier octets see ISO/IEC 8825-1:2008, 8.1.2. For compliance with existing implementations, encoding of the [Application 31] tag on one byte (5F) instead of two bytes (5F 1F) is accepted when the 3-layer, connection-oriented, HDLC-based profile is used.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The A-XDR encoding of the xDLMS InitiateResponse APDU is as shown in Table D.3

**Table D.3 – A-XDR encoding of the xDLMS InitiateResponse APDU**

| ***-- A-XDR encoding of the xDLMS InitiateResponse APDU*** | **LN referencing** | **SN referencing** |
|---|---|---|
| *// encoding of the tag of the DLMS APDU CHOICE (InitiateResponse)* | 08 | 08 |
| *-- encoding of the negotiated-quality-of-service component ([0] **IMPLICIT** Integer8 **OPTIONAL**)* | | |
| *// usage flag(**FALSE**, not present)* | 00 | 00 |
| *-- encoding of the negotiated-dlms-version-number component (Unsigned8)* | | |
| *// value = 6, the encoding of an Unsigned8 is its value* | 06 | 06 |
| *-- encoding of the negotiated-conformance component (Conformance, [APPLICATION 31] **IMPLICIT BIT STRING** (SIZE(24))* | | |
| *// encoding of the [APPLICATION 31] tag (ASN.1 explicit tag)* | 5F 1F | 5F 1F |
| *// encoding of the length of the 'contents' field in octet (4)* | 04 | 04 |
| *// encoding of the number of unused bits in the final octet of the BIT STRING (0)* | 00 | 00 |
| *// encoding of the fixed length BIT STRING value* | 00 50 1F | 1C 03 20 |
| *-- encoding of the server-max-receive-pdu-size component (Unsigned16)* | | |
| *// value = 0x01F4, the encoding of an Unsigned16 is its value* | 01 F4 | 01 F4 |
| *-- encoding of the VAA-Name component (ObjectName, Integer16)* | | |
| *// value=0x0007 for LN and 0xFA00 for SN referencing; the encoding of a value constrained Integer16 is its value* | 00 07 | FA 00 |
| *-- resulting octet-string, to be inserted in the user-information field of the AARE APDU* | 08 00 06 5F 1F 04 00 00 50 1F 01 F4 00 07 | 08 00 06 5F 1F 04 00 1C 03 20 01 F4 FA 00 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## D.3 Specification of the AARQ and AARE APDUs

The AARQ and the AARE APDUs are specified in Clause 8 as follows:

```
AARQ-apdu::= [APPLICATION 0] IMPLICIT SEQUENCE
{
-- [APPLICATION 0] == [ 60H ] = [ 96 ]

   protocol-version                  [0] IMPLICIT    BIT STRING {version1 (0)}
                                                     DEFAULT {version1},
   application-context-name          [1]             Application-context-name,
   called-AP-title                   [2]             AP-title OPTIONAL,
   called-AE-qualifier               [3]             AE-qualifier OPTIONAL,
   called-AP-invocation-id           [4]             AP-invocation-identifier OPTIONAL,
   called-AE-invocation-id           [5]             AE-invocation-identifier OPTIONAL,
   calling-AP-title                  [6]             AP-title OPTIONAL,
   calling-AE-qualifier              [7]             AE-qualifier OPTIONAL,
   calling-AP-invocation-id          [8]             AP-invocation-identifier OPTIONAL,
   calling-AE-invocation-id          [9]             AE-invocation-identifier OPTIONAL,

-- The following field shall not be present if only the kernel is used.
   sender-acse-requirements          [10] IMPLICIT   ACSE-requirements OPTIONAL,

-- The following field shall only be present if the authentication functional unit is
-- selected.
   mechanism-name                    [11] IMPLICIT   Mechanism-name OPTIONAL,
```

```
-- The following field shall only be present if the authentication functional unit is
--selected.
   calling-authentication-value      [12] EXPLICIT   Authentication-value OPTIONAL,
    implementation-information       [29] IMPLICIT   Implementation-data OPTIONAL,
    user-information                 [30] EXPLICIT   Association-information OPTIONAL
}

-- The user-information field shall carry an InitiateRequest APDU encoded in A-XDR, and
-- then encoding the resulting OCTET STRING in BER.

AARE-apdu::= [APPLICATION 1] IMPLICIT SEQUENCE
{
-- [APPLICATION 1] == [ 61H ] = [ 97 ]

   protocol-version                  [0] IMPLICIT    BIT STRING {version1 (0)}
                                                     DEFAULT {version1},
   application-context-name          [1]             Application-context-name,
   result                            [2]             Association-result,
   result-source-diagnostic          [3]             Associate-source-diagnostic,
   responding-AP-title               [4]             AP-title OPTIONAL,
   responding-AE-qualifier           [5]             AE-qualifier OPTIONAL,
   responding-AP-invocation-id       [6]             AP-invocation-identifier OPTIONAL,
    responding-AE-invocation-id      [7]             AE-invocation-identifier OPTIONAL,

-- The following field shall not be present if only the kernel is used.
   responder-acse-requirements       [8] IMPLICIT    ACSE-requirements OPTIONAL,

-- The following field shall only be present if the authentication functional unit is
-- selected.
   mechanism-name                    [9] IMPLICIT    Mechanism-name OPTIONAL,

-- The following field shall only be present if the authentication functional unit is
-- selected.
   responding-authentication-value   [10] EXPLICIT   Authentication-value OPTIONAL,
   implementation-information        [29] IMPLICIT   Implementation-data OPTIONAL,
   user-information                  [30] EXPLICIT   Association-information OPTIONAL
}

-- The user-information field shall carry either an InitiateResponse (or, when the
-- proposed xDLMS context is not accepted by the server, a ConfirmedServiceError) APDU
-- encoded in A-XDR. The resulting OCTET STRING shall be encoded in BER.
```

## D.4 Data for the examples

In these examples:

- the protocol-version is the default ACSE version;
- the value of the application-context-name:
  - in the case of LN referencing, with no ciphering: 2, 16, 756, 5, 8, 1, 1;
  - in the case of SN referencing, with no ciphering: 2, 16, 756, 5, 8, 1, 2;
- the optional called-AP-title, called-AE-qualifier, called-AP-invocation-id, called-AE-invocation-id, calling-AP-title, calling-AE-qualifier, calling-AP-invocation-id, calling-AE-invocation-id fields of the AARQ, and the optional responding-AP-title, responding-AE-qualifier, responding-AP-invocation-id, responding-AE-invocation-id fields of the AARE are not present;
- the value of the mechanism-name:
  - in the case of low-level-security: 2, 16, 756, 5, 8, 2, 1;
  - in the case of high-level-security (5): 2, 16, 756, 5, 8, 2, 5;
- the calling-authentication-value:
  - in the case of low-level-security is 12345678 (encoded as 31 32 33 34 35 36 37 38);
  - in the case of high-level security, (challenge CtoS) is K56iVagY (encoded as 4B 35 36 69 56 61 67 59);
- the responding authentication-value (challenge StoC) is P6wRJ21F (encoded as 50 36 77 52 4A 32 31 46);

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- the optional implementation-information field in the AARQ and AARE APDUs is not present;
- the user-information field carries the xDLMS InitiateRequest / InitiateResponse APDUs as shown above.

The application-context-name and the (authentication) mechanism-name OBJECT IDENTIFIERS are encoded as follows:

- BER Encoding for OBJECT IDENTIFIER is a packed sequence of numbers representing the arc labels. Each number – except the first two, which are combined into one – is represented as a series of octets, with 7 bits being used from each octet and the most significant bit is set to 1 in all but the last octet. The fewest possible number of octets shall be used;
- in the case of the application context name LN referencing with no ciphering, the arc labels of the object identifier are (2, 16, 756, 5, 8, 1, 1);
  - the first octet of the encoding is the combination of the first two numbers into a single number, following the rule of 40*First+Second -> 40*2 + 16 = 96 = 0x60;
  - the third number of the Object Identifier (756) requires two octets: its hexadecimal value is 0x02F4, which is 00000010 11110100, but following the above rule, the MSB of the first octet shall be set to 1 and the MSB of the second (last) octet shall be set to 0, thus this bit shall be shifted into the LSB of the first octet. This gives binary 10000101 01110100, which is 0x8574;
  - each remaining numbers of the Object Identifier required to be encoded on one octet;
  - this results in the encoding 60 85 74 05 08 01 01.
- similarly, in the case of application context name SN referencing with no ciphering the BER encoding is 60 85 74 05 08 01 02;
- in the case of mechanism name low-level-security, the BER encoding is 60 85 74 05 08 02 01;
- in the case of mechanism name high-level-security (5), the BER encoding is 60 85 74 05 08 02 05.

## D.5 Encoding of the AARQ APDU

Here, six different cases are shown:

- LN referencing with no ciphering, no security, LLS and HLS;
- SN referencing with no ciphering, no security, LLS and HLS;

The encoding is shown in Table D.4. See also Table D.5.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table D.4 – BER encoding of the AARQ APDU**

| *-- BER encoding of the AARQ APDU* | LN referencing | | | SN referencing | | |
|---|---|---|---|---|---|---|
| | no sec. | LLS | HLS | no sec. | LLS | HLS |
| // encoding of the tag of the AARQ APDU *([APPLICATION 0], Application)* | 60 | | | 60 | | |
| // encoding of the length of the AARQ's content's field | 1D | 36 | 36 | 1D | 36 | 36 |
| *-- protocol-version field ([0], **IMPLICIT BIT STRING** { version1 (0) } **DEFAULT** { version1 }* | | | | | | |
| // no encoding, thus it is considered with its **DEFAULT** value | | | | | | |
| *-- encoding of the fields of the Kernel* | | | | | | |
| *-- application-context-name field ([1], Application-context-name, **OBJECT IDENTIFIER**)* | | | | | | |
| // encoding of the tag *([1], Context-specific)* | A1 | | | A1 | | |
| // encoding of the length of the tagged component's value field | 09 | | | 09 | | |
| // encoding of the choice for application-context-name *(**OBJECT IDENTIFIER**, Universal)* | 06 | | | 06 | | |
| // encoding of the length of the Object Identifier's value field | 07 | | | 07 | | |
| // encoding of the value of the Object Identifier | 60 85 74 05 08 01 01 | | | 60 85 74 05 08 01 02 | | |
| *-- encoding of the fields of the authentication functional unit* | | | | | | |
| *--sender-acse-requirements field ([10], ACSE-requirements, **BIT STRING** { authentication (0) } )* | | | | | | |
| // encoding of the tag of the acse-requirements field *([10], **IMPLICIT**, Context-specific)* | – | 8A | 8A | – | 8A | 8A |
| // encoding of the length of the tagged component's value field | – | 02 | 02 | – | 02 | 02 |
| // encoding of the number of unused bits in the last byte of the **BIT STRING** | – | 07 | 07 | – | 07 | 07 |
| // encoding of the authentication functional unit *(0)*<br>The number of bits coded may vary from client to client, but within the COSEM environment, only bit 0 set to 1 (indicating the requirement of the authentication functional unit) is to be respected. | – | 80 | 80 | – | 80 | 80 |
| *-- mechanism-name field ([11], **IMPLICIT** Mechanism-name **OBJECT IDENTIFIER**)* | | | | | | |
| // encoding of the tag *([11], **IMPLICIT**, Context-specific)* | – | 8B | 8B | – | 8B | 8B |
| // encoding of the length of the tagged component's value field | – | 07 | 07 | – | 07 | 07 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| *-- BER encoding of the AARQ APDU* | LN referencing | | | SN referencing | | |
|---|---|---|---|---|---|---|
| | **no sec.** | **LLS** | **HLS** | **no sec.** | **LLS** | **HLS** |
| // encoding of the value of the **OBJECT IDENTIFIER**:<br>- low-level-security-mechanism-name (1),<br>- high-level-security-mechanism-name (5) | – | 60 85 74 05 08 02 01 | 60 85 74 05 08 02 05 | – | 60 85 74 05 08 02 01 | 60 85 74 05 08 02 05 |
| *-- calling-authentication-value field ([12], Authentication-value **CHOICE**)* | | | | | | |
| // encoding of the tag *([12], **EXPLICIT**, Context-specific)* | – | AC | AC | – | AC | AC |
| // encoding of the length of the tagged component's value field | – | 0A | 0A | – | 0A | 0A |
| // encoding of the choice for Authentication-value *(charstring [0] **IMPLICIT GraphicString**)* | – | 80 | 80 | – | 80 | 80 |
| // encoding of the length of the Authentication-value's value field (8 octets) | – | 08 | 08 | – | 08 | 08 |
| // encoding of the calling-authentication-value:<br>- in the case of LLS, the value of the Password *"12345678"*<br>- in the case of HLS, the value of challenge CtoS *"K56iVagY"* | – | 31 32 33 34 35 36 37 38 | 4B 35 36 69 56 61 67 59 | – | 31 32 33 34 35 36 37 38 | 4B 35 36 69 56 61 67 59 |
| *-- encoding of the user-information field component (Association-information, **OCTET STRING**)* | | | | | | |
| // encoding of the tag *([30], Context-specific, Constructed)* | BE | BE | BE | BE | BE | BE |
| // encoding of the length of the tagged component's value field | 10 | 10 | 10 | 10 | 10 | 10 |
| // encoding of the choice for user-information *(**OCTET STRING**, Universal)* | 04 | 04 | 04 | 04 | 04 | 04 |
| // encoding of the length of the **OCTET STRING's** value field *(14 octets)* | 0E | 0E | 0E | 0E | 0E | 0E |
| // user-information: xDLMS InitiateRequest APDU | 01 00 00 00 06 5F 1F 04 00 00 7E 1F 04 B0 | | | 01 00 00 00 06 5F 1F 04 00 1C 03 20 04 B0 | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table D.5 – Complete AARQ APDU**

| | |
|---|---|
| LN referencing with no ciphering, lowest level security; | 60 1D A1 09 06 07 60 85 74 05 08 01 01 BE 10 04 0E 01 00 00 00 06 5F 1F 04 00 00 7E 1F 04 B0 |
| LN referencing with no ciphering, low level security; | 60 36 A1 09 06 07 60 85 74 05 08 01 01 8A 02 07 80 8B 07 60 85 74 05 08 02 01 AC 0A 80 08 31 32 33 34 35 36 37 38 BE 10 04 0E 01 00 00 00 06 5F 1F 04 00 00 7E 1F 04 B0 |
| LN referencing with no ciphering, high level security; | 60 36 A1 09 06 07 60 85 74 05 08 01 01 8A 02 07 80 8B 07 60 85 74 05 08 02 05 AC 0A 80 08 4B 35 36 69 56 61 67 59 BE 10 04 0E 01 00 00 00 06 5F 1F 04 00 00 7E 1F 04 B0 |
| SN referencing with no ciphering, lowest level security; | 60 1D A1 09 06 07 60 85 74 05 08 01 02 BE 10 04 0E 01 00 00 00 06 5F 1F 04 00 1C 03 20 04 B0 |
| SN referencing with no ciphering, low level security; | 60 36 A1 09 06 07 60 85 74 05 08 01 02 8A 02 07 80 8B 07 60 85 74 05 08 02 01 AC 0A 80 08 31 32 33 34 35 36 37 38 BE 10 04 0E 01 00 00 00 06 5F 1F 04 00 1C 03 20 04 B0 |
| SN referencing with no ciphering, high level security | 60 36 A1 09 06 07 60 85 74 05 08 01 02 8A 02 07 80 8B 07 60 85 74 05 08 02 05 AC 0A 80 08 4B 35 36 69 56 61 67 59 BE 10 04 0E 01 00 00 00 06 5F 1F 04 00 1C 03 20 04 B0 |

## D.6 Encoding of the AARE APDU

Here, six different cases are shown:

- LN referencing with no ciphering, no security or LLS, successful establishment of the AA;
- LN referencing with no ciphering, no security or LLS, failure because the proposed application-context-name does not fit the application-context-name supported by the server (failure case 1):
  - when the meter uses LN referencing, SN referencing is proposed;
  - when the meter uses SN referencing, LN referencing is proposed;
- LN referencing with no ciphering, no security or LLS, failure because the proposed-dlms-version-number is too low; (failure case 2)
- LN referencing with no ciphering, HLS, successful establishment of the AA;
- SN referencing with no ciphering, no security or LLS, successful establishment of the AA;
- SN referencing with no ciphering, HLS, successful establishment of the AA.

The encoding is shown in Table D.6. See also Table D.7.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table D.6 – BER encoding of the AARE APDU**

| *-- BER encoding of the AARE APDU* | LN referencing | | | | SN referencing | |
|---|---|---|---|---|---|---|
| | **No sec./LLS success** | **No sec./LLS failure 1** | **No sec./LLS failure 2** | **HLS success** | **No sec./LLS success** | **HLS success** |
| // encoding of the tag for the AARE APDU *([APPLICATION 1], Application)* | 61 | | | | 61 | |
| // encoding of the length of the AARE's contents field | 29 | 29 | 1F | 42 | 29 | 42 |
| *-- protocol-version field ([0], **IMPLICIT BIT STRING** { version1 (0) } **DEFAULT** { version1 }* | | | | | | |
| // no encoding, thus it is considered with its **DEFAULT** value | | | | | | |
| *-- application-context-name field ([1], Application-context-name, **OBJECT IDENTIFIER**)* | | | | | | |
| // encoding of the tag *([1], Context-specific)* | A1 | | | | A1 | |
| // encoding of the length of the tagged component's value field | 09 | | | | 09 | |
| // encoding of the choice for application-context-name ***(OBJECT IDENTIFIER**, Universal)* | 06 | | | | 06 | |
| // encoding of the length of the Object Identifier's value field | 07 | | | | 07 | |
| // encoding of the value of the Object Identifier:<br>NOTE when the proposed application-context does not fit the application-context supported by the server, the server responds either with the application-context name proposed or the application-context-name supported. | 60 85 74 05 08 01 01 | 60 85 74 05 08 01 01 | 60 85 74 05 08 01 01<br>or<br>60 85 74 05 08 01 02 | 60 85 74 05 08 01 01 | 60 85 74 05 08 01 02 | 60 85 74 05 08 01 02 |
| *-- result field ([2], Association-result, **INTEGER**)* | | | | | | |
| // encoding of the tag *([2], Context-specific)* | A2 | | | | A2 | |
| // encoding of the length of the tagged component's value field | 03 | | | | 03 | |
| // encoding of the choice for the result *(**INTEGER**, Universal)* | 02 | | | | 02 | |
| // encoding of the length of the result's value field | 01 | | | | 01 | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| -- ***BER encoding of the AARE APDU*** | **LN referencing** | | | | **SN referencing** | |
|---|---|---|---|---|---|---|
| | **No sec./LLS success** | **No sec./LLS failure 1** | **No sec./LLS failure 2** | **HLS success** | **No sec./LLS success** | **HLS success** |
| // encoding of the value of the Result:<br>// success: 0, accepted<br>// failure case 1 and 2: 1, rejected-permanent | 00 | 01 | 01 | 00 | 00 | 00 |
| *-- result-source-diagnostic field ([3], Associate-source-diagnostic,* ***CHOICE****)* | | | | | | |
| // encoding of the tag *([3], Context-specific)* | A3 | | | | A3 | |
| // encoding of the length of the tagged component's value field | 05 | | | | 05 | |
| // encoding of the tag for the acse-service-user CHOICE *(1)* | A1 | | | | A1 | |
| // encoding of the length of the tagged component's value field | 03 | | | | 03 | |
| // encoding of the choice for associate-source-diagnostics *(INTEGER, Universal)* | 02 | | | | 02 | |
| // encoding of the length of the value field | 01 | | | | 01 | |
| // encoding of the value:<br>- success, no security and LLS: 0, no diagnostics provided;<br>- failure 1: 2, application-context-name not supported;<br>- failure 2: 1, no-reason-given.<br>- success, HLS security (5): 14, authentication required; | 00 | 02 | 01 | 0E | 00 | 0E |
| ***-- encoding of the fields of the authentication functional unit*** | | | | | | |
| *-- responder-acse-requirements field ([8],* ***IMPLICIT****, ACSE-requirements,* ***BIT STRING*** *{ authentication (0) } )* | | | | | | |
| // encoding of the tag of the acse-requirements field *([8],* ***IMPLICIT****, Context-specific)* | – | | | 88 | – | 88 |
| // encoding of the length of the tagged component's value field. | – | | | 02 | – | 02 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| *-- BER encoding of the AARE APDU* | LN referencing | | | | SN referencing | |
|---|---|---|---|---|---|---|
| | **No sec./LLS success** | **No sec./LLS failure 1** | **No sec./LLS failure 2** | **HLS success** | **No sec./LLS success** | **HLS success** |
| // encoding of the number of unused bits in the last byte of the **BIT STRING** | – | | | 07 | – | 07 |
| // encoding of the authentication functional unit *(0)* | – | | | 80 | – | 80 |
| *-- mechanism-name field ([9], **IMPLICIT**, Mechanism-name **OBJECT IDENTIFIER**)* | | | | | | |
| // encoding of the tag *([9], **IMPLICIT**, Context-specific)* | – | | | 89 | – | 89 |
| // encoding of the length of the tagged component's value field | – | | | 07 | – | 07 |
| // encoding of the value of the object identifier: high-level-security-mechanism-name (5) | – | | | 60 85 74 05 08 02 05 | – | 60 85 74 05 08 02 05 |
| *-- responding-authentication-value field ([10], **EXPLICIT**, Authentication-value CHOICE)* | | | | | | |
| // encoding of the tag *([10], Context-specific)* | – | – | - | AA | – | AA |
| // encoding of the length of the tagged component's value field | – | | - | 0A | – | 0A |
| // encoding of the choice for Authentication-value *(charstring [0] **IMPLICIT** GraphicString)* | – | – | – | 80 | – | 80 |
| // encoding of the length of the Authentication-information's value *field (8 octets)* | – | – | – | 08 | – | 08 |
| // encoding of the value of the challenge StoC "P6wRJ21F" | – | – | - | 50 36 77 52 4A 32 31 46 | – | 50 36 77 52 4A 32 31 46 |
| *-- encoding of the user-information field component (Association-information, **OCTET STRING**)* | | | | | | |
| // encoding of the tag for the user-information field component *([30], Context-specific, Constructed)* | BE | BE | BE | BE | BE | BE |
| // encoding of the length of the tagged component's value field | 10 | 10 | 06 | 10 | 10 | 10 |
| // encoding of the choice for user-information *(OCTET STRING, Universal)* | 04 | 04 | 04 | 04 | 04 | 04 |
| // encoding of the length of the OCTET STRING's value field | 0E | 0E | 04 | 0E | 0E | 0E |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| -- *BER encoding of the AARE APDU* | **LN referencing** | | | | **SN referencing** | |
|---|---|---|---|---|---|---|
| | **No sec./LLS success** | **No sec./LLS failure 1** | **No sec./LLS failure 2** | **HLS success** | **No sec./LLS success** | **HLS success** |
| // failure case 1: xDLMS-InitiateResponse;<br>// failure case 2: ConfirmedServiceError ([14]), InitiateError [1], ServiceError, initiate [6], dlms-version-too-low (1)) | 08 00 06 5F 1F 04 00 00 50 1F 01 F4 00 07 | 08 00 06 5F 1F 04 00 00 50 1F 01 F4 00 07 | 0E 01 06 01 | 08 00 06 5F 1F 04 00 00 50 1F 01 F4 00 07 | 08 00 06 5F 1F 04 00 1C 03 20 01 F4 FA 00 | 08 00 06 5F 1F 04 00 1C 03 20 01 F4 FA 00 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table D.7 – The complete AARE APDU**

| | |
|---|---|
| LN referencing with no ciphering, no security or LLS, successful establishment of the AA | 61 29 A1 09 06 07 60 85 74 05 08 01 01 A2 03 02 01 00 A3 05 A1 03 02 01 00 BE 10 04 0E 08 00 06 5F 1F 04 00 00 50 1F 01 F4 00 07 |
| LN referencing with no ciphering, no security or LLS, failure because the proposed application-context-name does not fit the application-context-name supported by the server (failure case 1): | 61 29 A1 09 06 07 60 85 74 05 08 01 01 A2 03 02 01 01 A3 05 A1 03 02 01 02 BE 10 04 0E 08 00 06 5F 1F 04 00 00 50 1F 01 F4 00 07 or<br>61 29 A1 09 06 07 60 85 74 05 08 01 02 A2 03 02 01 01 A3 05 A1 03 02 01 02 BE 10 04 0E 08 00 06 5F 1F 04 00 00 50 1F 01 F4 00 07 |
| LN referencing with no ciphering, no security or LLS, failure because the proposed-dlms-version-number is too low; (failure case 2) | 61 1F A1 09 06 07 60 85 74 05 08 01 01 A2 03 02 01 01 A3 05 A1 03 02 01 01 BE 06 04 04 0E 01 06 01 |
| LN referencing with no ciphering, high level security; | 61 42 A1 09 06 07 60 85 74 05 08 01 01 A2 03 02 01 00 A3 05 A1 03 02 01 0E 88 02 07 80 89 07 60 85 74 05 08 02 05 AA 0A 80 08 50 36 77 52 4A 32 31 46 BE 10 04 0E 08 00 06 5F 1F 04 00 00 50 1F 01 F4 00 07 |
| SN referencing with no ciphering, lowest level security; | 61 29 A1 09 06 07 60 85 74 05 08 01 02 A2 03 02 01 00 A3 05 A1 03 02 01 00 BE 10 04 0E 08 00 06 5F 1F 04 00 1C 03 20 01 F4 FA 00 |
| SN referencing with no ciphering, high level security | 61 42 A1 09 06 07 60 85 74 05 08 01 02 A2 03 02 01 00 A3 05 A1 03 02 01 0E 88 02 07 80 89 07 60 85 74 05 08 02 05 AA 0A 80 08 50 36 77 52 4A 32 31 46 BE 10 04 0E 08 00 06 5F 1F 04 00 1C 03 20 01 F4 FA 00 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# Annex E
(informative)

# Encoding examples: AARQ and AARE APDUs using a ciphered application context

## E.1 A-XDR encoding of the xDLMS InitiateRequest APDU, carrying a dedicated key

NOTE The System Title is the same in each example. In reality, the System Title in the request and in the response APDUs are different, as they are originated by different systems.

In this example:

- the value of the dedicated key is 00112233445566778899AABBCCDDEEFF;
- the value of the Conformance block is 007E1F;
- the value of the client-max-receive-pdu-size is 1 200 bytes (0x04B0).

The A-XDR encoding of the xDLMS InitiateRequest APDU carrying a dedicated key is shown in Table E.1.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table E.1 – A-XDR encoding of the xDLMS InitiateRequest APDU**

| | |
|---|---|
| // encoding of the tag of the DLMS APDU CHOICE *(InitiateRequest)* | 01 |
| -- *encoding of the dedicated-key component (**OCTET STRING OPTIONAL**)* | |
| // usage flag *(**TRUE**, present)* | 01 |
| // length of the **OCTET STRING** | 10 |
| // contents of the **OCTET STRING** | 00112233445566778899AABBCCDDEEFF |
| -- *encoding of the response-allowed component (**BOOLEAN DEFAULT TRUE**)* | |
| // usage flag *(**FALSE**, default value **TRUE** conveyed)* | 00 |
| -- encoding of the proposed-quality-of-service component *([0] **IMPLICIT** Integer8 **OPTIONAL**)* | |
| // usage flag *(**FALSE**, not present)* | 00 |
| -- *encoding of the proposed-dlms-version-number component (Unsigned8)* | |
| // value = 6; the A-XDR encoding of an Unsigned8 is its value | 06 |
| -- *encoding of the proposed-conformance component (Conformance, [APPLICATION 31] **IMPLICIT BIT STRING** (SIZE(24))* [1] | |
| // encoding of the [APPLICATION 31] tag *(ASN.1 explicit tag)* [2] | 5F1F |
| // encoding of the length of the 'contents' field in octet *(4)* | 04 |
| // encoding of the number of unused bits in the final octet of the BIT STRING *(0)* | 00 |
| // encoding of the fixed length BIT STRING value | 007E1F |
| -- *encoding of the client-max-receive-pdu-size component (Unsigned16)* | |
| // value = 0x04B0, the encoding of an Unsigned16 is its value | 04B0 |
| -- *resulting octet-string* | 0101100011223344556677889AABBCCDDEEFF0000065F1F0400007E1F04B0 |

[1] As specified in IEC 61334-6:2000, Annex C, Examples 1 and 2, the proposed-conformance element of the xDLMS InitiateRequest APDU and the negotiated-conformance element of the xDLMS InitiateResponse APDU are encoded in BER. That's why the length of the bit-string and the number of the unused bits are encoded.

[2] For encoding of identifier octets see ISO/IEC 8825-1:2008, 8.1.2. For compliance with existing implementations, encoding of the [Application 31] tag on one byte (5F) instead of two bytes (5F 1F) is accepted when the 3-layer, connection-oriented, HDLC-based profile is used.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## E.2 Authenticated encryption of the xDLMS InitiateRequest APDU

Table E.2 shows the encoding of an xDLMS InitiateRequest APDU which is also authenticated and encrypted.

**Table E.2 – Authenticated encryption of the xDLMS InitiateRequest APDU**

| | *X* | Contents | LEN(X) bytes | len(X) bits |
|---|---|---|---|---|
| **Security material** | | | | |
| Security suite | | GCM-AES-128 | | |
| System Title | *Sys-T* | **4D4D4D**0000BC614E<br>(here, the five last octets contain the manufacturing number in hexadecimal form) | 8 | 64 |
| Invocation Counter | *IC* | 01234567 | 4 | 32 |
| Initialization Vector | *IV* | *Sys-T* ‖ *IC*<br>4D4D4D0000BC614E01234567 | 12 | 96 |
| Block cipher key (global) | *EK* | 000102030405060708090A0B0C0D0E0F | 16 | 128 |
| Authentication Key | *AK* | D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF | 16 | 128 |
| **Security applied** | | **Authenticated encryption** | | |
| Security control byte (with unicast key) | *SC* | *SC-AE*<br>*30* | 1 | 8 |
| Security header | *SH* | *SH = SC-AE* ‖ *IC*<br>3001234567 | 5 | 40 |
| **Inputs** | | | | |
| xDLMS APDU to be protected | *APDU* | 0101100011223344556677889 9AABBCC DDEEFF0000065F1F040000 7E1F04B0 | 31 | 188 |
| Plaintext | *P* | 0101100011223344556 6778899AABBCC DDEEFF0000065F1F0400007E1F04B0 | 31 | 188 |
| Associated data | *A* | *SC* ‖ *AK*<br>30D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF | 17 | 136 |
| **Outputs** | | | | |
| Ciphertext | *C* | 80130FF8A7874133D414CED25B42534 D28DB0047720606B175BD52211BE68 | 31 | 188 |
| Authentication tag | *T* | 41DB204D39EE6FDB8E356855 | 12 | 96 |
| The complete ciphered APDU | | *TAG* ‖ *LEN* ‖ *SH* ‖ *C* ‖ *T*<br>2130300123456780 1302FF8A7874133D 414CED25B42534D28DB0047720606B17 5BD52211BE6841DB204D39EE6FDB8E35 6855 | 50 | 400 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## E.3 The AARQ APDU

In this example, the following values are used:

- Application-Context-Name: Logical_Name_Referencing_With_Ciphering;
- Calling-AP-Title (carries the System title): 4D4D4D0000BC614E;
- Mechanism-Name: COSEM_Low_Level_Security;
- Calling-Authentication-Value: 12345678.

The BER encoding of the AARQ APDU is shown in Table E.3.

**Table E.3 – BER encoding of the AARQ APDU**

| | |
|---|---|
| // encoding of the tag of the AARQ APDU ([APPLICATION 0], Application) | 60 |
| // encoding of the length of the AARQ's contents field (102 octets) | 66 |
| *-- protocol-version field ([0], **IMPLICIT BIT STRING** { version 1 (0) } **DEFAULT** { version1 }* | |
| // no encoding, thus it is considered with its **DEFAULT** value | |
| *-- encoding of the fields of the Kernel* | |
| *-- application-context-name field ([1], Application-context-name, **OBJECT IDENTIFIER**)* | |
| // encoding of the tag ([1], Context-specific) | A1 |
| // encoding of the length of the tagged component's value field | 09 |
| // encoding of the choice for application-context-name (**OBJECT IDENTIFIER**, Universal) | 06 |
| // encoding of the length of the Object Identifier's value field | 07 |
| // encoding of the value of the Object Identifier | 60857405080103 |
| *-- encoding of the calling-AP-title field* | |
| // encoding of the tag ([6], Context-specific) | A6 |
| // encoding of the length of the tagged component's value field | 0A |
| // encoding of the type ([4], Universal, Octetstring type) | 04 |
| // encoding of the length of the calling-AP-title-field | 08 |
| // encoding of the value | 4D4D4D0000BC614E |
| *-- encoding of the fields of the authentication functional unit* | |
| --sender-acse-requirements field ([10], ACSE-requirements, **BIT STRING** { authentication (0) } ) | |
| // encoding of the tag of the acse-requirements field ([10], **IMPLICIT**, Context-specific) | 8A |
| // encoding of the length of the tagged component's value field | 02 |
| // encoding of the number of unused bits in the last byte of the **BIT STRING** | 07 |
| // encoding of the authentication functional unit (0)<br>The number of bits coded may vary from client to client, but within the COSEM environment, only bit 0 set to 1 (indicating the requirement of the authentication functional unit) is to be respected. | 80 |
| *-- mechanism-name field ([11], **IMPLICIT** Mechanism-name **OBJECT IDENTIFIER**)* | |
| // encoding of the tag ([11], **IMPLICIT**, Context-specific) | 8B |
| // encoding of the length of the tagged component's value field | 07 |
| // encoding of the value of the **OBJECT IDENTIFIER**:<br>- low-level-security-mechanism-name, | 60857405080201 |
| *-- calling-authentication-value field ([12], Authentication-value **CHOICE**)* | |
| // encoding of the tag ([12], **EXPLICIT**, Context-specific) | AC |
| // encoding of the length of the tagged component's value field | 0A |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| | |
|---|---|
| // encoding of the choice for Authentication-value (charstring [0] **IMPLICIT GraphicString**) | 80 |
| // encoding of the length of the Authentication-value's value field (8 octets) | 08 |
| // encoding of the calling-authentication-value (12345678) | 3132333435363738 |
| *-- encoding of the user-information field component (Association-information, **OCTET STRING**)* | |
| // encoding of the tag ([30], Context-specific, Constructed) | BE |
| // encoding of the length of the tagged component's value field | 34 |
| // encoding of the choice for user-information (**OCTET STRING**, Universal) | 04 |
| // encoding of the length of the **OCTET STRING's** value field (32 octets) | 32 |
| *-- ciphered xDLMS InitiateRequest APDU* | 2130300123456780<br>1302FF8A7874133D<br>414CED25B42534D2<br>8DB5047720606B17<br>5BD52211BE6841DB<br>204D39EE6FDB8E35<br>6855 |

## E.4 A-XDR encoding of the xDLMS InitiateResponse APDU

In this example:

- the value of the Conformance block is 007C1F;
- the value of the server-max-receive-pdu-size is 1 024 bytes (0x0400).

The A-XDR encoding of the xDLMS InitiateResponse APDU is shown in Table E.4.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table E.4 – A-XDR encoding of the xDLMS InitiateResponse APDU**

| | |
|---|---|
| // encoding of the tag of the DLMS APDU CHOICE (InitiateResponse) | 08 |
| *-- encoding of the negotiated-quality-of-service component ([0] **IMPLICIT** Integer8 **OPTIONAL**)* | |
| // usage flag (**FALSE**, not present) | 00 |
| *-- encoding of the negotiated-dlms-version-number component (Unsigned8)* | |
| // value = 6, the A-XDR encoding of an Unsigned8 is its value | 06 |
| *-- encoding of the negotiated-conformance component (Conformance, [APPLICATION 31] **IMPLICIT BIT STRING** (SIZE(24))* | |
| // encoding of the [APPLICATION 31] tag (ASN.1 explicit tag) | 5F1F |
| // encoding of the length of the 'contents' field in octet (4) | 04 |
| // encoding of the number of unused bits in the final octet of the BIT STRING (0) | 00 |
| // encoding of the fixed length BIT STRING value | 007C1F |
| *-- encoding of the server-max-receive-pdu-size component (Unsigned16)* | |
| // value = 0x0400, the encoding of an Unsigned16 is its value | 0400 |
| *-- encoding of the VAA-Name component (ObjectName, Integer16)* | |
| // value=0x0007; the encoding of a value constrained Integer16 is its value | 0007 |
| *-- resulting octet-string, to be inserted in the user-information field of the AARE APDU* | 0800065F1F040000 7C1F04000007 |

## E.5 Authenticated encryption of the xDLMS InitiateResponse APDU

Table E.5 shows the encoding of the xDLMS InitiateResponse APDU which is also authenticated and encrypted.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table E.5 – Authenticated encryption of the xDLMS InitiateResponse APDU**

| | *X* | Contents | LEN(X) bytes | len(X) bits |
|---|---|---|---|---|
| **Security material** | | | | |
| Security suite | | GCM-AES-128 | | |
| System Title | *Sys-T* | 4D4D4D0000BC614E<br>(here, the five last octets contain the manufacturing number in hexadecimal form) | 8 | 64 |
| Invocation Counter | *IC* | 01234567 | 4 | 32 |
| Initialization Vector | *IV* | *Sys-T* ‖ *IC* | | |
| | | 4D4D4D0000BC614E01234567 | 12 | 96 |
| Block cipher key (global) | *EK* | 000102030405060708090A0B0C0D0E0F | 16 | 128 |
| Authentication Key | *AK* | D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF | 16 | 128 |
| **Security applied** | | **Authenticated encryption** | | |
| Security control byte (with unicast key) | *SC* | *SC-AE* | | |
| | | 30 | 1 | 8 |
| Security header | *SH* | *SH* = *SC-AE* ‖ *IC* | | |
| | | 3001234567 | 5 | 40 |
| **Inputs** | | | | |
| xDLMS APDU to be protected | *APDU* | 0800065F1F0400007C1F04000007 | 14 | 112 |
| Plaintext | *P* | 0800065F1F0400007C1F04000007 | 14 | 112 |
| Associated data | *A* | *SC* ‖ *AK*<br>30D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF | 17 | 136 |
| **Outputs** | | | | |
| Ciphertext | *C* | 891214A0845E475714383F65BC19 | 14 | 112 |
| Authentication tag | *T* | 745CA235906525E4F3E1C893 | 12 | 96 |
| The complete Ciphered APDU | | *TAG* ‖ *LEN* ‖ *SH* ‖ *C* ‖ *T*<br>281F3001234567891214A0845E475714<br>383F65BC19745CA235906525E4F3E1C8<br>93 | 33 | 264 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## E.6 The AARE APDU

The BER encoding of the AARE APDU is shown in Table E.6.

**Table E.6 – BER encoding of the AARE APDU**

| **-- BER encoding of the AARE APDU** | |
|---|---|
| // encoding of the tag for the AARE APDU ([APPLICATION 1], Application) | 61 |
| // encoding of the length of the AARE's contents field (72 octets) | 48 |
| *-- protocol-version field ([0], **IMPLICIT BIT STRING** { version1 (0) } **DEFAULT** { version1 }* | |
| // no encoding, thus it is considered with its **DEFAULT** value | |
| *-- encoding of the fields of the Kernel* | |
| -- application-context-name field ([1], Application-context-name, **OBJECT IDENTIFIER**) | |
| // encoding of the tag ([1], Context-specific) | A1 |
| // encoding of the length of the tagged component's value field | 09 |
| // encoding of the choice for application-context-name **(OBJECT IDENTIFIER**, Universal) | 06 |
| // encoding of the length of the Object Identifier's value field | 07 |
| // encoding of the value of the Object Identifier:<br>NOTE When the proposed application-context does not fit the application-context supported by the server, the server responds either with the application-context name proposed or the application-context-name supported. | 60857405080103 |
| *-- result field ([2], Association-result, **INTEGER**)* | |
| // encoding of the tag ([2], Context-specific) | A2 |
| // encoding of the length of the tagged component's value field | 03 |
| // encoding of the choice for the result (**INTEGER**, Universal) | 02 |
| // encoding of the length of the result's value field | 01 |
| // encoding of the value of the Result: 0, accepted | 00 |
| *-- result-source-diagnostic field ([3], Associate-source-diagnostic, **CHOICE**)* | |
| // encoding of the tag ([3], Context-specific) | A3 |
| // encoding of the length of the tagged component's value field | 05 |
| // encoding of the tag for the acse-service-user CHOICE (1) | A1 |
| // encoding of the length of the tagged component's value field | 03 |
| // encoding of the choice for associate-source-diagnostics (INTEGER, Universal) | 02 |
| // encoding of the length of the value field | 01 |
| // encoding of the value (0, no diagnostics provided) | 00 |
| *-- encoding of the responding-AP-title field* | |
| // encoding of the tag ([4], Context-specific) | A4 |
| // encoding of the length of the tagged component's value field | 0A |
| // encoding of the type ([4], Universal, Octetstring type) | 04 |
| // encoding of the length of the responding-AP-title-field | 08 |
| // encoding of the value | 4D4D4D0000BC614E |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| **-- BER encoding of the AARE APDU** | |
|---|---|
| *-- encoding of the fields of the authentication functional unit*<br>-- In this example the Authentication functional unit is not<br>-- present; it is not necessary in the case of LLS, but if it is<br>-- present it is also acceptable. | |
| *-- encoding of the user-information field component (Association-information, **OCTET STRING**)* | |
| // encoding of the tag ([30], Context-specific, Constructed) | BE |
| // encoding of the length of the tagged component's value field | 23 |
| // encoding of the choice for user-information (**OCTET STRING**, Universal) | 04 |
| // encoding of the length of the **OCTET STRING's** value field | 21 |
| *-- ciphered xDLMS InitiateResponse APDU* | 281F300123456789<br>12J4A0845E475714<br>383F65BC19745CA2<br>35906525E4F3E1C8<br>93 |

## E.7 The RLRQ APDU (carrying a ciphered xDLMS InitiateRequest APDU)

The BER encoding of the RLRQ APDU is shown in Table E.7.

**Table E.7 – BER encoding of the RLRQ APDU**

| **-- BER encoding of the RLRQ APDU** | |
|---|---|
| // encoding of the tag of the RLRQ APDU ([APPLICATION 2], Application) | 62 |
| // encoding of the length of the RLRQ's contents field | 39 |
| *-- reason field* | |
| // encoding of the tag [0], IMPLICIT] | 80 |
| // encoding of the length of the tagged component's value field | 01 |
| // encoding of the value (0, normal) | 00 |
| *-- encoding of the user-information field component (Association-information, **OCTET STRING**)* | |
| // encoding of the tag ([30], Context-specific, Constructed) | BE |
| // encoding of the length of the tagged component's value field | 34 |
| // encoding of the choice for user-information (**OCTET STRING**, Universal) | 04 |
| // encoding of the length of the **OCTET STRING's** value field (14 octets) | 32 |
| // user-information: xDLMS InitiateRequest APDU | 2130300123456780<br>1302FF8A7874133D<br>414CED25B42534D2<br>8DB0047720606B17<br>5BD52211BE6841DB<br>204D39EE6FDB8E35<br>6855 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## E.8 The RLRE APDU (carrying a ciphered xDLMS InitiateResponse APDU)

The BER encoding of the RLRQ APDU is shown in Table E.8.

**Table E.8 – BER encoding of the RLRE APDU**

| **-- BER encoding of the RLRE APDU** | |
|---|---|
| // encoding of the tag of the RLRE APDU ([APPLICATION 3], Application) | 63 |
| // encoding of the length of the RLRE's contents field | 28 |
| -- reason field | |
| // encoding of the tag ([0], IMPLICIT] | 80 |
| // encoding of the length of the tagged component's value field | 01 |
| // encoding of the value (0, normal) | 00 |
| *-- encoding of the user-information field component (Association-information, **OCTET STRING**)* | |
| // encoding of the tag ([30], Context-specific, Constructed) | BE |
| // encoding of the length of the tagged component's value field | 23 |
| // encoding of the choice for user-information (**OCTET STRING**, Universal) | 04 |
| // encoding of the length of the **OCTET STRING's** value field (14 octets) | 21 |
| // user-information: xDLMS InitiateResponse APDU | 281F300123456789<br>1214A0845E475714<br>383F65BC19745CA2<br>35906525E4F3E1C8<br>93 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# Annex F
(informative)

# Data transfer service examples

## F.1 GET / Read, SET / Write examples

Table F.2 to Table F.9 show examples for data exchange using xDLMS services with LN referencing (left column) and SN referencing (right column). Table F.1 shows the objects used in the examples.

**Table F.1 – The objects used in the examples**

```
Object 1:
-   Class: Data
-   Logical name: 0000800000FF
-   Short name of value attribute: 0100
-   Value: octet string of 50 elements
-   01020304050607080910111213141516
-   17181920212223242526272829303132
-   33343536373839404142434445464748
-   4950

Object 2:
-   Class: Data
-   Logical name: 0000800100FF
-   Short name of value attribute: 0110
-   Value: visible string of 3 elements 303030
```

In the case of block transfer, the negotiated APDU size is 40 bytes.

NOTE What is negotiated is the APDU size, not the block size. Therefore, the block size is smaller than the APDU size.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table F.2 – Example: Reading the value of a single attribute without block transfer**

<table>
<tr>
<td>C001C1<br>00010000800000FF0200<br><br><GetRequest><br>  <GetRequestNormal><br>  <InvokeIdAndPriority Value="C1" /><br>  <AttributeDescriptor><br>  <ClassId Value="0001" /><br>  <InstanceId Value="0000800000FF" /><br>  <AttributeId Value="02" /><br>  </AttributeDescriptor><br>  </GetRequestNormal><br></GetRequest></td>
<td>0501<br>020100<br><br><ReadRequest Qty="0001" ><br>  <VariableName Value="0100" /><br></ReadRequest></td>
</tr>
<tr>
<td>C401C1<br>00<br>0932<br>0102030405060708091011121314151617181920212223242526272829303132333435363738394041424344454647484950<br><br><GetResponse><br>  <GetResponsenormal><br>  <InvokeIdAndPriority Value="C1" /><br>  <Result><br>  <Data><br>  <OctetString Value="0102030405060708091011121314151617181920212223242526272829303132333435363738394041424344454647484950" /><br>  </Data><br>  </Result><br>  </GetResponsenormal><br></GetResponse></td>
<td>0C01<br>00<br>0932<br>0102030405060708091011121314151617181920212223242526272829303132333435363738394041424344454647484950<br><br><ReadResponse Qty="0001" ><br>  <Data><br>  <OctetString Value="0102030405060708091011121314151617181920212223242526272829303132333435363738394041424344454647484950" /><br>  </Data><br></ReadResponse></td>
</tr>
</table>

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table F.3 – Example: Reading the value of a list of attributes without block transfer**

| | |
|---|---|
| C003C1<br>02<br>00010000800000FF0200<br>00010000800100FF0200<br><br><GetRequestWithList><br><InvokeIdAndPriority Value="C1" /><br><AttributeDescriptorList Qty="0002" ><br><_AttributeDescriptorWithSelection><br><AttributeDescriptor><br><ClassId Value="0001" /><br><InstanceId Value="0000800000FF" /><br><AttributeId Value="02" /><br></AttributeDescriptor><br></_AttributeDescriptorWithSelection><br><_AttributeDescriptorWithSelection><br><AttributeDescriptor><br><ClassId Value="0001" /><br><InstanceId Value="0000800100FF" /><br><AttributeId Value="02" /><br></AttributeDescriptor><br></_AttributeDescriptorWithSelection><br></AttributeDescriptorList><br></GetRequestWithList><br></GetRequest> | 0502<br>020100<br>020110<br><br><ReadRequest Qty="0002" ><br><VariableName Value="0100" /><br><VariableName Value="0110" /><br></ReadRequest> |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
C403C1
02
00
0932
0102030405060708091011121314151 6
1718192021222324252627282930313 2
3334353637383940414243444546474 8
4950
00
0A03
303030

<GetResponse>
  <GetResponseWithList>
    <InvokeIdAndPriority Value="C1" />
    <Result Qty="0002" >
      <Data>
        <OctetString Value="0102030405060708091011121314151 6
                            1718192021222324252627282930313 2
                            3334353637383940414243444546474 8
                            4950" />
      </Data>
      <Data>
        <VisibleString Value="303030" />
      </Data>
    </Result>
  </GetResponseWithList>
</GetResponse>
```

```
0C02
00
0932
0102030405060708091011121314151 6
1718192021222324252627282930313 2
3334353637383940414243444546474 8
4950
00
0A03
303030

<ReadResponse Qty="0002" >
  <Data>
    <OctetString Value="0102030405060708091011121314151 6
                        1718192021222324252627282930313 2
                        3334353637383940414243444546474 8
                        4950" />
  </Data>
  <Data>
    <VisibleString Value="303030" />
  </Data>
</ReadResponse>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table F.4 – Example: Reading the value of a single attribute with block transfer**

| | |
|---|---|
| C001C1<br>00010000800000FF0200<br><br><GetRequest><br>  <GetRequestNormal><br>  <InvokeIdAndPriority Value="C1" /><br>  <AttributeDescriptor><br>  <ClassId Value="0001" /><br>  <InstanceId Value="0000800000FF" /><br>  <AttributeId Value="02" /><br>  </AttributeDescriptor><br>  </GetRequestNormal><br></GetRequest> | 0501<br>020100<br><br><ReadRequest Qty="0001" ><br>  <VariableName Value="0100" /><br></ReadRequest> |
| C402C1<br>00<br>00000001<br>00<br>1E<br>0932010203040506070809101112 13<br>1415161718192021222324252627 28<br><br><GetResponse><br>  <GetResponsewithDataBlock><br>  <InvokeIdAndPriority Value="C1" /><br>  <Result><br>  <LastBlock Value="00" /><br>  <BlockNumber Value="00000001" /><br>  <Result><br>  <RawData Value="0932010203040506070809101112 13141516171819202122232425262728" /><br>  </Result><br>  </Result><br>  </GetResponsewithDataBlock><br></GetResponse><br>// 30 bytes of raw-data contains data type, length and 28 bytes<br>// of data. Note that Data is encoded, not Get-Data-result. | 0C01<br>02<br>00<br>0001<br>21<br>0100093201020304050607080910111 2<br>1314151617181920212223242526272 8<br>29<br><br><ReadResponse Qty="0001" ><br>  <DataBlockResult><br>  <LastBlock Value="00" /><br>  <BlockNumber Value="0001" /><br>  <RawData Value="0100093201020304050607080910111 2<br>  1314151617181920212223242526272 8<br>  29" /><br>  </DataBlockResult><br></ReadResponse><br><br>// 33 bytes of raw-data contains number of data, success, data<br>// type, length and 29 bytes of data.<br>// As the raw-data contains the data encoded exactly as without<br>// block transfer, the number of results is encoded because the<br>// ReadResponse is a **SEQUENCE OF CHOICE**. |
| C002C1<br>00000001<br><br><GetRequest><br>  <GetRequestForNextDataBlock><br>  <InvokeIdAndPriority Value="C1" /><br>  <BlockNumber Value="00000001" /><br>  </GetRequestForNextDataBlock><br></GetRequest> | 0501<br>050001<br><br><ReadRequest Qty="0001" ><br>  <BlockNumberAccess><br>  <BlockNumber Value="0001" /><br>  </BlockNumberAccess><br></ReadRequest> |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
C402C1
01
00000002
00
16
2930313233343536373839404142434445
4647484950

<GetResponse>
  <GetResponsewithDataBlock>
    <InvokeIdAndPriority Value="C1" />
    <Result>
      <LastBlock Value="01" />
      <BlockNumber Value="00000002" />
      <Result>
        <RawData Value="2930313233343536373839404142434445
                        4647484950" />
      </Result>
    </Result>
  </GetResponsewithDataBlock>
</GetResponse>

// APDU length 32 bytes, 22 bytes of raw-data carries the
// remaining part of data requested.
```

```
0C01
02
01
0002
15
3031323334353637383940414243445
4647484950

<ReadResponse Qty="0001" >
  <DataBlockResult>
    <LastBlock Value="01" />
    <BlockNumber Value="0002" />
    <RawData Value="3031323334353637383940414243445
                    4647484950" />
  </DataBlockResult>
</ReadResponse>

// APDU length 28 bytes, 21 bytes of raw-data carries the
// remaining part of data requested.
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table F.5 – Example: Reading the value of a list of attributes with block transfer**

| | |
|---|---|
| C003C1<br>02<br>00010000800000FF0200<br>00010000800100FF0200<br><br><GetRequestWithList><br><InvokeIdAndPriority Value="C1" /><br><AttributeDescriptorList Qty="0002" ><br><_AttributeDescriptorWithSelection><br><AttributeDescriptor><br><ClassId Value="0001" /><br><InstanceId Value="0000800000FF" /><br><AttributeId Value="02" /><br></AttributeDescriptor><br></_AttributeDescriptorWithSelection><br><_AttributeDescriptorWithSelection><br><AttributeDescriptor><br><ClassId Value="0001" /><br><InstanceId Value="0000800100FF" /><br><AttributeId Value="02" /><br></AttributeDescriptor><br></_AttributeDescriptorWithSelection><br></AttributeDescriptorList><br></GetRequestWithList><br></GetRequest> | 0502<br>020100<br>020110<br><br><ReadRequest Qty="0002" ><br><VariableName Value="0100" /><br><VariableName Value="0110" /><br></ReadRequest> |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
C402C1
00
00000001
00
1E
0200093201020304050607080910111 2
1314151617181920212223242526

<GetResponse>
  <GetResponsewithDataBlock>
    <InvokeIdAndPriority Value="C1" />
    <Result>
      <LastBlock Value="00" />
      <BlockNumber Value="00000001" />
      <Result>
        <RawData Value="0200093201020304050607080910111 2
                        1314151617181920212223242526" />
      </Result>
    </Result>
  </GetResponsewithDataBlock>
</GetResponse>

// 30 bytes of raw-data contains the number of results and part
// of the data. The first one is success, octet-string of 32
// elements; the first 26 bytes fit in.
```

```
0C01
02
00
0001
21
0200093201020304050607080910111 2
1314151617181920212223242526272 8
29

<ReadResponse Qty="0001" >
  <DataBlockResult>
    <LastBlock Value="00" />
    <BlockNumber Value="0001" />
    <RawData Value="0200093201020304050607080910111 2
                    1314151617181920212223242526272 8
                    29" />
  </DataBlockResult>
</ReadResponse>

// 33 bytes of raw-data contains the number of results and part
// of the data. The first one is success, octet-string of 32
// elements; the first 29 bytes fit in.
```

```
C002C1
00000001

<GetRequest>
  <GetRequestForNextDataBlock>
    <InvokeIdAndPriority Value="C1" />
    <BlockNumber Value="00000001" />
  </GetRequestForNextDataBlock>
</GetRequest>
```

```
0501
05
0001

<ReadRequest Qty="0001" >
  <BlockNumberAccess>
    <BlockNumber Value="0001" />
  </BlockNumberAccess>
</ReadRequest>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
C402C1
01
00000002
00
1E
272829303132333435363738394041 42
434445464748495000 0A03303030

<GetResponse>
  <GetResponsewithDataBlock>
    <InvokeIdAndPriority Value="C1" />
    <Result>
      <LastBlock Value="01" />
      <BlockNumber Value="00000002" />
      <Result>
        <RawData Value="272829303132333435363738394041 42
                        434445464748495000 0A03303030" />
      </Result>
    </Result>
  </GetResponsewithDataBlock>
</GetResponse>

// 30 bytes of raw-data contains the remaining 24 bytes of the
// first result and it also contains the six bytes of the second
//result: success, visible string of 3 elements.
```

```
0C01
02
01
0002
1B
303132333435363738394041 42434445
4647484950000A03303030

<ReadResponse Qty="0001" >
  <DataBlockResult>
    <LastBlock Value="01" />
    <BlockNumber Value="0002" />
    <RawData Value="303132333435363738394041 42434445
                    4647484950000A03303030" />
  </DataBlockResult>
</ReadResponse>

// The APDU is 34 bytes. It contains the second and last block.
// 27 bytes of raw-data contains the remaining 21 bytes of the
// first result and the six bytes of the second result: success, //
visible string of 3 elements
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table F.6 – Example: Writing the value of a single attribute without block transfer**

| | |
|---|---|
| C101C1<br>00010000800000FF0200<br>0932<br>0102030405060708091011121314151 6<br>1718192021222324252627282930313 2<br>3334353637383940414243444546474 8<br>4950<br><br><SetRequest><br>  <SetRequestNormal><br>  <InvokeIdAndPriority Value="C1" /><br>  <AttributeDescriptor><br>  <ClassId Value="0001" /><br>  <InstanceId Value="0000800000FF" /><br>  <AttributeId Value="02" /><br>  </AttributeDescriptor><br>  <Value><br>  <OctetString Value="0102030405060708091011121314151 6<br>  1718192021222324252627282930313 2<br>  3334353637383940414243444546474 8<br>  4950" /><br>  </Value><br>  </SetRequestNormal><br></SetRequest> | 06<br>01<br>020100<br>01<br>0932<br>0102030405060708091011121314151 6<br>1718192021222324252627282930313 2<br>3334353637383940414243444546474 8<br>4950<br><br><WriteRequest><br>  <ListOfVariableAccessSpecification Qty="0001" ><br>  <VariableName Value="0100" /><br>  </ListOfVariableAccessSpecification><br>  <ListOfData Qty="0001" ><br>  <OctetString Value="0102030405060708091011121314151 6<br>  1718192021222324252627282930313 2<br>  3334353637383940414243444546474 8<br>  4950" /><br>  </ListOfData><br></WriteRequest> |
| C501C1<br>00<br><SetResponse><br>  <SetResponseNormal><br>  <InvokeIdAndPriority Value="C1" /><br>  <Result Value="Success" /><br>  </SetResponseNormal><br></SetResponse> | 0D01<br>00<br><br><WriteResponse Qty="0001" ><br>  <Success /><br></WriteResponse> |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table F.7 – Example: Writing the value of a list of attributes without block transfer**

```
C104C1
02
00010000800000FF0200
00010000800100FF0200
02
0932
0102030405060708091011121314151 6
1718192021222324252627282930313 2
3334353637383940414243444546474 8
4950
0A03
303030

<SetRequest>
  <SetRequestNormalWithList>
    <InvokeIdAndPriority Value="C1" />
    <AttributeDescriptorList Qty="0002" >
      <_AttributeDescriptorWithSelection>
        <AttributeDescriptor>
          <ClassId Value="0001" />
          <InstanceId Value="00008000 00FF" />
          <AttributeId Value="02" />
        </AttributeDescriptor>
      </_AttributeDescriptorWithSelection>
      <_AttributeDescriptorWithSelection>
        <AttributeDescriptor>
          <ClassId Value="0001" />
          <InstanceId Value="00008001 00FF" />
          <AttributeId Value="02" />
        </AttributeDescriptor>
      </_AttributeDescriptorWithSelection>
    </AttributeDescriptorList>
    <ValueList Qty="0002" >
      <OctetString Value="0102030405060708091011121314151 6
                          1718192021222324252627282930313 2
                          3334353637383940414243444546474 8
                          4950" />
      <VisibleString Value="303030" />
    </ValueList>
  </SetRequestNormalWithList>
</SetRequest>
```

```
0602
020100
020110
02
0932
0102030405060708091011121314151 6
1718192021222324252627282930313 2
3334353637383940414243444546474 8
4950
0A03
303030

<WriteRequest>
  <ListOfVariableAccessSpecification Qty="0002" >
    <VariableName Value="0100" />
    <VariableName Value="0110" />
  </ListOfVariableAccessSpecification>
  <ListOfData Qty="0002" >
    <OctetString Value="0102030405060708091011121314151 6
                        1718192021222324252627282930313 2
                        3334353637383940414243444546474 8
                        4950" />
    <VisibleString Value="303030" />
  </ListOfData>
</WriteRequest>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
C505C1
02
00
00
    <SetResponse>
      <SetResponseWithList>
        <InvokeIdAndPriority Value="C1" />
        <Result Qty="0002" >
          <_DataAccessResult Value="Success" />
          <_DataAccessResult Value="Success" />
        </Result>
      </SetResponseWithList>
    </SetResponse>
```

```
0D02
00
00
    <WriteResponse Qty="0002" >
      <Success />
      <Success />
    </WriteResponse>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table F.8 – Example: Writing the value of a single attribute with block transfer**

| | |
|---|---|
| C102C1<br>00010000800000FF0200<br>00<br>00000001<br>15<br>0932010203040506070809101112131 4<br>1516171819<br><br><SetRequest><br><SetRequestWithFirstDataBlock><br><InvokeIdAndPriority Value="C1" /><br><AttributeDescriptor><br><ClassId Value="0001" /><br><InstanceId Value="0000800000FF" /><br><AttributeId Value="02" /><br></AttributeDescriptor><br><DataBlock><br><LastBlock Value="00" /><br><BlockNumber Value="00000001" /><br><RawData Value="0932010203040506070809101112131 4<br>1516171819" /><br></DataBlock><br></SetRequestWithFirstDataBlock><br></SetRequest><br><br>// 21 bytes of raw-data contain the type, length and the first 19<br>// bytes of data to be written. | 0601<br>07<br>00<br>0001<br>01<br>091F<br>0102010001093201020304050607080 9<br>1011121314151617181920212223 24<br><br><WriteRequest><br><ListOfVariableAccessSpecification Qty="0001" ><br><WriteDataBlockAccess><br><LastBlock Value="00" /><br><BlockNumber Value="0001" /><br></WriteDataBlockAccess><br></ListOfVariableAccessSpecification><br><ListOfData Qty="0001" ><br><OctetString Value="0102010001093201020304050607080 9<br>1011121314151617181920212223 24" /><br></ListOfData><br></WriteRequest><br><br>// 31 bytes of octet-string contains raw-data: the sequence of<br>// Variable-Access-Specification, the sequence of data, the type,<br>// length and the first 24 bytes to be written. |
| C502C1<br>00000001<br><br><SetResponse><br><SetResponseForDataBlock><br><InvokeIdAndPriority Value="C1" /><br><BlockNumber Value="00000001" /><br></SetResponseForDataBlock><br></SetResponse> | 0D01<br>02<br>0001<br><br><WriteResponse Qty="0001" ><br><BlockNumber Value="0001" /><br></WriteResponse> |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

```
C103C1
01
00000002
1F
20212223242526272829303132333435
3637383940414243444546474849 50

<SetRequest>
  <SetRequestWithDataBlock>
    <InvokeIdAndPriority Value="C1" />
    <DataBlock>
      <LastBlock Value="01" />
      <BlockNumber Value="00000002" />
      <RawData Value="20212223242526272829303132333435
                      3637383940414243444546474849 50" />
    </DataBlock>
  </SetRequestWithDataBlock>
</SetRequest>

// 31 bytes of raw-data contains the remaining 21 bytes of the
// data to be written.
```

```
0601
07
01
0002
01
091A
252627282930313233343536373839 40
4142434445464748495 0

<WriteRequest>
  <ListOfVariableAccessSpecification Qty="0001" >
    <WriteDataBlockAccess>
      <LastBlock Value="01" />
      <BlockNumber Value="0002" />
    </WriteDataBlockAccess>
  </ListOfVariableAccessSpecification>
  <ListOfData Qty="0001" >
    <OctetString Value="252627282930313233343536373839 40
                        4142434445464748495 0" />
  </ListOfData>
</WriteRequest>

// The APDU is 35 bytes. 26 bytes of octet-string contains raw-
// data: the remaining 26 bytes of data to be written.
```

```
C503C10000000002

<SetResponse>
  <SetResponseForLastDataBlock>
    <InvokeIdAndPriority Value="C1" />
    <Result Value="Success" />
    <BlockNumber Value="00000002" />
  </SetResponseForLastDataBlock>
</SetResponse>
```

```
0D0100

<WriteResponse Qty="0001" >
  <Success />
</WriteResponse>
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table F.9 – Example: Writing the value of a list of attributes with block transfer**

```
C105C1
02
00010000800000FF0200
00010000800100FF0200
00
00000001
0A
02093201020304050607

<SetRequest>
  <SetRequestWithListAndWithFirstDatablock>
    <InvokeIdAndPriority Value="C1" />
    <AttributeDescriptorList Qty="0002" >
      <_AttributeDescriptorWithSelection>
        <AttributeDescriptor>
          <ClassId Value="0001" />
          <InstanceId Value="00000800000FF" />
          <AttributeId Value="02" />
        </AttributeDescriptor>
      </_AttributeDescriptorWithSelection>
      <_AttributeDescriptorWithSelection>
        <AttributeDescriptor>
          <ClassId Value="0001" />
          <InstanceId Value="00000800100FF" />
          <AttributeId Value="02" />
        </AttributeDescriptor>
      </_AttributeDescriptorWithSelection>
    </AttributeDescriptorList>
    <DataBlock>
      <LastBlock Value="00" />
      <BlockNumber Value="00000001" />
      <RawData Value="02093201020304050607" />
    </DataBlock>
  </SetRequestWithListAndWithFirstDatablock>
</SetRequest>

// The APDU is 40 bytes. It contains the two attribute
// descriptors and 10 bytes of raw-data containing the type and
// length of the first data and the first 7 bytes to be written.
```

```
0601
07
00
0001
01
091F
020201000201100209320102030405060708091011121314151617181920210

<WriteRequest>
  <ListOfVariableAccessSpecification Qty="0001" >
    <WriteDataBlockAccess>
      <LastBlock Value="00" />
      <BlockNumber Value="0001" />
    </WriteDataBlockAccess>
  </ListOfVariableAccessSpecification>
  <ListOfData Qty="0001" >
    <OctetString Value="020201000201100209320102030405060708091011121314151617181920210" />
  </ListOfData>
</WriteRequest>

// The APDU is 40 bytes. 31 bytes of octet-string contains raw-
// data: the number and the name of objects to be written, the
// number of data to be written and the first 21 bytes of the
// first data to be written.
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| | |
|---|---|
| C502C1<br>00000001<br><br><SetResponse><br>  <SetResponseForDataBlock><br>  <InvokeIdAndPriority Value="C1" /><br>  <BlockNumber Value="00000001" /><br>  </SetResponseForDataBlock><br></SetResponse> | 0D01<br>02<br>0001<br><br><WriteResponse Qty="0001" ><br>  <BlockNumber Value="0001" /><br></WriteResponse> |
| C103C1<br>00<br>00000002<br>1F<br>0809101112131415161718192021222324252627282930313233343536373839<br><br><SetRequest><br>  <SetRequestWithDataBlock><br>  <InvokeIdAndPriority Value="C1" /><br>  <DataBlock><br>  <LastBlock Value="00" /><br>  <BlockNumber Value="00000002" /><br>  <RawData Value="0809101112131415161718192021222324252627282930313233343536373839" /><br>  </DataBlock><br>  </SetRequestWithDataBlock><br></SetRequest><br><br>// The APDU is 40 bytes. 31 bytes of raw-data contain the second<br>// part of data to be written. | 0601<br>07<br>00<br>0002<br>01<br>091F<br>2223242526272829303132333435363738393940414243444546474849500A03<br><br><WriteRequest><br>  <ListOfVariableAccessSpecification Qty="0001" ><br>  <WriteDataBlockAccess><br>  <LastBlock Value="00" /><br>  <BlockNumber Value="0002" /><br>  </WriteDataBlockAccess><br>  </ListOfVariableAccessSpecification><br>  <ListOfData Qty="0001" ><br>  <OctetString Value="2223242526272829303132333435363738393940414243444546474849500A03" /><br>  </ListOfData><br></WriteRequest><br><br>// The APDU is 40 bytes. 31 bytes of octet-string contains raw-<br>// data: the second 29 bytes of the first data to be written and<br>// the data type and length of the second data to be written. The<br>// value follows in the next block. |
| C502C100000002<br><br><SetResponse><br>  <SetResponseForDataBlock><br>  <InvokeIdAndPriority Value="C1" /><br>  <BlockNumber Value="00000002" /><br>  </SetResponseForDataBlock><br></SetResponse> | 0D01<br>02<br>0002<br><br><WriteResponse Qty="0001" ><br>  <BlockNumber Value="0002" /><br></WriteResponse> |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| | |
|---|---|
| C103C1<br>01<br>00000003<br>11<br>3940414243444546474849500A033030<br>30<br><br><SetRequest><br><SetRequestWithDataBlock><br><InvokeIdAndPriority Value="C1" /><br><DataBlock><br><LastBlock Value="01" /><br><BlockNumber Value="00000003" /><br><RawData Value="3940414243444546474849500A033030<br>30" /><br></DataBlock><br></SetRequestWithDataBlock><br></SetRequest><br><br>// The APDU is 26 bytes. 17 bytes of raw-data contain the third // part of the first data and the second data to be written. | 0601<br>07<br>01<br>0003<br>01<br>0903<br>303030<br><br><WriteRequest><br><ListOfVariableAccessSpecification Qty="0001" ><br><WriteDataBlockAccess><br><LastBlock Value="01" /><br><BlockNumber Value="0003" /><br></WriteDataBlockAccess><br></ListOfVariableAccessSpecification><br><ListOfData Qty="0001" ><br><OctetString Value="303030" /><br></ListOfData><br></WriteRequest><br><br>// The APDU is 12 bytes. 3 bytes of octet-string contains<br>// raw-data: the value of the second attribute. |
| C504C1<br>02<br>00<br>00<br>00000003<br><br><SetResponse><br><SetResponseForLastDataBlockWithList><br><InvokeIdAndPriority Value="C1" /><br><Result Qty="0002" ><br><_DataAccessResult Value="Success" /><br><_DataAccessResult Value="Success" /><br></Result><br><BlockNumber Value="00000003" /><br></SetResponseForLastDataBlockWithList><br></SetResponse> | 0D02<br>00<br>00<br><br><WriteResponse Qty="0002" ><br><Success /><br><Success /><br></WriteResponse> |

## F.2 ACCESS service example

Table F.10 shows an example of the ACCESS service without general block transfer.

**Table F.10 – Example: ACCESS service without block transfer**

| Message Elements (MAX APDU = 1024) | Contents | LEN (Bytes) |
|---|---|---|
| **Access-Request** | D9 | 1 |
| **long-invoke-id-and-priority** | 40000000 | 4 |
| **date-time** | 00 | 1 |
| **access-request-body** | | 0 |
| **access-request-specification** | | 0 |
| **SEQUENCE OF CHOICE** | 04 | 1 |
| **access-request-get** | 01 | 1 |
| **cosem-attribute-descriptor** | | 0 |
| **class-id** | 0001 | 2 |
| **instance-id** | 0000600100FF | 6 |
| **attr-id** | 02 | 1 |
| **access-request-get** | 01 | 1 |
| **cosem-attribute-descriptor** | | 0 |
| **class-id** | 0008 | 2 |
| **instance-id** | 0000010000FF | 6 |
| **attr-id** | 02 | 1 |
| **access-request-set** | 02 | 1 |
| **cosem-attribute-descriptor** | | 0 |
| **class-id** | 0014 | 2 |
| **instance-id** | 00000D0000FF | 6 |
| **attr-id** | 07 | 1 |
| **access-request-set** | 02 | 1 |
| **cosem-attribute-descriptor** | | 0 |
| **class-id** | 0014 | 2 |
| **instance-id** | 00000D0000FF | 6 |
| **attr-id** | 08 | 1 |
| **access-request-list-of-data** | | |
| **SEQUENCE OF Data** | 04 | 1 |
| **null-data** | 00 | 1 |
| **null-data** | 00 | 1 |
| **array** | 01040203090100090CFFFFFFFFFFFFFFFF<br>00000000000901FF0203090101090CFF<br>FFFFFFFFFFFF00000000000901FF0203<br>09010209 0CFFFFFFFFFFFFFF00000000<br>000901FF0203090103090CFFFFFFFFFF<br>FFFF00000000000901FF | 90 |
| **array** | 01040208090100 11FF11FF11FF11FF11<br>FF11FF11FF020809010111021101 1101<br>1101110111011101020809010211FF11<br>FF11FF11FF11FF11FF11FF0208090103<br>1101110211021102110211021102 | 78 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Message Elements (MAX APDU = 1024) | Contents | LEN (Bytes) |
|---|---|---|
| **Complete Access-Request APDU (encoded)** | D940000000000401000100006001 00FF020100080000010000FF020200140000D0000FF0702001400000D0000FF0804000001040203090100090CFFFFFFFFFFFFFFF00000000000901FF020309010109 0CFFFFFFFFFFFFFFF00000000000901FF0203090102090CFFFFFFFFFFFFFFF00000000000901FF0203090103090CFFFFFFFFFFFFFFF00000000000901FF01040208 09010011FF11FF11FF11FF11FF11FF11FF020809010111021101110111011011011101110210201090102 11FF11FF11FF11FF11FF11FF11FF11FF11FF11FF020809010311011102110211021102110211021102 | 218 |
| **Access-Response** | DA | 1 |
| **long-invoke-id-and-priority** | 40000000 | 4 |
| **date-time** | 00 | 1 |
| **access-response-body** | | 0 |
| **access-request-specification OPTIONAL** | 00 | 1 |
| **access-response-list-of-data** | | 0 |
| **SEQUENCE OF Data** | 04 | 1 |
| **octet-string** | 09083030303030303031 | 10 |
| **octet-string** | 090C07DC030C07161E0000FF8880 | 14 |
| **null-data** | 00 | 1 |
| **null-data** | 00 | 1 |
| **access-response-specification** | | 0 |
| **SEQUENCE OF CHOICE** | 04 | 1 |
| **access-response-get** | 01 | 1 |
| **result** | 00 | 1 |
| **access-response-get** | 01 | 1 |
| **result** | 00 | 1 |
| **access-response-set** | 02 | 1 |
| **result** | 00 | 1 |
| **access-response-set** | 02 | 1 |
| **result** | 00 | 1 |
| **Complete Access-Response APDU (encoded)** | DA400000000000040908303030303030303031090C07DC030C07161E0000FF8880000004010001000200200 | 43 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## F.3 Compact array encoding example

### F.3.1 General

Any series of data of the same type can be encoded as array or compact-array.

The compact-array data type is present from the beginning in the DLMS®/COSEM specification, but so far its interpretation was not unambiguous, therefore it has not been used.

The objective of this subclause is to facilitate the use of compact-array encoding.

### F.3.2 The specification of compact-array

Subclause 7.3.13 specifies the following:

```
Data::= CHOICE
{
    null-data                          [0]   IMPLICIT   NULL,
    array                              [1]   IMPLICIT   SEQUENCE OF Data,
    structure                          [2]   IMPLICIT   SEQUENCE OF Data,
    boolean                            [3]   IMPLICIT   BOOLEAN,
    bit-string                         [4]   IMPLICIT   BIT STRING,
    double-long                        [5]   IMPLICIT   Integer32,
    double-long-unsigned               [6]   IMPLICIT   Unsigned32,
    floating-point                     [7]   IMPLICIT   OCTET STRING(SIZE(4)) ,
    octet-string                       [9]   IMPLICIT   OCTET STRING,
    visible-string                     [10]  IMPLICIT   VisibleString,
    bcd                                [13]  IMPLICIT   Integer8,
    utf8-string                        [12]  IMPLICIT   NULL,
    integer                            [15]  IMPLICIT   Integer8,
    long                               [16]  IMPLICIT   Integer16,
    unsigned                           [17]  IMPLICIT   Unsigned8,
    long-unsigned                      [18]  IMPLICIT   Unsigned16,
    compact-array                      [19]  IMPLICIT   SEQUENCE
    {
        contents-description           [0]              TypeDescription,
        array-contents                 [1]   IMPLICIT   OCTET STRING
    },
    long64                             [20]  IMPLICIT   Integer64,
    long64-unsigned                    [21]  IMPLICIT   Unsigned64,
    enum                               [22]  IMPLICIT   Unsigned8,
    float32                            [23]  IMPLICIT   OCTET STRING (SIZE(4)),
    float64                            [24]  IMPLICIT   OCTET STRING (SIZE(8)),
    date_time                          [25]  IMPLICIT   OCTET STRING (SIZE(12)),
    date                               [26]  IMPLICIT   OCTET STRING (SIZE(5)),
    time                               [27]  IMPLICIT   OCTET STRING (SIZE(4)),
    dont-care                          [255] IMPLICIT   NULL
}

-- The following TypeDescription relates to the compact-array data Type

TypeDescription::= CHOICE
{
    null-data                          [0]   IMPLICIT  NULL,
    array                              [1]   IMPLICIT  SEQUENCE
    {
        number-of-elements      Unsigned16,
        type-description        TypeDescription
    },
    structure                          [2]   IMPLICIT  SEQUENCE OF TypeDescription,
    boolean                            [3]   IMPLICIT  NULL,
    bit-string                         [4]   IMPLICIT  NULL,
    double-long                        [5]   IMPLICIT  NULL,
    double-long-unsigned               [6]   IMPLICIT  NULL,
    floating-point                     [7]   IMPLICIT  NULL,
    octet-string                       [9]   IMPLICIT  NULL,
    visible-string                     [10]  IMPLICIT  NULL,
    bcd                                [13]  IMPLICIT  NULL,
    integer                            [15]  IMPLICIT  NULL,
    long                               [16]  IMPLICIT  NULL,
    unsigned                           [17]  IMPLICIT  NULL,
    long-unsigned                      [18]  IMPLICIT  NULL,

    long64                             [20]  IMPLICIT  NULL,
    long64-unsigned                    [21]  IMPLICIT  NULL,
    enum                               [22]  IMPLICIT  NULL,
    float32                            [23]  IMPLICIT  NULL,
    float64                            [24]  IMPLICIT  NULL,
    date_time                          [25]  IMPLICIT  NULL,
    date                               [26]  IMPLICIT  NULL,
    time                               [27]  IMPLICIT  NULL,
    dont-care                          [255] IMPLICIT  NULL
}
```

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Notice that in the compact-array type:

- contents-description / TypeDescription specifies the data type of the elements in the compact array:
- in the case of simple data types it contains the tag of the type. In the case of string types, the length is not part of the TypeDescription: it is conveyed as part of the array contents;

  NOTE For example if the data includes octet-strings, only the tag [9] is included in the contents-description. The length of the octet-string is included in the array-contents. With this, string type data with different lengths (including a length of 0) can be encoded in the compact-array.

- in the case of *array*, it includes the tag of the array [1], the number of elements in the array (Unsigned16) and the TypeDescription of the elements in the array;
- in the case of *structure* the TypeDescription is specified as a SEQUENCE OF TypeDescription. Therefore, the contents-description includes the tag of the structure [2], the number of elements in the structure and the TypeDescription of each element in the structure.
- the array-contents includes the series of data – when relevant (in the case of string types) together with its length, without repeating the data type – as an octet string;

Note also that although the contents-description and the array-contents elements of the compact-array type are tagged, these tags do not have to be encoded, as specified in IEC 61334-6:2000, 6.9:

> *A-XDR encoding of a SEQUENCE value shall be the A-XDR encoding of one data value from each of the types listed in the ASN.1 definition of the SEQUENCE type, in the order of their appearance in the definition, unless the type was referenced with the keyword "OPTIONAL" or the keyword "DEFAULT".*
>
> *Tags of explicitly tagged components of a SEQUENCE value represent redundant information, therefore are not encoded: A-XDR encoding of an explicit tagged component value is the A-XDR encoding of the component value.*

### F.3.3 Example 1: Compact array encoding an array of five long-unsigned values

An array of 5 elements of type long-unsigned has to be encoded.

The values of the five elements are: 11 11, 22 22, 33 33, 44 44, 55 55.

| Encoding as array | Encoding as compact-array |
|---|---|
| 01 // tag of array | 13 // tag of compact-array |
| 05 // number of elements | // contents-description |
| 12 11 11 // tag of long-unsigned type and first value | 12 // tag of the long-unsigned type |
| 12 22 22 // tag of long-unsigned type and second value | // array-contents |
| 12 33 33 // etc. | 0A // length of octet-string |
| 12 44 44 | 11 11 22 22 33 33 44 44 55 55 |
| 12 55 55 | // the five values |

The length of the encoded data in the two cases is shown in the table below. In the case of the long-unsigned type, 33 % per element can be saved.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| | Array | Compact array | Gain compared to array |
|---|---|---|---|
| Header | 2 | 3 | -1 |
| First element | 3 | 2 | 1 |
| Second element | 3 | 2 | 1 |
| Third element | 3 | 2 | 1 |
| Fourth element | 3 | 2 | 1 |
| Fifth element | 3 | 2 | 1 |
| **Total** | **17** | **13** | **4** |

### F.3.4 Example 2: Compact-array encoding of five octet-string values

An array of five octet-string values has to be encoded, of which one is of zero length.

- 31 32 33 34 35 36 37 38
- 41 42 43 44 45 46 47 48
- -
- 31 32 33 34 35 36 37 38
- 41 42 43 44 45 46 47 48

| Encoding as array | Encoding as compact-array |
|---|---|
| 01 // tag of array | 13 // tag of compact array |
| 05 // number of elements | // contents description |
| | 09 // tag for octet-string |
| | // array contents |
| 09 08 31 32 33 34 35 36 37 38 // type – length – value | 25 // length of octet-string, 37 bytes |
| 09 08 41 42 43 44 45 46 47 48 // second value | 08 31 32 33 34 35 36 37 38 // length – value |
| 09 00 // third value, octet-string of length 0 | 08 41 42 43 44 45 46 47 48 // second length – value |
| 09 08 31 32 33 34 35 36 37 38 // fourth value | 00 // third value octet-string of length 0 |
| 09 08 41 42 43 44 45 46 47 48 // fifth value | 08 31 32 33 34 35 36 37 38 // fourth length – value |
| | 08 41 42 43 44 45 46 47 48 // fifth length – value |

The length of the encoded data in the two cases is shown in the table below.

In the case of octet-string, the gain depends on the length of the octet-string. In the case of octet-string of length zero (null-data) the gain is 50 % per element.

| | Array | Compact array | Gain compared to array |
|---|---|---|---|
| Header | 2 | 3 | -1 |
| First element | 10 | 9 | 1 |
| Second element | 10 | 9 | 1 |
| Third element | 2 | 1 | 1 |
| Fourth element | 10 | 9 | 1 |
| Fifth element | 10 | 9 | 1 |
| **Total** | **44** | **40** | **4** |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### F.3.5 Example 3: Encoding of the buffer of a Profile generic object

The profile has a time stamp column, a status column, and two columns carrying a double-long-unsigned value each (e.g. A+ and A-, import and export active energy). The capture period is 900 s. There are 96 entries (one day).

NOTE If instead of register readings just delta values would be stored, they could be represented as long-unsigned instead of double-long-unsigned.

| Entry | Timestamp | Status | Value | Value | Bytes |
|---|---|---|---|---|---|
| 1 | 07D00101FF000000FF800000 | 80 | 00000101 | 00000001 | 21 |
| 2 | 07D00101FF000F00FF800000 | 00 | 00000102 | 00000002 | 21 |
| 3 | 07D00101FF001E00FF800000 | 00 | 00000103 | 00000003 | 21 |
| 4 | 07D00101FF002D00FF800000 | 00 | 00000104 | 00000004 | 21 |
| ..... | | | | | ... |
| 96 | 07D00101FF172D00FF800000 | 00 | 00000196 | 00000096 | 21 |
| | | | | **Total bytes** | **2 016** |

| Encoding as array (using the null-data feature) | Encoding as compact-array |
|---|---|
| 01 60 // array of 96 elements | 13 // compact-array<br>// contents-description<br>02 04 09110606<br>// structure of four elements:<br>// octet-string,<br>// unsigned,<br>// double-long-unsigned,<br>// double-long-unsigned |
| | // array-contents<br>8203CC // length of octet-string 972 bytes |
| // first structure, 28 bytes<br>0204 // structure of 4 elements<br>090C07D00101FF000000FF800000<br>1180 // status value is 80<br>0600000101<br>0600000001 | // first structure, 22 bytes<br><br>0C07D00101FF000000FF800000<br>80<br>00000101<br>00000001 |
| // second structure, 15 bytes<br>0204<br>00 // null-data for time stamp<br>1100 // status value is 0<br>0600000102<br>0600000002 | // second structure, 10 bytes<br>00 // an octet-string of length 0 has the same effect as null-data<br>00 // status value is 0<br>00000102<br>00000002 |
| // third structure 14 bytes<br>0204<br>00 // null-data for time stamp<br>00 // null-data for status<br>0600000103<br>0600000003 | // third structure, 10 bytes<br><br>00 // octet-string of length 0<br>00 // status value is 0<br>00000103<br>00000003 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Encoding as array (using the null-data feature) | Encoding as compact-array |
|---|---|
| // fourth structure 14 bytes<br>0204<br>00 // null-data for time stamp<br>00 // null-data for status<br>0600000104<br>0600000004 | // fourth structure, 10 bytes<br><br>00 // octet-string of length 0<br>00 // status value is 0<br>00000104<br>00000004 |
| … | … |
| // ninety-sixth structure 14 bytes<br>0204<br>00 // null-data for time stamp<br>00 // null-data for status<br>0600000196<br>0600000096 | // ninety-sixth structure, 10 bytes<br><br>00 // octet-string of length 0<br>00 // status value is 0<br>00000196<br>00000096 |
| In the case of using compact array, it is not possible to know the number of elements from the length of the octet string of the array-contents.<br>In the case of compact array encoding, the null-data feature can be applied only for string type data. | |

The length of the encoded data in the two cases is shown in the table below.

| | Array (using null-data) | Compact array | Gain compared to array |
|---|---|---|---|
| Header | 2 | 10 | -8 |
| First structure | 28 | 22 | 6 |
| Second structure | 15 | 10 | 5 |
| Third structure | 14 | 10 | 4 |
| Fourth structure | 14 | 10 | 4 |
| | | | |
| 96th structure | 14 | 10 | 4 |
| **Total** | **1361** | **982** | **375** |
| **Encoded data in % of raw data** | **68%** | **49%** | |

When encoding the data as an array, the use of the null data feature allows compression of 32 % in this example. When encoding as a compact array the compression is 51 %.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## F.4 Profile generic IC buffer attribute encoding examples

### F.4.1 General

The following tables show encoding examples of reading a Profile generic IC buffer attribute with:

- normal encoding;
- null-data encoding;
- compact-array encoding;
- null-data and delta-value encoding combined;
- the various methods combined with compression.

### F.4.2 Get-response with Profile generic normal encoding example

Table F.11 shows an encoding example for a get-response-normal APDU reading a Profile generic object buffer attribute with normal encoding. The content of the buffer is determined by the capture_objects attribute that references:

- the Clock object time attribute;
- the value attribute of a status object; and
- the value attribute of an energy register.

By selective access parameters, one day is selected with 24 entries. The size of the APDU is 558 bytes.

**Table F.11 – Profile generic buffer – get-response with normal encoding**

| Message Elements | Index | Value | Contents | LEN (Bytes) |
|---|---|---|---|---|
| **get-response** | | | C4 | 1 |
| **get-response-normal** | | | 01 | 1 |
| **invoke-id-and-priority** | | | 00 | 1 |
| result CHOICE | | | | 0 |
| data | | | 00 | 1 |
| array [24] | | | 0118 | 2 |
| structure [3] | 0 | | 0203 | 2 |
| octet-string | | 12-2-2018 0:00:00 | 090C07E2020C050000 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 100000 | 06000186A0 | 5 |
| structure [3] | 1 | | 0203 | 2 |
| octet-string | | 12-2-2018 1:00:00 | 090C07E2020C050100 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 100416 | 0600018840 | 5 |
| structure [3] | 2 | | 0203 | 2 |
| octet-string | | 12-2-2018 2:00:00 | 090C07E2020C050200 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 100832 | 06000189E0 | 5 |
| structure [3] | 3 | | 0203 | 2 |
| octet-string | | 12-2-2018 3:00:00 | 090C07E2020C050300 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 101248 | 0600018B80 | 5 |
| structure [3] | 4 | | 0203 | 2 |
| octet-string | | 12-2-2018 4:00:00 | 090C07E2020C050400 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 101664 | 0600018D20 | 5 |
| structure [3] | 5 | | 0203 | 2 |
| octet-string | | 12-2-2018 5:00:00 | 090C07E2020C050500 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 102080 | 0600018EC0 | 5 |
| structure [3] | 6 | | 0203 | 2 |
| octet-string | | 12-2-2018 6:00:00 | 090C07E2020C050600 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 102496 | 0600019060 | 5 |
| structure [3] | 7 | | 0203 | 2 |
| octet-string | | 12-2-2018 7:00:00 | 090C07E2020C050700 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 102912 | 0600019200 | 5 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Message Elements | Index | Value | Contents | LEN (Bytes) |
|---|---|---|---|---|
| structure [3] | 8 | | 0203 | 2 |
| octet-string | | 12-2-2018 8:00:00 | 090C07E2020C050800 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 103328 | 06000193A0 | 5 |
| structure [3] | 9 | | 0203 | 2 |
| octet-string | | 12-2-2018 9:00:00 | 090C07E2020C050900 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 103744 | 0600019540 | 5 |
| structure [3] | 10 | | 0203 | 2 |
| octet-string | | 12-2-2018 10:00:00 | 090C07E2020C050A00 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 104160 | 06000196E0 | 5 |
| structure [3] | 11 | | 0203 | 2 |
| octet-string | | 12-2-2018 11:00:00 | 090C07E2020C050B00 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 104576 | 0600019880 | 5 |
| structure [3] | 12 | | 0203 | 2 |
| octet-string | | 12-2-2018 12:00:00 | 090C07E2020C050C00 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 104992 | 0600019A20 | 5 |
| structure [3] | 13 | | 0203 | 2 |
| octet-string | | 12-2-2018 13:00:00 | 090C07E2020C050D00 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 105408 | 0600019BC0 | 5 |
| structure [3] | 14 | | 0203 | 2 |
| octet-string | | 12-2-2018 14:00:00 | 090C07E2020C050E00 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 105824 | 0600019D60 | 5 |
| structure [3] | 15 | | 0203 | 2 |
| octet-string | | 12-2-2018 15:00:00 | 090C07E2020C050F00 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 106240 | 0600019F00 | 5 |
| structure [3] | 16 | | 0203 | 2 |
| octet-string | | 12-2-2018 16:00:00 | 090C07E2020C051000 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 106656 | 060001A0A0 | 5 |
| structure [3] | 17 | | 0203 | 2 |
| octet-string | | 12-2-2018 17:00:00 | 090C07E2020C051100 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 107072 | 060001A240 | 5 |
| structure [3] | 18 | | 0203 | 2 |
| octet-string | | 12-2-2018 18:00:00 | 090C07E2020C051200 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 107488 | 060001A3E0 | 5 |
| structure [3] | 19 | | 0203 | 2 |
| octet-string | | 12-2-2018 19:00:00 | 090C07E2020C051300 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 107904 | 060001A580 | 5 |
| structure [3] | 20 | | 0203 | 2 |
| octet-string | | 12-2-2018 20:00:00 | 090C07E2020C051400 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Message Elements | Index | Value | Contents | LEN (Bytes) |
|---|---|---|---|---|
| double-long-unsigned | | 108320 | 060001A720 | 5 |
| structure [3] | 21 | | 0203 | 2 |
| octet-string | | 12-2-2018 21:00:00 | 090C07E2020C051500 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 108736 | 060001A8C0 | 5 |
| structure [3] | 22 | | 0203 | 2 |
| octet-string | | 12-2-2018 22:00:00 | 090C07E2020C051600 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 109152 | 060001AA60 | 5 |
| structure [3] | 23 | | 0203 | 2 |
| octet-string | | 12-2-2018 23:00:00 | 090C07E2020C051700 0000800000 | 14 |
| unsigned | | 0 | 1100 | 2 |
| double-long-unsigned | | 109568 | 060001AC00 | 5 |
| The complete get-response APDU:<br><br>C401000001180203090C07E2020C050000000080000011000600018 6A00203090C07E2020C050100000800000110006000188400203090C07E2020C050200000080000011000600189E00203090C07E2020C050300000080000011000600018B800203090C07E2020C050400000080000011000600018D200203090C07E2020C050500000080000011000600018EC00203090C07E2020C050600000080000011000600019060020 3090C07E2020C050700000080000011000600019200020 3090C07E2020C050800000080000011000600019 3A00203090C07E2020C050900000080000011000600019540020 3090C07E2020C050A00000080000011000600019 6E00203090C07E2020C050B00000080000011000600019 8800203090C07E2020C050C00000080000011000600019A200203090C07E2020C050D00000080000011000600019BC00203090C07E2020C050E00000080000011000600019D600203090C07E2020C050F00000080000011000600019F000203090C07E2020C05100000008000001100060001A0A00203090C07E2020C05110000008000001100060001A2400203090C07E2020C05120000008000001100060001A3E00203090C07E2020C05130000008000001100060001A5800203090C07E2020C05140000008000001100060001A7200203090C07E2020C05150000008000001100060001A8C00203090C07E2020C05160000008000001100060001AA600203090C07E2020C05170000008000001100060001AC00 | | | | 558 |

### F.4.3 Get-response with Profile generic null-data compressed encoding example

Table F.12 shows an encoding example for the same get-response APDU but using null-data encoding. The time stamp is provided for the first reading only. For the subsequent entries null-data are provided since from the capture_period attribute it is known that the capture occurs hourly. Similarly, the status is provided only in the first entry because it is the same for each subsequent entry. This method results in a buffer size of 236 bytes compared to 558 for normal encoding.

**Table F.12 – Profile generic buffer – get-response with null-data compression**

| Message Elements | Index | Value | Contents | LEN (Bytes) |
|---|---|---|---|---|
| **get-response** | | | C4 | 1 |
| **get-response-normal** | | | 01 | 1 |
| **invoke-id-and-priority** | | | 00 | 1 |
| **result CHOICE** | | | | 0 |
| **data** | | | 00 | 1 |
| **array [24]** | | | 0118 | 2 |
| **structure [3]** | 0 | | 0203 | 2 |
| **octet-string** | | 12-2-2018 0:00:00 | 090C07E2020C050000 0000800000 | 14 |
| **unsigned** | | 0 | 1100 | 2 |
| **double-long-unsigned** | | 100000 | 06000186A0 | 5 |
| **structure [3]** | 1 | | 0203 | 2 |
| **null-data** | | 12-2-2018 1:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Message Elements | Index | Value | Contents | LEN (Bytes) |
|---|---|---|---|---|
| **double-long-unsigned** | | 100416 | 0600018840 | 5 |
| **structure [3]** | 2 | | 0203 | 2 |
| **null-data** | | 12-2-2018 2:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 100832 | 06000189E0 | 5 |
| **structure [3]** | 3 | | 0203 | 2 |
| **null-data** | | 12-2-2018 3:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 101248 | 0600018B80 | 5 |
| **structure [3]** | 4 | | 0203 | 2 |
| **null-data** | | 12-2-2018 4:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 101664 | 0600018D20 | 5 |
| **structure [3]** | 5 | | 0203 | 2 |
| **null-data** | | 12-2-2018 5:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 102080 | 0600018EC0 | 5 |
| **structure [3]** | 6 | | 0203 | 2 |
| **null-data** | | 12-2-2018 6:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 102496 | 0600019060 | 5 |
| **structure [3]** | 7 | | 0203 | 2 |
| **null-data** | | 12-2-2018 7:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 102912 | 0600019200 | 5 |
| **structure [3]** | 8 | | 0203 | 2 |
| **null-data** | | 12-2-2018 8:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 103328 | 06000193A0 | 5 |
| **structure [3]** | 9 | | 0203 | 2 |
| **null-data** | | 12-2-2018 9:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 103744 | 0600019540 | 5 |
| **structure [3]** | 10 | | 0203 | 2 |
| **null-data** | | 12-2-2018 10:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 104160 | 06000196E0 | 5 |
| **structure [3]** | 11 | | 0203 | 2 |
| **null-data** | | 12-2-2018 11:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 104576 | 0600019880 | 5 |
| **structure [3]** | 12 | | 0203 | 2 |
| **null-data** | | 12-2-2018 12:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 104992 | 0600019A20 | 5 |
| **structure [3]** | 13 | | 0203 | 2 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Message Elements | Index | Value | Contents | LEN (Bytes) |
|---|---|---|---|---|
| **null-data** | | 12-2-2018 13:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 105408 | 0600019BC0 | 5 |
| **structure [3]** | 14 | | 0203 | 2 |
| **null-data** | | 12-2-2018 14:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 105824 | 0600019D60 | 5 |
| **structure [3]** | 15 | | 0203 | 2 |
| **null-data** | | 12-2-2018 15:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 106240 | 0600019F00 | 5 |
| **structure [3]** | 16 | | 0203 | 2 |
| **null-data** | | 12-2-2018 16:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 106656 | 060001A0A0 | 5 |
| **structure [3]** | 17 | | 0203 | 2 |
| **null-data** | | 12-2-2018 17:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 107072 | 060001A240 | 5 |
| **structure [3]** | 18 | | 0203 | 2 |
| **null-data** | | 12-2-2018 18:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 107488 | 060001A3E0 | 5 |
| **structure [3]** | 19 | | 0203 | 2 |
| **null-data** | | 12-2-2018 19:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 107904 | 060001A580 | 5 |
| **structure [3]** | 20 | | 0203 | 2 |
| **null-data** | | 12-2-2018 20:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 108320 | 060001A720 | 5 |
| **structure [3]** | 21 | | 0203 | 2 |
| **null-data** | | 12-2-2018 21:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 108736 | 060001A8C0 | 5 |
| **structure [3]** | 22 | | 0203 | 2 |
| **null-data** | | 12-2-2018 22:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 109152 | 060001AA60 | 5 |
| **structure [3]** | 23 | | 0203 | 2 |
| **null-data** | | 12-2-2018 23:00:00 | 00 | 1 |
| **null-data** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 109568 | 060001AC00 | 5 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Message Elements | Index | Value | Contents | LEN (Bytes) |
|---|---|---|---|---|
| The complete get-response APDU:<br><br>C401000001180203090C07E2020C0500000000800000110006000186A0020300000600018840020300<br>0006000189E0020300000600018B8002030000060001 8D2002030000060001 8EC00203000006000190<br>6002030000060001920002030000060001 93A0020300000600019540020300000600019 6E002030000<br>0600019880020300000600019A2002030000060001 9BC0020300000600019D6002030000060001 9F00<br>02030000060001A0A002030000060001A2400203000006000 1A3E002030000060001A5800203000006<br>0001A7200203000006000 1A8C002030000060001AA6002030000060001AC00 | | | | 236 |

### F.4.4 Get-response with Profile generic compact-array encoding example

Table F.13 shows an encoding example for the same get-response APDU but using compact-array encoding. This method results in a buffer size of 168 bytes compared to 558 for normal encoding.

**Table F.13 – Profile generic buffer – get-response with compact-array encoding**

| Message Elements | Index | Value | Contents | LEN (Bytes) |
|---|---|---|---|---|
| **get-response** | | | C4 | 1 |
| **get-response-normal** | | | 01 | 1 |
| **invoke-id-and-priority** | | | 00 | 1 |
| **result CHOICE** | | | | 0 |
| **data** | | | 00 | 1 |
| **compact-array** | | | 13 | 1 |
| **contents-description** | | | | 0 |
| **structure [3]** | | | 0203 | 2 |
| **octet-string** | | | 09 | 1 |
| **unsigned** | | | 11 | 1 |
| **double-long-unsigned** | | | 06 | 1 |
| **array-contents** | | 156 | 819C | 2 |
| **structure [3]** | 0 | | | 0 |
| **octet-string** | | 12-2-2018 0:00:00 | 0C07E2020C05000 00000800000 | 13 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 100000 | 000186A0 | 4 |
| **structure [3]** | 1 | | | 0 |
| **octet-string** | | 12-2-2018 1:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 100416 | 00018840 | 4 |
| **structure [3]** | 2 | | | 0 |
| **octet-string** | | 12-2-2018 2:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 100832 | 000189E0 | 4 |
| **structure [3]** | 3 | | | 0 |
| **octet-string** | | 12-2-2018 3:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 101248 | 00018B80 | 4 |
| **structure [3]** | 4 | | | 0 |
| **octet-string** | | 12-2-2018 4:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 101664 | 00018D20 | 4 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Message Elements | Index | Value | Contents | LEN (Bytes) |
|---|---|---|---|---|
| **structure [3]** | 5 | | | 0 |
| **octet-string** | | 12-2-2018 5:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 102080 | 00018EC0 | 4 |
| **structure [3]** | 6 | | | 0 |
| **octet-string** | | 12-2-2018 6:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 102496 | 00019060 | 4 |
| **structure [3]** | 7 | | | 0 |
| **octet-string** | | 12-2-2018 7:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 102912 | 00019200 | 4 |
| **structure [3]** | 8 | | | 0 |
| **octet-string** | | 12-2-2018 8:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 103328 | 000193A0 | 4 |
| **structure [3]** | 9 | | | 0 |
| **octet-string** | | 12-2-2018 9:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 103744 | 00019540 | 4 |
| **structure [3]** | 10 | | | 0 |
| **octet-string** | | 12-2-2018 10:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 104160 | 000196E0 | 4 |
| **structure [3]** | 11 | | | 0 |
| **octet-string** | | 12-2-2018 11:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 104576 | 00019880 | 4 |
| **structure [3]** | 12 | | | 0 |
| **octet-string** | | 12-2-2018 12:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 104992 | 00019A20 | 4 |
| **structure [3]** | 13 | | | 0 |
| **octet-string** | | 12-2-2018 13:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 105408 | 00019BC0 | 4 |
| **structure [3]** | 14 | | | 0 |
| **octet-string** | | 12-2-2018 14:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 105824 | 00019D60 | 4 |
| **structure [3]** | 15 | | | 0 |
| **octet-string** | | 12-2-2018 15:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 106240 | 00019F00 | 4 |
| **structure [3]** | 16 | | | 0 |
| **octet-string** | | 12-2-2018 16:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 106656 | 0001A0A0 | 4 |
| **structure [3]** | 17 | | | 0 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Message Elements | Index | Value | Contents | LEN (Bytes) |
|---|---|---|---|---|
| **octet-string** | | 12-2-2018 17:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 107072 | 0001A240 | 4 |
| **structure [3]** | 18 | | | 0 |
| **octet-string** | | 12-2-2018 18:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 107488 | 0001A3E0 | 4 |
| **structure [3]** | 19 | | | 0 |
| **octet-string** | | 12-2-2018 19:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 107904 | 0001A580 | 4 |
| **structure [3]** | 20 | | | 0 |
| **octet-string** | | 12-2-2018 20:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 108320 | 0001A720 | 4 |
| **structure [3]** | 21 | | | 0 |
| **octet-string** | | 12-2-2018 21:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 108736 | 0001A8C0 | 4 |
| **structure [3]** | 22 | | | 0 |
| **octet-string** | | 12-2-2018 22:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 109152 | 0001AA60 | 4 |
| **structure [3]** | 23 | | | 0 |
| **octet-string** | | 12-2-2018 23:00:00 | 00 | 1 |
| **unsigned** | | 0 | 00 | 1 |
| **double-long-unsigned** | | 109568 | 0001AC00 | 4 |
| The complete get-response APDU:<br><br>C401000013020309110681 9C0C07E2020C0500000000800000000000186A0000000018840000000018 9E0000000018B8000000001 8D20000000018EC0000000019060000000019200000000193A000000001 954000000000196E000000001 9880000000019A20000000019BC0000000019D60000000019F000000000 01A0A000000001A240000000001A3E000000001A58000000001A72000000001A8C000000001AA600000 0001AC00 | | | | 168 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### F.4.5 Get-response with Profile generic null-data and delta-value encoding example

Table F.14 shows an encoding example for the same get-response APDU but using null-data encoding combined with delta-value encoding. The first entry contains the full time stamp, the status and the full register value. For subsequent entries, null-data are provided for the time stamp and the status and delta-value is provided for the register value. In this example the change is a constant value of 41 which can be encoded as a delta--unsigned instead of a double-long-unsigned. This method results in a buffer size of 167 bytes compared to 558 for normal encoding.

**Table F.14 – Profile generic buffer – Get-response with null-data and delta-value encoding**

| Message Elements | Index | Value | Delta | Contents | LEN (Bytes) |
|---|---|---|---|---|---|
| **get-response** | | | | C4 | 1 |
| **get-response-normal** | | | | 01 | 1 |
| **invoke-id-and-priority** | | | | 00 | 1 |
| **result CHOICE** | | | | | 0 |
| **data** | | | | 00 | 1 |
| **array [24]** | | | | 0118 | 2 |
| **structure [3]** | 0 | | | 0203 | 2 |
| **octet-string** | | 12-2-2018 0:00:00 | | 090C07E2020C 050000000080 0000 | 14 |
| **unsigned** | | 0 | | 1100 | 2 |
| **double-long-unsigned** | | 100000 | | 06000186A0 | 5 |
| **structure [3]** | 1 | 0 | | 0203 | 2 |
| **null-data** | | 12-2-2018 1:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100041 | 41 | 1F29 | 2 |
| **structure [3]** | 2 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 2:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100082 | 41 | 1F29 | 2 |
| **structure [3]** | 3 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 3:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100123 | 41 | 1F29 | 2 |
| **structure [3]** | 4 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 4:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100164 | 41 | 1F29 | 2 |
| **structure [3]** | 5 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 5:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100205 | 41 | 1F29 | 2 |
| **structure [3]** | 6 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 6:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100246 | 41 | 1F29 | 2 |
| **structure [3]** | 7 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 7:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100287 | 41 | 1F29 | 2 |
| **structure [3]** | 8 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 8:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100328 | 41 | 1F29 | 2 |
| **structure [3]** | 9 | | | 0203 | 2 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Message Elements | Index | Value | Delta | Contents | LEN (Bytes) |
|---|---|---|---|---|---|
| **null-data** | | 12-2-2018 9:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100369 | 41 | 1F29 | 2 |
| **structure [3]** | 10 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 10:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100410 | 41 | 1F29 | 2 |
| **structure [3]** | 11 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 11:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100451 | 41 | 1F29 | 2 |
| **structure [3]** | 12 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 12:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100492 | 41 | 1F29 | 2 |
| **structure [3]** | 13 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 13:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100533 | 41 | 1F29 | 2 |
| **structure [3]** | 14 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 14:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100574 | 41 | 1F29 | 2 |
| **structure [3]** | 15 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 15:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100615 | 41 | 1F29 | 2 |
| **structure [3]** | 16 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 16:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100656 | 41 | 1F29 | 2 |
| **structure [3]** | 17 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 17:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100697 | 41 | 1F29 | 2 |
| **structure [3]** | 18 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 18:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100738 | 41 | 1F29 | 2 |
| **structure [3]** | 19 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 19:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100779 | 41 | 1F29 | 2 |
| **structure [3]** | 20 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 20:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Message Elements | Index | Value | Delta | Contents | LEN (Bytes) |
|---|---|---|---|---|---|
| **unsigned-delta** | | 100820 | 41 | 1F29 | 2 |
| **structure [3]** | 21 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 21:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100861 | 41 | 1F29 | 2 |
| **structure [3]** | 22 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 22:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100902 | 41 | 1F29 | 2 |
| **structure [3]** | 23 | | | 0203 | 2 |
| **null-data** | | 12-2-2018 23:00:00 | | 00 | 1 |
| **null-data** | | 0 | | 00 | 1 |
| **unsigned-delta** | | 100943 | 41 | 1F29 | 2 |
| The complete get-response APDU:<br><br>C401000001180203090C07E2020C0500000000800000110006000186A0020300001F29020300001F290<br>20300001F29020300001F29020300001F29020300001F29020300001F29020300001F29020300001F29<br>020300001F29020300001F29020300001F29020300001F29020300001F29020300001F29020300001F2<br>9020300001F29020300001F29020300001F29020300001F29020300001F29020300001F29020300001F<br>29 | | | | | 167 |

### F.4.6 Comparison of various encoding methods for Get-response APDU

Table F.15 summarizes the result of the various encoding methods presented, also showing the results when data for 2, 4 or 7 days are captured.

### F.4.7 Combination of the various encoding methods and V.44 compression

V.44 compression can be used to further decrease the size of the APDU. Table F.16 shows the resulting APDU sizes for the same examples shown in Table F.15.

**Table F.15 – Comparison of various encoding methods for get-response APDU**

| Profile Generic Parameters | | | | Profile Generic Encoding | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Records | Period | Interval | Values | Normal | | Compressed null-data | | compact array | | Compressed null-data & delat values | | Compressed null-data & delta values (const) | |
| Number | Minutes | Days | Number | Size | Ratio | Size | Ratio | Size | Ratio | Size | Ratio | Size | Ratio |
| 24 | 60 | 1 | 1 | 558 | 100% | 236 | 42% | 168 | 30% | 168 | 30% | 167 | 30% |
| 48 | 60 | 2 | 1 | 1110 | 100% | 452 | 41% | 313 | 28% | 336 | 30% | 311 | 28% |
| 96 | 60 | 4 | 1 | 2214 | 100% | 884 | 40% | 601 | 27% | 669 | 30% | 599 | 27% |
| 168 | 60 | 7 | 1 | 3871 | 100% | 1533 | 40% | 1033 | 27% | 1177 | 30% | 1032 | 27% |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table F.16 – Combination of the various encoding methods and V.44 compression for get-response APDU**

| Profile Generic Parameters | | | | Uncompressed | Profile Generic Compression with V.44 | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Records | Period | Interval | Values | Normal | Normal | | Compressed null-data | | compact array | | Compressed null-data & delat values | | Compressed null-data & delta values (const) | |
| Number | Min | Days | Number | Size | Size | Ratio | Size | Ratio | Size | Ratio | Size | Ratio | Size | Ratio |
| 24 | 60 | 1 | 1 | 558 | 164 | 29% | 111 | 20% | 105 | 19% | 82 | 15% | 34 | 6% |
| 48 | 60 | 2 | 1 | 1110 | 329 | 30% | 193 | 17% | 190 | 17% | 137 | 12% | 33 | 3% |
| 96 | 60 | 4 | 1 | 2214 | 560 | 25% | 377 | 17% | 352 | 16% | 255 | 12% | 34 | 2% |
| 168 | 60 | 7 | 1 | 3871 | 1146 | 30% | 676 | 17% | 649 | 17% | 458 | 12% | 34 | 1% |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# Annex G
(normative)

## NSA Suite B elliptic curves and domain parameters

NOTE This information is reproduced from NSA2.

Domain parameters $D$ for ECC schemes are of the form: $(q, FR, a, b\{, SEED\}, G, n, h)$, where $q$ is the field size; $FR$ is an indication of the basis used; $a$ and $b$ are two field elements that define the equation of the curve; $SEED$ is an optional bit string that is included if the elliptic curve was randomly generated in a verifiable fashion; $G$ is a generating point consisting of $(x_G, y_G)$ of prime order on the curve; $n$ is the order of the point $G$; and $h$ is the cofactor (which is equal to the order of the curve divided by $n$).

Suite B requires the use of one of the following two sets of domain parameters, see Table G.1 and Table G.2:

**Table G.1 – ECC_P256_Domain_Parameters**

| Parameter name | Symbol | Value |
|---|---|---|
| Field size | q | FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFF<br>$=(2^{224}(2^{32}-1)+2^{192}+2^{96}-1$ |
| Field representation indicator | FR | NULL |
| Curve parameter | a | FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFC |
| Curve parameter | b | 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B |
| Seed used to generate parameter b: | SEED | C49D3608 86E70493 6A6678E1 139D26B7 819F7E90 |
| x-coordinate of base point G | $x_G$ | 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296 |
| y-coordinate base point G | $y_G$ | 4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068 37BF51F5 |
| Order of point G | n | FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551 |
| Cofactor | h | 1 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Table G.2 – ECC_P384_Domain_Parameters**

| Parameter name | Symbol | Value |
|---|---|---|
| Field size | q | FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF 00000000 00000000 FFFFFFFF<br>$=2^{384}-2^{128}-2^{96}+2^{32}-1$ |
| Field representation indicator | FR | NULL |
| Curve parameter | a | FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF 00000000 00000000 FFFFFFFC |
| Curve parameter | b | B3312FA7 E23EE7E4 988E056B E3F82D19 181D9C6E FE814112 0314088F 5013875A C656398D 8A2ED19D 2A85C8ED D3EC2AEF |
| Seed used to generate parameter b: | SEED | A335926A A319A27A 1D00896A 6773A482 7ACDAC73 |
| x-coordinate of base point G | $x_G$ | AA87CA22 BE8B0537 8EB1C71E F320AD74 6E1D3B62 8BA79B98 59F741E0 82542A38 5502F25D BF55296C 3A545E38 72760AB7 |
| y-coordinate base point G | $y_G$ | 3617DE4A 96262C6F 5D9E98BF 9292DC29 F8F41DBD 289A147C E9DA3113 B5F0B8C0 0A60B1CE 1D7E819D 7A431D7C 90EA0E5F |
| Order of point G | n | FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF C7634D81 F4372DDF 581A0DB2 48B0A77A ECEC196A CCC52973 |
| Cofactor | h | 1 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# Annex H
(informative)

# Example of an End entity signature certificate using P-256 signed with P-256

## H.1 Fields of public key certificates

Table H.1 shows the fields of public key Certificates. For details, see 5.6.4.2.

**Table H.1 – Fields of public key Certificates using P-256 signed with P-256**

| Field | Value | Comments |
|---|---|---|
| *begin tbsCertificate* | | |
| Version | 2 | X.509 version 3 |
| Serial Number | INTEGER | Assigned by the CA, up to 20 octets |
| Signature | 1.2.840.10045.4.3.2 | ecdsa-with-SHA256 (same as signatureAlgorithm) |
| Issuer | CN, O, C | Distinguished name (DN) of the certificate issuer |
| Validity | Not before <date><br>Not after <date> | Validity of the Certificate. Follows RFC 5280. |
| Subject | CN, O, C | Distinguished name (DN) of the certificate subject |
| Subject Public Key Info | | |
| AlgorithmIdentifier | | |
| algorithm | 1.2.840.10045.2.1 | ECDSA and ECDH Public Key |
| parameters | 1.2.840.10045.3.1.7 | NIST P-256 named curve |
| subjectPublicKey | ECC public key<br>bit string 528 bits | 1st byte = 0, 2nd byte = 4 (uncompressed)<br>256 bit x, 256 bit y coordinates |
| Unique Identifier | | |
| subjectUniqueID | Bit string | Optional in end device certificates other than server certificates. |
| Extensions | | |
| **BasicConstraints**<br>Identifier<br>Value<br>Critical | <br>2.5.29.19<br>Boolean \| Integer<br>TRUE | Only required for CA certificates.<br><br>CA indication and maximum path length |
| **Subject Key Identifier**<br>Identifier<br>Value<br>Critical | <br>2.5.29.14<br>octet string<br>FALSE | Optional in end entity certificates<br><br>Follows RFC 5280, 8 or 20 octets. |
| **Authority Key Identifier**<br>Identifier<br>Value<br>Critical | <br>2.5.29.35<br>octet string<br>FALSE | Mandatory in end entity certificates<br><br>Follows RFC 5280, 8 or 20 octets. |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Field | Value | Comments |
|---|---|---|
| **Key Usage**<br>Identifier<br>Value<br>Critical | <br>2.5.29.15<br>DER encoded bit string<br>TRUE | Mandatory in all certificates.<br>Values according to RFC 5280, 4.2.1.3:<br>0 digitalSignature<br>4 keyAgreement<br>5 keyCertSign<br>6 cRLSign<br>At least one bit must be 1. |
| CertificatePolicies<br>Identifier<br>Value<br>Critical | <br>2.5.29.32<br>OID<br>FALSE | Optional in end entity certificates |
| Subject Alt Names<br>Identifier<br>Value<br>Critical | <br>2.5.29.17<br>To be assigned by the PKI or the CA<br>TRUE when Subject is absent. | Optional in end entity certificates |
| *end tbsCertificate* | | |
| signatureAlgorithm | 1.2.840.10045.4.3.2 | ecdsa-with-SHA256 |
| signatureValue | Bit string | Encoded bit string value of a DER encoded sequence of 2 integers; each a maximum of 33 bytes. |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## H.2 Example of a Root-CA Certificate using P-256 signed with P-256

This Clause shows an example Root-CA Certificate used to generate the end entity Certificate shown in Clause H.3.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 156 (0x9c)
    Signature Algorithm: ecdsa-with-SHA256
        Issuer: C=CH, O=DLMS USER ASSOCIATION, CN=Root_CA
        Validity
            Not Before: Jan  1 00:00:00 2010 GMT
            Not After: Dec 31 23:59:59 2049 GMT
        Subject: C=CH, O=DLMS USER ASSOCIATION, CN=Root_CA
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:09:5d:5d:4d:08:69:82:87:6a:a6:25:f6:c1:67:
                    c2:e3:a1:5f:e0:d0:4c:ac:15:c3:f8:66:75:d0:21:
                    0a:96:52:fb:3f:96:1c:b6:f3:99:3e:e1:fe:86:7d:
                    33:f3:b4:4b:9c:87:60:da:fb:6c:de:63:36:2c:05:
                    2c:34:2f:1b:f7
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:1
            X509v3 Subject Key Identifier:
                D3:72:AB:23:C7:E0:A4:2C:49:71:EA:B7:D1:B8:3F:20:AB:52:4F:CC
            X509v3 Key Usage: critical
```

```
            Certificate Sign, CRL Sign
    Signature Algorithm: ecdsa-with-SHA256
         30:46:02:21:00:aa:d4:0b:10:8d:98:b7:08:4b:a4:ab:9c:f4:
         95:f2:0f:39:c5:0d:1f:32:88:41:36:a7:1c:99:05:ea:67:3d:
         21:02:21:00:ad:f3:67:b0:66:d3:72:53:67:09:ad:26:3f:b8:
         4c:38:73:ae:d8:a7:fd:55:d8:15:14:d4:11:31:6a:20:ad:40
```

## H.3 Example of an end entity digital signature Certificate using P-256 signed with P-256

This clause shows an example end entity Certificate generated using the Root-CA certificate shown in Clause H.2.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 170 (0xaa)
    Signature Algorithm: ecdsa-with-SHA256
        Issuer: C=CH, O=DLMS USER ASSOCIATION, CN=Root_CA
        Validity
            Not Before: Jan  1 00:00:00 2010 GMT
            Not After: Dec 31 23:59:59 2049 GMT
        Subject: C=CH, O=DLMS USER ASSOCIATION, CN=4354543030303030
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:f4:a3:26:87:23:43:25:7f:57:00:b3:34:88:4b:
                    25:97:80:9c:5e:da:ef:0e:eb:b8:9a:1d:c7:5c:6d:
                    75:84:0f:4a:12:8b:15:21:5d:b9:3c:51:8c:dd:fe:
                    af:25:b2:9a:12:6d:00:e2:d1:98:66:e7:1b:aa:3e:
                    51:1b:82:86:81
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Authority Key Identifier:
                keyid:D3:72:AB:23:C7:E0:A4:2C:49:71:EA:B7:D1:B8:3F:20:AB:52:4F:CC

            X509v3 Key Usage: critical
                Digital Signature
    Signature Algorithm: ecdsa-with-SHA256
         30:46:02:21:00:96:fe:e8:56:dd:53:68:d9:c7:70:c0:da:d5:
         2f:4f:a4:97:12:9a:b7:2c:52:3a:35:66:0e:b6:91:e5:ce:97:
         89:02:21:00:8d:f5:9e:87:58:21:24:7b:ca:6f:18:a2:b3:34:
         aa:39:7b:b8:c0:c4:81:25:7a:f6:17:f5:bb:04:a3:8d:75:a9
```

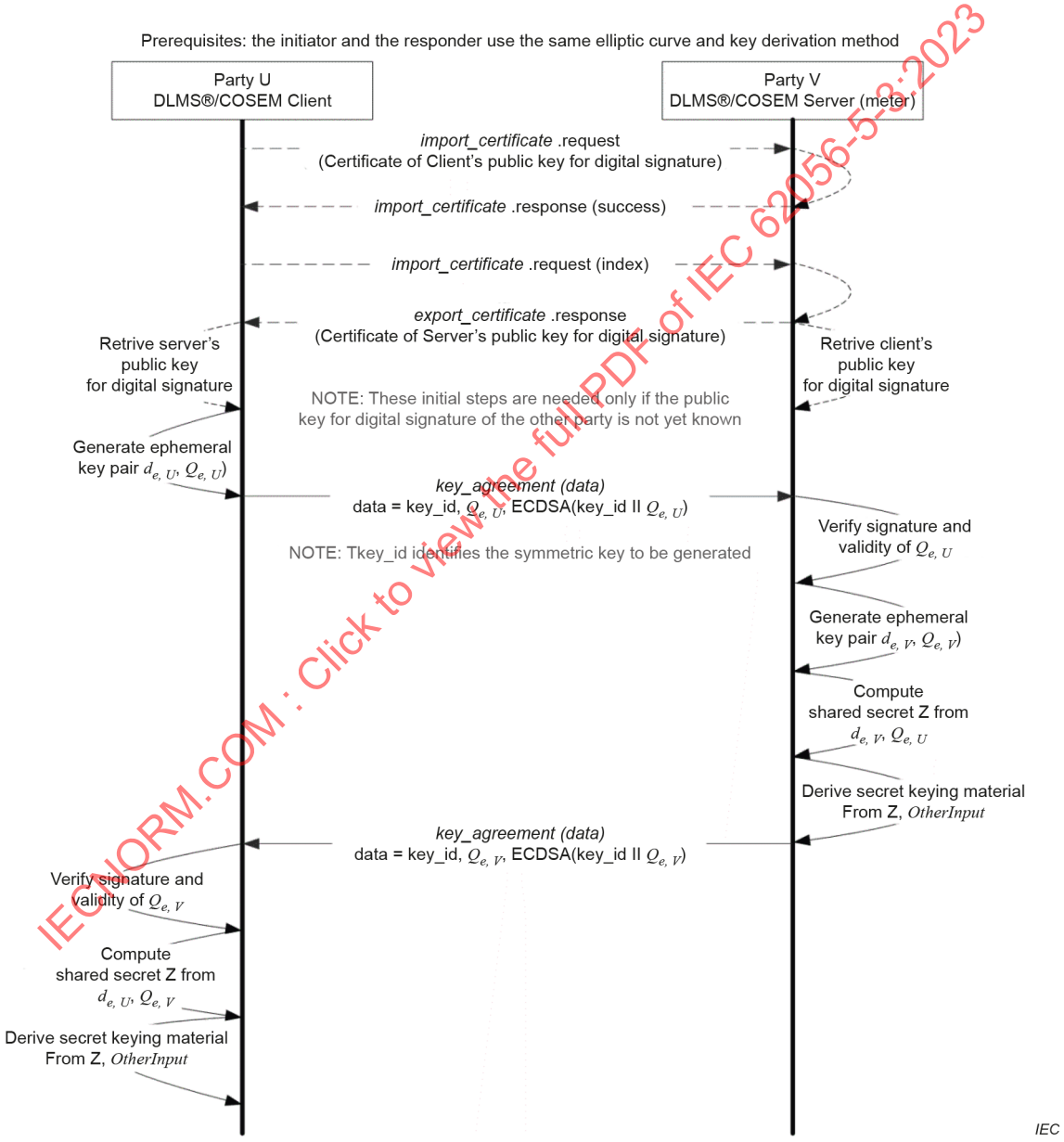
IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# Annex I
(normative)

# Use of key agreement schemes in DLMS®/COSEM

## I.1 Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme

Figure I.1 shows how the Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme – specified in 5.3.4.6.2 – is used in DLMS®/COSEM, by invoking the appropriate methods of the "Security setup" IC. See also 5.5.5.




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure I.1 – MSC for key agreement using the Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme**

The steps are the following (for details, please refer to NIST SP 800-56A Rev. 2: 2013, 6.1.2.2 and NSA2:2009, 3.1):

- Step 1 (optional): the client sends to the server the certificate of its public key for digital signature by invoking the *import_certificate* method;
- Step 2 (optional): the client retrieves from the server the certificate of the public key for digital signature by invoking the *export_certificate* method;
- Step 3: The client generates an ephemeral key pair ($d_{e,U}$, $Q_{e,U}$). It signs (key_Id, $Q_{e,U}$) with its private digital signature key and sends it to the server by invoking the *key_agreement* method;
- Step 4: The server verifies the signature and the validity of $Q_{e,U}$. It computes shared secret $Z$ from ($d_{e,v}$, $Q_{e,U}$) and derives the secret key from $Z$ and *OtherInput;*
- Step 5: If the key has been successfully derived the server generates then an ephemeral key pair ($d_{e,v}$, $Q_{e,v}$). It signs (key_id, $Q_{e,v}$) and sends it to the client in the response to the invocation of the *key_agreement* method;
- Step 6: The client computes shared secret $Z$ from ($d_{e,U}$, $Q_{e,V}$) and derives the secret key from $Z$ and *OtherInput.*

Table I.1 provides a test vector.

**Table I.1 – Test vector for key agreement using the Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme**

| Security material | Symbol | Contents | LEN(X) Bytes | LEN(X) Bits |
|---|---|---|---|---|
| Security Suite | | ECDH-ECDSA-AES-GCM-128-SHA-256 | | |
| Curve | | P-256 | | |
| Domain Parameters | *D* | See Annex G. | | |
| System Title Client | Sys-TC | 4D4D4D0000BC614E | 8 | 64 |
| System Title Server | Sys-TS | 4D4D4D0000000001 | 8 | 64 |
| Private Signing Key Client | Pri-KC | 418073C239FA6125011DE4D6CD2E645780289F761BB21BFB0835CB5585E8B373 | 32 | 256 |
| Private Signing Key Server | Pri-KS | AE55414FFE079F9FC95649536BD1C2B5653D200813727E07D501A8B550C69207 | 32 | 256 |
| Public Signing Key Client | Pub-KC | BAAFFDE06A8CB1C9DAE8D94023C601DBBB249254BA22EDD827E820BCA2BCC64362FBB83D86A82B87BB8B7161D2AAB5521911A946B97A284A90F7785CD9047D25 | 64 | 512 |
| Public Signing Key Server | Pub-KS | 933ACF15B03A9248E029B2787FB52A0AECAF635F07C42A0019FB3197E38F8F549A125EA36781B0CA96BE89A0E1FE2CF9B7361ED48B3C5E24592B9C0F4EDD31D1 | 64 | 512 |
| Ephemeral Public Key Client | Epub-KC | 2914D60E10AB705F62ED6CC349D7CB99B9AB3F3978E59278C7AF595B3AF987941372DAB6D5AF1FA867E134167E6F23DE664A6693E05F43414611058D1B48F894 | 64 | 512 |
| Ephemeral Public Key Server | Epub-KS | 95F41066009B185B074F5FFFF736B71C325FCADB2BC0CF1A4F4B17BBE7AB81D62946506BC8169C7B539B39A5D8463787F449C9BD2583FA67A1075B0DBFC638BA | 64 | 512 |
| Ephemeral Private Key Client | Epri-KC | 1BAC19FC1D52A1E5102622EDFA36584C05E12FA8CDEAA450F2F1E9A7DCCF7628 | 32 | 256 |
| Ephemeral Private Key Server | Epri-KS | 34A8C23A34DBB519D09B245754C85A6CFE05D14A063EFA5AA41545AA8241EFAE | 32 | 256 |

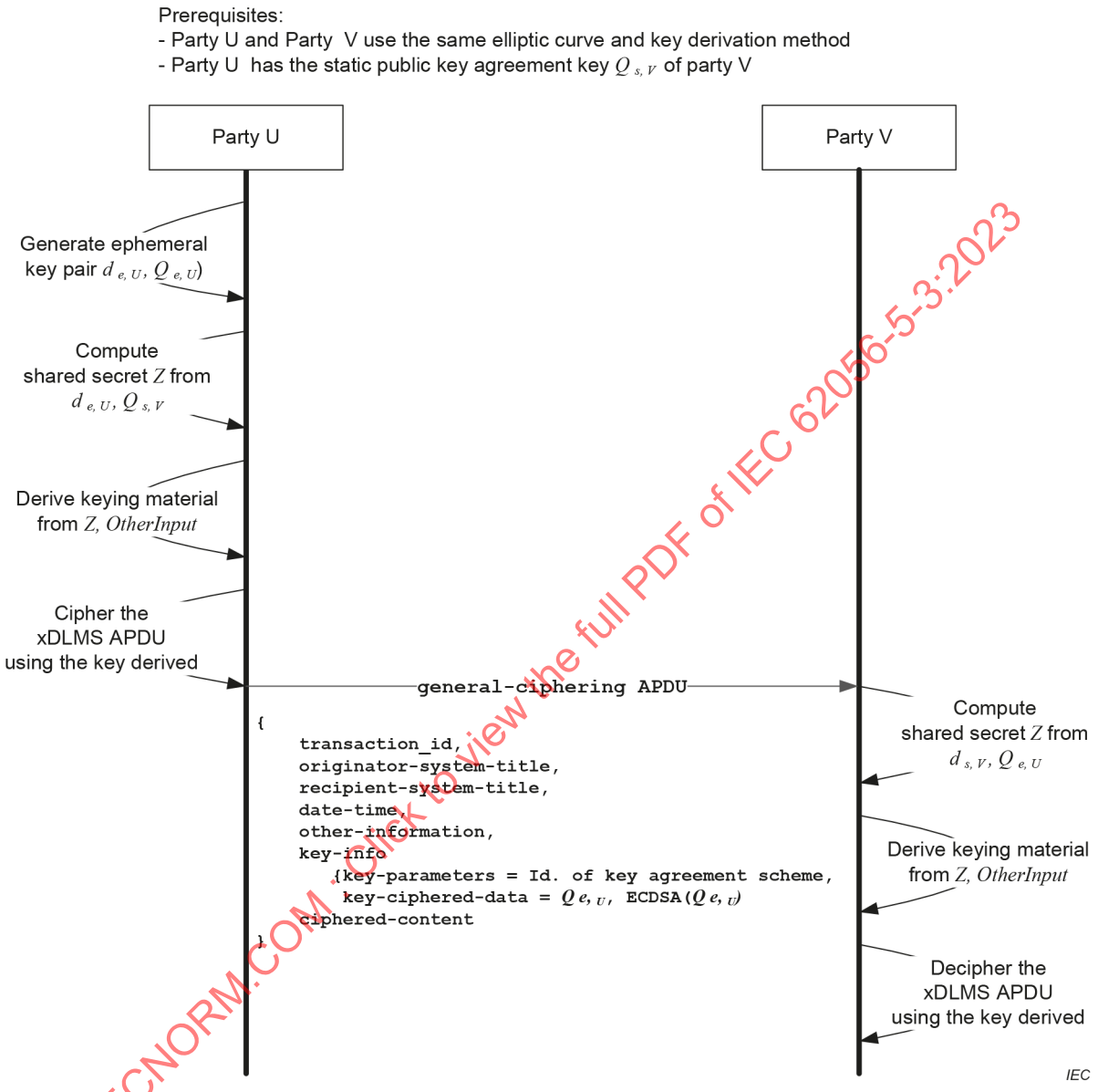
IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Security material | Symbol | Contents | LEN(X) Bytes | LEN(X) Bits |
|---|---|---|---|---|
| Ephemeral Public Key Signature Client | Epub-K-Sig-C | 06F0607702AA0E2435A183E2F6B1ECD19629712E389A213610C03F77B2590860EA840AF5C3FA1F2BCDF055D4744E9A01CE9A0E55026BCAA4EEBEB764CED64BB3<br>// The key_id \|\| Epub-KC are included in signature | 64 | 512 |
| Ephemeral Public Key Signature Server | Epri-K-Sig-S | A92995225CEE004ED4376057EEE9536E97EE6F5BAE43E59BDBBD515A89FB2CB83F2A270871A31B09338DCF0D1797466087908BA4A6ED8FD48B9EA067DA67DC4D<br>// The key_id \|\| Epub-KS are included in signature | 64 | 512 |
| key_agreement(data) Client | ACTION-Request | C30140<br>0040 00002B0000FF 03 // method id<br>01 // optional flag<br>0101 // array of 1<br>0202 // structure of 2<br>1600 // key_id = 0, global unicast encryption key<br>098180<br>2914D60E10AB705F62ED6CC349D7CB99B9AB3F3978E59278C7AF595B3AF987941372DAB6D5AF1FA867E134167E6F23DE664A6693E05F43414611058D1B48F894<br>// ephemeral public key client 64 bytes<br>06F0607702AA0E2435A183E2F6B1ECD19629712E389A213610C03F77B2590860EA840AF5C3FA1F2BCDF055D4744E9A01CE9A0E55026BCAA4EEBEB764CED64BB34<br>// ephemeral public key signature client 64 bytes | 150 | 1200 |
| global_key_agreement(data) Server | ACTION-Response | C70140<br>00 // success<br>01 // optional Get-Data-result present<br>00 // data CHOICE<br>0101 // array of 1<br>0202 // structure of 2<br>1600 // key id = 0<br>098180 // octet string 128 bytes<br>95F41066009B185B074F5FFFF736B71C325FCADB2BC0CF1A4F4B17BBE7AB81D62946506BC8169C7B539B39A5D8463787F449C9BD2583FA67A1075B0DBFC638BA<br>// ephemeral public key server 64 bytes<br>A92995225CEE004ED4376057EEE9536E97EE6F5BAE43E59BDBBD515A89FB2CB83F2A270871A31B09338DCF0D1797466087908BA4A6ED8FD48B9EA067DA67DC4D<br>// ephemeral public key signature server 64 bytes | 143 | 1144 |
| Shared Secret | Z | C1CF8FE7891AEF3617D7190795E61FE6C24EFC3CCA2E08469BAD1A225CE6EA08 | 32 | 256 |
| AlgorithmID | AlgID | 60857405080300 // AES-GCM-128 | 7 | 56 |
| KDF(Z,AlgID, Sys-TC,Sys-TS) | KDF | C5F4512846EDE51CFB8CCF59F08A694E002EDF66B4CB1739AC26A74E49712F46 | 32 | 256 |
| Global Unicast Encryption Key | GUEK | C5F4512846EDE51CFB8CCF59F08A694E | 16 | 128 |
| NOTE The values of the public keys are represented here as FE2OS(xp)II FE2OS(yp). | | | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## I.2 One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme

Figure I.2 shows how the One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme, specified in 5.3.4.6.3 is used in DLMS®/COSEM to protect an xDLMS APDU. See also 5.5.5.




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure I.2 – Ciphered xDLMS APDU protected by an ephemeral key established using the One-pass Diffie-Hellman (1e, 1s, ECC CDH) scheme**

The process is the following:

- Step 1: The originator, taking the role of the party U of the key agreement process generates an ephemeral key pair ($d_{e,U}$, $Q_{e,U}$);
- Step 2: It computes shared secret $Z$ from $d_{e,U}$, $Q_{s,V}$;
- Step 3: It derives the secret key from $Z$ and $OtherInfo$;
- Step 4: It ciphers the xDLMS APDU as required by the security policy in force and by the access rights, using the key derived;
- Step 5: It sends a general-ciphering APDU to the recipient. The use of the fields of the APDU shall be as follows (see also 5.3.4.6.5):

  - transaction-id: as required; not needed for the key derivation process;
  - originator-system-title: this is used as the *PartyUInfo* element of *OtherInfo;*
  - recipient-system-title: this is used as the *PartyVInfo* element of *OtherInfo;*
  - date-time: as required; not needed for the key derivation process;
  - other-information: as required; not needed for the key derivation process;
  - key-info:
    - key-parameters: Identifier of the key agreement scheme: 0x01, see Table 11;
    - key-ciphered-data = $Q_{e,\ U}$ signed by the digital signature private key of Party U;
  - ciphered-content: carries the ciphered xDLMS APDU that is protected using the key.
- Step 6: The recipient, taking the role of party V of the key agreement process computes shared secret $Z$ from $d_{s,\ V}$, $Q_{e,\ U}$;
- Step 7: It derives the secret key from $Z$ and *OtherInfo*;
- Step 8: It deciphers the xDLMS APDU using the key derived.

Table I.2 provides a test vector.

**Table I.2 – Test vector for key agreement using the One-pass Diffie-Hellman (1e, 1s, ECC CDH) scheme**

| Security material | Symbol | Contents | LEN(X) Bytes | LEN(X) Bits |
|---|---|---|---|---|
| Security Suite | | ECDH-ECDSA-AES-GCM-128-SHA-256 | | |
| Curve | | P-256 | | |
| Domain Parameters | *D* | See Annex G. | | |
| System Title Client | Sys-TC | 4D4D4D0000BC614E | 8 | 64 |
| System Title Server | Sys-TS | 4D4D4D0000000001 | 8 | 64 |
| Private Signing Key Client | Pri-KC | 418073C239FA6125011DE4D6CD2E6[4]5780289F761BB21BFB0885CB5585E8B373 | 32 | 256 |
| Private Signing Key Server | Pri-KS | AE55414FFE079F9FC95649536BD1C2B5653D200813727E07D501A8B550C69207 | 32 | 256 |
| Public Signing Key Client | Pub-KC | BAAFFDE06A8CB1C9DAE8D94023C601DBBB249254BA22EDD827E820BCA2BCC64362FBB83D86A82B87BB8B7161D2AAB5521911A946B97A284A90F7785CD9047D25 | 64 | 512 |
| Public Signing Key Server | Pub-KS | 933ACF15B03A9248E029B2787FB52A0AECAF635F07C42A0019FB3197E38F8F549A125EA36781B0CA96BE89A0E1FE2CF9B7361ED48B3C5E24592B9C0F4EDD31D1 | 64 | 512 |
| Private Key Agreement Key Client | Pri-AKC | A51C16FF5C498FCC89323D4A9267CD71BF81FD6F6A891CD240DA7F3D6F283E65 | 32 | 256 |
| Private Key Agreement Key Server | Pri-AKS | AAD3FD0732E991CF52A74C66C1F2827DDC53522A2E0A169D7C4FFCC0FB5D6A4D | 32 | 256 |
| Public Key Agreement Key Client | Pub-AKC | 07C56DE2DCAF0FD793EF29F019C89B4A0CC1E001CE94F4FFBE10BC05E7E66F7671A13FBCF9E662B9826FFF6A6938546D524ED6D3405F020296BDE16B04F7A7C2 | 64 | 512 |
| Public Key Agreement Key Server | Pub-AKS | A653565B0E06070BAE9FBE140A5D2156812AEE2DD525053E3EFC850BF13BFDFFCB240BC7B77BFF5883344E7275908D2287BEFA3725017295A096989D2338290B | 64 | 512 |
| Ephemeral Public Key Client | Epub-KC | C323C2BD45711DE4688637D919F92E9DB8FB2DFC213A88D21C9DC8DCBA917D8170511DE1BADB360D50058F794B0960AE11FA28D392CFF907A62D13E3357B1DC0 | 64 | 512 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Security material | Symbol | Contents | LEN(X) Bytes | LEN(X) Bits |
|---|---|---|---|---|
| Ephemeral Public Key Server | Epub-KS | 6439724714B47CD9CB988897D8424AB946DCD083D37A95463<br>7616011B9C2378773295F0F850D8DAFD1BBE9FE666E53E4F0<br>97CD10B38B69622152724A90987444 | 64 | 512 |
| Ephemeral Private Key Client | Epri-KC | 47DAB03842E5B6E74828EF4F449B378D7DD1A5DAE1FFCA5AE<br>0B0BE0AD18EC57E | 32 | 256 |
| Ephemeral Private Key Server | Epri-KS | 819B1BEACC955E29139E368BF4119C126FF799EE16BCBA3F4<br>5C1EF16749BCB95 | 32 | 256 |
| Ephemeral Public Key Signature Client | Epub-K-Sig-C | B51BE089D0B682863B2217201E73A1A9031968A9B4121DCBC<br>3281A69739AF87429F5B3AC5471E7B6A04A2C0F2F8A25FD77<br>2A317DF97FC5463FEAC248EB8AB8BE<br>// Epub-KC is included in signature | 64 | 512 |
| Ephemeral Public Key Signature Server | Epri-K-Sig-S | E1FF47974A1F6931A6502F58147463F0E8CC517D47F55B0AC<br>56DD8AC5C9D0E481934F2D90F9893016BD82B6E3FFE21FF15<br>88F3278B4E9D98EB4FB62ADD64B380<br>// Epub-KS are included in signature | 64 | 512 |
| general-ciphering (Access-Request) | GC-C | DD<br>080102030405060708 // transaction-id<br>084D4D4D0000BC614E // originator-system-title<br>084D4D4D0000000001 // recipient-system-title<br>00 // date-time not present<br>00 // other-information not present<br>01 // optional flag<br>02 // agreed-key CHOICE<br>0101 // key-parameters<br>8180C323C2BD45711DE4688637D919F92E9DB8FB2DFC213A8<br>8D21C9DC8DCBA917D8170511DE1BADB360D50058F794B0960<br>AE11FA28D392CFF907A62D13E3357B1DC0B51BE089D0B6828<br>63B2217201E73A1A9031968A9B4121DCBC3281A69739AF874<br>29F5B3AC5471E7B6A04A2C0F2F8A25FD772A317DF97FC5463<br>FEAC248EB8AB8BE // key-ciphered-data<br>81EB3100000000F435B9679270C5BF4425EE5777402A6C8D<br>51C620EED52DBB1B878B836E2857D5C053E6DDF27FA87409<br>AEF502CD9618AE47017C010224FD109CC0BEB21E742D44AB4<br>0CD11908743EC90EC8C40E221D517F72228E1A26E827F43DC<br>18ED27B5F458D66508B05A2A4CC6FED178C881AFC3BC67064<br>689BE8BB1C80ABB3C114A31F4CB03B8B64C7E0B4CE77B239<br>9C93347858888F92239713B38DF01C4858245827A92EF3341<br>72EA636B31CBBDF2A96AD5D035F66AA38F1A2D97D4BBA9962<br>2E6B5F18789CECB2DFB3937D9F3E17F8B472098E6563238F3<br>7528374809836002AEA6E7012D2ADFAA7 // ciphered-content | 401 | 3208 |
| general-ciphering(Access-Response) | GC-S | DD<br>080123456789012345 // transaction-id<br>084D4D4D0000000001 // originator-system-title<br>084D4D4D0000BC614E // recipient-system-title<br>00 // date-time not present<br>00 // other-information not present<br>01 // optional flag<br>02 // agreed-key CHOICE<br>0101 // key-parameters<br>81806439724714B47CD9CB988897D8424AB946DCD083D37A9<br>54637616011B9C2378773295F0F850D8DAFD1BBE9FE666E53<br>E4F097CD10B38B69622152724A90987444E1FF47974A1F693<br>1A6502F58147463F0E8CC517D47F55B0AC56DD8AC5C9D0E48<br>1934F2D90F9893016BD82B6E3FFE21FF1588F3278B4E9D98E<br>B4FB62ADD64B380 // key-ciphered-data<br>3D3100000000B3FFCAA594642D8319CEC6B2A233E2BF4621D<br>6991B97E4565B986E8CCBE9A299D8E7869723638FF6BB20E6<br>6E175E6F2D762CFD26B3D58733 // ciphered-content | 226 | 1808 |
| Shared Secret GC-C | Z-GC-C | 0D4385BA0DD756CBCAB9887EB538396EE8F090A14C1079B43<br>59F115B977F4615 | 32 | 256 |
| AlgorithmID GC-C | AlgID-GC-C | 60857405080300 // AES-GCM-128 | 7 | 56 |
| KDF(Z,AlgID, Sys-TC, Sys-TS) GC-C | KDF-GC-C | 59A71FD81C929A86A99438DA17A66C058C6A93FD3065F5EE1<br>6A05D775927659B | 32 | 256 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Security material | Symbol | Contents | LEN(X) Bytes | LEN(X) Bits |
|---|---|---|---|---|
| Encryption Key GC-C | EK-GC-C | 59A71FD81C929A86A99438DA17A66C05 | 16 | 128 |
| Shared Secret GC-S | Z-GC-S | 2B4302DC49790E2E78D990CFB52ED6E2F273DECE441A2D95E4301B93812A9FAC | 32 | 256 |
| AlgorithmID GC-S | AlgID-GC-S | 60857405080300 | 7 | 56 |
| KDF(Z,AlgID, Sys-TS, Sys-TC) GC-S | KDF-GC-S | F0184BDA9466BFA4601A64A7EF46504AB1A40C4851A0A644503599DF298B2E14 | 32 | 256 |
| Encryption Key GC-S | EK-GC-S | F0184BDA9466BFA4601A64A7EF46504A | 16 | 128 |
| NOTE The values of the public keys are represented here as FE2OS(xp)II FE2OS(yp). | | | | |

## I.3 Static Unified Model C(0e, 2s, ECC CDH) scheme

Figure I.3 shows how Static Unified Model C(0e, 2s, ECC CDH) schemes specified in 5.3.4.6.4 is used in DLMS®/COSEM to protect an xDLMS APDU. See also 5.5.5.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

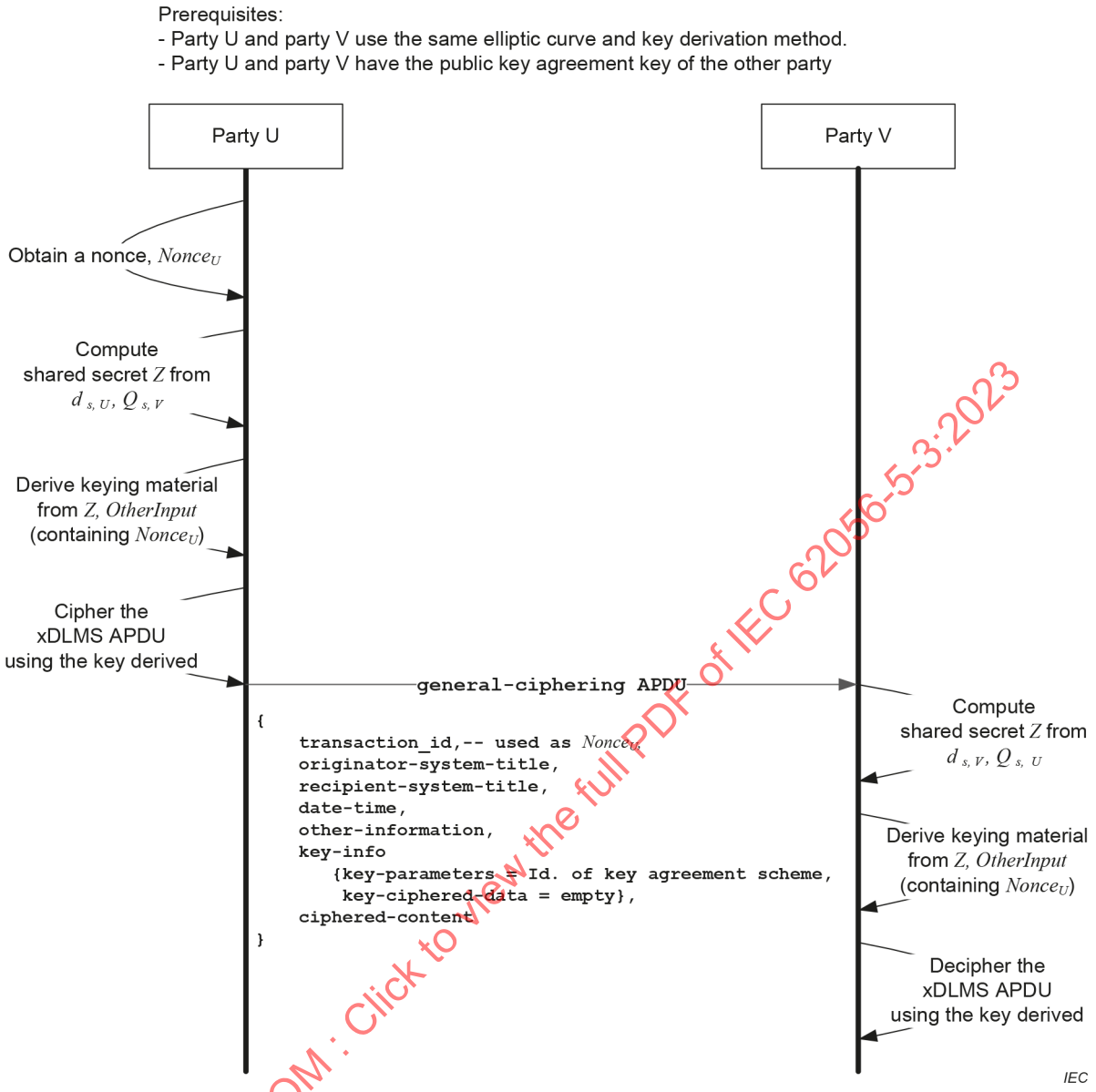



**Figure I.3 – Ciphered xDLMS APDU protected by an ephemeral key established using the Static Unified Model C(0e, 2s, ECC CDH) scheme**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

The process is the following:

- Step 1: The originator, taking the role of the Party U of the key agreement process obtains a nonce, $Nonce_U$;
- Step 2: It computes shared secret $Z$ from $d_{s,U}$, $Q_{s,V}$;

  NOTE See also Note to Table 6.
- Step 3: It derives the secret key from $Z$, and $OtherInput$ that contains $Nonce_U$;
- Step 4: It ciphers the xDLMS APDU APDU as required by the security policy in force and by the access rights, using the key derived;
- Step 5: It sends a general-ciphering APDU to the recipient. The use of the fields of the APDU shall be as follows (see also 5.3.4.6.5):
  - transaction-id: this field is used as $Nonce_U$;
  - originator-system-title: this is used as the $PartyUInfo$ element of $OtherInfo$;

  - recipient-system-title: this is used as the *PartyVInfo* element of *OtherInfo;*
  - date-time: as required; not needed for the key derivation process;
  - key-info:
  - key-parameters: Identifier of the key agreement scheme: 0x02, see Table 11.
  - key-ciphered-data = empty;
  - ciphered-content carries the ciphered xDLMS APDU that is protected using the key.
- Step 6: The recipient, taking the role of party V of the key agreement process computes shared secret $Z$ from $d_{s,\ V}$, $Q_{s,\ U}$;
- Step 7: It derives the secret key from $Z$ and *OtherInput* that contains $Nonce_U$;
- Step 8: It deciphers the xDLMS APDU using the key derived.

Table I.3 provides a test vector.

**Table I.3 – Test vector for key agreement using the Static-Unified Model (0e, 2s, ECC CDH) scheme**

| Security material | Symbol | Contents | LEN(X) Bytes | LEN(X) Bits |
|---|---|---|---|---|
| Security Suite | | ECDH-ECDSA-AES-GCM-128-SHA-256 | | |
| Curve | | P-256 | | |
| Domain Parameters | *D* | See Annex G. | | |
| System Title Client | Sys-TC | 4D4D4D0000BC614E | 8 | 64 |
| System Title Server | Sys-TS | 4D4D4D0000000001 | 8 | 64 |
| Private Key Agreement Key Client | Pri-AKC | A51C16FF5C498FCC89323D4A9267CD71BF81FD6F6A891CD240DA7F3D6F283E65 | 32 | 256 |
| Private Key Agreement Key Server | Pri-AKS | AAD3FD0732E991CF52A74C66C1F2827DDC53522A2E0A169D7C4FFCC0FB5D6A4D | 32 | 256 |
| Public Key Agreement Key Client | Pub-AKC | 07C56DD2DCAF0FD793EF29F019C89B4A0CC1E001CE94F4FFBE10BC05E7E66F7671A13FBCF9E662B9826FFF6A6938546D524ED6D3405F020296BDE16B04F7A7C2 | 64 | 512 |
| Public Key Agreement Key Server | Pub-AKS | A653565B0E06070BAE9FBE140A5D2156812AEE2DD525053E3EFC850BF13BFDFFCB240BC7B77BFF5883344E7275908D2287BEFA3725017295A096989D2338290B | 64 | 512 |
| Nonce Client | Nonce-C | 080102030405060708<br><br>// Nonce-C is transaction-id with length | 9 | 72 |
| Nonce Server | Nonce-S | 080123456789012345<br><br>// Nonce-S is transaction-id with length | 9 | 72 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Security material | Symbol | Contents | LEN(X) Bytes | LEN(X) Bits |
|---|---|---|---|---|
| general-ciphering (Access-Request) | GC-C | DD<br>080102030405060708 // transaction-id<br>084D4D4D0000BC614E // originator-system-title<br>084D4D4D0000000001 // recipient-system-title<br>00 // date-time not present<br>00 // other-information not present<br>01 // optional flag<br>02 // agreed-key CHOICE<br>0102 // key-parameters<br>00 // key-ciphered-data not present<br>81EB3100000000607581D83815281B561904E6A72B83BF3FB<br>0B1F2A0A23A82A804B39911F6CB1B4EF9B6F76C6338ED0580<br>14FFBF4A13A162E0EED11EEB8F597757A18215F28E1D3FC2D<br>44586C4B8AFE4500B535B579506CDB925CBEFF7F1CF6BF96C<br>583B9CF588FE8F6B01A574824D7CEC597F057FFA8700AF12A<br>D63A7FA72040439F4392C089B265F9EB1AC308239906DC04E<br>1A8712ABE383CF92349842EBA166EF2E9EB2F53D0DBA025D7<br>9875463398BBC3007BC4A6FC12780C21EE1ABF080BF0FA926<br>242834C0AC30D55A1EF8856E7DED48621F17FDF4D5B54EDDA<br>DD874119251049B6AEE111C12FBC2FEAF // ciphered-content | 272 | 2176 |
| general-ciphering (Access-Response) | GC-S | DD<br>080123456789012345 // transaction-id<br>084D4D4D0000000001 // originator-system-title<br>084D4D4D0000BC614E // recipient-system-title<br>00 // date-time not present<br>00 // other-information not present<br>01 // optional flag<br>02 // agreed-key CHOICE<br>0102 // key-parameters<br>00 // key-ciphered-data not present<br>3D31000000003CF971CD09E0B1C5B89E7BB52C1B39923D14C<br>4A160D7DDB3F2DE8AD255C625F407F04031CD95F4F261E23E<br>823B73490CD6CB593A140CD05F ciphered-content | 97 | 776 |
| Shared Secret GC-C | Z-GC-C | B1455B2AD5F68BCFFE6AD5412BA89548ACA7E0CBF4B1560D6<br>A57496F15E931AD | 32 | 256 |
| AlgorithmID GC-C | AlgID-GC-C | 60857405080300 // AES-GCM-128 | 7 | 56 |
| KDF(Z,AlgID, Sys-TC, Nonce-C, Sys-TS) GC-C | KDF-GC-C | 56C46B57DF675515C31025455822514AFA2CDEB3E0BF1CADA<br>84576159E84DE7E | 32 | 256 |
| Encryption Key GC-C | EK-GC-C | 56C46B57DF675515C31025455822514A | 16 | 128 |
| Shared Secret GC-S | Z-GC-S | B1455B2AD5F68BCFFE6AD5412BA89548ACA7E0CBF4B1560D6<br>A57496F15E931AD | 32 | 256 |
| AlgorithmID GC-S | AlgID-GC-S | 60857405080300 | 7 | 56 |
| KDF(Z,AlgID, Sys-TS, Nonce-S, Sys-TC) GC-S | KDF-GC-S | FC1314F7DE033B7DD19C80DBCF9FF2C5286DC8F76E87877BD<br>90B9B7F00CD5613 | 32 | 256 |
| Encryption Key GC-S | EK-GC-S | FC1314F7DE033B7DD19C80DBCF9FF2C5 | 16 | 128 |
| NOTE The values of the public keys are represented here as FE2OS(xp)II FE2OS(yp). | | | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# Annex J
(informative)

# Exchanging protected xDLMS APDUs between TP and server

## J.1 General

This use case shows exchanging protected xDLMS APDUs between a third party (TP) and a server via a client.

## J.2 Example 1: Protection is the same in the two directions

In the first example, the security policy of the server requires that the request is digitally signed and authenticated and the response is also digitally signed and authenticated.

In the .request, the digital signature is applied by the TP and the authentication is applied by the client:

- the TP sends the .request to the client in a general-signing APDU for the server: the recipient-system-title is that of the server;
- the client verifies the digital signature and if correct, encapsulates the general-signing APDU in a general-ciphering APDU.

The server checks and removes first the authentication and then the digital signature.

If both are correct, it prepares the .response APDU. The protection is applied to the same parties as in the request:

- the server first encapsulates the .response APDU in a general-signing APDU for the TP: the destination-system-title is that of the TP;
- it encapsulates then this general-signing APDU in a general-ciphering APDU for the client: the destination-system-title is that of the client.

The protection to be applied by each party is subject to project specific companion specifications, but the overall protection shall meet the security policy configured in the server. For example, the server would accept any of the following:

– digital signature applied by the TP and authentication applied by the client;
– authentication applied by the TP and digital signature applied by the client;
– both digital signature and authentication applied by the client;
– both digital signature and authentication applied by the TP.

The process is shown in Figure J.1.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

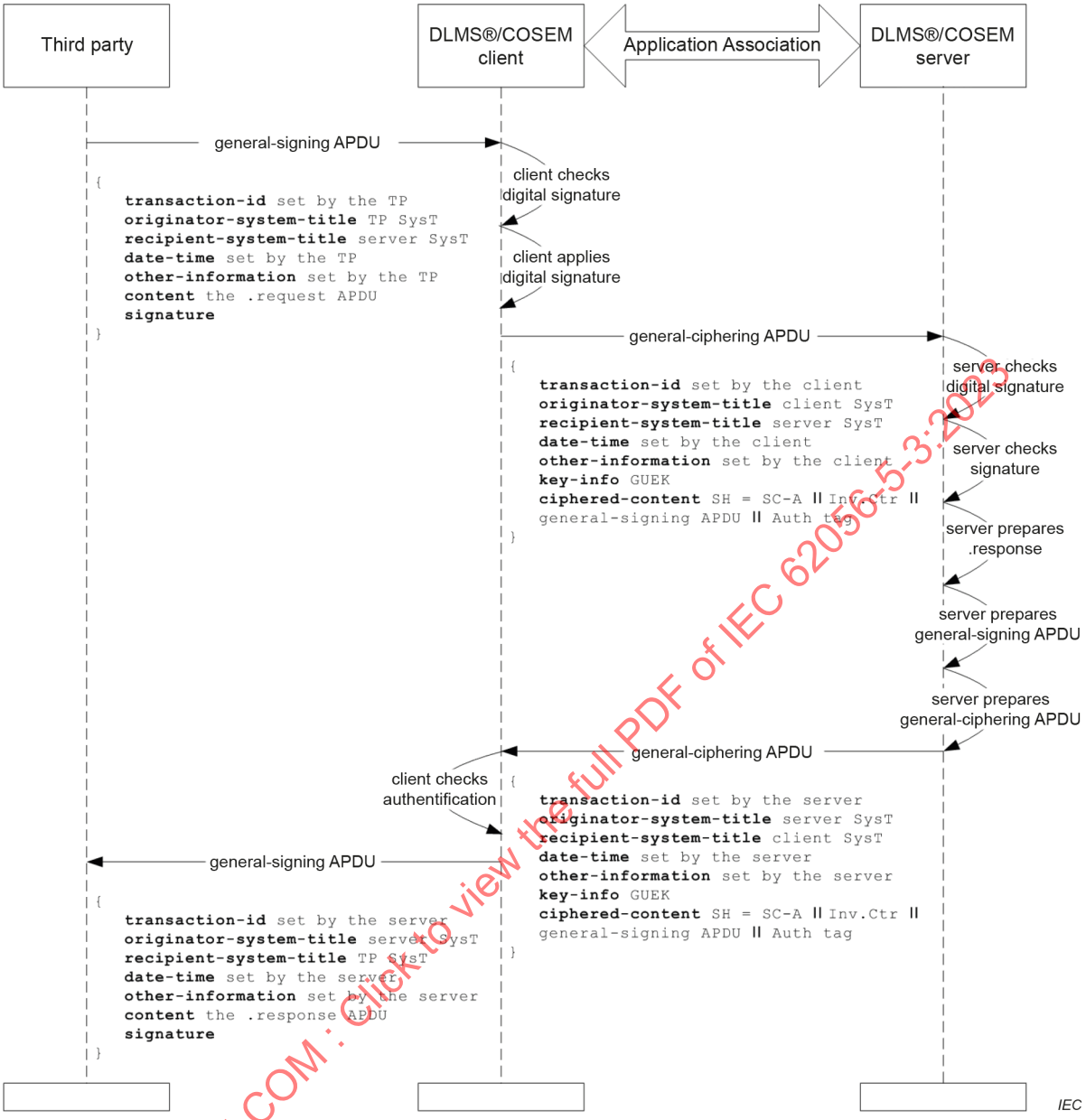



**Figure J.1 – Exchanging protected xDLMS APDUs between TP and server: example 1**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## J.3 Example 2: Protection is different in the two directions

In the second example, the security policy of the server requires that the request is digitally signed and authenticated and the response is only authenticated.

In the .request, the digital signature is applied by the TP and the authentication is applied by the client:

- the TP sends the .request to the client in a general-signing APDU for the server: the recipient-system-title is that of the server;
- the client verifies the digital signature and if correct, encapsulates the general-signing APDU in a general-ciphering APDU.

The server checks and removes first the authentication tag and then the digital signature.

If both are correct, it prepares the .response APDU. The protection is applied to the same parties as in the request:

- the server first encapsulates the .response APDU in a general-ciphering APDU for the TP: the destination-system-title is that of the TP, but no protection is applied: in the Security Control Byte, the bits indicating authentication and encryption are set to 0;
- it encapsulates then this general-ciphering APDU in a general-ciphering APDU for the client: the destination-system-title is that of the client.

The process is shown in Figure J.2.

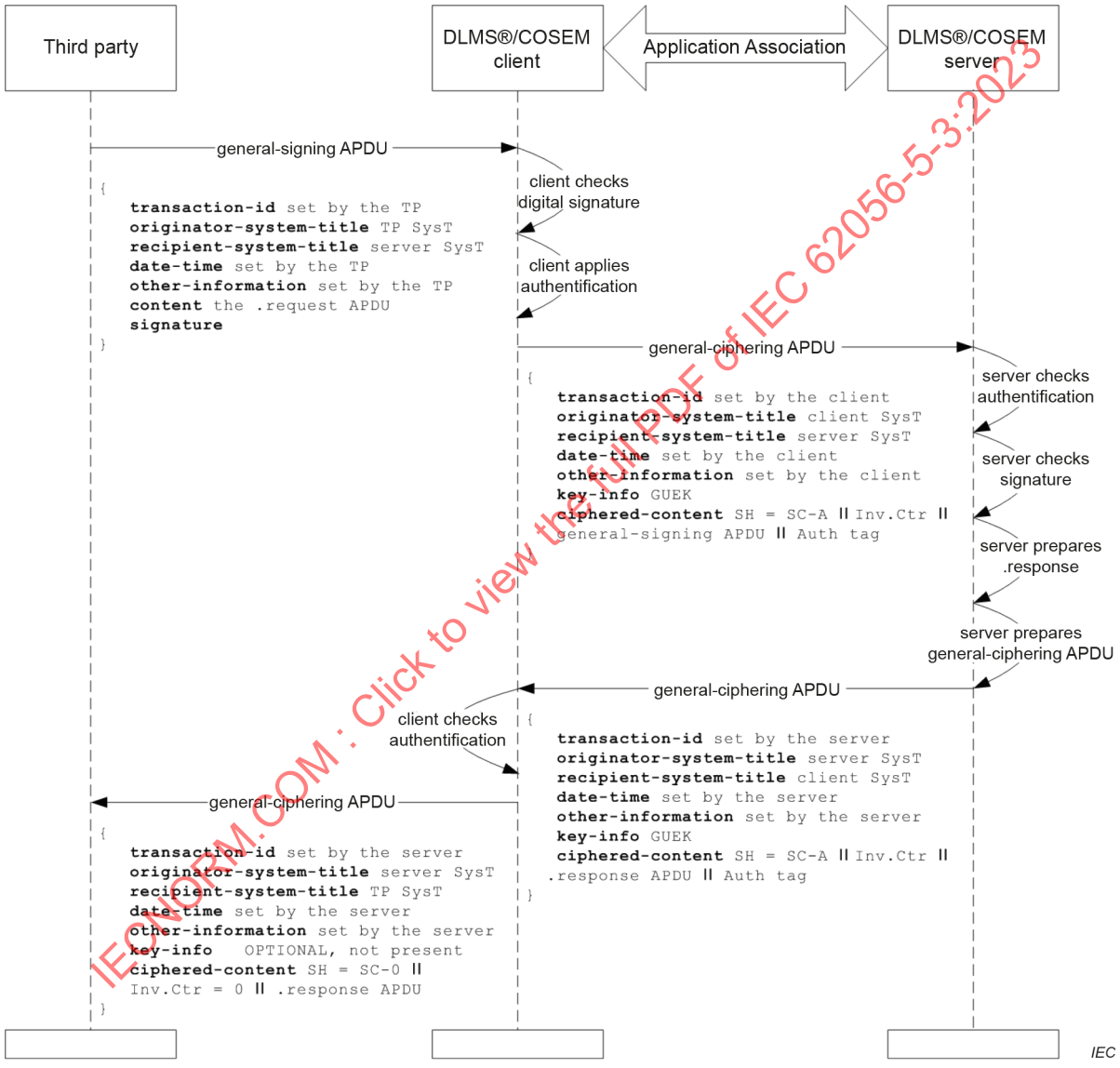



**Figure J.2 – Exchanging protected xDLMS APDUs between TP and server: example 2**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# Annex K
(informative)

## Significant technical changes with respect to IEC 62056-5-3:2017

This fourth edition of IEC 62056-5-3 includes the following significant technical changes with respect to the third edition of IEC 62056-5-3:2017.

| Item | Clause | |
|---|---|---|
| 1. | 2 | New references added. |
| 2. | 3.1 | General DLMS®/COSEM definitions added |
| 3. | 3.2 | Definitions related to cryptographic security added |
| 4. | 3.4 | General abbreviations: new abbreviations added |
| 5. | 3.3 | Definitions, abbreviations, symbols and notation relevant for the Galois/Counter Mode brought here from Green Book Ed.7 9.2.4.8.2 |
| 6. | 3.6 | Definitions, abbreviations, symbols and notation relevant for the ECDSA algorithm added. |
| 7. | 3.7 | Definitions, abbreviations, symbols and notation relevant for the key agreement algorithms added |
| 8. | 4.1 | Specifies the general concepts of information exchange in DLMS®/COSEM, extending IEC 62056-5-3 Ed.3.0:2017, Clause 4. |
| 9. | 4.1.1 | General: Text updated, key characteristics of DLMS®/COSEM listed here. |
| 10. | 4.1.2 | Communication model: new text introducing APs, AEs, ASEs, AAs and their relationship. Client/server model is explained here. |
| 11. | 4.1.3 | Naming and addressing: new text bringing all related elements together, and bringing in the concept of data exchange with third parties. Table of SAPs added. |
| 12. | 4.1.3.4 | System title: IEC 62056-5-3 Ed. 3.0:2017, 5.4.8.3.4.6 on exchanging system titles brought here. |
| 13. | 4.1.3.6 | Client user identification added. |
| 14. | 4.1.4 | Connection oriented operation explained. |
| 15. | 4.1.5 | Application associations. New text, bringing all related elements together. |
| 16. | 4.1.6 | Messaging patterns: new text bringing in the concept of push operation. |
| 17. | 4.1.7 | Data exchange between third parties and DLMS®/COSEM servers: new text. |
| 18. | 4.1.8 | Communication profiles added. |
| 19. | 4.1.9 | Model of a DLMS®/COSEM metering system |
| 20. | 4.1.10 | Model of DLMS®/COSEM servers added. |
| 21. | 4.1.11 | Model of DLMS®/COSEM clients added. |
| 22. | 4.1.12 | Interoperability and interconnectivity in COSEM added. |
| 23. | 4.1.13 | Ensuring interconnectivity: the protocol identification service added. |
| 24. | 4.1.14 | System integration and meter installation added. |
| 25. | 4.2 | The whole clause has been revised to present the additional services and mechanisms compared to DLMS® as specified in IEC 61334-4-41:1996. |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Item | Clause | |
|---|---|---|
| 26. | 5 | Information security in DLMS®/COSEM: The whole clause has been reworked to describe the extended security features of DLMS®/COSEM. Main elements:<br>– the DLMS®/COSEM security concept is presented;<br>– not only xDLMS APDUs but also COSEM data – carried by the APDUs – can be protected;<br>– the protection can be applied not only between clients and servers but also between third parties and servers via clients;<br>– symmetric and public key algorithms are available to provide any combination of authentication, encryption and digital signature;<br>– multiple layers of protection can be applied and verified by multiple entities;<br>– key-transport has been complemented by key agreement;<br>– compression (managed together with symmetric key algorithms) added. |
| 27. | 6.2 | The COSEM-OPEN service: Service parameters and their use updated to support client user identification and transportation of Certificates. |
| 28. | 6.5 | Protection and general block transfer parameters: clause reworked to cover the new general protection and general block transfer mechanisms and related parameters. |
| 29. | 6.6, 6.7 | GET, SET, ACTION service: either the service-specific or the general block transfer mechanism can be used. |
| 30. | 6.8 | The ACTION service: Text on method invocation parameters – use of null-data – precised. |
| 31. | 6.9 | The ACCESS service: the new unified service added, with the main features presented. |
| 32. | 6.10 | The DataNotification service: The new service added. |
| 33. | 6.14, 6.15 | Read, Write service: Either the service-specific or the general block transfer mechanism can be used. |
| 34. | 6.16 | UnconfirmedWrite service: General block transfer mechanism can be used. |
| 35. | 6.19 | Summary of services and LN/SN data transfer service mapping: New services added. |
| 36. | 7.1.1, 7.1.2 | State definitions: Updated to include the new services and APDUs. |
| 37. | 7.2.1 | ACSE functional units, services and service parameters: Updated to be in line with ISO/IEC 15953:1999 and ISO/IEC 15954:1999 replacing ISO/IEC 8649 and ISO/IEC 8650. |
| 38. | 7.2.2.3 | The COSEM authentication mechanism name: New mechanisms names added. |
| 39. | 7.2.2.4 | Cryptographic algorithm ID-s: Added, these IDs are used in the KDF function. |
| 40. | 7.2.3 | Protocol for the ACCESS service: Encoding of the ACSE APDUs, the xDLMS APDUs and XML. |
| 41. | 7.2.4 | Protocol for the establishment of confirmed application associations: text amended to include parsing of the new parameters. |
| 42. | 7.3.1 | Negotiation of services and options – the conformance block: New elements added. |
| 43. | 7.3.3, 7.3.4, 7.3.5 | Protocol of the GET, SET, and ACTION service: It is specified that either the service-specific or the general block transfer mechanism can be used. |
| 44. | 7.3.6 | Protocol for the ACCESS service added. |
| 45. | 7.3.7 | Protocol of the DataNotification service added. |
| 46. | 7.3.9, 7.3.10 | Protocol of the Read and Write service: It is specified that either the service-specific or the general block transfer mechanism can be used. |
| 47. | 7.3.13 | Protocol of general block transfer mechanism added. |
| 48. | 8 | Abstract syntax of the new services and APDUs added. |
| 49. | 9 | COSEM APDU XML schema added. |
| 50. | Annex B | SMS short wrapper added. |
| 51. | Annex C | Gateway protocol added. |
| 52. | F.2 | Example for the ACCESS service added. |
| 53. | F.3 | Compact array example added. |
| 54. | Annex G | NSA Suite B elliptic curves and domain parameters added. |
| 55. | Annex H | Certificate examples added. |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Item | Clause | |
|---|---|---|
| 56. | Annex I | Use of key agreement schemes in DLMS®/COSEM added. |
| 57. | Annex J | Exchanging protected xDLMS APDUs between TP and server added. |
| 58. | Bibliography | New references added. |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# Bibliography

The DLMS® UA "Books" are available for the members of the DLMS® User Association. See www.dlms.com

DLMS UA 1000-1, the "Blue Book" Ed. 14.0: 2020, *COSEM interface classes and OBIS identification system*

DLMS UA 1000-1, the "Blue Book" Ed. 13.0: 2019, *COSEM interface classes and OBIS identification system*

DLMS UA 1000-1, the "Blue Book" Ed. 12.0: 2017, *COSEM interface classes and OBIS identification system*

DLMS UA 1000-1, the "Blue Book" Ed. 11.0: 2013, *COSEM interface classes and OBIS identification system*

DLMS UA 1000-1, the "Blue Book" Ed. 12.0: 2014, *COSEM interface classes and OBIS identification system*

DLMS UA 1000-2, the "Green Book" Ed. 10.0 Amendment 1.0:2021, *DLMS®/COSEM Architecture and Protocols*

DLMS UA 1000-2, the "Green Book" Ed. 10.0:2020, *DLMS®/COSEM Architecture and Protocols*

DLMS UA 1000-2, the "Green Book" Ed. 9.01:2019, *DLMS®/COSEM Architecture and Protocols*

DLMS UA 1000-2, the "Green Book" Ed. 8.3:2017, *DLMS®/COSEM Architecture and Protocols*

DLMS UA 1000-2, the "Green Book" Ed. 7.0:2009, *DLMS®/COSEM Architecture and Protocols*

DLMS UA 1000-2, the "Green Book" Ed. 7.0, Amendment 3: 2013, *DLMS®/COSEM Architecture and Protocols, (cancels and replaces Amendment 1 and 2)*

DLMS UA 1000-2, the "Green Book" Ed. 8.1:2015, *DLMS®/COSEM Architecture and Protocols*

DLMS UA 1001-1 the "Yellow Book", Ed. 4.0:2007, *DLMS®/COSEM Conformance test and certification process*

DLMS UA 1002, the "White Book", Ed. 1.0:2003, *COSEM Glossary of terms*

ISO 3166 (all parts), *Codes for the representation of names of countries and their subdivisions*

ISO/IEC 8802-2:1998, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 2: Logical link control*

ISO/IEC 9545:1994, *Information technology – Open Systems Interconnection – Application layer structure*

ISO/IEC 9798-1:2010, *Information technology – Security techniques – Entity authentication – Part 1: General Evaluation of ISO/IEC 9798 Protocols Version 2.0 David Basin and Cas Cremers April 7, 2011*

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

ISO/IEC 9798-2:2008, *Information technology – Security techniques – Entity authentication – Part 2: Mechanisms using symmetric encipherment algorithms*

ISO/IEC 9798-3:1998, *Information technology – Security techniques – Entity authentication – Part 3: Mechanisms using digital signature techniques*

ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

ISO/IEC 13239:2002, *Information technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures*

ISO/IEC 15945:2002, *Information technology – Security techniques – Specification of TTP services to support the application of digital signatures*

ITU-T V.44:2000, *SERIES V: DATA COMMUNICATION OVER THE TELEPHONE NETWORK – Error control – V.44:2000, Data compression procedures*

ITU-T X.211:1995, *Information technology – Open Systems Interconnection – Physical service definition*

ITU-T X.811:1995, *Information technology – Open Systems Interconnection – Security frameworks for open systems: Authentication framework*

IEEE 802.1 AE:2006, *IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security*

IEEE 802.15.4:2006, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*

FIPS PUB 197:2001, Advanced Encryption Standard (AES)

NIST SP 800-47:2002, *Security Guide for Interconnecting Information Technology Systems*

*The Galois/Counter Mode of Operation (GCM)* – David A. McGrew, Cisco Systems, Inc. 170, West Tasman Drive, San Jose, CA 95032, mcgrew@cisco.com, John Viega, Secure Software, 4100 Lafayette Center Drive, Suite 100, Chantilly, VA 20151, viega@securesoftware.com, May 31, 2005

RFC 1321, *The MD5 Message-Digest Algorithm.* Edited by R. Rivest (MIT Laboratory for Computer Science and RSA Data Security, Inc.) April 1992 *http://www.ietf.org/rfc/rfc1321.txt*

RFC 1662, *PPP in HDLC-like Framing, 1984, http://www.ietf.org/rfc/rfc1662.txt*

RFC 2104, *HMAC: Keyed-Hashing for Message Authentication, 2004, http://www.ietf.org/rfc/rfc2104.txt*

RFC 2119, *Key words for use in RFCs to Indicate Requirement Levels, 1997,* http://www.ietf.org/rfc/rfc2119.txt

RFC 2315, PKCS #7, *Cryptographic Message Syntax Version 1.5, 1998,* https://www.ietf.org/rfc/rfc2315

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

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

RFC 2822, *Internet Message Format, 2001, http://www.ietf.org/rfc/rfc2822*

RFC 2986, PKCS #10 v1.7, *Certification Request Syntax Standard,* http://www.ietf.org/rfc/rfc2986

RFC 3268, *Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS), 2002, http://tools.ietf.org/html/rfc3268*

RFC 4210, *Elliptic Curve Cryptography (ECC) Support for Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)*, 2008, https://tools.ietf.org/html/rfc5349

RFC 4211, *Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF), 2005, http://www.ietf.org/rfc/rfc4211*

RFC 4308, *Cryptographic Suites for IPsec, 2005*, *https://www.google.hu/#q=RFC%204308*

RFC 4835, *Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH),2007, http://tools.ietf.org/html/rfc4335*

RFC 5084, *Internet Engineering Task Force (IETF) Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS).* Edited by R. Housley November 2007. Available from: http://www.rfc-editor.org/rfc/rfc5084.txt

RFC 5349, *Elliptic Curve Cryptography (ECC) Support for Public Key Cryptography for Initial Authentication in Kerberos* (PKINIT), 2008, https://tools.ietf.org/html/rfc5349

ANSI C12.21:1999, *Protocol Specification for Telephone Modem Communication*

SEC2:2010, *Standards for Efficient Cryptography: Recommended Elliptic Curve Domain Parameters,* Version 2.0. Certicom Research

NIST SP 800-38D:2007, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*

NSA1, *Suite B Implementer's Guide to FIPS 186-3 (ECDSA)*, Feb 3rd 2010

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# SOMMAIRE

AVANT-PROPOS ..... 386
INTRODUCTION ..... 388
1 Domaine d'application ..... 389
2 Références normatives ..... 390
3 Termes, définitions, abréviations et symboles ..... 392
3.1 Définitions générales concernant DLMS®/COSEM ..... 392
3.2 Définitions liées à la sécurité cryptée ..... 396
3.3 Définitions et abréviations liées au mode Galois/Counter ..... 407
3.4 Abréviations générales ..... 409
3.5 Symboles liés au mode Galois/Counter ..... 413
3.6 Symboles liés à l'algorithme ECDSA ..... 413
3.7 Symboles liés aux algorithmes à agrément de clé ..... 414
4 Vue d'ensemble de DLMS®/COSEM ..... 414
4.1 Échange d'informations dans DLMS®/COSEM ..... 414
4.1.1 Généralités ..... 414
4.1.2 Modèle de communication ..... 415
4.1.3 Nommage et adressage ..... 416
4.1.4 Opération orientée connexion ..... 419
4.1.5 Associations d'applications ..... 420
4.1.6 Type de messagerie ..... 422
4.1.7 Échange de données entre des tierces parties et des serveurs DLMS®/COSEM ..... 423
4.1.8 Profils de communication ..... 423
4.1.9 Modèle de système de comptage DLMS®/COSEM ..... 424
4.1.10 Modèle de serveurs DLMS®/COSEM ..... 425
4.1.11 Modèle d'un client DLMS®/COSEM ..... 427
4.1.12 Interopérabilité et interconnectivité dans DLMS®/COSEM ..... 428
4.1.13 Assurance d'interconnectivité: service d'identification de protocole ..... 428
4.1.14 Intégration de système et installation de comptage ..... 428
4.2 Principales caractéristiques de la couche application DLMS®/COSEM ..... 429
4.2.1 Généralités ..... 429
4.2.2 Structure de la couche application DLMS®/COSEM ..... 429
4.2.3 Élément de service de contrôle d'association (ACSE) ..... 431
4.2.4 Élément de service d'application xDLMS ..... 432
4.2.5 Services de gestion de couche ..... 440
4.2.6 Récapitulatif des services de la couche application DLMS®/COSEM ..... 440
4.2.7 Protocoles de la couche application DLMS®/COSEM ..... 441
5 Sécurité des informations dans DLMS®/COSEM ..... 442
5.1 Vue d'ensemble ..... 442
5.2 Concept de sécurité DLMS®/COSEM ..... 442
5.2.1 Vue d'ensemble ..... 442
5.2.2 Identification et authentification ..... 442
5.2.3 Contexte de sécurité ..... 446
5.2.4 Droits d'accès ..... 446
5.2.5 Sécurité des messages de la couche application ..... 447
5.2.6 Sécurité des données COSEM ..... 449
5.3 Algorithmes cryptographiques ..... 449

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

5.3.1 Vue d'ensemble ..... 449
5.3.2 Fonction de hachage ..... 450
5.3.3 Algorithmes à clé symétrique ..... 451
5.3.4 Algorithmes à clé publique ..... 458
5.3.5 Génération de nombres aléatoires ..... 470
5.3.6 Compression ..... 470
5.3.7 Suite de sécurité ..... 470
5.4 Clés cryptographiques — Vue d'ensemble ..... 471
5.5 Clés utilisées avec des algorithmes à clé symétrique ..... 471
5.5.1 Types de clés symétriques ..... 471
5.5.2 Informations relatives aux clés avec APDU general-ciphering et protection des données ..... 474
5.5.3 Identification de clé ..... 475
5.5.4 Enveloppement de clé ..... 475
5.5.5 Agrément de clé ..... 475
5.5.6 Périodes cryptographiques à clé symétrique ..... 476
5.6 Clés utilisées avec des algorithmes à clé publique ..... 476
5.6.1 Vue d'ensemble ..... 476
5.6.2 Génération de paires de clés ..... 477
5.6.3 Certificats de clé publique et infrastructure à clé publique ..... 477
5.6.4 Certificat et profil d'extension de certificat ..... 480
5.6.5 Types de certificats d'entités finales de la Suite B à prendre en charge par les serveurs DLMS®/COSEM ..... 488
5.6.6 Gestion des certificats ..... 489
5.7 Application de la protection cryptographique ..... 494
5.7.1 Vue d'ensemble ..... 494
5.7.2 Protection des APDU xDLMS ..... 494
5.7.3 Protection multicouche par plusieurs parties ..... 507
5.7.4 Mécanismes d'authentification HLS ..... 508
5.7.5 Protection des données COSEM ..... 511
6 Spécification de service de la couche application DLMS®/COSEM ..... 512
6.1 Primitives de service et paramètres ..... 512
6.2 Service COSEM-OPEN ..... 514
6.3 Service COSEM-RELEASE ..... 519
6.4 Service COSEM-ABORT ..... 522
6.5 Paramètres de protection et de transfert de bloc général ..... 523
6.6 Service GET ..... 529
6.7 Service SET ..... 532
6.8 Service ACTION ..... 537
6.9 Service ACCESS ..... 541
6.9.1 Vue d'ensemble — Principales fonctionnalités ..... 541
6.9.2 Spécification de service ..... 543
6.10 Service DataNotification ..... 548
6.11 Service EventNotification ..... 549
6.12 Service TriggerEventNotificationSending ..... 551
6.13 Spécification d'accès variable ..... 551
6.14 Service Read ..... 552
6.15 Service Write ..... 556
6.16 Service UnconfirmedWrite ..... 560

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

6.17 Service InformationReport ..... 561
6.18 Services de gestion de couches côté client: Service SetMapperTable.request ..... 562
6.19 Récapitulatif des services et de la mise en correspondance de services de transfert de données LN/SN ..... 562
7 Spécification du protocole de couche application DLMS®/COSEM ..... 564
7.1 Fonction de commande ..... 564
7.1.1 Définitions des états de la fonction de commande côté client ..... 564
7.1.2 Définitions des états de la fonction de commande côté serveur ..... 565
7.2 Services ACSE et APDU ..... 567
7.2.1 Unités fonctionnelles ACSE, services et paramètres de service ..... 567
7.2.2 Noms COSEM enregistrés ..... 570
7.2.3 Règles de codage d'APDU ..... 573
7.2.4 Protocole d'établissement d'association d'applications ..... 573
7.2.5 Protocole de libération d'association d'applications ..... 579
7.3 Protocole des services de transfert de données ..... 582
7.3.1 Négociation de services et d'options — Bloc de conformité ..... 582
7.3.2 Appels de service confirmés et non confirmés ..... 583
7.3.3 Protocole du service GET ..... 585
7.3.4 Protocole du service SET ..... 588
7.3.5 Protocole du service ACTION ..... 591
7.3.6 Protocole du service ACCESS ..... 593
7.3.7 Protocole du service DataNotification ..... 595
7.3.8 Protocole du service EventNotification ..... 598
7.3.9 Protocole du service Read ..... 599
7.3.10 Protocole du service Write ..... 603
7.3.11 Protocole du service UnconfirmedWrite ..... 607
7.3.12 Protocole du service InformationReport ..... 608
7.3.13 Protocole du mécanisme de transfert de bloc général ..... 609
7.3.14 Protocole de mécanisme d'exception ..... 632
8 Syntaxe abstraite des APDU ACSE et COSEM ..... 633
9 Schéma XML des APDU COSEM ..... 652
9.1 Généralités ..... 652
9.2 Schéma XML ..... 652
Annexe A (normative) Utilisation de la couche application DLMS®/COSEM dans différents profils de communication ..... 674
A.1 Généralités ..... 674
A.2 Environnements de communication ciblés ..... 674
A.3 Structure du profil ..... 674
A.4 Schémas d'identification et d'adressage ..... 674
A.5 Services de couche de support et mise en correspondance de services ..... 675
A.6 Paramètres spécifiques au profil de communication des services d'AL COSEM ..... 675
A.7 Considérations / contraintes spécifiques à l'utilisation de certains services dans un profil donné ..... 675
A.8 Profil de communication à 3 couches, orienté connexion et basé sur HDLC ..... 675
A.9 Profils de communication basés sur TCP-UDP/IP (COSEM_on_IP) ..... 675
A.10 Profils de communication M-Bus filaire et sans fil ..... 675
A.11 Profil PLC S-FSK ..... 675

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Annexe B (normative) Couche d'adaptation réduite pour SMS ..... 676
Annexe C (normative) Protocole de passerelle ..... 677
C.1 Généralités ..... 677
C.2 Protocole de passerelle ..... 678
C.3 HES dans le WAN/NN agissant comme initiateur (opération Pull) ..... 679
C.4 Dispositifs finaux dans le LAN agissant comme initiateurs (opération Push) ..... 680
C.4.1 Généralités ..... 680
C.4.2 Dispositif final ayant des connaissances sur le WAN/NN ..... 680
C.4.3 Dispositifs finaux sans connaissances sur le WAN/NN ..... 681
C.5 Sécurité ..... 681
Annexe D (informative) Exemples de codages AARQ et AARE ..... 682
D.1 Généralités ..... 682
D.2 Codage des APDU xDLMS InitiateRequest/InitiateResponse ..... 682
D.3 Spécification des APDU AARQ et AARE ..... 685
D.4 Données pour les exemples ..... 686
D.5 Codage de l'APDU AARQ ..... 687
D.6 Codage de l'APDU AARE ..... 690
Annexe E (informative) Exemples de codage: APDU AARQ et AARE utilisant un contexte chiffré d'application ..... 696
E.1 Codage A-XDR de l'APDU xDLMS InitiateRequest contenant une clé dédiée ..... 696
E.2 Cryptage authentifié de l'APDU xDLMS InitiateRequest ..... 697
E.3 APDU AARQ ..... 698
E.4 Codage A-XDR de l'APDU xDLMS InitiateResponse ..... 700
E.5 Cryptage authentifié de l'APDU xDLMS InitiateResponse ..... 701
E.6 APDU AARE ..... 702
E.7 APDU RLRQ (contenant une APDU xDLMS InitiateRequest chiffrée) ..... 704
E.8 APDU RLRE (contenant une APDU xDLMS InitiateResponse chiffrée) ..... 705
Annexe F (informative) Exemples de services de transfert de données ..... 706
F.1 Exemples GET / Read, SET / Write ..... 706
F.2 Exemple de service ACCESS ..... 723
F.3 Exemple de codage compact-array ..... 724
F.3.1 Généralités ..... 724
F.3.2 Spécification de compact-array ..... 725
F.3.3 Exemple 1: codage compact-array d'un array de cinq valeurs long-unsigned ..... 726
F.3.4 Exemple 2: codage compact-array de cinq valeurs octet-string ..... 727
F.3.5 Exemple 3: codage du tampon d'un objet générique Profile ..... 728
F.4 Exemples de codage de l'attribut buffer de l'IC "Profile generic" ..... 729
F.4.1 Généralités ..... 729
F.4.2 Exemple de codage normal pour Get-response avec Profile generic ..... 730
F.4.3 Exemple de codage null-data compressées pour Get-response avec Profile generic ..... 732
F.4.4 Exemple de codage compact-array pour Get-response avec Profile generic ..... 735
F.4.5 Exemple de codage null-data et delta-value pour Get-response avec Profile generic ..... 737
F.4.6 Comparaison de différentes méthodes de codage pour l'APDU get-response ..... 740
F.4.7 Combinaison des différentes méthodes de codage et compression V.44 ..... 740

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Annexe G (normative) Courbes elliptiques et paramètres de domaine de la Suite B NSA ..... 742
Annexe H (informative) Exemple de certificat de signature d'entité finale utilisant P-256 signé avec P-256 ..... 744
H.1 Champs des certificats de clé publique ..... 744
H.2 Exemple de certificat Root-CA utilisant P-256 signé avec P-256 ..... 745
H.3 Exemple de certificat de signature numérique d'entité finale utilisant P-256 signé avec P-256 ..... 746
Annexe I (normative) Utilisation des mécanismes d'agrément de clé dans DLMS®/COSEM ..... 747
I.1 Schéma Ephemeral Unified Model C(2e, 0s, ECC CDH) ..... 747
I.2 Schéma One-pass Diffie-Hellman C(1e, 1s, ECC CDH) ..... 750
I.3 Schéma de modèle unifié statique C(0e, 2s, ECC CDH) ..... 753
Annexe J (informative) Échange d'APDU xDLMS protégées entre TP et serveur ..... 757
J.1 Généralités ..... 757
J.2 Exemple 1: protection similaire dans les deux sens ..... 757
J.3 Exemple 2: protection différente dans les deux sens ..... 758
Annexe K (informative) Modifications techniques majeures par rapport à l'IEC 62056-5-3:2017 ..... 760
Bibliographie ..... 763

Figure 1 – Modèle client/serveur et protocoles de communication ..... 416
Figure 2 – Nommage et adressage dans DLMS®/COSEM ..... 417
Figure 3 – Session complète de communication dans l'environnement CO ..... 420
Figure 4 – Types de messagerie DLMS®/COSEM ..... 422
Figure 5 – Profil générique de communication DLMS®/COSEM ..... 424
Figure 6 – Modèle de système de comptage DLMS®/COSEM ..... 425
Figure 7 – Modèle de serveur DLMS®/COSEM ..... 426
Figure 8 – Modèle de client DLMS®/COSEM utilisant plusieurs piles de protocoles ..... 427
Figure 9 – Structure des couches d'application DLMS®/COSEM ..... 430
Figure 10 – Concept de messages xDLMS composables ..... 438
Figure 11 – Récapitulatif des services de l'AL DLMS®/COSEM ..... 441
Figure 12 – Mécanismes d'authentification ..... 444
Figure 13 – Conception de sécurité des messages client-serveur ..... 447
Figure 14 – Concept de sécurité de bout en bout sur les messages ..... 448
Figure 15 – Fonction de hachage ..... 450
Figure 16 – Cryptage et décryptage ..... 451
Figure 17 – Codes d'authentification de message (MAC) ..... 453
Figure 18 – Fonctions du GCM ..... 454
Figure 19 – Signatures numériques ..... 461
Figure 20 – Schéma C(2e, 0s): chaque partie apporte uniquement une paire de clés éphémères ..... 463
Figure 21 – Schémas C(1e, 1s): la partie U apporte une paire de clés éphémères, et la partie V apporte une paire de clés statiques ..... 465
Figure 22 – Schéma C(0e, 2s): chaque partie apporte uniquement une paire de clés statiques ..... 467
Figure 23 – Architecture d'une infrastructure à clé publique (exemple) ..... 479

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Figure 24 – MSC pour l'approvisionnement du serveur en certificats de la CA ..... 490
Figure 25 – MSC pour la personnalisation de sécurité du serveur ..... 491
Figure 26 – Approvisionnement du serveur en certificat du client ..... 492
Figure 27 – Approvisionnement du client/de la tierce partie en certificat du serveur ..... 493
Figure 28 – Suppression de certificat du serveur ..... 493
Figure 29 – Protection cryptographique des informations à l'aide d'AES-GCM ..... 497
Figure 30 – Structure des APDU xDLMS de chiffrement global spécifique au service / de chiffrement dédié spécifique au service ..... 499
Figure 31 – Structure des APDU xDLMS general-glo-ciphering et general-ded-ciphering ..... 500
Figure 32 – Structure des APDU xDLMS general-ciphering ..... 501
Figure 33 – Structure des APDU general-signing ..... 507
Figure 34 – Primitives de service ..... 512
Figure 35 – Diagrammes de séquences temporelles ..... 513
Figure 36 – Paramètres de service supplémentaires pour contrôler la protection cryptographique et le GBT ..... 524
Figure 37 – Diagramme d'états partiel pour la fonction de commande côté client ..... 564
Figure 38 – Diagramme d'états partiel pour la fonction de commande côté serveur ..... 566
Figure 39 – MSC pour l'établissement réussi d'une AA précédé de l'établissement réussi d'une connexion de couches inférieures ..... 575
Figure 40 – Libération d'AA sans perte de données à l'aide du service A-RELEASE ..... 580
Figure 41 – Libération d'AA sans perte de données par déconnexion de la couche de support ..... 581
Figure 42 – Abandon d'une AA après la primitive PH-ABORT.indication ..... 582
Figure 43 – MSC du service GET ..... 586
Figure 44 – MSC du service GET avec transfert de blocs ..... 586
Figure 45 – MSC du service GET avec transfert de blocs, transfert long abandonné ..... 588
Figure 46 – MSC du service SET ..... 589
Figure 47 – MSC du service SET avec transfert de blocs ..... 590
Figure 48 – MSC du service ACTION ..... 592
Figure 49 – MSC du service ACTION avec transfert de bloc ..... 593
Figure 50 – Service ACCESS avec réponse longue ..... 594
Figure 51 – Service ACCESS avec demande et réponse longues ..... 595
Figure 52 – MSC pour le service DataNotification, situation 1) ..... 596
Figure 53 – MSC pour le service DataNotification, situation 2) ..... 597
Figure 54 – MSC pour le service DataNotification, situation 3) ..... 598
Figure 55 – MSC du service Read utilisé pour lire un attribut ..... 601
Figure 56 – MSC du service Read utilisé pour appeler une méthode ..... 601
Figure 57 – MSC du service Read utilisé pour lire un attribut, avec transfert de blocs ..... 602
Figure 58 – MSC du service Write utilisé pour écrire un attribut ..... 606
Figure 59 – MSC du service Write utilisé pour appeler une méthode ..... 606
Figure 60 – MSC du service Write utilisé pour écrire un attribut, avec transfert de blocs ..... 607
Figure 61 – MSC du service UnconfirmedWrite utilisé pour écrire un attribut ..... 608
Figure 62 – Appels de service partiels et APDU GBT ..... 611
Figure 63 – Procédure de GBT ..... 614

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Figure 64 – Sous-procédure Envoyer le flux d'APDU GBT ..... 618
Figure 65 – Sous-procédure Traiter l'APDU GBT ..... 620
Figure 66 – Sous-procédure Vérifier la RQ et combler les trous ..... 622
Figure 67 – Service GET avec GBT, passage à la diffusion en flux ..... 623
Figure 68 – Service GET avec appels partiels, GBT et en flux, récupération du 4e bloc envoyé dans le 2e flux ..... 625
Figure 69 – Service GET avec appels partiels, GBT et en flux, récupération des 4e et 5e blocs ..... 626
Figure 70 – Service GET avec appels partiels, GBT et en flux, récupération du dernier bloc ..... 627
Figure 71 – Service SET avec GBT, avec serveur ne prenant pas en charge du flux, récupération du 3e bloc ..... 628
Figure 72 – Service ACTION-WITH-LIST avec GBT bidirectionnel et récupération de blocs ..... 629
Figure 73 – Service DataNotification avec GBT, avec appel partiel ..... 631
Figure B.1 – Couche d'adaptation réduite ..... 676
Figure C.1 – Architecture générale avec passerelle ..... 677
Figure C.2 – Champs utilisés pour le préfixage des APDU COSEM ..... 678
Figure C.3 – Tableau de séquence de messages Pull ..... 679
Figure C.4 – Tableau de séquence de messages Push ..... 680
Figure I.1 – MSC pour agrément de clé utilisant le schéma Ephemeral Unified Model C(2e, 0s, ECC CDH) ..... 747
Figure I.2 – APDU xDLMS chiffrée protégée par une clé éphémère établie à l'aide d'un schéma One-pass Diffie-Hellman (1e, 1s, ECC CDH) ..... 750
Figure I.3 – APDU xDLMS chiffrée protégée par une clé éphémère établie à l'aide du schéma de modèle unifié statique C(0e, 2s, ECC CDH) ..... 754
Figure J.1 – Échange d'APDU xDLMS protégées entre TP et serveur: exemple 1 ..... 758
Figure J.2 – Échange d'APDU xDLMS protégées entre TP et serveur: exemple 2 ..... 759

Tableau 1 – SAP client et serveur ..... 418
Tableau 2 —Explication de la signification des paramètres PDU Size pour DLMS®/COSEM ..... 440
Tableau 3 – Courbes elliptiques dans les suites de sécurité DLMS®/COSEM ..... 459
Tableau 4 – Récapitulatif de mécanisme d'agrément de clé Ephemeral Unified Model ..... 464
Tableau 5 – Récapitulatif de mécanisme d'agrément de clé One-pass Diffie-Hellman ..... 466
Tableau 6 – Récapitulatif du mécanisme d'agrément de clé du modèle unifié statique ..... 468
Tableau 7 – Sous-champs et sous-chaînes *OtherInfo* ..... 469
Tableau 8 – ID d'algorithmes de sécurité ..... 470
Tableau 9 – Suites de sécurité DLMS®/COSEM ..... 471
Tableau 10 – Types de clés symétriques ..... 473
Tableau 11 – Informations relatives aux clés avec APDU general-ciphering et protection des données ..... 474
Tableau 12 – Types de clés asymétriques et leur utilisation ..... 476
Tableau 13 – Structure de certificat X.509 v3 ..... 481
Tableau 14 – Champs du tbsCertificate X.509 v3 ..... 482

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Tableau 15 – Schéma de nommage pour l'instance de la Root-CA (informatif) ..... 483
Tableau 16 – Schéma de nommage pour l'instance de la Sub-CA (informatif) ..... 483
Tableau 17 – Schéma de nommage pour l'instance de l'entité finale ..... 483
Tableau 18 – Extensions de certificat X.509 v3 ..... 485
Tableau 19 – Extensions KeyUsage ..... 486
Tableau 20 – Valeurs Subject Alternative Name (nom alternatif d'objet) ..... 487
Tableau 21 – Valeurs Issuer Alternative Name (nom alternatif de l'émetteur) ..... 487
Tableau 22 – Valeurs de l'extension Basic Constraints ..... 488
Tableau 23 – Certificats traités par des entités finales DLMS®/COSEM ..... 489
Tableau 24 – Valeurs de la politique de sécurité ("Security setup" version 1) ..... 494
Tableau 25 – Valeurs des droits d'accès ("Association LN" ver 3, "Association SN" ver 4) ..... 495
Tableau 26 – APDU xDLMS chiffrées ..... 496
Tableau 27 – Octet de contrôle de sécurité ..... 498
Tableau 28 – Texte brut et données supplémentaires authentifiées ..... 498
Tableau 29 – Utilisation des champs des APDU xDLMS de chiffrement ..... 502
Tableau 30 – Exemple: APDU xDLMS glo-get-request ..... 503
Tableau 31 – Service ACCESS avec le mécanisme d'agrément de clé One-pass Diffie-Hellman C(1e, 1s, ECC CDH) et general-ciphering ..... 505
Tableau 32 – Mécanismes d'authentification HLS DLMS®/COSEM ..... 509
Tableau 33 – Exemple de HLS utilisant le mécanisme d'authentification 5 avec GMAC ..... 510
Tableau 34 – Exemple de HLS utilisant le mécanisme d'authentification 7 avec ECDSA ..... 511
Tableau 35 – Codes des paramètres de service de l'AL ..... 514
Tableau 36 – Paramètres de service des primitives de service COSEM-OPEN ..... 515
Tableau 37 – Paramètres de service des primitives de service COSEM-RELEASE ..... 520
Tableau 38 – Paramètres de service des primitives de service COSEM-ABORT ..... 523
Tableau 39 – Paramètres de service supplémentaires ..... 525
Tableau 40 – Paramètres de sécurité ..... 526
Tableau 41 – APDU utilisées avec les types de protections de sécurité (Security_Protection_Type) ..... 528
Tableau 42 – Paramètres de service du service GET ..... 530
Tableau 43 – Types de demandes et de réponses du service GET ..... 531
Tableau 44 – Paramètres de service du service SET ..... 533
Tableau 45 – Types de demandes et de réponses du service SET ..... 534
Tableau 46 – Paramètres de service du service ACTION ..... 537
Tableau 47 – Types de demandes et de réponses du service ACTION ..... 538
Tableau 48 – Paramètres de service du service ACCESS ..... 545
Tableau 49 – Paramètres de service des primitives de service DataNotification ..... 548
Tableau 50 – Paramètres de service des primitives de service EventNotification ..... 550
Tableau 51 – Paramètres de service de la primitive de service TriggerEventNotificationSending.request ..... 551
Tableau 52 – Spécification d'accès variable ..... 551
Tableau 53 – Paramètres de service du service Read ..... 552
Tableau 54 – Utilisation des variantes du paramètre Variable_Access_Specification et des choix pour Read.response ..... 554

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Tableau 55 – Paramètres de service du service Write ..... 557
Tableau 56 – Utilisation des variantes de Variable_Access_Specification et des choix pour Write.response ..... 558
Tableau 57 – Paramètres de service du service UnconfirmedWrite ..... 560
Tableau 58 – Utilisation des variantes de Variable_Access_Specification ..... 560
Tableau 59 – Paramètres de service du service InformationReport ..... 562
Tableau 60 – Paramètres de service des primitives de service SetMapperTable.request ..... 562
Tableau 61 – Récapitulatif des services ACSE ..... 563
Tableau 62 – Récapitulatif des services xDLMS ..... 563
Tableau 63 – APDU d'unité fonctionnelle et leurs champs ..... 568
Tableau 64 – Noms de contexte d'application COSEM ..... 572
Tableau 65 – Noms de mécanismes d'authentification COSEM ..... 572
Tableau 66 – ID d'algorithmes cryptographiques ..... 573
Tableau 67 – Bloc de conformité xDLMS ..... 583
Tableau 68 – Types de service GET et APDU ..... 585
Tableau 69 – Types de service SET et APDU ..... 589
Tableau 70 – Types de service ACTION et APDU ..... 592
Tableau 71 – Mise en correspondance du service GET et du service Read ..... 599
Tableau 72 – Mise en correspondance du service ACTION et du service Read ..... 600
Tableau 73 – Mise en correspondance du service SET et du service Write ..... 603
Tableau 74 – Mise en correspondance du service ACTION et du service Write ..... 604
Tableau 75 – Mise en correspondance du service SET et du service UnconfirmedWrite ..... 608
Tableau 76 – Mise en correspondance du service ACTION et du service UnconfirmedWrite ..... 608
Tableau 77 – Mise en correspondance des services EventNotification et InformationReport ..... 609
Tableau 78 – Variables d'états de la procédure de GBT ..... 616
Tableau 79 – Mécanisme d'exception xDLMS ..... 632
Tableau B.1 – Processus d'application réservés ..... 676
Tableau D.1 – Bloc de conformité ..... 683
Tableau D.2 – Codage A-XDR de l'APDU xDLMS InitiateRequest ..... 684
Tableau D.3 – Codage A-XDR de l'APDU xDLMS InitiateResponse ..... 685
Tableau D.4 – Codage BER de l'APDU AARQ ..... 688
Tableau D.5 – APDU AARQ complète ..... 690
Tableau D.6 – Codage BER de l'APDU AARE ..... 691
Tableau D.7 – APDU AARE complète ..... 695
Tableau E.1 – Codage A-XDR de l'APDU xDLMS InitiateRequest ..... 697
Tableau E.2 – Cryptage authentifié de l'APDU xDLMS InitiateRequest ..... 698
Tableau E.3 – Codage BER de l'APDU AARQ ..... 699
Tableau E.4 – Codage A-XDR de l'APDU xDLMS InitiateResponse ..... 701
Tableau E.5 – Cryptage authentifié de l'APDU xDLMS InitiateResponse ..... 702
Tableau E.6 – Codage BER de l'APDU AARE ..... 703
Tableau E.7 – Codage BER de l'APDU RLRQ ..... 704
Tableau E.8 – Codage BER de l'APDU RLRE ..... 705

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Tableau F.1 – Objets utilisés dans les exemples ..... 706
Tableau F.2 – Exemple: lecture de la valeur d'un attribut unique sans transfert de blocs ..... 707
Tableau F.3 – Exemple: lecture de la valeur d'une liste d'attributs sans transfert de blocs ..... 708
Tableau F.4 – Exemple: lecture de la valeur d'un attribut unique avec transfert de blocs ..... 710
Tableau F.5 – Exemple: lecture de la valeur d'une liste d'attributs avec transfert de blocs ..... 712
Tableau F.6 – Exemple: écriture de la valeur d'un attribut unique sans transfert de blocs ..... 715
Tableau F.7 – Exemple: écriture de la valeur d'une liste d'attributs sans transfert de blocs ..... 716
Tableau F.8 – Exemple: écriture de la valeur d'un attribut unique avec transfert de blocs ..... 718
Tableau F.9 – Exemple: écriture de la valeur d'une liste d'attributs avec transfert de blocs ..... 720
Tableau F.10 – Exemple: service ACCESS sans transfert de bloc général ..... 723
Tableau F.11 – Tampon "Profile generic" – get-response avec codage normal ..... 730
Tableau F.12 – Tampon "Profile generic" – get-response avec compression null-data ..... 732
Tableau F.13 – Tampon "Profile generic" – get-response avec codage compact-array ..... 735
Tableau F.14 – Tampon "Profile generic" – get-response avec codage null-data et delta-value ..... 738
Tableau F.15 – Comparaison de différentes méthodes de codage pour l'APDU get-response ..... 740
Tableau F.16 – Combinaison des différentes méthodes de codage et compression V.44 pour l'APDU get-response ..... 741
Tableau G.1 – ECC_P256_Domain_Parameters ..... 742
Tableau G.2 – ECC_P384_Domain_Parameters ..... 743
Tableau H.1 – Champs des certificats de clé publique utilisant P-256 signé avec P-256 ..... 744
Tableau I.1 – Vecteur d'essai pour agrément de clé utilisant le schéma Ephemeral Unified Model C(2e, 0s, ECC CDH) ..... 748
Tableau I.2 – Vecteur d'essai pour agrément de clé utilisant le schéma One-pass Diffie-Hellman (1e, 1s, ECC CDH) ..... 751
Tableau I.3 – Vecteur d'essai pour agrément de clé utilisant le schéma de modèle unifié statique (0e, 2s, ECC CDH) ..... 755

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

# ÉCHANGE DES DONNÉES DE COMPTAGE DE L'ÉLECTRICITÉ – LA SUITE DLMS®/COSEM –

## Partie 5-3: Couche application DLMS®/COSEM

## AVANT-PROPOS

1) La Commission Électrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. À cet effet, l'IEC — entre autres activités — publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments du présent document de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

L'IEC 62056-5-3 a été établie par le comité d'études 13 de l'IEC: Comptage et pilotage de l'énergie électrique. Il s'agit d'une Norme internationale.

Cette quatrième édition annule et remplace la troisième édition parue en 2017. Cette édition constitue une révision technique.

Les modifications techniques majeures par rapport à l'édition précédente sont énumérées à l'Annexe K (informative).

Le texte de cette Norme internationale est issu des documents suivants:

| Projet | Rapport de vote |
|---|---|
| 13/1890/FDIS | 13/1904/RVD |

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à son approbation.

La version française de la norme n'a pas été soumise au vote.

La langue employée pour l'élaboration de cette Norme internationale est l'anglais.

Ce document a été rédigé selon les Directives ISO/IEC, Partie 2, il a été développé selon les Directives ISO/IEC, Partie 1 et les Directives ISO/IEC, Supplément IEC, disponibles sous www.iec.ch/members_experts/refdocs. Les principaux types de documents développés par l'IEC sont décrits plus en détail sous www.iec.ch/publications.

Une liste de toutes les parties de la série IEC 62056, publiées sous le titre général *Échange des données de comptage de l'électricité — La suite DLMS®/COSEM*, se trouve sur le site Web de l'IEC.

Le comité a décidé que le contenu de ce document ne sera pas modifié avant la date de stabilité indiquée sur le site Web de l'IEC sous webstore.iec.ch dans les données relatives au document recherché. À cette date, le document sera

- reconduit,
- supprimé,
- remplacé par une édition révisée, ou
- amendé.

**IMPORTANT — Le logo "colour inside" qui se trouve sur la page de couverture de ce document indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# INTRODUCTION

Cette quatrième édition de l'IEC 62056-5-3 a été établie par le groupe de travail 14 du comité d'études 13 de l'IEC avec la contribution significative de la DLMS® User Association, son partenaire de liaison de type A.

La présente édition est conforme à DLMS® UA 1000-2, the "Green Book" Ed. 10:2020 et DLMS® UA 1000-2, the "Green Book" Ed. 10 Amendment 1 2021.

La Commission Électrotechnique Internationale (IEC) attire l'attention sur le fait qu'il est déclaré que la conformité avec les dispositions de la présente Norme internationale peut impliquer l'utilisation d'un service de maintenance concernant la pile de protocoles sur laquelle est basée la présente norme IEC 62056-5-3.

L'IEC ne prend pas position quant à la preuve, la validité et la portée de ce service de maintenance.

Le fournisseur du service de maintenance a donné l'assurance à l'IEC qu'il consent à fournir des services à des demandeurs du monde entier, en des termes et à des conditions raisonnables et non discriminatoires. À cet égard, la déclaration du fournisseur du service de maintenance est enregistrée à l'IEC. Des informations peuvent être obtenues auprès de:

DLMS[1] User Association (Association des utilisateurs de la DLMS)
www.dlms.com

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

[1] Device Language Message Specification (Spécification des messages de langage d'équipement).

# ÉCHANGE DES DONNÉES DE COMPTAGE DE L'ÉLECTRICITÉ – LA SUITE DLMS®/COSEM –

## Partie 5-3: Couche application DLMS®/COSEM

## 1 Domaine d'application

La présente partie de l'IEC 62056 spécifie la couche application DLMS®/COSEM concernant la structure, les services et les protocoles pour les clients et serveurs DLMS®/COSEM, et définit les règles de spécification des profils de communication DLMS®/COSEM.

Elle définit les services permettant d'établir et de libérer des associations d'applications, ainsi que les services de communication de données permettant d'accéder aux méthodes et aux attributs des objets d'interface COSEM, définis dans l'IEC 62056-6-2:2021, à l'aide du référencement par nom logique (LN, Logical Name) ou par nom abrégé (SN, Short Name).

L'Annexe A (normative) définit comment utiliser la couche application COSEM dans différents profils de communication. Elle indique comment différents profils de communication peuvent être construits de sorte à échanger des données avec le matériel de comptage à l'aide du modèle d'interface COSEM, ainsi que les éléments nécessaires à indiquer dans chaque profil de communication. Les profils de communication réels, spécifiques au support, sont spécifiés dans des parties distinctes de la série IEC 62056.

L'Annexe B (normative) spécifie la couche d'adaptation réduite pour le système de messages courts (SMS, Short Message Service).

L'Annexe C (normative) spécifie le protocole de passerelle.

L'Annexe D, l'Annexe E et l'Annexe F (informatives) incluent des exemples de codage d'APDU.

L'Annexe G (normative) spécifie des courbes elliptiques et des paramètres de domaine de la Suite B de la NSA.

L'Annexe H (informative) donne un exemple de certificat de signature d'entité finale utilisant P-256 signé avec P-256.

L'Annexe I (normative) spécifie l'utilisation de mécanismes d'agrément de clé dans DLMS®/COSEM.

L'Annexe J (informative) donne des exemples d'échanges d'APDU xDLMS protégés entre une tierce partie et un serveur.

L'Annexe K (informative) énumère les modifications techniques majeures contenues dans cette quatrième édition.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 2 Références normatives

Les documents suivants sont cités dans le texte de sorte qu'ils constituent, pour tout ou partie de leur contenu, des exigences du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

IEC 61334-4-41:1996, *Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs – Partie 4:Protocoles de communication de données – Section 41:Protocoles d'application – Spécification des messages de ligne de distribution*

IEC 61334-6:2000, *Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs – Partie 6:Règles d'encodage A-XDR*

IEC TR 62051:1999, *Lecture des compteurs électriques – Glossaire de termes*

IEC TR 62051-1:2004, *Electricity metering – Data exchange for meter reading, tariff and load control – Glossary of terms – Part 1:Terms related to data exchange with metering equipment using DLMS®/COSEM* (disponible en anglais seulement)

IEC 62056-6-2:2021, *Échange des données de comptage de l'électricité – La suite DLMS/COSEM – Partie 6-2:Classes d'interfaces COSEM*

IEC 62056-7-3:2017, *Échange des données de comptage de l'électricité – La suite DLMS/COSEM – Partie 7-3: Profils de communication M-Bus filaire et sans fil pour les réseaux locaux et les réseaux de voisinage*

IEC 62056-7-6:2013, *Échange des données de comptage de l'électricité – La suite DLMS/COSEM – Partie 7-6:Profil de communication à 3 couches, orienté connexion et basé sur HDLC*

IEC 62056-8-3:2013, *Échange des données de comptage de l'électricité – La suite DLMS/COSEM – Partie 8-3:Profil de communication pour réseaux de voisinage CPL S-FSK*

IEC 62056-8-11:–[2] *Electricity metering data exchange – The DLMS®/COSEM suite – Part 8-11:Communication profile for Wi-SUN field area mesh networks* (disponible en anglais seulement)

IEC 62056-8-12:–[3] *Electricity metering data exchange – The DLMS®/COSEM suite – Part 8-12:Communication profile for Low Power Wide Area Networks (LPWAN)* (disponible en anglais seulement)

IEC 62056-9-7:2013, *Échange des données de comptage de l'électricité – La suite DLMS/COSEM – Partie 9-7:Profil de communication pour réseaux TCP-UDP/IP*

ISO/IEC 8824-1:2008, *Technologies de l'information – Notation de syntaxe abstraite numéro un (ASN.1) – Partie 1:Spécification de la notation de base*

ISO/IEC 8825-1:2008, *Technologies de l'information – Règles de codage ASN.1 – Partie 1:Spécification des règles de codage de base (BER), des règles de codage canoniques (CER) et des règles de codage distinctives (DER)*

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

---

2 En cours d'élaboration. Stade au moment de la publication: IEC CDV.

3 En cours d'élaboration. Stade au moment de la publication: 13/1877/CDV:2023.

ISO/IEC 15953:1999, *Technologies de l'information – Interconnexion des systèmes ouverts – Définition du service pour l'élément de service de contrôle d'association des objets de service d'application*

NOTE Cette norme annule et remplace l'ISO/IEC 8649:1996 ainsi que ses amendements A1:1997 et A2:1998, dont elle constitue une révision technique.

ISO/IEC 15954:1999, *Technologies de l'information – Interconnexion des systèmes ouverts – Protocole en mode connexion pour l'élément de service de contrôle d'association des objets de service d'application*

NOTE Cette norme annule et remplace l'ISO/IEC 8650-1:1999 ainsi que ses amendements A1:1997 et A2:1998, dont elle constitue une révision technique.

ISO/IEC 7498-1:1994, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base:le modèle de base*

UIT-T X.509:2008, *SÉRIE X:RÉSEAUX DE DONNÉES ET COMMUNICATION ENTRE SYSTÈMES OUVERTS ET SÉCURITÉ – Technologies de l'information – Interconnexion des systèmes ouverts – L'annuaire: cadre général des certificats de clé publique et d'attribut*

UIT-T X.693 (11/2008), *Technologies de l'information – Règles de codage ASN.1: Règles de codage XML (XER)*

UIT-T X.693 Corrigendum 1 (10/2011), *Technologies de l'information – Règles de codage ASN.1:Règles de codage XML (XER) – Corrigendum technique 1*

UIT-T X.694 (11/2008), *Technologies de l'information – Règles de codage ASN.1:Mappage en ASN.1 des définitions de schéma XML du W3C*

UIT-T X.694 Corrigendum 1 (10/2011), *Technologies de l'information – Règles de codage ASN.1:Mappage en ASN.1 des définitions de schéma XML du W3C – Corrigendum technique 1*

FIPS PUB 180-4:2012, *Secure hash standard (SHS)* (disponible en anglais seulement)

FIPS PUB 186-4:2013, *Digital Signature Standard (DSS)* (disponible en anglais seulement)

NIST SP 800-21:2005, *Guideline for Implementing Cryptography in the Federal Government* (disponible en anglais seulement)

NIST SP 800-32:2001, *Introduction to Public Key Technology and the Federal PKI Infrastructure* (disponible en anglais seulement)

NIST SP 800-56A Rev. 2:2013, *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography* (disponible en anglais seulement)

NIST SP 800-57:2012, *Recommendation for Key Management – Part 1: General* (Révision 3) (disponible en anglais seulement)

NSA2, *Suite B Implementer's Guide to NIST SP800-56A*, 28 juillet 2009 (disponible en anglais seulement)

NSA3, *NSA Suite B Base Certificate and CRL Profile*, 27 mai 2008 (disponible en anglais seulement)

SEC1:2009, *Standards for Efficient Cryptography:Elliptic Curve Cryptography* (disponible en anglais seulement).SECG. Version 2.0

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

RFC 3394, *Advanced Encryption Standard (AES) Key Wrap Algorithm*. Éditée par J. Schaad (Soaring Hawk Consulting) et R. Housley (RSA Laboratories), septembre 2002, http://tools.ietf.org/html/rfc3394

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

RFC 4108, *Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages*, 2005, http://www.ietf.org/rfc/rfc4108

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

## 3 Termes, définitions, abréviations et symboles

Pour les besoins du présent document, les termes et définitions de l'IEC TR 62051:1999, de l'IEC TR 62051-1, de la RFC 4106 ainsi que les suivants s'appliquent.

### 3.1 Définitions générales concernant DLMS®/COSEM

**3.1.1**
**APDU ACSE**
APDU utilisée par l'élément de service de contrôle d'association (ACSE, Association Control Service Element)

**3.1.2**
**association d'applications**
relation de coopération entre deux entités d'application, formée par l'échange de leurs informations de contrôle de protocole d'application au moyen de leur utilisation des services de présentation

**3.1.3**
**contexte d'application**
ensemble d'éléments de service d'application, d'options associées et de toute autre information nécessaire à l'interfonctionnement d'entités d'application dans une association d'applications

**3.1.4**
**entité d'application**
activités d'application indépendantes du système mises à disposition en tant que services d'application à l'agent d'application, par exemple, ensemble d'éléments de service d'application qui, réunis, effectuent tout ou partie des aspects de communication d'un processus d'application

**3.1.5**
**processus d'application**
élément d'un système ouvert réel effectuant un traitement d'information pour une application particulière

[SOURCE: ISO/IEC 7498-1:1994, 4.1.4]

**3.1.6**
**mécanisme d'authentification**
spécification d'un ensemble précis de règles relatives à la fonction d'authentification et applicables à la définition, au traitement et au transfert de valeurs d'authentification

[SOURCE: ISO/IEC 15953:1999, 3.5.11]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.1.7**
**bloc**
une partie d'une APDU xDLMS; la charge utile d'une APDU GBT

**3.1.8**
**client**
processus d'application fonctionnant dans le système de collecte de données

**3.1.9**
**client/serveur**
relation entre deux programmes informatiques dans laquelle un programme, le client, formule une demande de service à un autre programme, le serveur, qui répond à cette demande

**3.1.10**
**procédure de GBT confirmé**
procédure dans laquelle l'expéditeur envoie des flux d'APDU GBT et à la fin de chaque flux, le destinataire accuse réception des blocs reçus et tente de récupérer des blocs manquants

Note 1 à l'article: Un flux GBT est constitué d'une ou plusieurs APDU GBT.

Note 2 à l'article: Dans le cas d'un flux GBT confirmé, la fin du flux est indiquée par le bit de flux positionné à FALSE (0). Dans le cas d'un flux GBT non confirmé, la fin du flux est indiquée par le bit Final positionné à TRUE (1).

**3.1.11**
**COSEM**
spécification d'accompagnement pour le comptage de l'énergie; fait référence au modèle d'objet COSEM

**3.1.12**
**APDU COSEM**
comprend les APDU ACSE et les APDU xDLMS

**3.1.13**
**données COSEM**
valeurs d'attributs d'objets COSEM, appel de méthode et paramètres de retour

**3.1.14**
**classe d'interface COSEM**
entité comprenant un ensemble spécifique d'attributs et de méthodes modélisant une fonction particulière par elle-même ou en relation avec d'autres classes d'interfaces COSEM

**3.1.15**
**objet COSEM**
instance d'une classe d'interface COSEM

**3.1.16**
**DLMS®/COSEM**
fait référence à la couche application qui fournit des services xDLMS pour accéder aux attributs d'objets d'interfaces COSEM. Fait également référence à la couche application DLMS®/COSEM et au modèle de données COSEM pris ensemble

**3.1.17**
**contexte DLMS®**
spécification d'éléments de service associée à une sémantique de communication en usage pendant la durée de vie d'une association d'applications

[SOURCE: IEC 61334-4-41:1996, 3.3.5]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.1.18**
**authentification d'entité**
confirmation qu'une entité est bien l'entité déclarée

[SOURCE: ISO/IEC 9798-1:2010, 3.14]

**3.1.19**
**trou**
espace vide, c'est-à-dire blocs manquants dans la file d'attente de réception (RQ)

Note 1 à l'article: Une file d'attente de réception (RQ) peut avoir un ou plusieurs trous. Dans chaque trou, un ou plusieurs blocs peuvent être manquants.

**3.1.20**
**APDU GBT**
APDU xDLMS avec des informations de contrôle transportant un bloc d'une autre APDU xDLMS ou un bloc vide

**3.1.21**
**échange GBT**
échange d'APDU GBT qui transportent les primitives de service du même service

**3.1.22**
**flux GBT**
une séquence d'APDU GBT

**3.1.23**
**transfert de blocs général**
**GBT**
mécanisme de couche application DLMS®/COSEM qui peut transférer en plusieurs blocs une autre APDU xDLMS quelconque, qui serait autrement trop longue pour convenir avec la taille maximale d'APDU négociée

Note 1 à l'article: GBT peut être forcé en incluant les paramètres GBT dans la primitive de service .request.

**3.1.24**
**dispositif logique**
entité abstraite dans un dispositif physique, représentant un sous-ensemble de la fonctionnalité modélisée avec des objets COSEM

**3.1.25**
**station maître**
station centrale qui prend l'initiative et contrôle le flux de données

**3.1.26**
**message**
APDU xDLMS transportant une primitive de service sous une forme codée, qui peut également être protégée par cryptographie

**3.1.27**
**authentification mutuelle**
authentification d'entité qui assure à deux entités la conformité de l'identité de leur homologue

Note 1 à l'article: Le mécanisme d'authentification HLS DLMS®/COSEM fournit l'authentification mutuelle.

[SOURCE: ISO/IEC 9798-1:2010, 3.18, modifié par l'ajout de la Note 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.1.28**
**débordement**
réception en un seul flux d'APDU GBT et dont la taille est supérieure à la taille de la fenêtre GBT

**3.1.29**
**dispositif physique**
matériel de comptage physique, élément de niveau le plus élevé utilisé dans le modèle d'interface COSEM du matériel de comptage

**3.1.30**
**opération Pull**
style de communication dans lequel le client lance la demande d'une transaction donnée

**3.1.31**
**opération Push**
style de communication dans lequel le serveur lance la demande d'une transaction donnée

**3.1.32**
**file d'attente de réception**
**RQ**
espace réservé pour les blocs de l'APDU reçus dans un flux GBT

Note 1 à l'article: L'abréviation "RQ" est dérivée du terme anglais développé correspondant "Receive Queue"

**3.1.33**
**serveur**
processus d'application qui s'exécute dans un matériel de comptage

**3.1.34**
**file d'attente d'envois**
**SQ**
espace réservé pour les blocs de l'APDU à envoyer

Note 1 à l'article: L'abréviation "SQ" est dérivée du terme anglais développé correspondant "Send Queue"

**3.1.35**
**transfert de blocs spécifique au service**
mécanisme de couche application DLMS®/COSEM qui peut transférer en plusieurs blocs une APDU xDLMS correspondant à une primitive de service spécifique, qui serait autrement trop longue pour la taille maximale d'APDU négociée

**3.1.36**
**fenêtre de flux**
nombre d'APDU GBT qui peuvent être reçues dans un flux

**3.1.37**
**station esclave**
station qui répond aux interrogations d'une station maître

Note 1 à l'article: Un compteur est normalement une station esclave.

**3.1.38**
**titre système**
identificateur unique du système

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.1.39**
**procédure de GBT non confirmé**
procédure dans laquelle l'expéditeur envoie et le destinataire reçoit un seul flux d'APDU GBT, le destinataire n'accuse pas réception des blocs reçus et n'essaie pas de récupérer des blocs perdus

Note 1 à l'article: Ceci est utilisé pour transporter des demandes de service non confirmées du client vers le serveur ou des demandes de service non sollicitées du serveur vers le client.

**3.1.40**
**authentification unidirectionnelle**
authentification d'entité qui fournit à une entité la garantie de l'identité de son homologue, mais l'inverse n'est pas vrai

Note 1 à l'article: Le mécanisme d'authentification LLS DLMS®/COSEM fournit l'authentification unidirectionnelle.

[SOURCE: ISO/IEC 9798-1:2010, 3.39]

**3.1.41**
**xDLMS**
DLMS® étendu; fait référence au protocole DLMS® avec les extensions spécifiées dans le présent document

**3.1.42**
**APDU xDLMS**
APDU utilisée par l'élément de service d'application xDLMS (ASE xDLMS)

**3.1.43**
**message xDLMS**
APDU xDLMS échangée entre un client et un serveur ou entre une tierce partie et un serveur

### 3.2 Définitions liées à la sécurité cryptée

**3.2.1**
**contrôle d'accès**
limite l'accès aux ressources aux entités privilégiées uniquement

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.2**
**algorithme à clé asymétrique**
voir algorithme cryptographique à clé publique

**3.2.3**
**authentification**
processus qui établit la source d'informations, fournit la garantie de l'identité d'une entité ou fournit l'intégrité des sessions de communication, des messages, des documents ou des données stockées

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.4**
**code d'authentification**
somme de contrôle cryptographique fondée sur une fonction approuvée de sécurité (aussi appelée "code d'authentification de message")

[SOURCE: NIST SP 800-57:2012, Partie 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.5**
**certificat**
voir certificat de clé publique

**3.2.6**
**autorité de certification**
**CA**
entité, située dans une infrastructure à clé publique (PKI), qui est responsable de l'émission de certificats de clé publique et de la conformité à une politique PKI

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.7**
**politique de certification**
**CP**
forme particulière de politique d'administration relative aux transactions électroniques effectuées pendant la gestion des certificats. Une politique de certification traite tous les aspects relatifs à la génération, à la production, à la distribution, à la comptabilité, à la récupération de compromission et à la gestion des certificats numériques. Une politique de certification peut également, de manière indirecte, régir les transactions effectuées à l'aide d'un système de communication protégé par un système de sécurité à certificats. En contrôlant les extensions critiques de certificats, ces politiques et la technologie d'application associée peuvent prendre en charge les services de sécurité exigés par des applications particulières

[SOURCE: NIST SP 800-32:2001]

**3.2.8**
**épreuve**
paramètre variable dans le temps produit par un vérificateur

[SOURCE: ITU-T X.811:1995, 3.8]

**3.2.9**
**chiffrement**
authentification et/ou cryptage utilisant des algorithmes à clé symétrique

**3.2.10**
**texte chiffré**
données sous forme cryptée

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.11**
**cofacteur**
ordre du groupe de courbes elliptiques divisé par l'ordre (premier) du point générateur (c'est-à-dire le point de base) spécifié dans les paramètres de domaine

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.12**
**confidentialité**
propriété selon laquelle les informations sensibles n'ont pas été divulguées à des entités non autorisées

[SOURCE: NIST SP 800-57:2012, Partie 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.13**
**algorithme cryptographique**
procédure informatique bien définie qui produit un résultat à partir de valeurs d'entrée variables, y compris une clé cryptographique

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.14**
**clé cryptographique**
**clé**
paramètre, utilisé conjointement avec un algorithme cryptographique, qui détermine son fonctionnement de sorte qu'une entité connaissant la clé puisse reproduire ou inverser le fonctionnement, alors qu'une entité sans connaissance de la clé ne peut pas

Les exemples comprennent:

1. la transformation de données en texte brut en des données en texte chiffré;
2. la transformation de données en texte chiffré en des données en texte brut;
3. le calcul d'une signature numérique à partir de données;
4. la vérification d'une signature numérique;
5. le calcul d'un code d'authentification à partir de données;
6. la vérification d'un code d'authentification à partir de données et d'un code d'authentification reçu;
7. le calcul d'un secret partagé qui est utilisé pour dériver le matériel de chiffrement.

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.15**
**période cryptographique**
intervalle de temps pendant lequel l'utilisation d'une clé spécifique est autorisée ou dans lequel les clés pour un système ou une application donnés peuvent rester en vigueur

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.16**
**clé dédiée**
dans DLMS®/COSEM, clé symétrique utilisée dans une instance unique d'une association d'applications. Voir aussi clé de session

**3.2.17**
**obsolète**
non recommandé pour de nouvelles implémentations

**3.2.18**
**signature numérique**
résultat d'une transformation cryptographique de données qui, lorsqu'elle est correctement mise en œuvre avec une infrastructure et une politique adéquates, fournit les services suivants:

1) authentification de l'origine;
2) intégrité des données;
3) non-répudiation du signataire.

[SOURCE: NIST SP 800-57:2012, Partie 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.19**
**autorité CA de confiance directe**
une autorité CA de confiance directe est une autorité CA dont la clé publique a été obtenue et est en cours de mémorisation par une entité finale, en toute confiance et sécurité, et dont la clé publique est acceptée par cette entité finale dans le contexte d'une ou plusieurs applications

[SOURCE: ISO/IEC 15945:2002, 3.4]

**3.2.20**
**clé de CA de confiance directe**
une clé de CA de confiance directe est une clé publique d'une autorité CA de confiance directe. Elle a été obtenue et est en cours de mémorisation par une entité finale, en toute confiance et sécurité. Elle sert à vérifier des certificats sans être elle-même vérifiée au moyen d'un certificat créé par une autre autorité

Note 1 à l'article: Les autorités CA de confiance directe et les clés de CA de confiance directe peuvent varier d'une entité à l'autre. Une entité peut considérer plusieurs autorités CA comme des autorités CA de confiance directe.

[SOURCE: ISO/IEC 15945:2002, 3.5]

**3.2.21**
**distribution**
voir distribution de clés

**3.2.22**
**paramètres de domaine**
paramètres, utilisés avec un algorithme cryptographique, qui sont communs à un domaine d'utilisateurs

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.23**
**cryptage**
processus consistant à modifier le texte brut en texte chiffré à l'aide d'un algorithme cryptographique et d'une clé

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.24**
**clé éphémère**
clé cryptographique qui est générée pour chaque exécution d'un processus d'établissement de clé et qui satisfait à d'autres exigences relatives au type de clé (par exemple, unique à chaque message ou session). Dans certains cas, les clés éphémères sont utilisées plus d'une fois, dans une même session (par exemple, applications en diffusion) au cours de laquelle l'expéditeur génère une seule paire de clés éphémères par message et la clé privée est combinée séparément avec chaque clé publique de destinataire

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.25**
**clé globale**
clé conçue pour être utilisée pendant un laps de temps relativement long et généralement conçue pour être utilisée dans de nombreuses instances d'une association d'applications DLMS®/COSEM (voir aussi clé symétrique statique)

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.26**
**fonction de hachage**
fonction qui met en correspondance une chaîne binaire de longueur arbitraire avec une chaîne binaire de taille fixe. Les fonctions approuvées de hachage satisfont aux propriétés suivantes:

1) unidirectionnelle: il est irréalisable par le calcul de trouver une valeur d'entrée qui correspond avec une quelconque valeur de sortie préalablement spécifiée;
2) résistant aux collisions: Il est irréalisable par le calcul de trouver deux valeurs d'entrée distinctes qui correspondent avec la même valeur de sortie

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.27**
**valeur de hachage**
résultat obtenu par l'application d'une fonction de hachage à des informations

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.28**
**vecteur d'initialisation**
**IV**
vecteur utilisé pour définir le point de départ d'un processus de cryptographie

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.29**
**identification**
processus consistant à vérifier l'identité d'un utilisateur, processus, ou dispositif, généralement comme condition préalable à l'octroi d'un accès aux ressources dans un système TI

[SOURCE: NIST SP 800-47:2002]

**3.2.30**
**clé**
voir clé cryptographique

**3.2.31**
**agrément de clé**
(par paires) procédure d'établissement de clé au cours de laquelle le matériel de chiffrement établit les clés secrètes comme une fonction des informations fournies par les deux participants, de sorte qu'aucune partie ne puisse prédéterminer la valeur des clés secrètes de chiffrement indépendamment des apports de l'autre partie. Concept opposé à celui de transport de clé

[SOURCE: NIST SP 800-56A Rev. 2:2013]

**3.2.32**
**confirmation de clé**
procédure permettant d'assurer à une partie (le destinataire de la confirmation de clé) qu'une autre partie (l'émetteur de la confirmation de clé) détient effectivement la bonne clé secrète de chiffrement et/ou le secret partagé

[SOURCE: NIST SP 800-56A Rev. 2:2013]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.33**
**fonction de dérivation de clés**
fonction permettant de dériver le matériel de chiffrement à partir d'un secret partagé (ou d'une clé) et d'autres informations

[SOURCE: NIST SP 800-56A Rev. 2:2013]

**3.2.34**
**distribution de clés**
transport d'une clé et autre matériel de chiffrement d'une entité qui soit est propriétaire de la clé, soit génère la clé, vers une autre entité qui est conçue pour utiliser la clé

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.35**
**clé de cryptage de clé**
clé cryptographique qui est utilisée pour le cryptage ou le décryptage d'autres clés

Note 1 à l'article: Dans DLMS®/COSEM, il s'agit de la clé principale.

[SOURCE: NIST SP 800-57:2012, Partie 1, modifiée par l'ajout de la Note]

**3.2.36**
**établissement de clé**
procédure qui donne lieu à un matériel de chiffrement partagé entre différentes parties

[SOURCE: NIST SP 800-56A Rev. 2:2013]

**3.2.37**
**paire de clés**
clé publique et sa clé privée correspondante; une paire de clés est utilisée avec un algorithme à clé publique

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.38**
**révocation de clé**
fonction survenant au cours du cycle de vie du matériel de chiffrement; processus par lequel est rendu disponible à des entités affectées un avis selon lequel il convient de retirer le matériel de chiffrement de l'utilisation opérationnelle avant la fin de la période cryptographique établie de ce matériel de chiffrement

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.39**
**transport de clé**
(par paires) procédure d'établissement de clé au cours de laquelle une partie (l'expéditeur) choisit une valeur pour la clé secrète de chiffrement, puis distribue cette valeur en toute sécurité à une autre partie (le destinataire). Concept opposé à celui d'agrément de clé

[SOURCE: NIST SP 800-56A Rev. 2:2013]

**3.2.40**
**enveloppement de clé**
méthode de cryptage de matériel de chiffrement (avec les informations d'intégrité associées) qui fournit une protection par confidentialité et par intégrité à l'aide d'une clé symétrique

[SOURCE: NIST SP 800-57:2012, Partie 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.41**
**code d'authentification de message**
**MAC**
somme de contrôle cryptographique des données qui utilise une clé symétrique pour détecter les modifications accidentelles et intentionnelles de données

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.42**
**synthèse de message**
résultat de l'application d'une fonction de hachage à un message. Également appelée "valeur de hachage"

[SOURCE: FIPS PUB 186-4:2013]

**3.2.43**
**courbe nommée**
ensemble de paramètres de domaine ECDH, également appelé "courbe". Une courbe est une "courbe nommée" si les paramètres de domaine sont bien connus et définis et s'ils peuvent être identifiés par un identificateur d'objet; dans le cas contraire, elle est appelée "courbe personnalisée"

[SOURCE: RFC 5349]

**3.2.44**
**valeur unique**
valeur fonction du temps qui a tout au plus une chance négligeable de se répéter. Par exemple, la valeur unique peut être une valeur aléatoire régénérée à chaque utilisation, un horodatage, un numéro de séquence, ou une combinaison de ces éléments

[SOURCE: NIST SP 800-56A Rev. 2:2013]

**3.2.45**
**non-répudiation**
service utilisé pour garantir l'intégrité et l'origine des données de sorte qu'une tierce partie puisse vérifier leur intégrité et vérifier qu'elles proviennent d'une entité spécifique détenant la clé privée du signataire prétendu

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.46**
**mot de passe**
chaîne de caractères (lettres, nombres et autres symboles) utilisés pour authentifier une identité ou pour vérifier une autorisation d'accès, ou pour dériver des clés cryptographiques

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.47**
**texte brut**
données intelligibles qui ont un sens et qui peuvent être comprises sans appliquer de décryptage

[SOURCE: NIST SP 800-57:2012, Partie 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.48**
**clé privée**
clé cryptographique, utilisée avec un algorithme cryptographique à clé publique, qui est uniquement associée à une entité et n'est pas rendue publique. Dans un système cryptographique (public) asymétrique, la clé privée est associée à une clé publique. En fonction de l'algorithme, la clé privée peut être utilisée, par exemple, pour:

1) calculer la clé publique correspondante;
2) calculer une signature numérique qui peut être vérifiée par la clé publique correspondante;
3) décrypter des clés qui avaient été cryptées par la clé publique correspondante;
4) calculer un secret partagé au cours d'une transaction par agrément de clé.

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.49**
**protégé**
chiffré et/ou signé numériquement. La protection peut être appliquée aux APDU xDLMS et/ou aux données COSEM

**3.2.50**
**clé publique**
clé cryptographique, utilisée avec un algorithme cryptographique à clé publique, qui est uniquement associée à une entité et qui peut être rendue publique. Dans un système cryptographique (public) asymétrique, la clé publique est associée à une clé privée. La clé publique peut être connue par toute personne et, en fonction de l'algorithme, elle peut être utilisée, par exemple, pour:

1) vérifier une signature numérique émanant de la clé privée correspondante;
2) crypter des clés pouvant être décryptées à l'aide de la clé privée correspondante; ou
3) calculer un secret partagé au cours d'une transaction par agrément de clé.

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.51**
**certificat de clé publique**
structure de données qui contient un ou des identificateurs d'entité, la clé publique de l'entité (y compris une indication concernant l'ensemble des paramètres de domaine associé) et éventuellement d'autres informations, ainsi qu'une signature sur cet ensemble de données qui est générée par un acteur de confiance, c'est-à-dire une autorité de certification, associant ainsi la clé publique à ou aux identificateurs inclus

[SOURCE: NIST SP 800-56A Rev. 2:2013]

**3.2.52**
**algorithme cryptographique (asymétrique) à clé publique**
algorithme cryptographique qui utilise deux clés associées, une clé publique et une clé privée. Les deux clés ont pour caractéristique qu'il est impossible de dériver par calcul la clé privée à partir de la clé publique

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.53**
**infrastructure à clé publique**
**PKI**
cadre établi pour émettre, conserver et révoquer des certificats de clé publique

[SOURCE: NIST SP 800-57:2012, Partie 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.54**
**récepteur <transport de clé>**
partie qui reçoit la clé secrète de chiffrement par le biais d'une transaction par transport de clé. Concept opposé à celui d'expéditeur

[SOURCE: NIST SP 800-56A Rev. 2: 2013]

**3.2.55**
**révoquer un certificat**
terminer de manière prématurée la période d'exploitation d'un certificat en vigueur à une date et une heure spécifiques

[SOURCE: NIST SP 800-32:2001]

**3.2.56**
**autorité de certification racine**
dans une infrastructure à clés publiques hiérarchique, autorité de certification dont la clé publique sert de référentiel de confiance (c'est-à-dire le point de départ des chemins de confiance) pour un domaine de sécurité

[SOURCE: NIST SP 800-32:2001]

**3.2.57**
**clé secrète**
clé cryptographique qui est utilisée avec un algorithme cryptographique (symétrique) à clé secrète et qui est uniquement associée à une ou plusieurs entités et n'est pas rendue publique. L'utilisation du terme "secrète" dans ce contexte n'implique pas de niveau de classification, mais implique plutôt le besoin de protéger la clé de toute divulgation

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.58**
**services de sécurité**
mécanismes utilisés pour fournir la confidentialité, l'intégrité des données, l'authentification ou la non-répudiation des informations

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.59**
**force de la sécurité**
**(aussi "bits de sécurité")**
nombre associé à la quantité de travail (c'est-à-dire le nombre d'opérations) exigée pour casser un algorithme ou un système cryptographique

[SOURCE: NIST SP 800-56A Rev. 2:2013]

**3.2.60**
**certificat autosigné**
certificat de clé publique dont la signature numérique peut être vérifiée par la clé publique contenue dans le certificat. La signature sur un certificat autosigné protège l'intégrité des données, mais ne garantit pas l'authenticité des informations. La confiance des certificats autosignés se base sur les procédures sécurisées utilisées pour les distribuer

[SOURCE: NIST SP 800-57:2012, Partie 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.61**
**expéditeur <transport de clé>**
partie qui envoie une clé secrète de chiffrement au récepteur lors d'une transaction par transport de clé. Concept opposé à celui de récepteur

[SOURCE: NIST SP 800-56A Rev. 2:2013]

**3.2.62**
**clé de session**
clé cryptographique conçue pour être utilisée pendant un laps de temps relativement court

Note 1 à l'article:Dans DLMS®/COSEM, la clé dédiée est une clé de session.

**3.2.63**
**secret partagé**
valeur secrète calculée à l'aide d'un mécanisme d'agrément de clé et utilisée comme valeur d'entrée pour une fonction/méthode de dérivation de clés

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.64**
**génération de signature**
utilisation d'un algorithme de signature numérique et d'une clé privée pour créer une signature numérique sur des données

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.65**
**vérification de signature**
utilisation d'un algorithme de signature numérique et d'une clé publique pour vérifier une signature numérique sur des données

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.66**
**données signées**
données sur lesquelles une signature numérique a été calculée

**3.2.67**
**clé symétrique statique**
clé conçue pour être utilisée pendant un laps de temps relativement long et généralement conçue pour être utilisée dans de nombreuses instances d'une association d'applications DLMS®/COSEM

Note 1 à l'article: Dans DLMS®/COSEM, elle est appelée "clé globale".

**3.2.68**
**clé statique**
clé conçue pour être utilisée pendant un laps de temps relativement long et généralement conçue pour être utilisée dans de nombreuses instances d'un mécanisme d'établissement de clé cryptographique. Concept opposé à celui de clé éphémère

[SOURCE: NIST SP 800-57:2012, Partie 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.69**
**autorité de certification subordonnée**
dans une PKI hiérarchique, autorité de certification (CA) dont la clé de signature du certificat est certifiée par une autre CA, et dont les activités sont limitées par cette autre CA

[SOURCE: NIST SP 800-32:2001]

**3.2.70**
**clé symétrique**
clé cryptographique unique utilisée avec un algorithme à clé secrète (symétrique)

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.71**
**algorithme à clé symétrique**
algorithme cryptographique qui utilise la même clé secrète pour une opération et son complément (par exemple, cryptage et décryptage)

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.72**
**ancre de confiance**
clé publique et nom d'une autorité de certification utilisés pour valider le premier certificat d'une séquence de certificats. La clé publique de l'ancre de confiance est utilisée pour vérifier la signature sur un certificat émis par une autorité de certification d'ancres de confiance. La sécurité du processus de validation dépend de l'authenticité et de l'intégrité de l'ancre de confiance. Les ancres de confiance sont souvent distribuées en tant que certificats autosignés

[SOURCE: NIST SP 800-57:2012, Partie 1]

**3.2.73**
**acteur de confiance**
partie en laquelle une entité a confiance pour effectuer fidèlement certains services pour cette entité. Une entité peut être son propre acteur de confiance

[SOURCE: NIST SP 800-56A Rev. 2:2013]

**3.2.74**
**tiers de confiance**
tierce partie, telle qu'une CA, en laquelle ses clients ont confiance pour effectuer certains services. (En revanche, dans une transaction d'établissement de clé, les participants, parties U et V, sont considérés comme étant la première partie et la deuxième partie.)

[SOURCE: NIST SP 800-56A Rev. 2:2013]

**3.2.75**
**certificat X.509**
certificat de clé publique X.509 ou certificat d'attribut X.509, tel que défini par la norme ISO/UIT-T X.509. Plus généralement (y compris dans le présent document), un certificat X.509 fait référence au certificat de clé publique X.509

[SOURCE: NIST SP 800-57:2012, Partie 1]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.2.76**
**certificat de clé publique X.509**
certificat numérique contenant une clé publique pour une entité et un nom pour cette entité, ainsi que certaines autres informations rendues infalsifiables par la signature numérique de l'autorité de certification qui a émis le certificat, codé dans le format défini dans la norme ISO/UIT-T X.509

[SOURCE: NIST SP 800-57:2012, Partie 1]

### 3.3 Définitions et abréviations liées au mode Galois/Counter

**3.3.1**
**données supplémentaires authentifiées**
**AAD**
données entrantes de la fonction de cryptage authentifié qui sont authentifiées, mais pas cryptées

Note à l'article: L'abréviation "AAD" est dérivée du terme anglais développé correspondant "Additional Authenticated Data"

[SOURCE: NIST SP 800-38D:2007]

**3.3.2**
**décryptage authentifié**
fonction du GCM dans laquelle le texte chiffré est décrypté en texte brut, et l'authenticité du texte chiffré et des AAD est vérifiée

[SOURCE: NIST SP 800-38D:2007]

**3.3.3**
**cryptage authentifié**
fonction du GCM dans laquelle le texte brut est crypté pour obtenir un texte chiffré et une balise d'authentification est générée sur les AAD et le texte chiffré

[SOURCE: NIST SP 800-38D:2007]

**3.3.4**
**balise d'authentification**
**balise, T**
somme de contrôle cryptographique appliquée à des données, qui est conçue pour révéler les erreurs accidentelles et les modifications intentionnelles des données

[SOURCE: NIST SP 800-38D:2007]

**3.3.5**
**chiffrement par blocs**
famille paramétrée de permutations sur des chaînes binaires d'une longueur fixe; le paramètre qui détermine la permutation est une chaîne binaire appelée la clé

[SOURCE: NIST SP 800-38D:2007]

**3.3.6**
**texte chiffré**
forme cryptée du texte brut

[SOURCE: NIST SP 800-38D:2007]

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.3.7**
**champ fixe**
dans la construction déterministe des IV, champ qui identifie le dispositif ou le contexte de l'instance de la fonction de cryptage authentifié

[SOURCE: NIST SP 800-38D:2007]

**3.3.8**
**fraîche**
pour une clé récemment générée, propriété qui consiste à être différente de toutes les clés précédemment utilisées

[SOURCE: NIST SP 800-38D:2007]

**3.3.9**
**GCM**
mode Galois/Counter

[SOURCE: NIST SP 800-38D:2007]

**3.3.10**
**vecteur d'initialisation**
**IV**
valeur unique associée à l'appel d'un cryptage authentifié pour un texte brut spécifique et des AAD

Note 1 à l'article: Pour les besoins du présent document, le champ d'appel correspond au compteur d'invocations.

[SOURCE: NIST SP 800-38D:2007]

**3.3.11**
**champ d'appel**
dans la construction déterministe des IV, champ qui identifie les ensembles de valeurs d'entrée dans la fonction de cryptage authentifié dans un dispositif ou un contexte spécifique

[SOURCE: NIST SP 800-38D:2007]

**3.3.12**
**clé**
paramètre du chiffrement par blocs qui détermine la sélection de la fonction de transmission de chiffrement dans la famille de permutations

[SOURCE: NIST SP 800-38D:2007]

**3.3.13**
**texte brut**
***P***
données entrantes pour la fonction de cryptage authentifié, qui sont authentifiées et cryptées

[SOURCE: NIST SP 800-38D:2007]

**3.3.14**
**octet de contrôle de sécurité**
***SC***
octet qui fournit des informations sur le chiffrement appliqué

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**3.3.15**
**en-tête de sécurité**
***SH***
concaténation de l'octet de contrôle de sécurité (*SC*) et du compteur d'invocations: $SH = SC$ II *IC*.

## 3.4 Abréviations générales

| Abréviation | Signification |
|---|---|
| .cnf | primitive de service .confirm |
| .ind | primitive de service .indication |
| .req | primitive de service .request |
| .res | primitive de service .response |
| AA | Application Association (Association d'applications) |
| AARE | A-Associate Response (Réponse d'association d'applications) – une APDU de l'ACSE |
| AARQ | A-Associate Request (Demande d'association d'applications) – une APDU de l'ACSE |
| ACPM | Association Control Protocol Machine (Machine de protocole de commande d'association) |
| ACSE | Association Control Service Element (Élément de service de contrôle d'association) |
| AE | Application Entity (Entité d'application) |
| AES | Advanced Encryption Standard (Norme de cryptage avancé) |
| AL | Application Layer (Couche application) |
| AP | Processus d'application |
| APDU | Application Layer Protocol Data Unit (Unité de données de protocole d'application) |
| API | Application Programming Interface (Interface de programmes d'applications) |
| ASE | Application Service Element (Élément de service d'application) |
| ASO | Application Service Object (Objet de service d'application) |
| ATM | Asynchronous Transfer Mode (Mode de transfert asynchrone) |
| A-XDR | Adapted Extended Data Representation (Représentation de données étendues adaptée) |
| base_name | Attribut short_name correspondant au premier attribut ("logical_name") d'un objet COSEM |
| BER | Basic Encoding Rules (Règles de codage de base) |
| BD | Block Data (Données de bloc) |
| BN | Block Number (Numéro de bloc) |
| BNA | Block Number Acknowledged (Numéro de bloc acquitté) |
| BS | Bit String (Chaîne binaire) |
| BTS | Block Transfer Streaming (Transfert de flux par blocs) |
| BTW | Block Transfer Window (Fenêtre de transfert de blocs) |
| CA | Certification Authority (Autorité de certification) |
| CF | Control Function (Fonction de commande) |
| CL | ConnectionLess (Sans connexion) |
| class_id | Code d'identification de la classe d'interface COSEM |
| CMP | Certificate Management Protocol (Protocole de gestion des certificats). Voir RFC 4210. |
| CO | Connection-Oriented (Orienté connexion) |
| COSEM | COmpanion Specification for Energy Metering (Spécification d'accompagnement pour le comptage de l'énergie) |
| COSEM_on_IP | Profil de communication COSEM basé sur TCP-UDP/IP |
| CRC | Cyclic Redundancy Check (Contrôle de redondance cyclique) |
| CRL | Certificate Revocation List (Liste de révocation des certificats). Voir RFC 5280. |
| CSR | Certificate Signing Request (Demande de signature de certificat) |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Abréviation | Signification |
|---|---|
| DCE | Data Communication Equipment (Équipement de transmission de données; interface ou modem de communication) |
| DCS | Data Collection System (Système de collecte de données) |
| DISC | DISConnect (Déconnecter; trame de type HDLC) |
| DLMS® | Device Language Message Specification (Spécification des messages de langage d'équipement) |
| DM | Disconnected Mode (Mode déconnecté; trame de type HDLC) |
| DSA | Digital Signature Algorithm (Algorithme de signature numérique) spécifié dans le document FIPS PUB 186-4:2013 |
| DSAP | Data Link Service Access Point (Point d'accès au service liaison de données) |
| DSO | energy Distribution System Operator (Opérateur de système de distribution d'énergie) |
| DTE | Data Terminal Equipment (Équipement terminal de traitement de données; ordinateurs, terminaux ou imprimantes) |
| ECC | Elliptic Curve Cryptography (Cryptographie à courbe elliptique) |
| ECDH | Elliptic Curve Diffie-Hellman key agreement protocol (Protocole d'agrément de clé Diffie-Hellman à courbe elliptique) |
| ECDSA | Elliptic Curve Digital Signature Algorithm (Algorithme de signature numérique à courbe elliptique) spécifié dans les documents ANSI X9.62 et FIPS PUB 186-4:2013 |
| ECP | Elliptic Curve Point (Point de courbe elliptique) |
| EUI-64 | 64-bit Extended Unique Identifier (Identifiant unique étendu à 64 bits) |
| FCS | Frame Check Sequence (Séquence de contrôle de trame) |
| FDDI | Fibre Distributed Data Interface (Interface de données distribuées par fibre) |
| FE | Field Element (Élément de champ; en relation avec les algorithmes à clé publique) |
| FIPS | Federal Information Processing Standard (Norme fédérale de traitement des informations) |
| FRMR | FRaMe Reject (Rejet de trame; trame de type HDLC) |
| FTP | File Transfer Protocol (Protocole de transfert de fichiers) |
| GAK | Global Authentication Key (Clé globale d'authentification) |
| GBEK | Global Broadcast Encryption Key (Clé globale de cryptage de diffusion) |
| GBT | General Block Transfer (Transfert de bloc général) |
| GCM | Galois/Counter Mode (Mode Galois/Counter); algorithme de cryptage authentifié avec données associées |
| GMAC | Spécialisation du GCM pour la génération de codes d'authentification de messages (MAC) sur des données non cryptées |
| GMT | Greenwich Mean Time (Heure de Greenwich) |
| GSM | Global System for Mobile communications (Système mondial de communications mobiles) |
| GUEK | Global Unicast Encryption Key (Clé globale de cryptage de monodiffusion) |
| GW | GateWay (Passerelle) |
| HCS | Header Check Sequence (Séquence de contrôle d'en-tête) |
| HDLC | High-level Data Link Control (Commande de liaison de données à haut niveau) |
| HES | Head End System (Système tête de réseau), aussi appelé "système de collecte de données"<br>NOTE Le HES peut appartenir au fournisseur d'énergie ou au réseau public de distribution de l'électricité |
| HHU | Hand Held Unit (Unité portative) |
| HLS | High Level Security (Niveau de sécurité élevé; COSEM) |
| HMAC | keyed-Hash Message Authentication Code (Code d'authentification de message avec hachage par clé) spécifié dans la norme FIPS 198-1 |
| HSM | Hardware Security Module (Module de sécurité matérielle) |
| HTTP | HyperText Transfer Protocol (Protocole de transfert hypertexte) |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Abréviation | Signification |
|---|---|
| I | Information (trame de type HDLC) |
| IANA | Internet Assigned Numbers Authority (Autorité chargée de la gestion de l'adressage sur Internet) |
| IC | Interface Class (Classe d'interfaces) |
| IEEE | Institute of Electrical and Electronics Engineers (Institut des ingénieurs électriciens et électroniciens) |
| IETF | Internet Engineering Task Force (Groupe de travail d'ingénierie Internet) |
| IP | Internet Protocol (Protocole Internet) |
| ISO | International Organization for Standardization (Organisation internationale de normalisation) |
| IV | Initialization Vector (Vecteur d'initialisation) |
| KEK | Key Encrypting Key (Clé de cryptage de clé) |
| LAN | Local Area Network (Réseau local) |
| LB | Last Block (Dernier bloc) |
| LDN | Logical Device Name (Nom de dispositif logique) |
| LLC | Logical Link Control (Commande de liaison logique) — sous-couche |
| LLS | Low Level Security (Niveau de sécurité faible) |
| LNAP | Local Network Access Point (Point d'accès au réseau local) |
| LPDU | LLC Protocol Data Unit (Unité de données de protocole LLC) |
| L-SAP | LLC sublayer Service Access Point (Point d'accès au service de la sous-couche LLC) |
| LSB | Least Significant Bit (Bit de poids faible) |
| LSDU | LLC Service Data Unit (Unité de données de service LLC) |
| O | Obligatoire, utilisé conjointement aux définitions d'attributs et de méthodes |
| MAC | Medium Access Control (Contrôle d'accès au support) – sous-couche |
| MAC | Message Authentication Code (Code d'authentification de message) – cryptographie |
| MIB | Management Information Base (Base d'informations de gestion) |
| MSAP | MAC sublayer Service Access Point (Point d'accès au service de la sous-couche MAC) (dans le profil HDLC, il est égal à l'adresse HDLC) |
| MSB | Most Significant Bit (Bit de poids fort) |
| MSC | Message Sequence Chart (Diagramme de séquence de message) |
| MSDU | MAC Service Data Unit (Unité de données de service MAC) |
| N(R) | Receive sequence Number (Numéro de séquence en réception) |
| N(S) | Send sequence Number (Numéro de séquence d'envoi) |
| NDM | Normal Disconnected Mode (Mode déconnecté normal) |
| NIST | National Institute of Standards and Technology (Institut national des normes et de la technologie) |
| NNAP | Neighbourhood Network Access Point (Point d'accès aux réseaux de voisinage) |
| NRM | Normal Response Mode (Mode normal de réponse) |
| F | Facultatif, utilisé conjointement aux définitions d'attributs et de méthodes |
| OBIS | Object Identification System (Système d'identification des objets) |
| OCSP | Online Certificate Status Protocol (Protocole de statut en ligne de certificat) |
| OID | Object Identifier (Identificateur d'objet) |
| OOB | Out Of Band (Hors bande) |
| OS | Octet String (Chaîne d'octets) |
| OSI | Open System Interconnection (Interconnexion de systèmes ouverts) |
| OTA | Over The Air (Par radio) |
| P/F | Poll/Final (D'interrogation/Final) |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Abréviation | Signification |
|---|---|
| PAR | Positive Acknowledgement with Retransmission (Acquittement positif avec retransmission) |
| PDU | Protocol Data Unit (Unité de données de protocole) |
| PhL | Physical Layer (Couche physique) |
| PHSDU | PH SDU (SDU PH) |
| PKCS | Public Key Cryptography Standard (Norme cryptographique de clé publique), établie par des laboratoires RSA |
| PKI | Public Key Infrastructure (Infrastructure à clé publique) |
| PLC | Power Line Carrier (Courant porteur en ligne) |
| PPP | Point-to-Point Protocol (Protocole point à point) |
| PSDU | Physical layer Service Data Unit (Unité de données de service de la couche physique) |
| PSTN | Public Switched Telephone Network (Réseau téléphonique public commuté) |
| RA | Registration Authority (Autorité d'enregistrement) |
| RLRE | A-Release Response (Réponse de libération A) – une APDU de l'ACSE |
| RLRQ | A-Release Request (Demande de libération A) – une APDU de l'ACSE |
| RNG | Random Number Generator (Générateur de nombres aléatoires) |
| RNR | Receive Not Ready (Non prêt à recevoir; trame de type HDLC) |
| RR | Receive Ready (Prêt à recevoir; trame de type HDLC) |
| RSA | Algorithme développé par Rivest, Shamir et Adelman; spécifié dans les normes ANS X9.31 et PKCS #1 |
| SAP | Service Access Point (Point d'accès au service) |
| SDU | Service Data Unit (Unité de données de service) |
| SHA | Secure Hash Algorithm (Algorithme de hachage sécurisé); spécifié dans le document FIPS PUB 180-4:2012 |
| SNMP | Simple Network Management Protocol (Protocole simple de gestion de réseau) |
| SNRM | Set Normal Response Mode (Mode normal de réponse; trame de type HDLC) |
| STR | Streaming (Diffusion en flux) |
| tbsCertificate | To be signed certificate (Certificat à signer) |
| TCP | Transmission Control Protocol (Protocole de contrôle de transmission) |
| TDEA | Triple Data Encryption Algorithm (Triple algorithme de cryptage de données) |
| TL | Transport Layer (Couche transport) |
| TPDU | Transport layer Protocol Data Unit (Unité de données de protocole de la couche transport) |
| TWA | Two Way Alternate (Bidirectionnel à l'alternat) |
| UA | Unnumbered Acknowledge (Accusé de réception non numéroté; trame de type HDLC) |
| UDP | User Datagram Protocol (Protocole de datagramme utilisateur) |
| UI | Unnumbered Information (Information non numérotée, trame de type HDLC) |
| UNC | Unbalanced operation Normal response mode Class (Classe de mode normal de réponse en fonctionnement asymétrique) |
| USS | Unnumbered Send Status (Statut d'envoi non numéroté) |
| V(R) | Receive state Variable (Variable d'état en réception) |
| V(S) | Send state Variable (Variable d'état d'envoi) |
| VAA | Virtual Application Association (Association d'applications virtuelles) |
| WPDU | Wrapper Protocol Data Unit (Unité de données de protocole de la couche d'adaptation) |
| ASE xDLMS | extended DLMS® Application Service Element (Élément de service d'application DLMS® étendu) |
| Voir aussi la liste des abréviations spécifiques à un algorithme cryptographique dans les articles correspondants. | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 3.5 Symboles liés au mode Galois/Counter

| Symbole | Signification |
|---|---|
| *A* | Données supplémentaires authentifiées, AAD |
| *AK* | Clé d'authentification, paramètre faisant partie des AAD |
| *C* | Texte chiffré |
| *EK* | Clé de cryptage, c'est-à-dire la clé de chiffrement par blocs |
| *IC* | Compteur d'invocations, partie du vecteur d'initialisation. Voir également champ d'appel. |
| *IV* | Vecteur d'initialisation |
| len*(X)* | Longueur en bits de la chaîne binaire *X* |
| *LEN(X)* | Longueur en octets de la chaîne d'octets *X* |
| *P* | Texte brut |
| *SC* | Octet de contrôle de sécurité |
| *SH* | En-tête de sécurité |
| *Sys-T* | Titre système |
| *T* | Balise d'authentification |
| *t* | Longueur en bits de la balise d'authentification<br>NOTE Elle est égale à len*(T)* |
| *X II Y* | Concaténation de deux chaînes, *X* et *Y* |

## 3.6 Symboles liés à l'algorithme ECDSA

| Symbole | Signification |
|---|---|
| *d* | Clé privée ECDSA, qui est un entier dans l'intervalle $[1, n - 1]$. |
| $Q = (x_Q, y_Q)$ | Clé publique ECDSA. Les coordonnées $x_Q$ et $y_Q$ sont des entiers dans l'intervalle $[0, q - 1]$, et $Q = dG$. |
| *k* | Numéro secret ECDSA par message, qui est un entier dans l'intervalle $[1, n - 1]$. |
| *r* | Composant d'une signature numérique ECDSA. Il s'agit d'un entier dans l'intervalle $[1, n - 1]$. Voir la définition de *(r, s)*. |
| *s* | Composant d'une signature numérique ECDSA. Il s'agit d'un entier dans l'intervalle $[1, n - 1]$. Voir la définition de *(r, s)*. |
| *(r, s)* | Signature numérique ECDSA, dans laquelle *r* et *s* sont des composants de signature numérique. |
| *M* | Message signé à l'aide de l'algorithme de signature numérique. |
| Hash*(M)* | Résultat d'un calcul de hachage (synthèse de message ou valeur de hachage) sur le message M à l'aide d'une fonction de hachage approuvée. |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 3.7 Symboles liés aux algorithmes à agrément de clé

| Symbole | Signification |
|---|---|
| $d_{e,\ U}$, $d_{e\ ,V}$ | Clés privées éphémères de la partie U et de la partie V. Ce sont des entiers dans la plage [1, n-1]. |
| $d_{s,\ U}$, $d_{s\ ,V}$ | Clés privées statiques de la partie U et de la partie V. Ce sont des entiers dans la plage [1, n-1]. |
| $ID_{\mathrm{U}}$ | Identificateur de la partie U (l'initiateur) |
| $ID_{\mathrm{V}}$ | Identificateur de la partie V (le répondeur) |
| $\mathrm{Q}_{e,\ U}$, $\mathrm{Q}_{e\ ,V}$ | Clés publiques éphémères de la partie U et de la partie V. Il s'agit de points sur la courbe elliptique définie par les paramètres du domaine. |
| $Q_{s,\ U}$, $Q_{s\ ,V}$ | Clés publiques statiques de la partie U et de la partie V. Il s'agit de points sur la courbe elliptique définie par les paramètres du domaine. |
| U, V | Représentent les deux parties d'un mécanisme d'établissement de clé (par paires). |
| $Z$ | Secret partagé (représenté comme une chaîne d'octets) utilisé pour dériver la clé secrète de chiffrement à l'aide d'une méthode de dérivation de clés. *Source: NIST SP 800-56A Rev. 2: 2013* |

## 4 Vue d'ensemble de DLMS®/COSEM

### 4.1 Échange d'informations dans DLMS®/COSEM

#### 4.1.1 Généralités

Le paragraphe 4.1 introduit les principaux concepts d'échange d'informations dans DLMS®/COSEM.

La suite DLMS®/COSEM a pour objet la spécification d'un modèle d'objet d'interface orienté domaine de métier, destiné aux dispositifs et aux systèmes de comptage, ainsi que des services permettant d'accéder aux objets. Les profils de communication permettant le transport des messages par différents supports de communication sont également spécifiés.

L'expression "dispositifs de comptage" est une abstraction; par conséquent, "dispositif de comptage" peut se référer à tout type de dispositif pour lequel cette abstraction est appropriée.

Le modèle objet COSEM est spécifié dans l'IEC 62056-6-2:2021. Les objets COSEM donnent un aperçu de la fonctionnalité des dispositifs de comptage au moyen de leurs interfaces de communication.

Le présent document spécifie la couche application DLMS®/COSEM et les règles de spécification des profils de communication DLMS®/COSEM (voir Annexe A).

Les principales caractéristiques de l'échange de données à l'aide de DLMS®/COSEM sont les suivantes:

- les dispositifs de comptage peuvent être accessibles par différentes parties: les clients et les tierces parties;
- des mécanismes permettant de contrôler l'accès aux ressources du dispositif de comptage sont fournis; ces mécanismes sont disponibles au moyen de l'AL DLMS®/COSEM et des objets COSEM (objet "Association SN/LN", objet "Security setup");
- la sécurité et la confidentialité sont assurées par l'application d'une protection cryptographique aux messages xDLMS et aux données COSEM;
- une charge utile faible ainsi que l'efficacité sont assurées par plusieurs mécanismes, dont l'accès sélectif, le codage compact et la compression;

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- sur un site de comptage, il peut exister un dispositif de comptage unique ou plusieurs dispositifs de comptage. Dans le cas de dispositifs de comptage multiples sur un site de comptage, un seul point d'accès peut être rendu accessible;
- l'échange de données peut avoir lieu à distance ou localement. Selon les capacités du dispositif de comptage, un échange local de données et un échange distant de données peuvent être effectués simultanément sans interférer l'un avec l'autre;
- plusieurs supports de communication peuvent être utilisés sur les réseaux locaux (LN, Local Network), les réseaux de voisinage (NN, Neighbourhood Network) et les réseaux étendus (WAN, Wide Area Network).

L'élément clé permettant d'assurer que les exigences susmentionnées sont satisfaites est l'association d'applications (AA) (qui détermine le contexte de l'échange de données) fournie par l'AL DLMS®/COSEM. Pour de plus amples informations, voir les articles correspondants ci-dessous.

### 4.1.2 Modèle de communication

DLMS®/COSEM utilise les concepts du modèle d'interconnexion de systèmes ouverts (OSI) pour modéliser l'échange d'informations entre les compteurs et les systèmes de collecte de données.

NOTE Dans ce contexte, les informations comprennent les messages xDLMS et les données COSEM.

Les concepts, les noms et la terminologie utilisés ci-dessous se rapportent au modèle OSI de référence décrit dans l'ISO/IEC 7498-1:1994. Leur utilisation est présentée dans le présent article et développée davantage dans d'autres articles.

Les fonctions d'application des dispositifs de comptage et des systèmes de collecte de données sont modélisées par les processus d'application (AP).

La communication entre les AP est modélisée par la communication entre des entités d'application (AE). Une AE représente les fonctions de communication d'un AP. Il peut exister plusieurs ensembles de fonctions de communication OSI dans un AP; par conséquent, un AP unique peut être représenté par plusieurs AE. Cependant, chaque AE représente un AP unique. Une AE contient un ensemble de capacités de communication appelé "éléments de service d'application" (ASE). Un ASE est un ensemble cohérent de fonctions intégrées. Ces ASE peuvent être utilisés de manière indépendante ou combinés. Voir aussi 4.2.2.

L'échange de données entre les systèmes de collecte de données et les dispositifs de comptage se base sur le modèle client/serveur dans lequel les systèmes de collecte de données jouent le rôle du client et les dispositifs de comptage jouent le rôle du serveur. Le client envoie des demandes de service au serveur, qui envoie des réponses de service. En outre, le serveur peut initier des demandes de service non sollicitées afin d'informer le client concernant des événements ou d'envoyer des données concernant des conditions préconfigurées. Voir aussi 4.1.6.

En règle générale, les AP client et serveur sont situés dans des dispositifs séparés. Par conséquent, l'échange de messages se déroule au moyen d'une pile de protocoles, comme représenté à la Figure 1.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

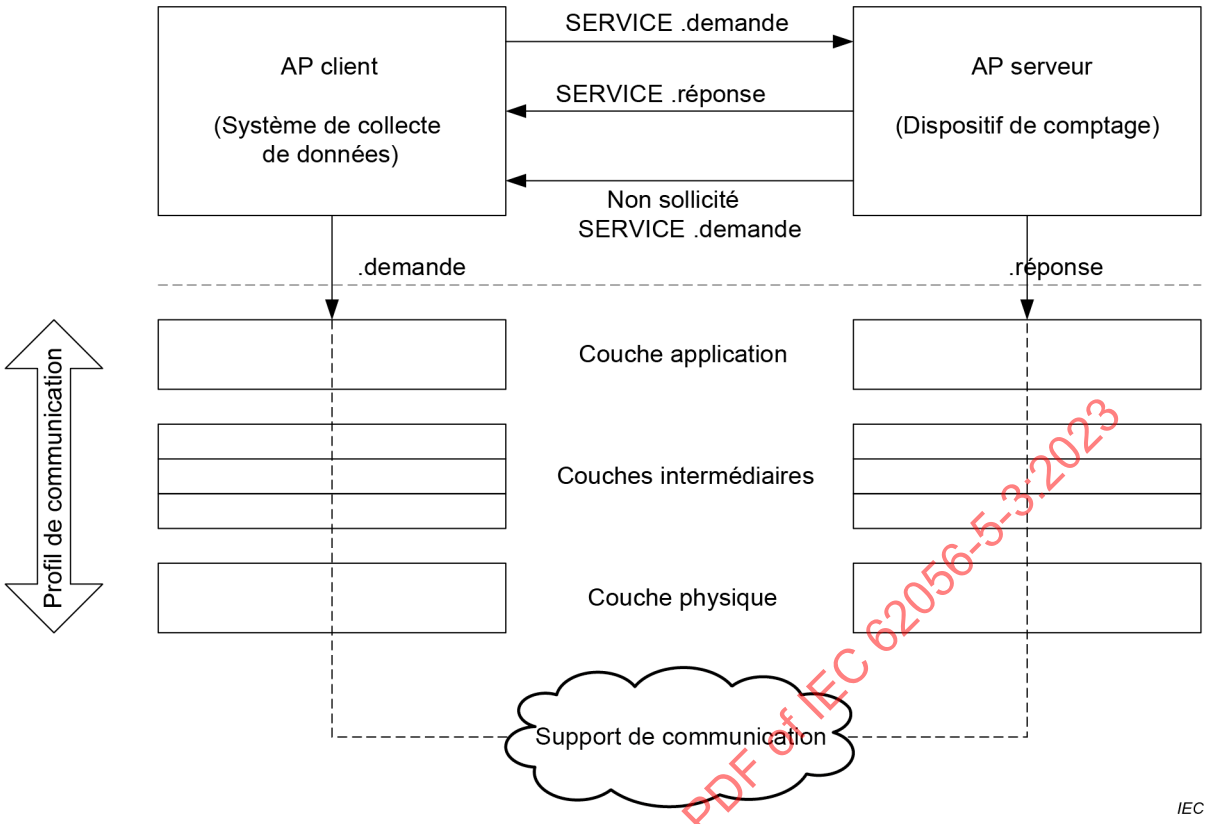



**Figure 1 – Modèle client/serveur et protocoles de communication**

### 4.1.3 Nommage et adressage

#### 4.1.3.1 Généralités

Le nommage et l'adressage sont des aspects importants des systèmes de communication. Un nom identifie une entité de communication. Une adresse identifie l'emplacement dans lequel cette entité peut être localisée. Les noms sont mappés sur les adresses; ceci est connu sous l'appellation "processus de liaison". La Figure 2 représente les principaux éléments du nommage et de l'adressage dans DLMS®/COSEM.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

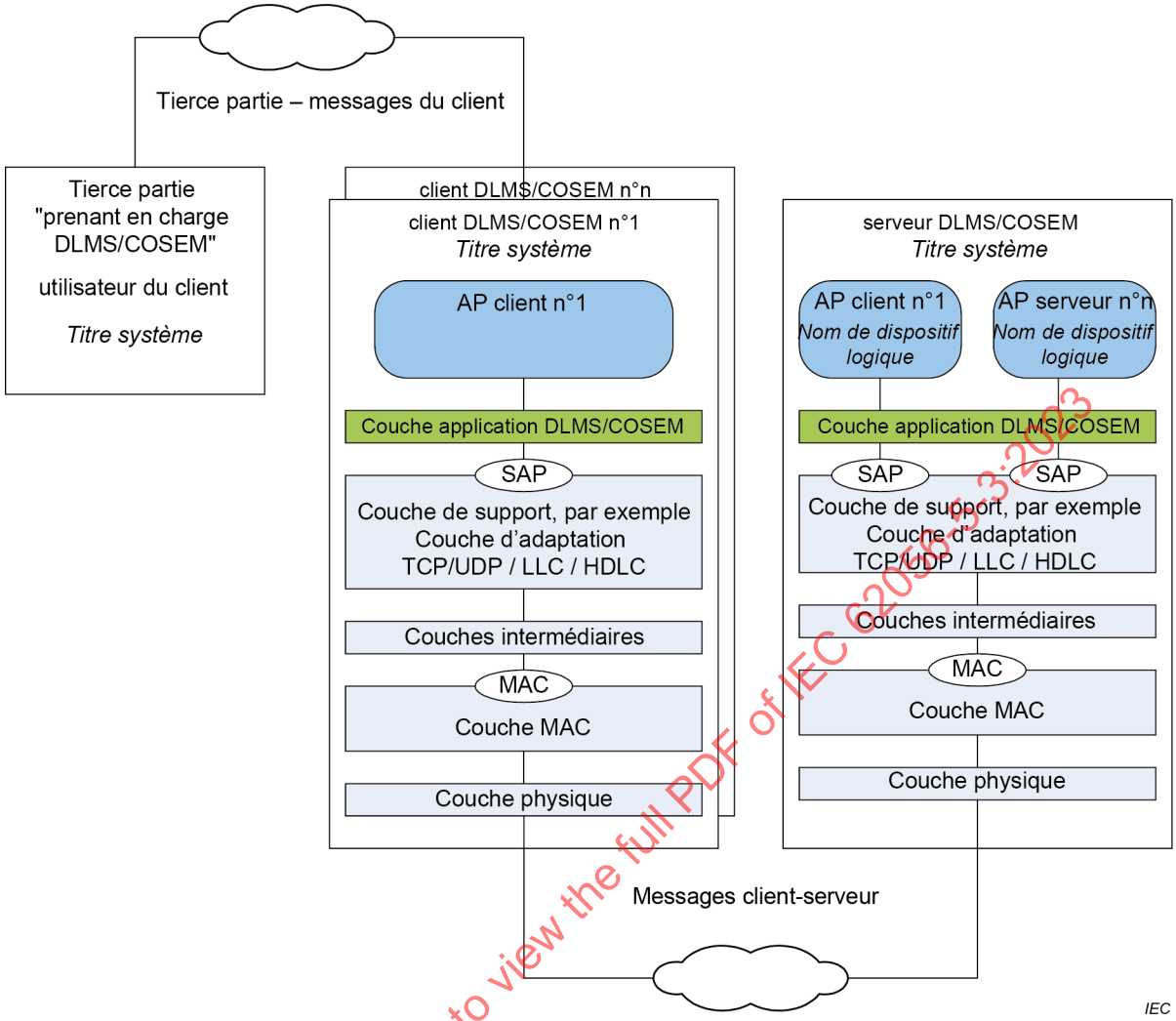



**Figure 2 – Nommage et adressage dans DLMS®/COSEM**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 4.1.3.2 Nommage

Les entités DLMS®/COSEM, y compris les clients, les serveurs et les systèmes de tierces parties doivent être uniquement désignées par leur *titre système*. Les titres du système doivent être assignés en permanence.

Les dispositifs physiques du serveur peuvent uniquement héberger un ou plusieurs dispositifs logiques (LD, Logical Device). Les LD doivent uniquement être identifiés par leur nom de dispositif logique (LDN). Les LD hébergés par le même dispositif physique partagent le même titre système. Les titres système sont spécifiés en 4.1.3.4. Les noms de dispositifs logiques sont spécifiés en 4.1.3.5.

#### 4.1.3.3 Adressage

Chaque dispositif physique doit avoir une adresse appropriée. Elle dépend du profil de communication et peut être un numéro de téléphone, une adresse MAC, une adresse de réseau IP ou une combinaison de ces éléments.

NOTE Par exemple, dans le cas du profil de communication à 3 couches, orienté connexion et basé sur HDLC, l'adresse HDLC basse est l'adresse MAC.

Les adresses de dispositifs physiques peuvent être préconfigurées ou peuvent être assignées au cours d'un processus d'enregistrement, qui implique également une liaison entre les adresses et les titres système.

Chaque client DLMS®/COSEM et chaque serveur (un dispositif logique COSEM) est lié à un point d'accès au service (SAP). Les SAP se trouvent dans la couche de support de l'AL DLMS®/COSEM. En fonction du profil de communication, le SAP peut être une adresse de couche d'adaptation TCP-UDP/IP, une adresse HDLC haute, une adresse LLC, etc. Du côté serveur, cette liaison est modélisée par l'IC "SAP Assignment" (voir IEC 62056-6-2:2021, 4.4.5).

Les valeurs des SAP côté client et côté serveur sont spécifiées dans le Tableau 1. La longueur des SAP dépend du profil de communication.

**Tableau 1 – SAP client et serveur**

| **SAP client** | |
|---|---|
| Aucune station | 0x00 |
| Processus de gestion des clients / CIASE [1] | 0x01 |
| Client public | 0x10 |
| Ouvert aux affectations d'AP client | 0x02 …0x0F |
| | 0x11 et supérieur |
| **SAP serveur** | |
| Aucune station / CIASE [1] | 0x00 |
| Dispositif logique de gestion | 0x01 |
| Réservé pour utilisation ultérieure | 0x02…0x0F |
| Ouvert aux affectations de SAP serveur | 0x10 et supérieur |
| Toutes les stations (diffusion) | Spécifique au profil de communication |
| [1] Dans le cas de profil PLC S-FSK DLMS®/COSEM, voir IEC 62056-8-3. | |
| NOTE En fonction de la couche de support, les SAP peuvent être représentés sur un ou plusieurs octets. | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 4.1.3.4 Titre système

Le titre système *Sys-T* doit identifier de manière unique chaque entité DLMS®/COSEM pouvant être un serveur, un client ou une tierce partie pouvant accéder aux serveurs par les clients. Le titre système:

- doit avoir une longueur de 8 octets;
- doit être unique.

Il convient que les premiers octets (c'est-à-dire les 3 octets le plus à gauche) comportent l'ID à trois lettres du fabricant[4]. Ils sont identiques aux trois premiers octets du nom de dispositif logique (voir 4.1.3.5). Les 5 octets restants doivent assurer l'unicité.

NOTE Il peut être dérivé, par exemple, des 12 derniers chiffres du numéro de fabrication, jusqu'à 999 999 999 999. Cette valeur est convertie en 0xE8D4A50FFF. Des valeurs supérieures, jusqu'à 0xFFFFFFFFFF (valeur décimale: 1 099 511 627 775), peuvent également être utilisées, mais ne peuvent pas être mises en correspondance avec les 12 derniers chiffres du numéro de fabrication.

Des spécifications auxiliaires spécifiques à un projet donné peuvent indiquer une structure différente. Dans ce cas, il convient que les détails soient spécifiés par l'autorité de nommage désignée comme telle pour ledit projet.

L'utilisation du titre système dans la protection cryptographique des messages xDLMS et des données COSEM est spécifiée davantage en 5.3 et en 5.7.

---

[4] Géré par la FLAG Association en collaboration avec l'UA DLMS.

Avant de pouvoir utiliser les algorithmes de sécurité cryptée (ceci exige un contexte d'application chiffré), les homologues sont tenus d'échanger les titres système. Les possibilités suivantes sont disponibles:

- au cours du processus d'enregistrement spécifique au support de communication. Par exemple, lorsque le profil PLC S-FSK est utilisé, les titres du système sont échangés pendant le processus d'enregistrement à l'aide du protocole CIASE (voir IEC 62056-8-3);
- dans tous les profils de communication, les titres système peuvent être échangés pendant l'établissement d'une AA à l'aide d'un service COSEM-OPEN (voir 6.2) contenant l'APDU AARQ/AARE. Si les titres système envoyés/reçus pendant l'établissement d'une AA ne sont pas identiques à ceux échangés pendant le processus d'enregistrement, l'AA doit être rejetée;
- par l'écriture de l'attribut *client_system_title* et par la lecture de l'attribut *server_system_title* des objets "Security setup" (voir IEC 62056-6-2:2021, 4.4.7).

En cas de communication de diffusion, seul le client envoie le titre système au serveur.

#### 4.1.3.5 Nom de dispositif logique

Le nom de dispositif logique (LDN) doit être tel que spécifié dans l'IEC 62056-6-2:2021, 4.1.4.6.

#### 4.1.3.6 Identification des utilisateurs du client

Le mécanisme d'identification des utilisateurs du client permet à un serveur d'établir une distinction entre différents utilisateurs du côté client et d'enregistrer leurs activités d'accès au compteur. Ce mécanisme est spécifié dans l'IEC 62056-6-2:2021, 4.4.2. Le nommage des utilisateurs client n'entre pas dans le domaine d'application de la présente Norme internationale.

### 4.1.4 Opération orientée connexion

L'AL DLMS®/COSEM est orientée connexion. Voir aussi 4.2.3.

Une session de communication est composée de trois phases, telles que représentées à la Figure 3:

en premier lieu, une connexion au niveau de l'application, appelée "associations d'applications" (AA), est établie entre une AE client et une AE serveur (voir aussi 4.2.3). Avant le début de l'établissement d'une AA, les PhL homologues des piles de protocoles côté client et côté serveur sont tenus d'être connectées. Les couches intermédiaires peuvent nécessiter une connexion ou non. Chaque couche nécessitant une connexion peut prendre en charge une ou plusieurs connexions simultanément;

- une fois que l'AA est établie, l'échange de messages peut commencer;
- à la fin de l'échange de données, l'AA est libérée.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

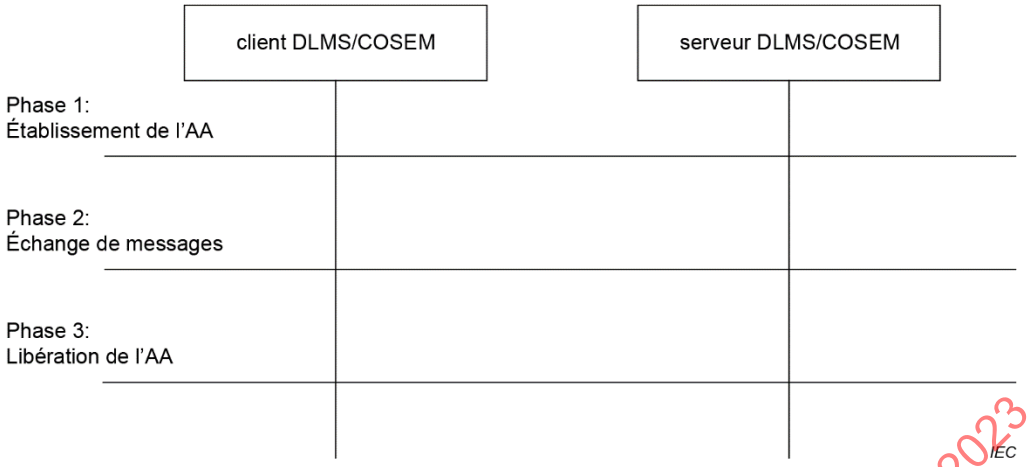



**Figure 3 – Session complète de communication dans l'environnement CO**

Pour les besoins des dispositifs très simples et des dispositifs à communication unidirectionnelle, et pour des besoins de multidiffusion et de diffusion, des AA préétablies sont également autorisées. Pour ces AA, la session entière de communication peut comprendre uniquement la phase d'échange de messages: il peut être envisagé que la phase d'établissement de connexion a déjà été effectuée auparavant. Les AA préétablies ne peuvent pas être libérées. Voir aussi 7.2.4.4.

### 4.1.5 Associations d'applications

#### 4.1.5.1 Généralités

Les associations d'applications (AA) sont des connexions logiques entre une AE client et une AE serveur. Les AA peuvent être établies à la demande d'un client à l'aide des services de l'ACSE orienté connexion de l'AL ou peuvent être préétablies. Elles peuvent être confirmées ou non confirmées. Voir aussi 4.2.3.

NOTE 1 Une AA préétablie peut être considérée comme ayant été établie dans le passé.

NOTE 2 Les serveurs ne peuvent pas lancer l'établissement d'une AA.

Un dispositif logique COSEM peut prendre en charge un ou plusieurs AA, chaque AA ayant un client différent. Chaque AA détermine le contexte dans lequel l'échange d'informations se déroule.

Une AA confirmée est proposée par le client et acceptée par le serveur dans la mesure où:

- l'utilisateur client est connu du serveur (voir 4.1.3.6);
- le contexte d'application proposé par le client (voir 4.1.5.2) est acceptable pour le serveur;
- le mécanisme d'authentification proposé par le client (voir 4.1.5.3) est acceptable pour le serveur et l'authentification est effectuée avec succès;
- les éléments du contexte xDLMS (voir 4.1.5.4) peuvent être négociés avec succès entre le client et le serveur.

Une AA non confirmée est également proposée par un client en prenant pour hypothèse une acceptation par le serveur. Aucune négociation n'a lieu. Les AA non confirmées sont utiles pour l'envoi de messages de diffusion du client aux serveurs.

Les AA sont modélisées par des objets COSEM "Association SN/LN" qui comprennent les SAP identifiant les partenaires associées, le nom du *contexte d'application*, le nom du *mécanisme d'authentification*, et le *contexte xDLMS*.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Les objets "Association SN/LN" déterminent également un ensemble spécifique de droits d'accès aux attributs et aux méthodes relatifs aux objets COSEM et ils indiquent (font référence à) un objet "Security setup" qui comporte les éléments du contexte de sécurité. Les droits d'accès et le contexte de sécurité peuvent différer dans chaque AA. Ces objets sont spécifiés dans l'IEC 62056-6-2:2021, 4.4.3, 4.4.4.

#### 4.1.5.2 Contexte d'application

Le contexte d'application détermine:

- l'ensemble des éléments de services d'application (ASE) présents dans l'AL;
- le style de référencement des attributs et des méthodes de l'objet COSEM: référencement par nom abrégé (SN, Short Name) ou par nom logique (LN). Voir aussi 4.2.4.3.1;
- la syntaxe de transfert;
- si le chiffrement est utilisé ou non.

Les contextes d'application sont identifiés par nom (voir 7.2.2.2).

#### 4.1.5.3 Authentification

Dans les systèmes de communication, l'authentification d'entité est un service de sécurité d'importance fondamentale. Le but de l'authentification d'entité est de définir si une certaine identité est bien l'identité qu'elle prétend être. Afin d'atteindre ce but, il convient qu'il existe une relation préexistante liant l'entité à un secret.

Dans DLMS®/COSEM, l'authentification a lieu pendant l'établissement de l'AA.

Dans les AA confirmées, le client (authentification unidirectionnelle) ou le client et le serveur (authentification mutuelle) peuvent s'authentifier.

Dans une AA non confirmée, seul le client peut s'authentifier.

Dans les AA préétablies, l'authentification des partenaires de communication n'est pas disponible.

Une fois que l'AA est établie, les attributs et méthodes de l'objet COSEM peuvent être accessibles à l'aide des services xDLMS soumis au contexte de sécurité qui prévaut et des droits d'accès se trouvant dans l'AA donnée.

Les mécanismes d'authentification sont spécifiés en 5.2.2.2. Les mécanismes d'authentification sont identifiés par nom (voir 7.2.2.3).

#### 4.1.5.4 Contexte xDLMS

Le contexte xDLMS détermine l'ensemble de services et de capacités xDLMS pouvant être utilisé dans une AA donnée. Voir 4.2.4.

#### 4.1.5.5 Contexte de sécurité

Le contexte de sécurité est pertinent lorsque le contexte d'application spécifie un chiffrement. Il comprend la suite de sécurité, la politique de sécurité, les clés de sécurité et d'autres matériaux de sécurité. Voir aussi 5.2.3. Il est géré par des objets "Security setup".

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 4.1.5.6 Droits d'accès

Les droits d'accès déterminent les droits du ou des clients à accéder aux attributs et méthodes de l'objet COSEM dans une AA. L'ensemble de droits d'accès dépend du rôle du client et est préconfiguré dans le serveur. Voir aussi 5.2.4.

NOTE Les rôles et les droits d'accès inhérents sont soumis aux spécifications auxiliaires spécifiques à un projet donné. Releveur de compteur, fournisseur de compteurs/de services de communication/d'énergie, fabricant, utilisateur final, etc. sont des exemples de rôles.

### 4.1.6 Type de messagerie

Les types de messagerie disponibles entre un client et un serveur DLMS®/COSEM sont représentés à la Figure 4.

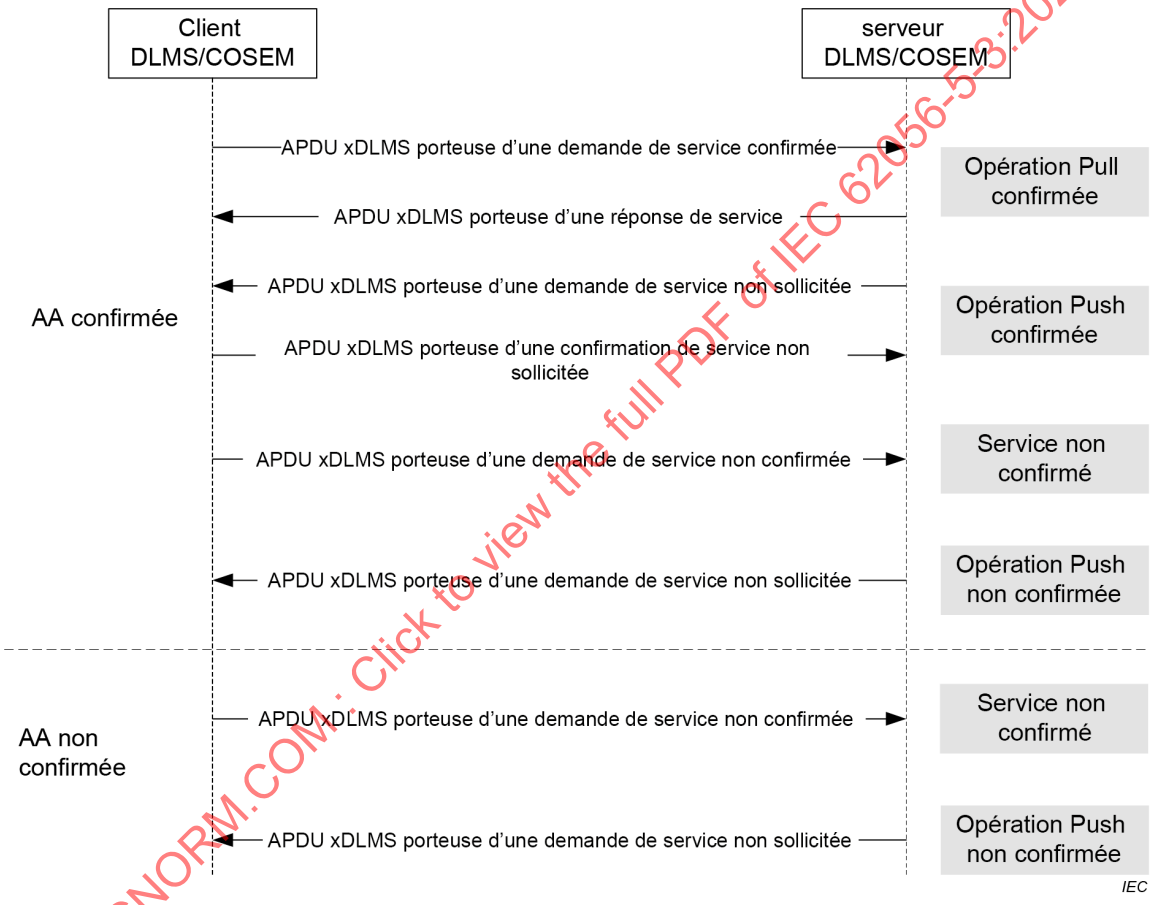



**Figure 4 – Types de messagerie DLMS®/COSEM**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Dans les AA confirmées:

- le client peut envoyer des demandes de service confirmées et le serveur répond: *opération Pull*;
- le client peut envoyer des demandes de service non confirmées. Le serveur ne répond pas;
- le serveur peut envoyer des demandes de services non sollicitées au client: *opération Push*.

NOTE Les services non sollicités peuvent être InformationReport (avec référencement SN), EventNotification (avec référencement LN) ou DataNotification (utilisé avec référencement SN et LN).

Dans les AA non confirmées:

- seul le client peut initier des demandes de service et uniquement des demandes non confirmées. Le serveur ne peut pas répondre et ne peut pas initier de demandes de service.

### 4.1.7 Échange de données entre des tierces parties et des serveurs DLMS®/COSEM

Les tierces parties (qui sont en dehors de la relation client-serveur DLMS®/COSEM) peuvent également échanger des informations avec des serveurs, en utilisant un client comme courtier. Pour la prise en charge de la sécurité de bout en bout, ces tierces parties doivent "être au fait de DLMS®/COSEM", ce qui signifie qu'elles doivent être en mesure d'envoyer au client des messages contenant des APDU xDLMS correctement formatées qui transportent des données COSEM correctement formatées, et qu'elles doivent être en mesure de traiter les messages reçus de la part du serveur par le biais du client. Voir aussi 5.2.5, Figure 14.

Les messages du serveur à la tierce partie peuvent être sollicités ou non sollicités.

### 4.1.8 Profils de communication

Les profils de communication spécifient la manière dont l'AL DLMS®/COSEM et le modèle de données COSEM modélisant le processus d'application (AP) sont pris en charge par les couches inférieures de protocole spécifiques au support de communication.

Les profils de communication comprennent un certain nombre de couches de protocole. Chaque couche a une tâche distincte et fournit des services à sa couche supérieure et utilise les services de sa ou de ses couches de support. Les AP COSEM client et serveur utilisent les services de la couche supérieure de protocole, celle de l'AL DLMS®/COSEM. Il s'agit de la seule couche de protocole contenant un ou plusieurs éléments spécifiques à COSEM: les ASE xDLMS (voir 4.2.4). Elle peut être prise en charge par toute couche en mesure de fournir les services exigés par l'AL DLMS®/COSEM. Le nombre et le type des couches inférieures dépendent du support de communication utilisé.

Un ensemble donné de couches de protocole avec l'AL DLMS®/COSEM et le modèle de l'objet COSEM constitue un profil de communication DLMS®/COSEM particulier. Chaque profil est caractérisé par les couches de protocole comprises ainsi que par leurs paramètres.

La Figure 5 représente un profil générique de communication DLMS®/COSEM, comprenant:

- le modèle de l'objet COSEM modélisant le processus d'application. Pour chaque support de communication, des classes d'interface de configuration spécifiques au support sont spécifiées;
- la couche application DLMS®/COSEM;
- la couche transport DLMS®/COSEM, présente dans les profils prenant en charge Internet;
- les couches de convergence qui lient la couche MAC à l'AL DLMS®/COSEM directement ou par le biais de la couche transport DLMS®/COSEM;
- la couche physique spécifique au support et la couche MAC;
- les gestionnaires de connexion.

Un dispositif physique unique peut prendre en charge plusieurs profils de communication pour permettre l'échange de données à l'aide de différents supports de communication. Dans de tels cas, il revient à l'AP côté client de déterminer le profil de communication qu'il convient d'utiliser.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

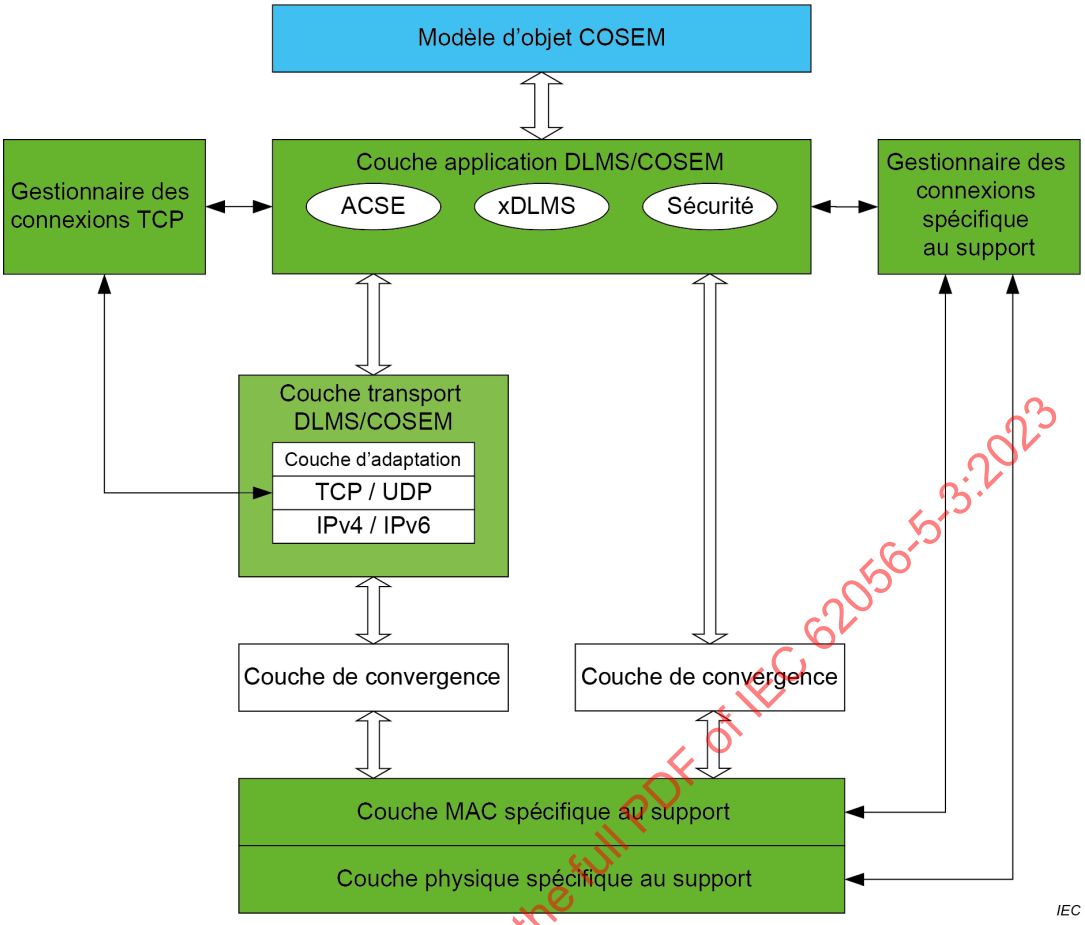



**Figure 5 – Profil générique de communication DLMS®/COSEM**

L'utilisation d'une couche application DLMS®/COSEM dans plusieurs profils de communication est spécifiée à l'Annexe A. Les normes relatives aux profils de communication sont spécifiées dans les autres parties de l'IEC 62056 relatives à la suite DLMS®/COSEM:

- le profil de communication à 3 couches, orienté connexion et basé sur HDLC, est spécifié dans l'IEC 62056-7-6;
- les profils de communication basés sur TCP-UDP/IP (COSEM_on_IP) sont spécifiés dans l'IEC 62056-9-7;
- le profil PLC S-FSK est spécifié dans l'IEC 62056-8-3;
- le profil M-Bus filaire et sans fil est spécifié dans l'IEC 62056-7-3:2017;
- le profil LPWAN qui peut être utilisé sur les réseaux LoRaWAN, spécifiés dans l'IEC 62056-8-11:–;
- le profil Wi-SUN spécifié dans l'IEC 62056-8-12:–.

NOTE D'autres profils de communication peuvent être spécifiés ultérieurement.

### 4.1.9 Modèle de système de comptage DLMS®/COSEM

La Figure 6 représente un modèle de système de comptage DLMS®/COSEM.

Les matériels de comptage sont modélisés comme un ensemble de dispositifs logiques hébergés dans un dispositif physique unique. Chaque dispositif logique représente un AP serveur et modélise un sous-ensemble de la fonctionnalité des matériels de comptage, étant donné que ces matériels sont perçus par le biais de leurs interfaces de communication. Les différentes fonctions sont modélisées à l'aide d'objets COSEM.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

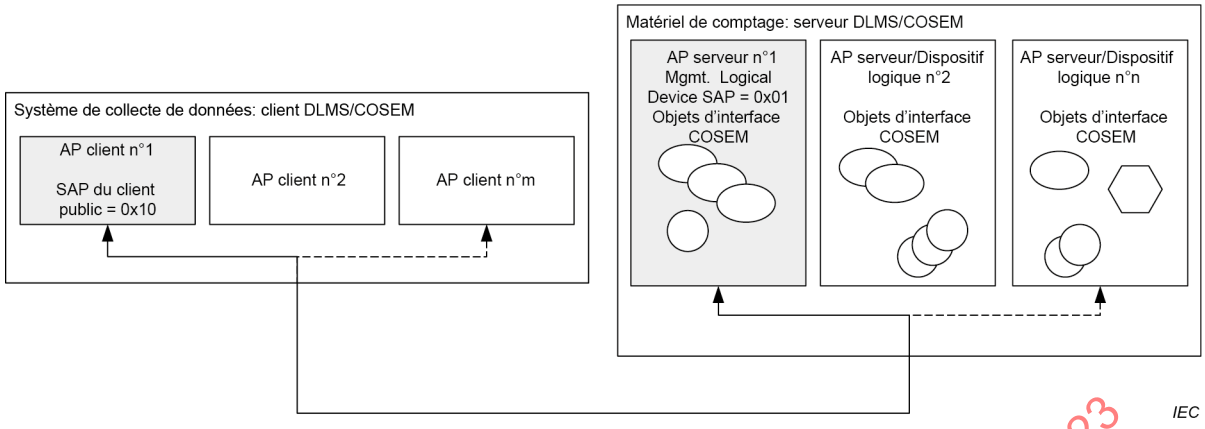



**Figure 6 – Modèle de système de comptage DLMS®/COSEM**

Les systèmes de collecte de données sont modélisés comme un ensemble d'AP client qui peut être hébergé par un ou plusieurs dispositifs physiques. Chaque AP client peut avoir différents rôles et droits d'accès, accordés par le matériel de comptage.

Les AP du client public et les AP du dispositif logique de gestion ont un rôle particulier et ils doivent toujours être présents.

Pour de plus amples informations, voir IEC 62056-6-2:2021, 4.1.7 et 4.1.8.

### 4.1.10 Modèle de serveurs DLMS®/COSEM

La Figure 7 représente le modèle de deux serveurs DLMS®/COSEM à titre d'exemple. L'un d'entre eux utilise un profil de communication à 3 couches, orienté connexion et basé sur HDLC, et l'autre utilise un profil de communication basé sur TCP-UDP/IP.

Le matériel de comptage du côté gauche comprend "n" dispositifs logiques et prend en charge le profil de communication à 3 couches, orienté connexion et basé sur HDLC.

L'AL DLMS®/COSEM est prise en charge par la couche liaison de données basée sur HDLC. Son rôle principal est de fournir un transfert de données fiable entre les couches homologues. Elle assure également l'adressage des dispositifs logiques de sorte que chaque dispositif logique soit lié à une adresse HDLC unique. Le dispositif logique de gestion est toujours lié à l'adresse 0x01. Afin de permettre la création d'un réseau local de sorte que plusieurs dispositifs de comptage sur un site de comptage donné puissent être atteints par le biais d'un point d'accès unique, une autre adresse, l'adresse physique est également fournie par la couche liaison de données. Les adresses de dispositifs logiques sont appelées "adresses HDLC hautes", tandis que l'adresse du dispositif physique est appelée "adresse HDLC basse". Voir également IEC 62056-7-6.

La PhL qui prend en charge la couche liaison de données assure la transmission de bits en série entre des dispositifs physiques hébergeant les applications client et serveur. Ceci permet l'utilisation de plusieurs interfaces, telles que RS 232, RS 485, boucle de courant 20 mA, etc. pour transférer des données localement par le biais des réseaux PSTN et GSM, etc.

Le matériel de comptage du côté droit comprend "m" dispositifs logiques.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

L'AL DLMS®/COSEM est prise en charge par la TL DLMS®/COSEM, comprenant la couche Internet TCP ou UDP et une couche d'adaptation. Le rôle principal de la couche d'adaptation consiste à adapter l'ensemble de services de style OSI, fourni par la TL DLMS®/COSEM vers et depuis des appels de fonction TCP et UDP. Elle assure également l'adressage des dispositifs logiques, en les liant à un SAP appelé "port de couche d'adaptation". Le dispositif logique de gestion est toujours lié au port de couche d'adaptation 0x01. Pour finir, la couche d'adaptation fournit des informations concernant la longueur des APDU transmises, afin d'aider l'homologue à reconnaître la fin de l'APDU. Ceci s'avère nécessaire en raison de la nature de diffusion en flux continu du TCP.

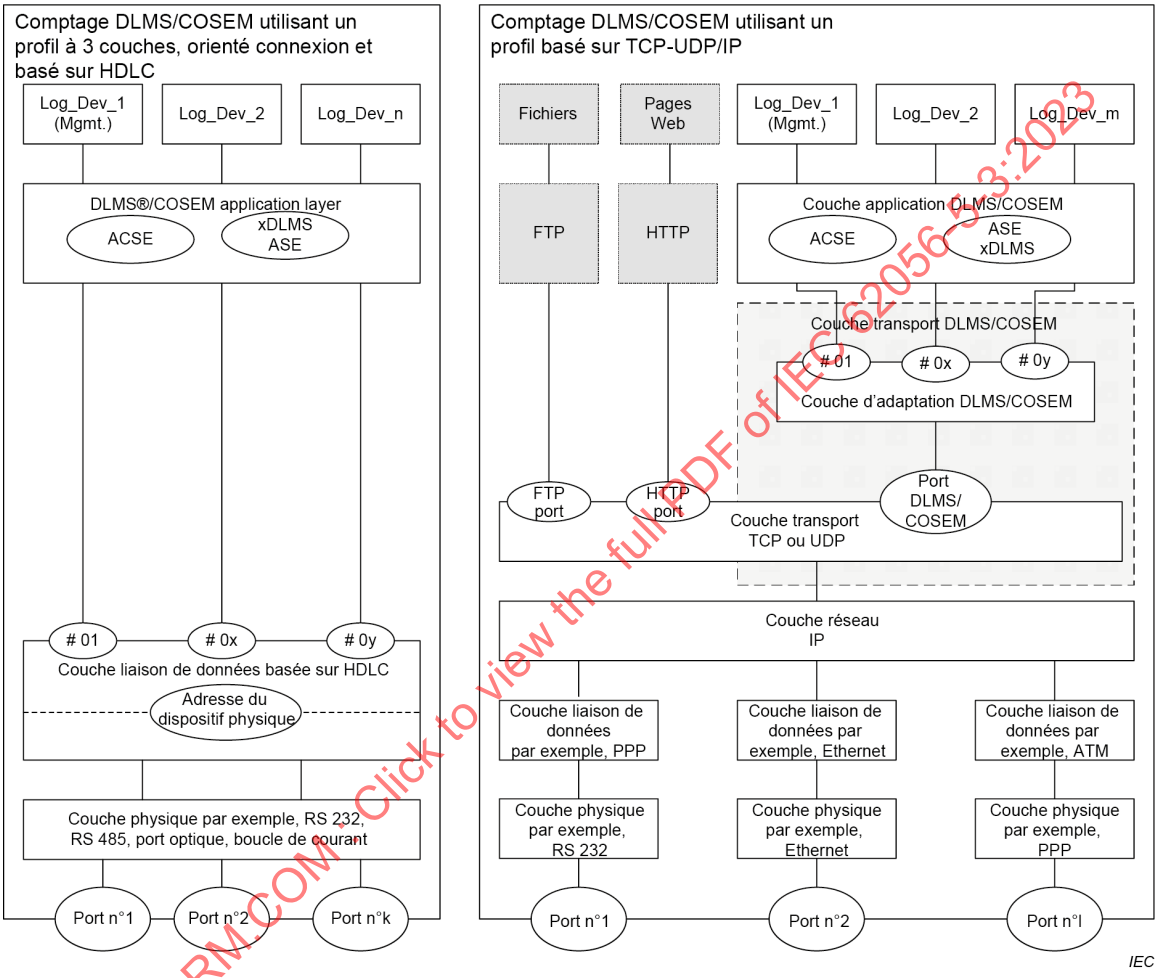



**Figure 7 – Modèle de serveur DLMS®/COSEM**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Au moyen de la couche d'adaptation, l'AL DLMS®/COSEM est liée à un numéro de port TCP ou UDP, qui est utilisé pour l'application DLMS®/COSEM. La présence des couches TCP et UDP permet l'intégration d'autres applications Internet, telles que FTP ou HTTP, liées respectivement à leurs ports normaux.

La couche TCP est prise en charge par la couche IP, qui peut elle-même être prise en charge par un ensemble quelconque de couches inférieures en fonction du support de communication à utiliser (par exemple, Ethernet, PPP, IEEE 802, ou les couches inférieures PLC prenant en charge IP, etc.).

Il est évidemment possible dans un serveur unique de mettre en œuvre plusieurs piles de protocoles, avec l'AL DLMS®/COSEM commune prise en charge par différents ensembles distincts de couches inférieures. Ceci permet au serveur d'échanger des données au moyen de différents supports de communication avec des clients situés dans différentes AA. Une telle structure est similaire à la structure d'un client DLMS®/COSEM tel que représenté ci-dessous.

### 4.1.11 Modèle d'un client DLMS®/COSEM

La Figure 8 représente à titre d'exemple le modèle d'un client DLMS®/COSEM.

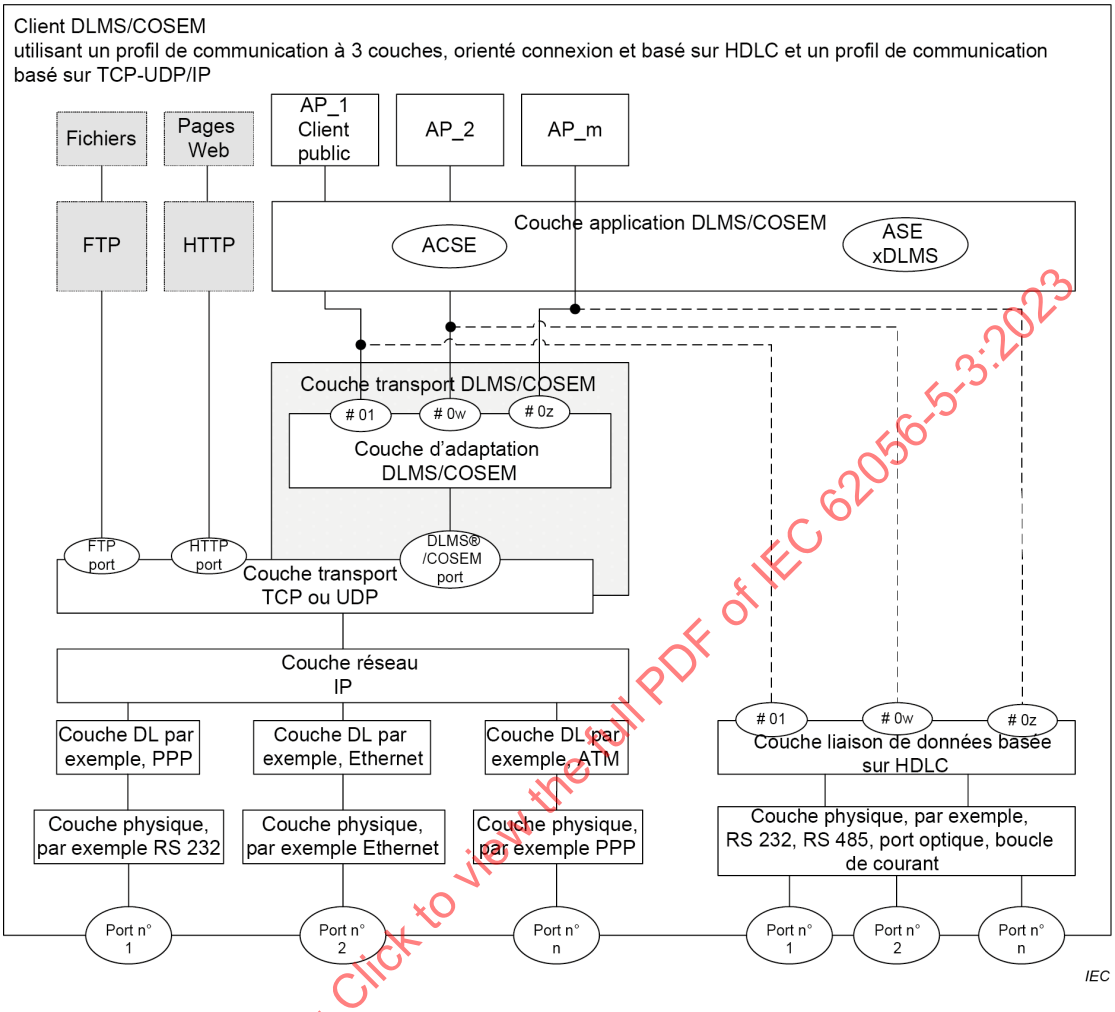



**Figure 8 – Modèle de client DLMS®/COSEM utilisant plusieurs piles de protocoles**

Le modèle du client est (évidemment) très similaire au modèle des serveurs:

- dans ce modèle particulier, l'AL DLMS®/COSEM est prise en charge soit par la couche liaison de données basée sur HDLC, soit par la TL DLMS®/COSEM, ce qui signifie que l'AL utilise les services de l'une ou de l'autre comme déterminé par les AP. En d'autres termes, les APDU sont reçues ou envoyées par l'intermédiaire de la couche de support appropriée, qui elles-mêmes utilisent les services de leur couche de support respective;
- à la différence du côté serveur, l'adressage fourni par la couche HDLC n'a qu'un niveau unique, celui des points d'accès aux services (SAP) de chaque processus d'application (AP).

Comme expliqué précédemment, les AP client et les AP serveur sont identifiés par leurs SAP. Par conséquent, une AA entre un AP client et un AP serveur peut être identifiée par une paire de SAP client et serveur.

L'AL DLMS®/COSEM peut être en mesure de prendre en charge une ou plusieurs AA simultanément. De même, les couches inférieures peuvent être en mesure de prendre en charge plusieurs connexions avec leurs couches homologues. Ceci permet l'échange de données entre des clients et des serveurs de manière simultanée par l'intermédiaire de différents ports et supports de communication.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 4.1.12 Interopérabilité et interconnectivité dans DLMS®/COSEM

Dans l'environnement DLMS®/COSEM, l'interopérabilité et l'interconnectivité sont définies entre les AE client et serveur. Une AE client et une AE serveur doivent être interopérables et interconnectables afin d'assurer l'échange de données entre les deux systèmes.

L'utilisation du modèle d'objet COSEM pour modéliser le comptage de toutes les sortes d'énergies sur tous les supports de communication assure l'interopérabilité sémantique, c'est-à-dire une signification non ambiguë partagée entre des clients et des serveurs utilisant différents supports de communication. Les éléments sémantiques sont les objets COSEM et leur nom logique, c'est-à-dire le code OBIS, la définition de leurs attributs et méthodes et les types de données pouvant être utilisés.

L'utilisation de l'AL DLMS®/COSEM sur tous les supports de communication assure l'interopérabilité syntaxique, qui est préalablement exigée pour l'interopérabilité sémantique. L'interopérabilité syntaxique comprend la capacité à établir des AA entre des clients et un serveur à l'aide de différents contextes d'application, mécanismes d'authentification, contextes xDLMS et contextes de sécurité ainsi que la structure et le codage normaux de tous les messages échangés.

L'interconnectivité est une notion de niveau protocole: afin d'être en mesure d'échanger des messages, il convient que les AE client et serveur soient interconnectables et interconnectées.

Avant que les deux AE ne puissent établir une AA, elles doivent être interconnectées. Les deux AE sont interconnectées si chaque couche de protocole homologue des deux côtés nécessitant d'être connectée est effectivement connectée. Pour qu'elles soient interconnectées, il convient que les AE client et serveur soient interconnectables et elles doivent établir les connexions exigées. Deux AE sont interconnectables si elles utilisent le même profil de communication.

Cela étant, l'interconnectivité dans DLMS®/COSEM est assurée par la capacité de l'AE DLMS®/COSEM à établir une connexion entre toutes les couches homologues nécessitant d'être connectées.

#### 4.1.13 Assurance d'interconnectivité: service d'identification de protocole

Dans DLMS®/COSEM, l'établissement d'une AA est toujours initié par l'AE client. Cependant, dans certains cas, elle peut ne pas avoir de connaissances concernant la pile de protocoles utilisée par un dispositif serveur inconnu (par exemple, lorsque le serveur a initié l'établissement d'une connexion physique). Dans de tels cas, il est nécessaire que l'AE client obtienne des informations concernant la pile de protocoles implémentée dans le serveur.

Un service spécifique au niveau de l'application est disponible à cet effet: le service d'identification de protocole. Il s'agit d'un service facultatif au niveau de l'application, permettant à l'AE client d'obtenir des informations (par suite de l'établissement d'une connexion physique) concernant la pile de protocoles implémentée dans le serveur. Le service d'identification de protocole utilise directement les services de transfert de données (PH-DATA.request/.indication) de la PhL; il contourne les autres couches de protocole. Sa prise en charge est recommandée dans tous les profils de communication ayant accès à la PhL.

#### 4.1.14 Intégration de système et installation de comptage

L'intégration de système est prise en charge par DLMS®/COSEM d'un certain nombre de manières.

Un processus possible est décrit dans le présent paragraphe.

Comme représenté à la Figure 6, la présence d'un client public (lié à l'adresse 0x10 dans tout profil) est obligatoire dans chaque système client. Son rôle principal est de révéler la structure d'un matériel de comptage inconnu (par exemple, récemment installé). Ceci se déroule dans le

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

cadre d'une AA obligatoire entre le client public et le dispositif logique de gestion, sans précaution de sécurité. Une fois la structure connue, les données peuvent être accessibles par l'utilisation de mécanismes appropriés d'authentification et de la protection cryptographique des messages xDLMS et des données COSEM.

Lorsqu'un nouveau compteur est installé dans le système, il peut générer un rapport d'événement au client. Une fois cet événement détecté, le client peut retrouver la structure interne du compteur, puis envoyer les informations de configuration nécessaires (par exemple, les barèmes tarifaires et les paramètres spécifiques à l'installation) au compteur. Cela étant, le compteur est prêt à l'emploi.

L'intégration de système est également facilitée par la disponibilité des essais de conformité DLMS®/COSEM décrits dans le Yellow Book (Livre jaune), DLMS® UA 1001-1. Ce faisant, l'implémentation correcte de la spécification dans le matériel de comptage peut être soumise à essai et certifiée.

## 4.2 Principales caractéristiques de la couche application DLMS®/COSEM

### 4.2.1 Généralités

Le présent paragraphe 4.2 donne une vue d'ensemble des principales caractéristiques de l'AL DLMS®/COSEM.

### 4.2.2 Structure de la couche application DLMS®/COSEM

La structure des couches application DLMS®/COSEM côtés client et serveur est représentée à la Figure 9.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

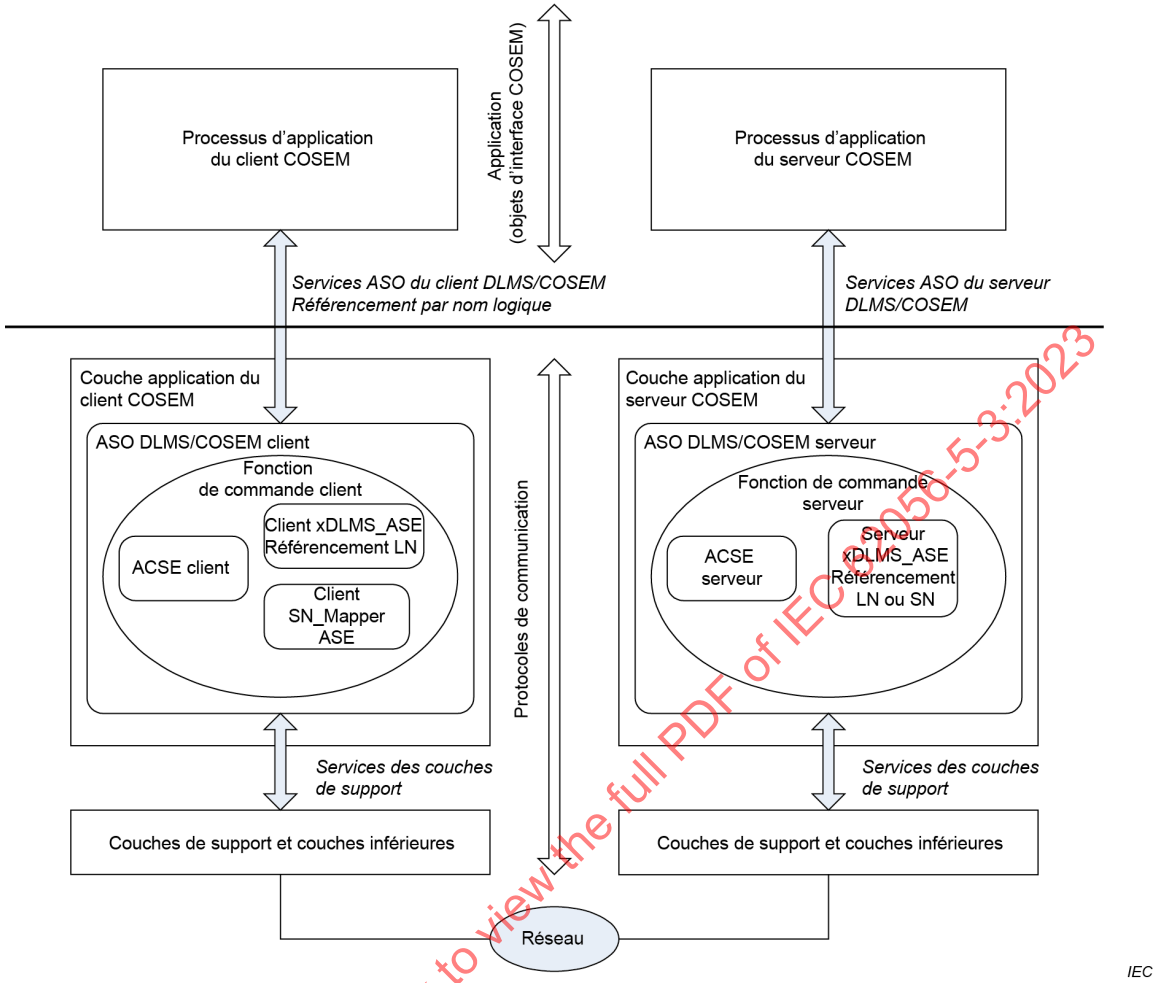



**Figure 9 – Structure des couches d'application DLMS®/COSEM**

Le composant principal de l'AL DLMS®/COSEM est l'objet de service d'application (ASO). Il fournit des services à son usager, le processus d'application (AP) COSEM et utilise les services fournis par la couche de support. Il contient les trois composants obligatoires, côté client et côté serveur:

- l'élément de service de contrôle d'association, l'ACSE;
- l'élément de service d'application DLMS® étendu, l'ASE xDLMS;
- la fonction de commande, CF.

Du côté client, il existe un quatrième élément facultatif, appelé l'ASE client SN_MAPPER.

L'ACSE fournit des services permettant d'établir et de libérer les associations d'applications (AA). Voir 4.2.3.

L'ASE xDLMS fournit des services de transport de données entre les AP COSEM. Voir 4.2.4.

L'élément de fonction de commande (CF) indique comment les services ASO appellent les primitives de service appropriées de l'ACSE, de l'ASE xDLMS et des services de la couche de support. Voir aussi 7.1.

Les ASO DLMS®/COSEM du client et du serveur peuvent tous deux contenir d'autres composants de protocole d'application facultatifs.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

L'ASE client SN_MAPPER facultatif est présent dans l'ASO de l'AL côté client, lorsque le serveur utilise le référencement SN. Il fournit une mise en correspondance entre les services utilisant le référencement LN et le référencement SN. Voir 4.2.5.

L'AL DLMS®/COSEM exécute également quelques fonctions de la couche présentation OSI:

- le codage et le décodage des APDU ACSE et des APDU xDLMS, voir aussi 0;
- en variante, la génération et l'utilisation de documents XML représentant les APDU ACSE et xDLMS;
- l'application d'une compression et d'une décompression;
- l'application, la vérification et la suppression d'une protection cryptographique.

### 4.2.3 Élément de service de contrôle d'association (ACSE)

Pour les besoins des profils de communication DLMS®/COSEM orientés connexion (CO), l'ACSE CO spécifié dans l'ISO/IEC 15953:1999 et l'ISO/IEC 15954:1999 est utilisé.

Les services fournis pour l'établissement et la libération des associations d'applications sont les suivants:

- COSEM-OPEN;
- COSEM-RELEASE;
- COSEM-ABORT.

Le service COSEM-OPEN permet d'établir des AA. Il est basé sur le service A-ASSOCIATE de l'ACSE. Il entraîne le démarrage de l'utilisation d'une AA par ces procédures ASE identifiées par la valeur des paramètres Application_Context_Name, Security_Mechanism_Name et de contexte xDLMS. Les AA peuvent être établies de différentes manières:

- les AA confirmées sont établies par l'intermédiaire d'un échange de messages, à l'aide du service COSEM-OPEN, entre le client et le serveur pour négocier les contextes. Les AA confirmées peuvent être établies entre un client et un serveur uniques;
- les AA non confirmées sont établies par l'intermédiaire d'un envoi de messages, à l'aide du service COSEM-OPEN, du client vers le serveur, en utilisant les paramètres des contextes qui sont par hypothèse pris en charge par le serveur. Les AA non confirmées peuvent être établies entre un client et un ou plusieurs serveurs;
- des AA préétablies peuvent préexister. Dans ce cas, le service COSEM-OPEN n'est pas utilisé. Le client est tenu d'avoir connaissance des contextes pris en charge par le serveur. Une AA préétablie peut être confirmée ou non confirmée.

Le service COSEM-RELEASE permet de libérer des AA. Si le service aboutit, il entraîne l'arrêt de l'utilisation de l'AA sans perte d'informations en transit (libération sans perte de données). Dans certains profils de communication (par exemple, dans le profil basé sur TCP-UDP/IP), le service COSEM-RELEASE est basé sur le service A-RELEASE de l'ACSE. Dans d'autres profils de communication (par exemple, dans le profil à 3 couches orienté connexion et basé sur HDLC), il existe une relation un-à-un entre une AA confirmée et la connexion à la couche de protocole de support. Dans ces profils, les AA peuvent être libérées simplement en déconnectant la connexion à la couche de support correspondante. Les AA préétablies ne peuvent pas être libérées.

Le service COSEM-ABORT entraîne la libération anormale d'une AA ainsi que la perte potentielle d'informations en transit. Il n'est pas basé sur le service A-ABORT de l'ACSE.

Le service COSEM-OPEN est spécifié en 6.2, le service COSEM-RELEASE en 6.3 et le service COSEM-ABORT en 6.4.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 4.2.4 Élément de service d'application xDLMS

#### 4.2.4.1 Vue d'ensemble

Pour accéder aux attributs et méthodes des objets COSEM, les services de l'ASE xDLMS sont utilisés. Ils sont basés sur la norme DLMS® IEC 61334-4-41:1996. La présente Norme internationale spécifie une plage d'extensions afin d'étendre les fonctionnalités tout en maintenant la rétrocompatibilité. Les extensions comprennent:

- des services supplémentaires (voir 4.2.4.3);
- des mécanismes supplémentaires (voir 4.2.4.4);
- des types supplémentaires de données (voir 4.2.4.5);
- un nouveau numéro de version DLMS® (voir 4.2.4.6);
- un nouveau bloc de conformité (voir 4.2.4.7);
- l'explication de la signification des paramètres PDU Size (voir 4.2.4.8).

#### 4.2.4.2 Service xDLMS Initiate

Pour établir le contexte xDLMS, le service xDLMS Initiate (spécifié dans l'IEC 61334-4-41:1996, 5.2) est utilisé. Ce service est intégré au service COSEM-OPEN (voir 6.2).

#### 4.2.4.3 Services xDLMS liés à l'objet COSEM

##### 4.2.4.3.1 Généralités

Les services xDLMS liés à l'objet COSEM permettent d'accéder aux attributs et méthodes de l'objet COSEM.

L'IEC 62056-6-2:2021, 4.1.2, spécifie deux méthodes de référencement:

- le référencement par nom logique (LN);
- le référencement par nom abrégé (SN).

Pour de plus amples informations concernant les méthodes de référencement, voir 4.2.4.4.2.

Ainsi, deux ensembles distincts de services xDLMS sont spécifiés: l'un utilisant exclusivement le référencement par nom logique (LN) et l'autre utilisant exclusivement le référencement par nom abrégé (SN). Deux ASE xDLMS différents peuvent être envisagés: le premier fournissant des services avec le référencement LN et l'autre avec le référencement SN. L'ASO client utilise toujours l'ASE xDLMS avec le référencement LN. L'ASO serveur peut utiliser l'ASE xDLMS avec le référencement LN, ou bien l'ASE xDLMS avec le référencement SN, ou les deux.

Ces services peuvent être:

- demandés / sollicités par le client;
- non sollicités: ces services sont toujours initiés par le serveur sans demande préalable de la part du client.

Les services demandés par le client peuvent également être (voir 7.3.2):

- confirmés: dans ce cas, le serveur fournit une réponse à la demande;
- non confirmés: dans ce cas, le serveur ne fournit pas de réponse à la demande.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Une DataNotification non sollicitée provenant du serveur peut également être (voir 9.3.10):

- confirmée: dans ce cas, le client fournit une réponse pour accuser réception de la DataNotification non sollicitée;
- non confirmée: dans ce cas, le serveur ne fournit pas de réponse à la demande.

Les services supplémentaires, qui ne sont pas les services basés sur DLMS® spécifiés dans l'IEC 61334-4-41:1996, sont:

- les services GET, SET, ACTION et ACCESS utilisés pour accéder aux attributs et aux méthodes des objets COSEM à l'aide du référencement LN;
- le service DataNotification utilisé par le serveur pour transmettre par opération Push des données au client;
- le service EventNotification utilisé par le serveur pour notifier au client les événements qui surviennent dans le serveur.

#### 4.2.4.3.2 Services xDLMS utilisés par le client avec le référencement LN

En cas de référencement LN, les attributs et méthodes de l'objet COSEM sont référencés par le biais de l'identificateur de l'instance de l'objet COSEM à laquelle ils appartiennent. Pour cette méthode de référencement, les services supplémentaires suivants sont spécifiés:

- le service GET est utilisé par le client pour demander au serveur de retourner la valeur d'un ou de plusieurs attributs (voir 6.6);
- le service SET est utilisé par le client pour demander au serveur de remplacer le contenu d'un ou de plusieurs attributs (voir 6.7);
- le service ACTION est utilisé par le client pour demander au serveur d'appeler une ou plusieurs méthodes. Les méthodes d'appel peuvent impliquer l'envoi de paramètres d'appel de méthode et la réception de paramètres de retour (voir 6.8);
- le service ACCESS est un service unifié pouvant être utilisé par le client pour accéder à plusieurs attributs et/ou méthodes avec une demande .request unique (voir 6.9).

Ces services peuvent être appelés par le client de manière confirmée ou non confirmée.

#### 4.2.4.3.3 Services xDLMS utilisés par le client avec le référencement SN

En cas de référencement SN, les attributs et méthodes de l'objet COSEM sont mis en correspondance avec les variables nommées selon la DLMS® qui sont spécifiées dans l'IEC 61334-4-41:1996, 10.1.2.

Les services xDLMS utilisant le référencement SN sont basés sur les services DLMS® d'accès aux variables, spécifiés dans l'IEC 61334-4-41:1996, 10.4 à 10.6. Il s'agit des services suivants:

- le service Read est utilisé par le client pour demander au serveur de retourner la valeur d'un ou de plusieurs attributs ou pour appeler une ou plusieurs méthodes lorsque les paramètres de retour sont attendus. Il s'agit d'un service confirmé. Voir 6.14;
- le service Write est utilisé par le client pour demander au serveur de remplacer le contenu d'un ou de plusieurs attributs ou d'appeler une ou plusieurs méthodes lorsqu'aucun paramètre de retour n'est attendu. Il s'agit d'un service confirmé. Voir 6.15;
- le service UnconfirmedWrite est utilisé par le client pour demander au serveur de remplacer le contenu d'un ou de plusieurs attributs ou d'appeler une ou plusieurs méthodes lorsqu'aucun paramètre de retour n'est attendu. Il s'agit d'un service non confirmé. Voir 6.16.

De nouvelles variantes du paramètre de service Variable_Access_Specification (voir 6.13), les services Read.response et Write.response, ont été ajoutées pour prendre en charge l'accès sélectif (voir 4.2.4.3.5) et le transfert de bloc (voir 4.2.4.4.5).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 4.2.4.3.4 Services non sollicités

Les services non sollicités sont initiés par le serveur dans des conditions prédéfinies, par exemple, à des instants programmés, déclenchements ou événements, afin d'informer le client de la valeur d'un ou de plusieurs attributs, comme s'ils avaient été demandés par le client.

Pour prendre en charge une opération Push, le service DataNotification est disponible (voir 6.10). Il peut être utilisé dans les contextes d'application à l'aide du référencement SN ou du référencement LN.

NOTE Le service DataNotification est utilisé conjointement aux objets COSEM "Push setup" (voir IEC 62056-6-2:2021, 4.4.8).

Pour prendre en charge la notification d'événements, les services non sollicités suivants sont disponibles:

- avec référencement LN, le service EventNotification (voir 6.11);
- avec référencement SN, le service InformationReport (voir 6.17). Ce service est basé sur l'IEC 61334-4-41:1996, 10.7.

#### 4.2.4.3.5 Accès sélectif

Dans le cas de certaines classes d'interface COSEM, un accès sélectif aux attributs est disponible, ce qui signifie que tout l'attribut ou une partie sélectionnée de celui-ci peut être accessible, selon ce qui est exigé. À cet effet, les sélecteurs et les paramètres d'accès sont spécifiés dans le cadre de la spécification des attributs concernés.

Pour utiliser cette option, les services liés aux attributs peuvent être appelés avec ces paramètres de sélection d'accès. Dans le cas du référencement LN, cette fonctionnalité est appelée Accès sélectif (voir 6.6 et 6.7). Il s'agit d'une fonctionnalité négociable (voir 7.3.1). Dans le cas du référencement SN, cette fonctionnalité est appelée Accès paramétré (voir 6.14, 6.15 et 6.16). Il s'agit d'une fonctionnalité négociable (voir 7.3.1).

#### 4.2.4.3.6 Références multiples

Dans un appel de services lié à un objet COSEM, il est possible de faire référence à une ou plusieurs variables nommées, une ou plusieurs méthodes et/ou un ou plusieurs attributs. L'utilisation de références multiples est une fonctionnalité négociable. Voir 7.3.1.

#### 4.2.4.3.7 Référencement Attribute_0

Avec les services GET, SET et ACCESS, une fonctionnalité spéciale appelée référencement Attribute_0 est disponible. Par convention, les attributs des objets COSEM sont numérotés de 1 à n, où Attribute_1 est le nom logique de l'objet COSEM. Attribute_0 a une signification spéciale: il référence tous les attributs possédant un index positif (attributs publics). L'utilisation du référencement Attribute_0 avec le service GET est expliquée en 6.6, et en 6.7 pour un référencement avec le service SET et en 6.9 pour un référencement avec le service ACCESS.

NOTE Comme indiqué dans l'IEC 62056-6-2:2021, 4.1.2, les fabricants peuvent ajouter des méthodes et/ou des attributs propriétaires à n'importe quel objet, à l'aide de numéros négatifs.

Le référencement Attribute_0 est une fonctionnalité négociable (voir 7.3.1).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 4.2.4.4 Mécanismes supplémentaires

##### 4.2.4.4.1 Vue d'ensemble

xDLMS spécifie plusieurs nouveaux mécanismes (en comparaison à DLMS®, comme spécifié dans l'IEC 61334-4-41:1996) pour améliorer la fonctionnalité, la flexibilité et l'efficacité. Les mécanismes supplémentaires sont:

- le référencement à l'aide de noms logiques;
- l'identification des appels de services;
- le traitement prioritaire;
- le transfert de longs messages d'application;
- des messages xDLMS composables;
- la compression et la décompression;
- la protection cryptographique générale;
- le transfert de bloc général.

##### 4.2.4.4.2 Méthodes de référencement et mise en correspondance de services

Pour accéder aux attributs et méthodes de l'objet COSEM avec les services xDLMS, ces attributs et méthodes sont tenus d'être référencés. Comme mentionné précédemment en 4.2.4.3.1, le paragraphe 4.1.2 de l'IEC 62056-6-2:2021 spécifie deux méthodes de référencement:

- le référencement par nom logique (LN);
- le référencement par nom abrégé (SN).

En cas de référencement LN, les attributs et méthodes de l'objet COSEM sont référencés par le biais du nom logique de l'identificateur de l'instance de l'objet COSEM auquel ils appartiennent. En cas de référencement SN, les attributs et méthodes de l'objet COSEM sont mis en correspondance avec des variables nommées selon la DLMS®.

En conséquence, deux ASE xDLMS sont spécifiées: un utilisant les services xDLMS avec le référencement LN et un utilisant les services xDLMS avec le référencement SN.

Du côté client, afin de traiter les différentes méthodes de référencement de manière transparente pour l'AP, l'AL utilise l'ASE xDLMS avec le référencement LN. L'utilisation d'un ensemble unique et normalisé de services entre les AP COSEM client et le protocole de communication, en masquant les caractéristiques des serveurs DLMS®/COSEM utilisant différentes méthodes de référencement, permet d'indiquer une interface de programmes d'applications (API). Il s'agit d'une interface explicitement spécifiée qui correspond à cet ensemble de services pour des applications s'exécutant dans un environnement informatique donné (par exemple, Windows, UNIX, etc.). À l'aide de cette spécification (publique) d'API, des applications client peuvent être développées sans connaître les caractéristiques d'un serveur donné.

Du côté serveur, l'ASE xDLMS avec référencement LN, ou bien l'ASE xDLMS avec référencement SN, ou ces deux ASE xDLMS, peuvent être utilisées.

Dans le cas des AA confirmées, la méthode de référencement est négociée durant la phase d'établissement de l'AA par l'intermédiaire du contexte d'application COSEM. Il ne doit pas changer pendant la durée de vie de l'AA établie. L'utilisation de services LN ou SN avec une AA donnée est exclusive.

Dans le cas d'AA non confirmées et préétablies, l'AL du client est susceptible de connaître la méthode de référencement prise en charge par le serveur.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Lorsque le serveur utilise le référencement LN, les services sont identiques des deux côtés. Lorsque le serveur utilise le référencement SN, l'ASE client SN_Mapper dans le client mappe le référencement SN sur le référencement LN, ou inversement. Voir 4.2.2 et 4.2.5.

#### 4.2.4.4.3 Identification des appels de services: paramètre Invoke_Id

Dans le modèle client/serveur, les demandes sont envoyées par le client et les réponses par le serveur. Le client est autorisé à envoyer plusieurs demandes avant de recevoir la réponse aux précédentes, dans la mesure où ceci est admis par les couches inférieures.

Par conséquent, pour être en mesure d'identifier quelle réponse correspond à quelle demande, il est nécessaire d'inclure une référence dans la demande.

Le paramètre Invoke_Id est utilisé à cet effet. La valeur de ce paramètre est attribuée par le client de sorte que chaque demande contient un Invoke_Id différent. Le serveur doit copier le paramètre Invoke_Id dans la réponse correspondante.

Dans les services ACCESS et DataNotification (voir 6.9 et 6.10), le paramètre Long-Invoke-Id est utilisé au lieu du paramètre Invoke_Id.

Le service EventNotification ne contient pas le paramètre Invoke_Id.

Cette fonctionnalité est disponible uniquement avec le référencement LN.

#### 4.2.4.4.4 Traitement prioritaire

Pour les services de transfert de données utilisant le référencement LN, deux niveaux de priorité sont disponibles: normal (FALSE) et élevé (TRUE). Cette fonctionnalité permet de recevoir une réponse à une nouvelle demande avant que la réponse à une demande précédente ne soit achevée.

En règle générale, le serveur traite les demandes de service entrantes dans l'ordre de réception (FIFS – First In, First Served ou Premier arrivé, premier traité). Cependant, une demande pour laquelle le paramètre Priority est défini sur élevé (TRUE) est traitée avant les demandes précédentes définies sur normal (FALSE). L'indicateur de priorité de la réponse est identique à celui de la demande correspondante. La gestion de la priorité est une fonctionnalité négociable (voir 7.3.1).

NOTE 1 Étant donné que les appels de services sont identifiés par un Invoke_Id, les services dont la priorité est identique peuvent être traités dans n'importe quel ordre.

NOTE 2 Si la fonctionnalité n'est pas prise en charge, les demandes dont la priorité est ÉLEVÉE sont traitées avec une priorité NORMALE.

Cette fonctionnalité n'est pas disponible avec les services utilisant le référencement SN. Le serveur traite les services sur une base FIFS.

#### 4.2.4.4.5 Transfert de messages longs

Les primitives de service xDLMS sont transportées sous une forme codée par des APDU xDLMS. Cette forme codée peut être plus longue que le paramètre Client/Server Max Receive PDU Size négocié. Pour transférer de longs messages, deux mécanismes de transport sont disponibles:

a) le mécanisme de transfert de bloc général (GBT) spécifié en 4.2.4.4.9;

b) le mécanisme de transfert de blocs spécifique au service. Ce mécanisme est disponible avec les services GET, SET, ACTION, Read et Write. Dans ce cas, les appels des primitives de service contiennent uniquement une partie (un bloc) des données (par exemple, les valeurs d'attribut) afin que la forme codée soit contenue dans une seule APDU.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

L'utilisation du mécanisme de transfert de bloc général ou de transfert de blocs spécifique au service est une fonctionnalité négociable (voir 7.3.1).

Une APDU susceptible d'être contenue dans le paramètre Client/Server Max Receive PDU Size négocié peut être trop longue pour être contenue dans une trame/un paquetage unique de la couche de protocole de support. Ces APDU peuvent être transportées si la couche de protocole de support fournit des segmentations.

Le mécanisme de transfert de blocs spécifique au service est disponible avec:

- les services confirmés utilisant le référencement LN: GET, SET, ACTION;
- les services confirmés utilisant le référencement SN: Read, Write;

et les APDU associées.

Le transfert de blocs spécifique au service n'est pas disponible avec:

- les services non confirmés;
- les services non sollicités (DataNotification, EventNotification et InformationReport);
- le service ACCESS;

et les APDU associées.

Avec le transfert de blocs spécifique au service, la réception de chaque bloc doit être acquittée avant que le bloc suivant puisse être envoyé. Aucun mécanisme de récupération des blocs perdus n'est disponible. Lorsqu'un chiffrement à clé globale ou à clé dédiée est exigé, il s'applique à l'APDU transportant un bloc (ou un acquittement de bloc) du message long. Cela crée des surcoûts de calcul et de transport considérables. La signature numérique spécifique au service n'est pas disponible.

Inversement, le mécanisme GBT peut être utilisé avec n'importe quelle APDU xDLMS, y compris les APDU general-ciphering et general-signing. Il assure le transfert bidirectionnel de blocs, la diffusion en flux et la récupération des blocs perdus. Lorsque la protection cryptographique est exigée, elle s'applique à l'APDU complète, puis l'APDU protégée est transférée en blocs, comme spécifié en 4.2.4.4.6.

#### 4.2.4.4.6 Messages xDLMS composables

Les trois aspects les plus importants du traitement de messages xDLMS sont le codage/décodage, l'application, la vérification/suppression de la protection cryptographique et le transfert de blocs.

Le concept de messages xDLMS composables distingue les trois aspects, comme représenté à la Figure 10. Voir aussi Figure 36.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

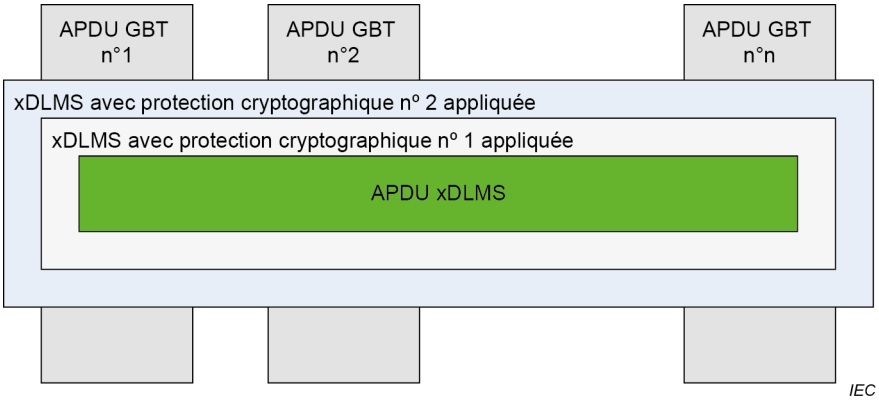



**Figure 10 – Concept de messages xDLMS composables**

Une fois que l'APDU correspondant à la primitive de service appelée par l'AP est générée par l'AL, le mécanisme de protection générale peut être utilisé pour appliquer la protection cryptographique. Lorsqu'une APDU non protégée ou protégée est trop longue par rapport à la taille d'APDU négociée, le mécanisme de transfert de bloc général peut être appliqué.

Ces mécanismes peuvent être appliqués avec toutes les APDU xDLMS.

NOTE 1 Avec les services GET, SET, ACTION, EventNotification, Read et Write et les APDU UnconfirmedWrite et InformationReport, la protection cryptographique spécifique au service est disponible à l'aide des types de protections et des APDU spécifiques au service.

NOTE 2 Avec les services GET, SET, ACTION, Read et Write et les APDU UnconfirmedWrite, le transfert de blocs spécifique au service est disponible à l'aide des types de demandes/réponses et des APDU spécifiques au service.

#### 4.2.4.4.7 Compression et décompression

Afin d'optimiser l'utilisation du support de communication, il est possible de compresser les APDU xDLMS à envoyer et de décompresser les APDU xDLMS reçues. Pour plus de détails, voir 5.3.6.

#### 4.2.4.4.8 Protection générale

Ce mécanisme peut être utilisé pour appliquer une protection cryptographique à une quelconque APDU xDLMS; ceci permet l'application de multiples couches de protection entre le client et le serveur ou entre une tierce partie et le serveur. Voir aussi 5.2.5.

À cet effet, les APDU suivantes sont disponibles (voir 5.7.2.3):

- les APDU general-ded-ciphering et les APDU general-glo-ciphering;
- les APDU general-ciphering;
- l'APDU general-signing.

L'utilisation du mécanisme général de protection est une fonctionnalité négociable (voir 7.3.1).

#### 4.2.4.4.9 Transfert de bloc général (GBT)

Le mécanisme GBT peut être utilisé pour transférer une quelconque APDU xDLMS courte ou longue dans des blocs. Avec le GBT, les blocs sont transportés par les APDU de GBT au lieu des APDU spécifiques au service "avec blocs de données".

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le mécanisme GBT prend en charge le transfert bidirectionnel de blocs, la diffusion en flux et la récupération des blocs perdus:

- le transfert de blocs bidirectionnel signifie que pendant qu'une partie envoie des blocs, l'autre partie non seulement confirme les blocs reçus, mais aussi, si elle a des blocs à envoyer, elle peut les envoyer pendant qu'elle continue de recevoir des blocs;

  NOTE Le transfert de blocs bidirectionnel est utile lorsqu'il est nécessaire de transporter des paramètres de services longs dans les deux directions.

- la diffusion en flux signifie que plusieurs blocs peuvent être envoyés (transmis en continu) par une partie sans acquittement de chaque bloc de la part de l'autre partie;
- la récupération de blocs perdus signifie que si la réception d'un bloc envoyé n'est pas confirmée, ce bloc peut être envoyé à nouveau. Si la diffusion en flux est utilisée, la récupération des blocs perdus a lieu à la fin de chaque fenêtre de flux.

Le mécanisme GBT est géré par l'AL utilisant les paramètres de flux du transfert de blocs, spécifiés en 6.5.

Le protocole du mécanisme de transfert de bloc général est spécifié en 0.

Le mécanisme GBT prend en charge les cas d'utilisation suivants:

a) demande courte du client qui conduit à une réponse longue provenant du serveur: la demande peut être envoyée avec ou sans utilisation du GBT. Si elle est envoyée à l'aide du GBT, le client peut faire connaître sa taille de fenêtre GBT privilégiée lorsque l'échange démarre. Le serveur répond à l'aide du GBT;

b) demande longue du client: le client envoie la demande et le serveur envoie la réponse en utilisant le GBT, quelle que soit la longueur de la réponse;

c) demande longue non confirmée du client: le client envoie la demande en utilisant le GBT;

d) demande longue non sollicitée du serveur: le serveur envoie le message non sollicité en utilisant le GBT.

#### 4.2.4.5 Types supplémentaires de données

Les types supplémentaires de données sont spécifiés à l'Article 8 et à l'Article 9.

#### 4.2.4.6 Numéro de version xDLMS

Le nouveau numéro de version DLMS®, qui correspond à la première version de l'ASE xDLMS, est 6.

#### 4.2.4.7 Bloc de conformité xDLMS

Le bloc de conformité xDLMS permet les implémentations de serveur DLMS®/COSEM optimisées, avec des fonctionnalités étendues. Il se distingue du bloc de conformité DLMS® par sa balise "Application 31". Voir 7.3.1, Article 8 et Article 9.

Le bloc de conformité xDLMS fait partie du contexte xDLMS.

En cas d'AA confirmées, le bloc de conformité est négocié au cours de la phase d'établissement des AA par le biais du contexte xDLMS. Il ne doit pas changer pendant la durée de vie de l'AA établie.

En cas d'AA non confirmées et préétablies, il est prévu que l'AL client ait connaissance du bloc de conformité pris en charge par le serveur.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 4.2.4.8 Max PDU size

Pour clarifier la signification des paramètres Max PDU Size utilisables par le client et le serveur, les modifications présentées dans le Tableau 2 ont été effectuées. Le service xDLMS Initiate utilise ces noms pour les paramètres PDU Size.

**Tableau 2 —Explication de la signification des paramètres PDU Size pour DLMS®/COSEM**

| avant: | maintenant: |
|---|---|
| **IEC 61334-4-41:1996, 5.2.2, Tableau 3** | |
| Proposed Max PDU Size | Client Max Receive PDU Size |
| Negotiated Max PDU Size | Server Max Receive PDU Size |
| **IEC 61334-4-41:1996, 5.2.3, 7e alinéa** | |
| Le paramètre Proposed Max PDU Size (taille maximale proposée de la PDU), de type Unsigned16 (nombre non signé de 16 bits), propose une longueur maximale exprimée en octets, pour les APDU de DLMS® échangées. La valeur proposée dans une demande Initiate doit être suffisante pour toujours permettre la transmission de la PDU "Initiate Error" (erreur d'initialisation). | Le paramètre Client Max Receive PDU Size, de type Unsigned16, contient la longueur maximale, exprimée en octets, pour une APDU DLMS® pouvant être envoyée par le serveur. Le client va jeter toute PDU reçue dont la taille est supérieure à cette longueur maximale. La valeur doit être suffisamment élevée pour toujours permettre la transmission des APDU AARE.<br>Les valeurs inférieures à 12 sont réservées. La valeur 0 indique qu'il n'existe aucune limite pour la taille de PDU. |
| **IEC 61334-4-41:1996, 5.2.3, dernier alinéa** | |
| Le paramètre Negotiated Max PDU Size (Taille maximale négociée de la PDU), de type Unsigned16, contient la longueur maximale exprimée en octets, pour les APDU de DLMS® échangées. Une PDU qui serait plus longue que cette longueur maximale ne serait pas prise en considération. La longueur maximale est traitée comme le minimum de Proposed Max PDU Size et le maximum de la taille de PDU que le VDE-handler peut supporter. | Le paramètre Server Max Receive PDU Size, de type Unsigned16, contient la longueur maximale exprimée en octets pour une APDU DLMS® pouvant être envoyée par le client. Le serveur supprime toute PDU reçue dont la taille est supérieure à cette longueur maximale.<br>Les valeurs inférieures à 12 sont réservées. La valeur 0 indique qu'il n'existe aucune limite pour la taille de PDU. |

### 4.2.5 Services de gestion de couche

Les services de gestion de couche ont une importance locale uniquement. Par conséquent, la spécification de ces services ne relève pas du domaine d'application du présent document.

Le service SetMapperTable spécifique est défini en 6.18.

### 4.2.6 Récapitulatif des services de la couche application DLMS®/COSEM

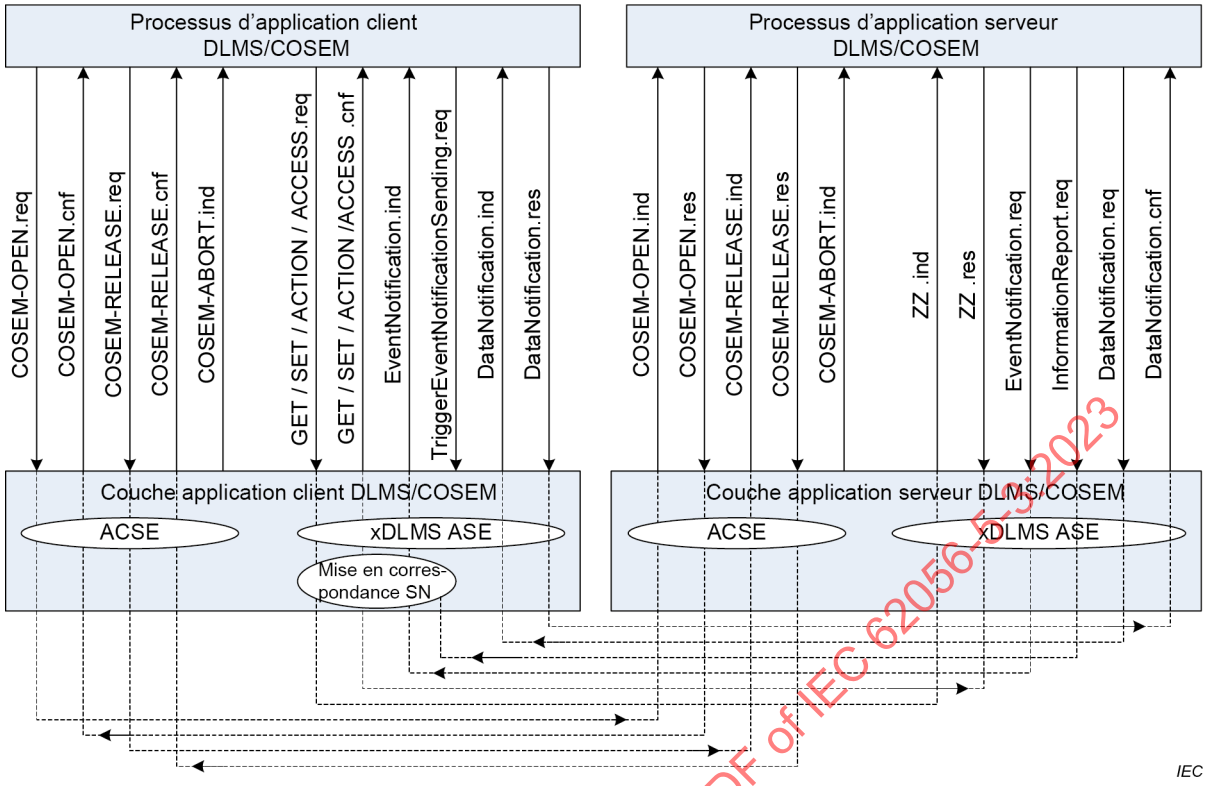
Un récapitulatif des services disponibles en haut de l'AL DLMS®/COSEM est représenté à la Figure 11. Les services de gestion de couche ne sont pas représentés. Même si les primitives de service sont différentes du côté client et du côté serveur, les APDU sont identiques.

NOTE 1 Par exemple, lorsque l'AP client appelle une primitive de service GET.request, l'AL client génère une APDU GET-Request. Lorsque cette APDU est reçue par l'AL du serveur, elle appelle une primitive de service GET.ind.

Les services de l'AL DLMS®/COSEM sont spécifiés à l'Article 6. Le protocole de l'AL DLMS®/COSEM est spécifié à l'Article 7. La syntaxe abstraite des APDU ACSE et xDLMS est spécifiée en 0. Le schéma XML est défini à l'Article 9.

Des exemples de codage sont fournis à l'Annexe D, l'Annexe E et l'Annexe F.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




NOTE 1 L'AP client utilise toujours le référencement LN. Si le serveur utilise le référencement SN, alors une mise en correspondance est effectuée par l'ASE client SN_Mapper. En conséquence, les primitives de service ZZ.ind et ZZ.res peuvent être des primitives de service LN ou SN. La mise en correspondance de services LN/SN est spécifiée à l'Article 8.

NOTE 2 Le service ACCESS ne peut pas être mis en correspondance avec des services à l'aide du référencement SN.

**Figure 11 – Récapitulatif des services de l'AL DLMS®/COSEM**

### 4.2.7 Protocoles de la couche application DLMS®/COSEM

Les protocoles de l'AL DLMS®/COSEM spécifient les procédures utilisées pour le transfert d'informations, pour le contrôle et l'authentification de l'AA à l'aide de procédures ACSE orientées connexion et pour le transfert de données entre les clients et les serveurs COSEM à l'aide de procédures xDLMS. Par conséquent, le protocole de l'AL DLMS®/COSEM est basé sur la norme régissant l'ACSE spécifiée dans l'ISO/IEC 15954:1999 et la norme régissant DLMS® spécifiée dans l'IEC 61334-4-41:1996 (y compris les extensions pour DLMS®/COSEM). Les procédures sont définies en termes:

- d'interactions entre les machines de protocole ACSE et xDLMS homologues par l'intermédiaire de l'utilisation des services de la couche de protocole de support;
- d'interactions entre les machines de protocole ACSE et xDLMS et leur usager.

Les protocoles de l'AL DLMS®/COSEM sont spécifiés à l'Article 7.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

# 5 Sécurité des informations dans DLMS®/COSEM

## 5.1 Vue d'ensemble

Le présent Article 5 décrit et spécifie:

- le concept de sécurité DLMS®/COSEM (voir 5.2);
- les algorithmes cryptographiques choisis (voir 5.3);
- les clés de sécurité (voir 5.4, 5.5 et 5.6);
- l'utilisation des algorithmes cryptographiques pour l'authentification d'entité, la protection des APDU xDLMS et la protection de données COSEM (voir 5.7).

## 5.2 Concept de sécurité DLMS®/COSEM

### 5.2.1 Vue d'ensemble

Les ressources des serveurs DLMS®/COSEM (attributs et méthodes des objets COSEM) peuvent être accessibles par des clients DLMS®/COSEM dans des associations d'applications (voir aussi 4.1.5).

Pendant l'établissement d'une AA, le client et le serveur sont tenus de s'identifier. Le serveur peut également exiger que l'*utilisateur* d'un client s'identifie. En outre, le serveur peut exiger que le client s'authentifie et le client peut également exiger que le serveur s'authentifie. Les mécanismes d'identification et d'authentification sont spécifiés en 5.2.2.

Une fois qu'une AA est établie, les services xDLMS peuvent être utilisés pour accéder aux attributs et méthodes de l'objet COSEM, soumis au contexte de sécurité et aux droits d'accès. Voir 5.2.3 et 5.2.4.

Les APDU xDLMS contenant des primitives de service peuvent être protégées par cryptographie. La protection exigée est déterminée par le contexte de sécurité et les droits d'accès. Pour la prise en charge de la sécurité de bout en bout entre des tierces parties et des serveurs, ces tierces parties peuvent également accéder aux ressources d'un serveur en utilisant un client comme courtier. Le concept de protection de message est expliqué plus en détail en 5.2.5.

En outre, les données COSEM contenues par les APDU xDLMS peuvent être protégées par cryptographie (voir 5.2.6).

Étant donné que les mécanismes de sécurité sont appliqués au niveau du processus d'application de la couche application, ils peuvent être utilisés dans tous les profils de communication DLMS®/COSEM.

NOTE Les couches inférieures peuvent fournir une sécurité supplémentaire.

### 5.2.2 Identification et authentification

#### 5.2.2.1 Identification

Comme spécifié en 4.1.3.3, les AE DLMS®/COSEM sont liées aux points d'accès aux services (SAP) dans la couche de protocole prenant en charge l'AL. Ces SAP sont présents dans les PDU contenant les APDU xDLMS dans une AA.

Le mécanisme d'identification des utilisateurs du client permet au serveur de distinguer les différents utilisateurs du côté client (pouvant être des opérateurs ou des tierces parties) afin d'enregistrer leurs activités d'accès au compteur. Voir aussi 4.1.3.6.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.2.2.2 Mécanismes d'authentification

##### 5.2.2.2.1 Vue d'ensemble

Les mécanismes d'authentification déterminent le protocole à utiliser par les entités de communication pour s'authentifier pendant l'établissement d'une AA. Il existe trois mécanismes d'authentification différents disponibles avec différents niveaux d'authentification de sécurité:

- authentification sans sécurité (niveau de sécurité le plus faible) (voir 5.2.2.2.2);
- authentification de niveau de sécurité faible (LLS) (voir 5.2.2.2.3);

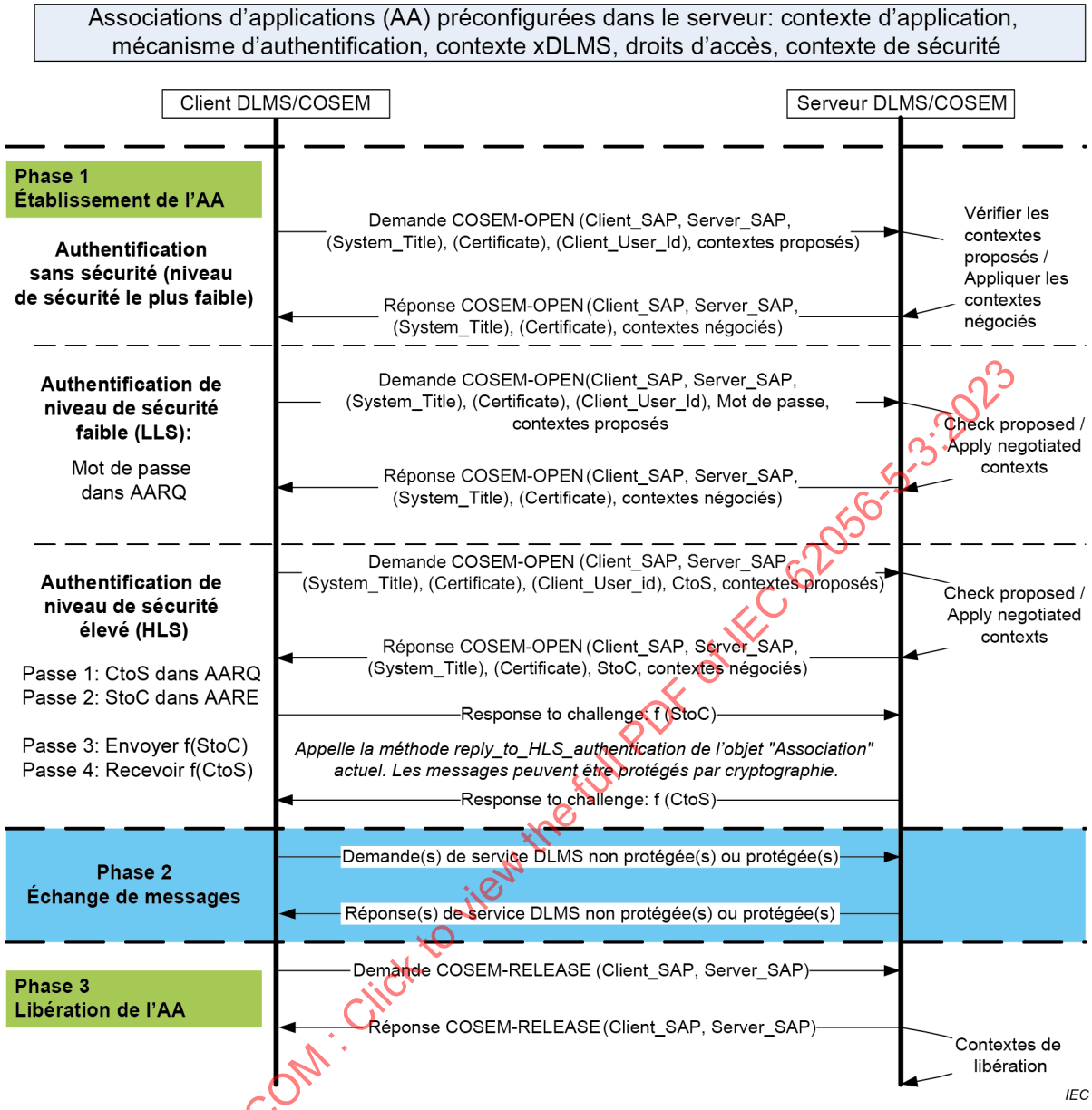
  NOTE 1 Dans l'UIT-T X.811, cette authentification est appelée "authentification unilatérale", mécanisme de classe 0.

- authentification de niveau de sécurité élevé (HLS) (voir 5.2.2.2.4);

  NOTE 2 Dans l'UIT-T X.811, cette authentification est appelée "authentification mutuelle" utilisant des mécanismes à épreuves.

Ces mécanismes sont représentés à la Figure 12. Les mécanismes d'authentification sont identifiés par leur nom (voir 7.2.2.3).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

NOTE 1 Les primitives de service COSEM-OPEN sont contenues dans des APDU AARQ/AARE. Les primitives de service COSEM-RELEASE sont contenues dans des APDU RLRQ/RLRE (le cas échéant). Voir 6.2 et 6.3.

NOTE 2 Les éléments (*System_Title*), (*Certificate*) et (*Client_User_Id*) sont facultatifs.

NOTE 3 Aucune authentification n'a lieu dans les AA préétablies.

NOTE 4 Le service COSEM-RELEASE peut être protégé par cryptographie en incluant une APDU xDLMS chiffrée Initiate .request/.response dans la RLRQ.

**Figure 12 – Mécanismes d'authentification**

La sécurité de l'échange de messages (dans la Phase 2) est indépendante de l'authentification client-serveur pendant l'établissement d'AA (Phase 1). Même en l'absence d'authentification client-serveur, les APDU protégées par cryptographie peuvent être utilisées pour assurer la sécurité des messages.

#### 5.2.2.2.2 Authentification sans sécurité (niveau de sécurité le plus faible)

L'objectif de l'authentification sans sécurité (niveau de sécurité le plus faible) est de permettre au client de récupérer du serveur quelques informations élémentaires. Ce mécanisme d'authentification n'exige pas d'authentification; le client peut accéder aux attributs et méthodes de l'objet COSEM dans le contexte de sécurité et conformément aux droits d'accès qui prévalent dans l'AA donnée.

#### 5.2.2.2.3 Authentification de niveau de sécurité faible (LLS)

Dans ce cas, le serveur exige que le client s'authentifie en saisissant un mot de passe connu du serveur. Le mot de passe est contenu dans l'objet "Association SN/LN" actuel qui modélise l'AA à établir. Les objets "Association SN/LN" fournissent un moyen de modifier le secret.

Si le mot de passe fourni est accepté, l'AA peut être établie; si ce n'est pas le cas, elle doit être rejetée.

L'authentification LLS est prise en charge par le service COSEM-OPEN (voir 6.2), comme suit:

- le client transmet un "secret" (un mot de passe) au serveur, à l'aide de la primitive de service COSEM-OPEN.request;
- le serveur vérifie si le "secret" est correct;
- si oui, le client est authentifié et l'AA peut être établie. À partir de ce moment, les contextes négociés sont valides;
- Sinon, l'AA doit être rejetée;
- le résultat de l'établissement de l'AA doit être renvoyé par le serveur à l'aide de la primitive de service COSEM-OPEN.response avec les informations de diagnostic.

#### 5.2.2.2.4 Authentification de niveau de sécurité élevé (HLS)

Dans ce cas, le client et le serveur sont tenus de s'authentifier avec succès pour établir une AA. L'authentification HLS est un processus en quatre passes qui est pris en charge par le service COSEM-OPEN et la méthode *reply_to_HLS_authentication* de la classe d'interface "Association SN/LN":

- Passe 1: le client transmet une "épreuve" $CtoS$ et, selon le mécanisme d'authentification, des informations supplémentaires au serveur;
- Passe 2: le serveur transmet une "épreuve" $StoC$ et, selon le mécanisme d'authentification, des informations supplémentaires au client;

Si $StoC$ est identique à $CtoS$, le client doit rejeter l'épreuve et doit annuler le processus d'établissement d'AA.

- Passe 3: le client traite $StoC$ et les informations supplémentaires en fonction des règles du mécanisme d'authentification HLS valide pour l'AA donnée et envoie le résultat au serveur. Le serveur vérifie si f($StoC$) est le résultat du traitement correct et, le cas échéant, accepte l'authentification du client.
- Passe 4: le serveur traite ensuite $CtoS$ et les informations supplémentaires en fonction des règles du mécanisme d'authentification HLS valide pour l'AA donnée et envoie le résultat au client. Le client vérifie si f($CtoS$) est le résultat du traitement correct et, le cas échéant, accepte l'authentification du serveur.

La Passe 1 et la Passe 2 sont prises en charge par le service COSEM-OPEN.

Après la Passe 2, dans la mesure où le contexte d'application et le contexte xDLMS proposés sont acceptables, le serveur octroie l'accès à la méthode *reply_to_HLS_authentication* de l'objet "Association SN/LN" actuel à l'aide du contexte d'application négocié.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La Passe 3 et la Passe 4 sont prises en charge par la méthode *reply_to_HLS_authentication* de l'objet ou des objets "Association SN/LN". Si la Passe 3 et la Passe 4 sont toutes deux exécutées avec succès, l'AA est établie avec le contexte d'application et le contexte xDLMS négociés.

La clé dédiée, si elle est transférée, peut être utilisée à partir de ce moment.

Dans le cas contraire, l'opération est annulée par le client ou par le serveur.

Plusieurs mécanismes d'authentification HLS sont disponibles. Ils sont spécifiés plus en détail en 5.7.4.

Dans certains mécanismes d'authentification HLS, le traitement des épreuves implique l'utilisation d'un secret HLS.

La classe d'interface "Association SN/LN" fournit une méthode de modification du "secret" HLS: *change_HLS_secret*.

REMARQUE Une fois que le client a émis la méthode change_HLS_secret () ou change_LLS_secret (), il s'attend à une réponse du serveur confirmant la modification du secret. Il est possible que le serveur transmette la confirmation, mais en raison de problèmes de communication, la confirmation n'est pas reçue du côté client. Par conséquent, le client ne sait pas si le secret a été modifié ou non. Pour des raisons de simplicité, le serveur n'offre pas de prise en charge spéciale pour ce cas, c'est-à-dire qu'il revient au client de résoudre ce problème.

### 5.2.3 Contexte de sécurité

Le contexte de sécurité définit les attributs de sécurité pertinents pour les transformations cryptographiques et comprend les éléments suivants:

- la suite de sécurité, qui détermine les algorithmes de sécurité disponibles (voir 5.3.7);
- la politique de sécurité, qui détermine le ou les types de protection à appliquer en règle générale à toutes les APDU xDLMS échangées dans une AA. Les politiques de sécurité possibles sont spécifiées en 5.7.2.2;
- le matériau de sécurité concerné pour les algorithmes de sécurité donnés, comprenant les clés de sécurité, les vecteurs d'initialisation, les certificats de clé publique et les éléments analogues. Étant donné que le matériau de sécurité est spécifique à chaque algorithme, les éléments sont spécifiés en détail dans les articles correspondants.

Le contexte de sécurité est géré par les objets "Security setup" (voir IEC 62056-6-2:2021, 4.4.7).

### 5.2.4 Droits d'accès

Les droits d'accès aux attributs peuvent être: *no_access*, *read_only*, *write_only* ou *read_and_write*. Les droits d'accès aux méthodes peuvent être *no_access* ou *access*.

En outre, les droits d'accès peuvent spécifier la protection cryptographique à appliquer aux APDU xDLMS contenant les primitives de service utilisées pour accéder à un attribut/une méthode particulier d'objet COSEM. La protection exigée pour .request et .response peut être configurée de manière indépendante.

Les droits d'accès sont contenus par les objets "Association SN/LN" pertinents (voir IEC 62056-6-2:2021, 4.4.3 et 4.4.4). Les droits d'accès possibles sont spécifiés en 5.7.2.2.

La protection à appliquer doit satisfaire à l'exigence la plus stricte spécifiée par la politique de sécurité et les droits d'accès.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.2.5 Sécurité des messages de la couche application

DLMS®/COSEM assure le niveau de sécurité de l'AL en fournissant un moyen de protection par cryptographie des APDU xDLMS. La protection peut être une quelconque combinaison quelconque d'authentification, de cryptage et de signature numérique, et peut être appliquée sur plusieurs couches par plusieurs parties. La protection est appliquée par l'émetteur et est vérifiée et supprimée par le destinataire.

Une demande ou une réponse reçue doit être traitée uniquement si la protection appliquée au message contenant la demande ou la réponse peut être vérifiée et supprimée avec succès.

Les spécifications d'accompagnement spécifiques au projet peuvent spécifier des critères supplémentaires pour l'acceptation et le traitement de messages.

Le concept de protection de message entre un client et un serveur est représenté à la Figure 13.

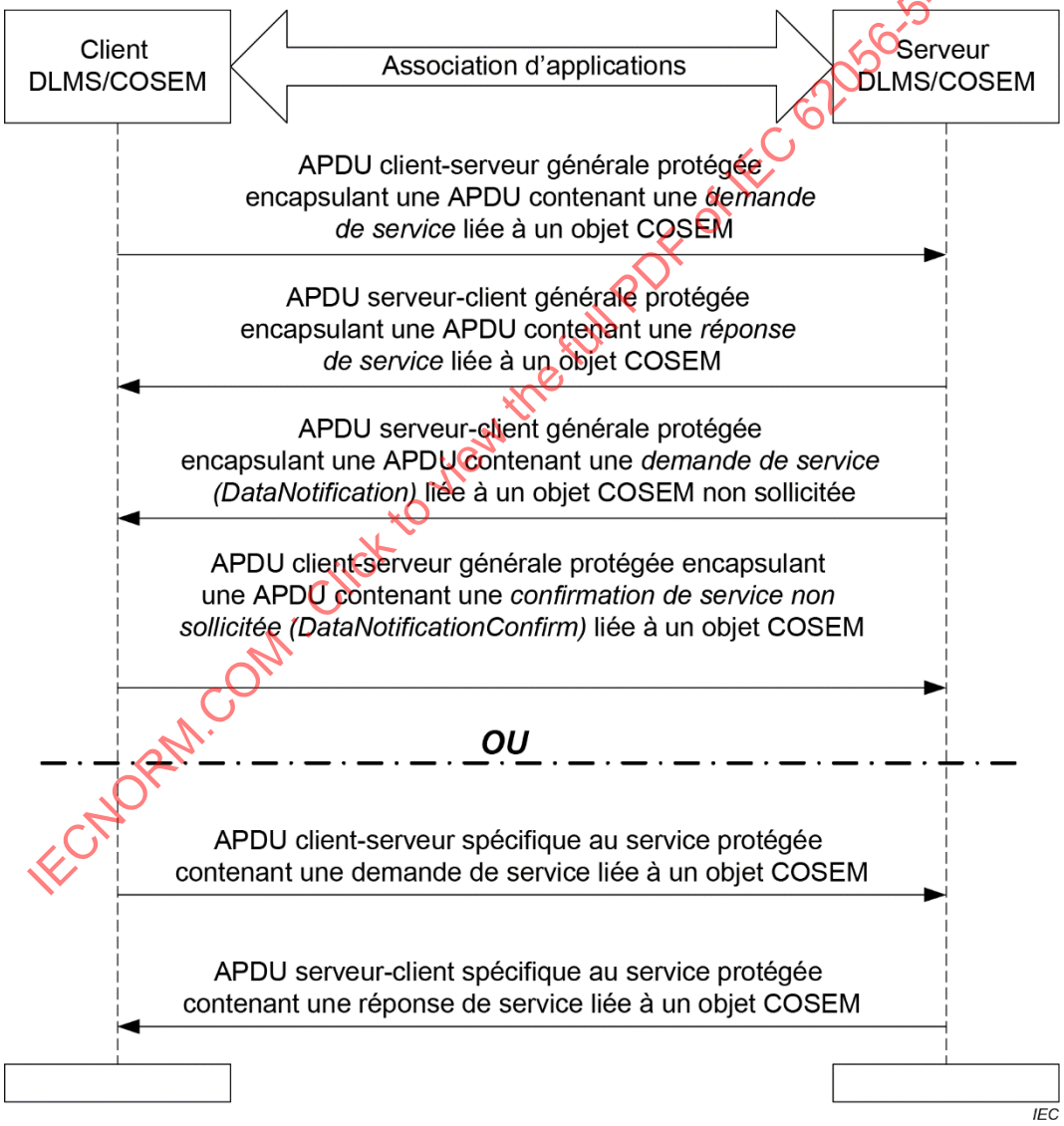



**Figure 13 – Conception de sécurité des messages client-serveur**

Pour assurer une sécurité de bout en bout des messages, les tierces parties sont tenues d'être en mesure d'échanger des demandes de service xDLMS protégées avec les serveurs DLMS®/COSEM. Dans ce cas, le client agit comme courtier, ce qui signifie qu'une tierce partie est un utilisateur de l'une des AA entre un client et un serveur que la tierce partie souhaite atteindre.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le concept de protection de messages entre une tierce partie et un serveur est représenté à la Figure 14.

La tierce partie:

- est au courant de DLMS®/COSEM, c'est-à-dire qu'elle peut générer et traiter des messages encapsulant des APDU xDLMS comprenant des demandes et des réponses de services liées à un objet COSEM;
- est en mesure d'appliquer sa propre protection à l'APDU xDLMS comprenant la demande;
- est en mesure de vérifier la protection appliquée sur la réponse par le serveur et/ou le client.

Le client DLMS®/COSEM:

- sert de courtier entre la tierce partie et le serveur;
- rend une AA appropriée disponible pour être utilisée par la tierce partie, d'après les informations comprises dans le message tierce partie (TP, Third Party)-client;
- vérifie que la TP a le droit d'utiliser cette AA;

NOTE La manière d'effectuer cette vérification ne relève pas du domaine d'application du présent document.

- peut vérifier la protection appliquée par la tierce partie;
- encapsule le message tierce partie-client dans une APDU xDLMS générale protégée;
- peut vérifier la protection appliquée par le serveur sur l'APDU encapsulant la réponse de service ou la demande non sollicitée de service liée à l'objet COSEM; (en cas d'opération Push);
- peut appliquer sa propre protection aux APDU xDLMS protégées envoyées à la TP.

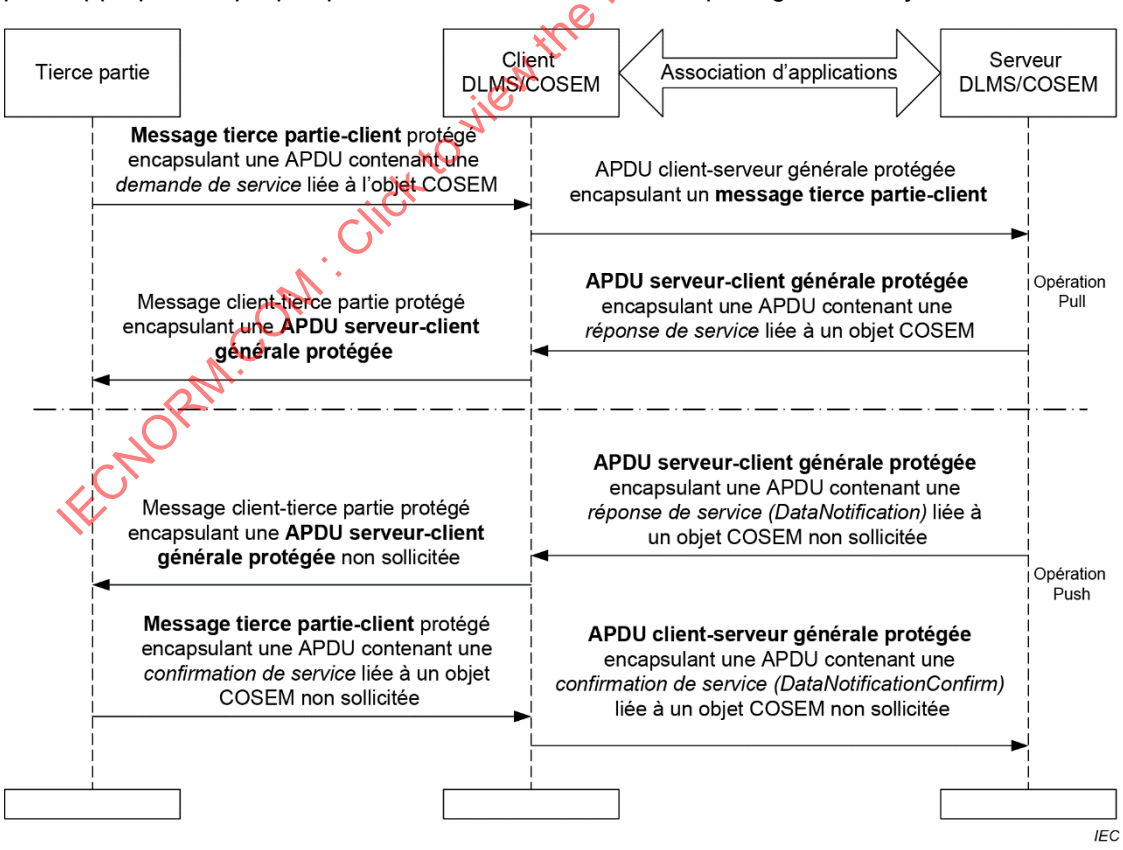



**Figure 14 – Concept de sécurité de bout en bout sur les messages**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le serveur:

- doit (pré)établir une AA avec le client utilisé par la tierce partie;
- peut vérifier l'identité de la tierce partie à l'aide de l'AA;
- doit fournir un accès aux attributs et méthodes de l'objet COSEM tel que déterminé par la politique de sécurité et les droits d'accès une fois que la ou les protections appliquées par le client et/ou la tierce partie ont été vérifiées avec succès;
- doit préparer la réponse (ou, dans le cas d'une opération Push, une demande de service non sollicitée) et appliquer la protection déterminée par la protection appliquée sur la demande entrante, les droits d'accès et la politique de sécurité.

L'application d'une protection cryptographique sur les APDU xDLMS est spécifiée en 5.7.2.

#### 5.2.6 Sécurité des données COSEM

Les données COSEM, c'est-à-dire les valeurs des attributs, des paramètres d'appel de méthode et les paramètres de retour de l'objet COSEM, peuvent également être protégées par cryptographie. Lorsque cela est exigé, les attributs et les méthodes concernés sont indirectement accessibles, par l'intermédiaire des objets "Data protection" qui appliquent et vérifient/retirent la protection des données COSEM (voir IEC 62056-6-2:2021, 4.4.9).

Voir aussi 0.

### 5.3 Algorithmes cryptographiques

#### 5.3.1 Vue d'ensemble

DLMS®/COSEM applique la cryptographie pour protéger les informations.

NOTE Le texte suivant est issu du document NIST SP 800-21:2005, 3.1.

La cryptographie est une branche des mathématiques basée sur la transformation de données et pouvant être utilisée pour fournir plusieurs services de sécurité: confidentialité, intégrité des données, authentification, autorisation et non-répudiation. La cryptographie repose sur deux composants de base: un *algorithme* (ou méthodologie de cryptographie) et une *clé*. L'algorithme est une fonction mathématique et la clé est un paramètre utilisé dans la transformation.

Un algorithme cryptographique et une clé sont utilisés pour appliquer la protection cryptographique aux données (par exemple crypter les données ou générer une signature numérique) et pour supprimer ou vérifier la protection (par exemple décrypter les données cryptées ou vérifier la signature numérique). Il existe trois types d'algorithmes cryptographiques de base approuvés:

- les fonctions de hachage cryptographique, qui n'exigent pas de clés (bien qu'elles puissent être utilisées dans un mode dans lequel des clés sont utilisées). Une fonction de hachage est souvent utilisée comme composant d'un algorithme pour fournir un service de sécurité. Voir 5.3.2;
- les algorithmes à clé symétrique (souvent appelés algorithmes à clé secrète), qui utilisent une clé unique (partagée par un expéditeur et un destinataire) pour appliquer la protection et supprimer ou vérifier cette protection. Les algorithmes à clé symétrique sont relativement simples à mettre en œuvre et fournissent un débit élevé. Voir 5.3.3;
- les algorithmes à clé asymétrique (souvent appelés algorithmes à clé publique), qui utilisent deux clés (c'est-à-dire une paire de clés): une clé publique et une clé privée mathématiquement liées. En comparaison avec les algorithmes à clé symétrique,

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

l'implémentation des algorithmes à clé asymétrique est complexe et exige beaucoup plus de calculs. Voir 5.3.4.

Afin d'utiliser la cryptographie, des clés cryptographiques doivent être "mises en place", c'est-à-dire qu'elles doivent être établies pour les parties qui utilisent la cryptographie. Voir 5.4.

### 5.3.2 Fonction de hachage

NOTE Le texte suivant est issu du document NIST SP 800-21:2005, 3.2.

Une fonction de hachage produit une représentation courte d'un message plus long. Une bonne fonction de hachage est une fonction unidirectionnelle: le calcul de la valeur de hachage à partir d'une valeur d'entrée précise est aisé. Toutefois, faire le processus inverse, partir de la valeur de hachage pour revenir à la valeur d'entrée, est très difficile. Avec une bonne fonction de hachage, il est également très difficile de trouver deux valeurs d'entrée spécifiques qui produisent la même valeur de hachage. En raison de ces caractéristiques, les fonctions de hachage sont souvent utilisées pour déterminer si les données ont été modifiées ou non.

Une fonction de hachage prend une valeur d'entrée de longueur arbitraire et obtient comme résultat une valeur de longueur fixe. Les noms communs pour la valeur de sortie d'une fonction de hachage incluent *valeur de hachage* et *synthèse de message*. La Figure 15 représente l'utilisation d'une fonction de hachage.

Une valeur de hachage (H1) est calculée sur les données (M1). M1 et H1 sont ensuite sauvegardées ou transmises. Ultérieurement, l'exactitude des données récupérées ou reçues est vérifiée en étiquetant les données reçues comme M2 (au lieu de M1) et en calculant une nouvelle valeur de hachage (H2) sur la valeur reçue. Si la nouvelle valeur de hachage calculée (H2) est égale à la valeur de hachage récupérée ou reçue (H1), il peut être pris pour hypothèse que les données récupérées ou reçues (M2) sont identiques aux données originales (M1) (c'est-à-dire M1 = M2).

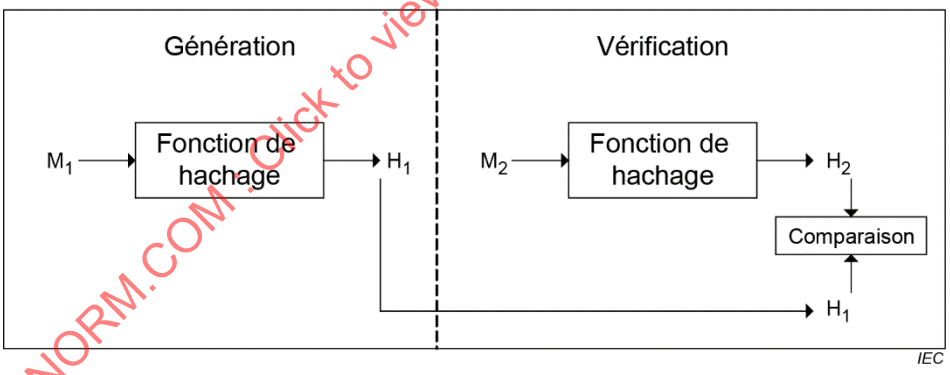



**Figure 15 – Fonction de hachage**

Des algorithmes de hachage sont utilisés dans DLMS®/COSEM pour les raisons suivantes:

- signature numérique (voir 5.3.4.4);
- agrément de clé (voir 5.3.4.6);
- authentification HLS. L'algorithme à utiliser dépend du mécanisme d'authentification (voir 5.7.4).

Pour la signature numérique et l'agrément de clé, l'algorithme doit être tel que spécifié par la suite de sécurité (voir Tableau 9).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.3.3 Algorithmes à clé symétrique

#### 5.3.3.1 Généralités

Des algorithmes à clé symétrique sont utilisés dans DLMS®/COSEM pour les raisons suivantes:

- authentification des partenaires de communication à l'aide de mécanismes d'authentification HLS (voir 5.7.4);
- authentification et cryptage de messages xDLMS (voir 5.7.2);
- authentification et cryptage de données COSEM (voir 0).

NOTE Le texte suivant est issu du document NIST SP 800-21:2005, 3.3.

Les algorithmes à clé symétrique (souvent appelés algorithmes à clé secrète) utilisent une clé unique pour appliquer la protection et supprimer ou vérifier cette protection. Par exemple, la clé utilisée pour crypter les données est également utilisée pour les décrypter. Cette clé doit rester secrète si la protection cryptographique des données est à conserver. Les algorithmes à clé symétrique permettent de fournir la confidentialité par le cryptage, ou bien une assurance d'authenticité ou d'intégrité par l'authentification, ou bien ils sont utilisés pendant l'établissement de la clé.

Les clés utilisées pour un objectif précis ne doivent pas être utilisées à d'autres fins. (voir NIST SP 800-57:2012).

#### 5.3.3.2 Cryptage et décryptage

NOTE Le texte suivant est issu du document NIST SP 800-21:2005, 3.3.1.

Le cryptage est utilisé pour fournir la confidentialité des données. Les données à protéger sont appelées *texte brut*. Le cryptage les transforme en *texte chiffré*. Le texte chiffré peut être retransformé en texte brut par l'intermédiaire du décryptage.

Les données en texte brut peuvent uniquement être récupérées à partir du texte chiffré en utilisant la même clé que celle utilisée pour crypter les données. Il convient que les destinataires non autorisés du texte chiffré, qui connaissent l'algorithme cryptographique, mais n'ont pas la clé correcte, ne soient pas en mesure de décrypter le texte chiffré. Cependant, toute personne qui détient la clé et l'algorithme cryptographique peut facilement décrypter le texte chiffré et obtenir les données originales en texte brut.

La Figure 16 représente les processus de cryptage et de décryptage. Le texte brut (P) et une clé (K) sont utilisés par le processus de cryptage pour produire le texte chiffré (C). Pour le décryptage, le texte chiffré (C) et la même clé (K) sont utilisés par le processus de décryptage pour récupérer le texte brut (P).

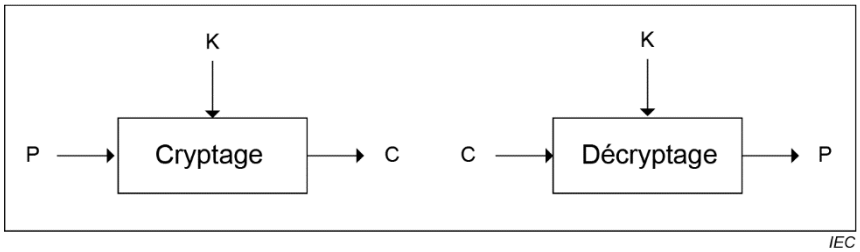



**Figure 16 – Cryptage et décryptage**

Avec les algorithmes de chiffrement par blocs par clé symétrique, un même bloc de texte brut et une même clé produisent toujours le même bloc de texte chiffré. La sécurité fournie par cette propriété n'est pas acceptable. C'est la raison pour laquelle des modes cryptographiques de fonctionnement ont été définis pour résoudre ce problème (voir 5.3.3.4).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.3.3.3 Advanced Encryption Standard (Norme de cryptage avancé)

Pour les besoins de DLMS®/COSEM, la norme Advanced Encryption Standard (AES) telle que définie dans le document FIPS PUB 197:2001 doit être utilisée. AES fonctionne par blocs (tranches) de données pendant une opération de cryptage ou de décryptage. C'est la raison pour laquelle l'AES est qualifiée d'"algorithme de chiffrement par blocs".

NOTE Le texte suivant est issu du document NIST SP 800-21:2005, 3.3.1.3.

AES crypte et décrypte les données par blocs de 128 bits, en utilisant des clés de 128 bits, 192 bits ou 256 bits. Les trois tailles de clés sont adéquates.

AES combine sécurité, performances, efficacité, facilité d'implémentation et flexibilité. L'algorithme a en particulier de bonnes performances pour le matériel et les logiciels dans une large plage d'environnements informatiques. De même, compte tenu de ses faibles exigences de mémoire, il est particulièrement adapté aux environnements soumis à des contraintes d'espace réduit.

#### 5.3.3.4 Modes de fonctionnement du cryptage

NOTE Le texte suivant est issu du document NIST SP 800-21:2005, 3.3.1.4.

Avec un algorithme de chiffrement par blocs par clé symétrique, le cryptage d'un même bloc de texte brut produit toujours le même bloc de texte chiffré lorsque la même clé symétrique est utilisée. Si les multiples blocs constituant un message type (flux de données) sont cryptés séparément, une partie adverse peut aisément remplacer des blocs individuels, probablement sans que cela soit détecté. En outre, certains types de modèles de données dans le texte brut, comme les blocs répétés, seraient apparents dans le texte chiffré.

Les modes cryptographiques de fonctionnement ont été définis pour résoudre ce problème en combinant l'algorithme cryptographique de base avec des valeurs variables d'initialisation (généralement appelées vecteurs d'initialisation) et des règles de retour pour les informations dérivées du fonctionnement cryptographique.

Le document NIST SP 800-38D:2007 spécifie le mode Galois/Counter (GCM), un algorithme de cryptage authentifié avec des données associées, et sa spécialisation, GMAC, pour générer un code d'authentification de message (MAC) sur les données non cryptées. GCM et GMAC sont des modes de fonctionnement pour un chiffrement par blocs par clé symétrique sous-jacent approuvé. Voir 5.3.3.3.

#### 5.3.3.5 Code d'authentification de message

NOTE Le texte suivant est issu du document NIST SP 800-21:2005, 3.3.2.

Les codes d'authentification de message (MAC) fournissent une garantie d'authenticité et d'intégrité. Un MAC est une somme de contrôle cryptographique des données utilisée pour assurer que les données n'ont pas été modifiées ou altérées et que le MAC a été calculé par la partie prévue (l'expéditeur). En règle générale, les MAC sont utilisés entre les deux parties qui partagent une clé secrète afin d'authentifier les informations qu'elles échangent.

La Figure 17 représente l'utilisation des codes d'authentification de message (MAC).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

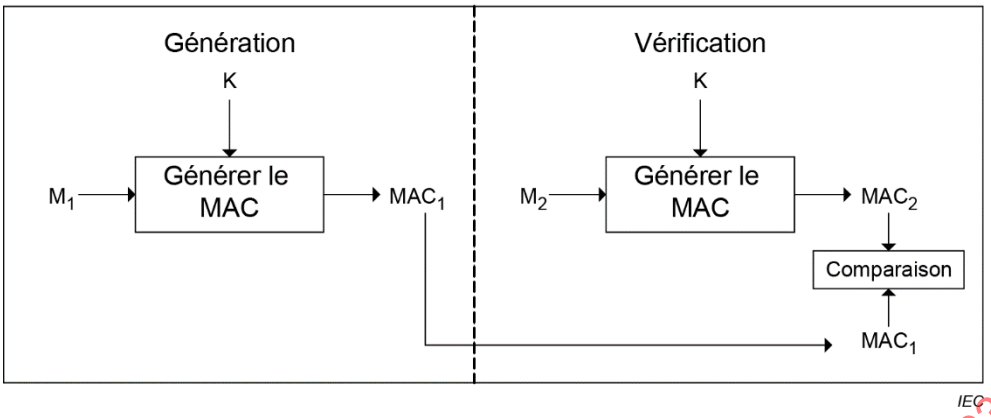



**Figure 17 – Codes d'authentification de message (MAC)**

Un MAC (MAC1) est calculé sur les données (M1) à l'aide d'une clé (K). M1 et MAC1 sont ensuite sauvegardées ou transmises. Ultérieurement, l'authenticité des données récupérées ou reçues est vérifiée en étiquetant les données récupérées ou reçues comme M2 et en calculant un MAC (MAC2) sur ces données à l'aide de la même clé (K). Si le MAC récupéré ou reçu (MAC1) est identique au MAC calculé (MAC2), il peut être pris pour hypothèse que les données récupérées ou reçues (M2) sont identiques aux données d'origine (M1) (c'est-à-dire M1 = M2). La partie qui vérifie connaît également l'identité de la partie émettrice, aucune autre partie ne connaît la clé.

Les MAC sont généralement utilisés pour détecter les modifications de données survenant entre la génération initiale du MAC et la vérification du MAC reçu. Ils ne détectent pas les erreurs qui surviennent avant la génération initiale du MAC.

L'intégrité du message est souvent fournie à l'aide de techniques non cryptographiques appelées codes de détection d'erreur. Ces codes peuvent toutefois être modifiés par une partie adverse à son avantage. L'utilisation d'un mécanisme cryptographique approuvé, tel que MAC, résout ce problème. L'intégrité fournie par un MAC repose sur l'hypothèse selon laquelle il est impossible de générer un MAC sans connaître la clé cryptographique. Une partie adverse qui ne connaît pas la clé est incapable de modifier des données puis de générer un MAC authentique sur les données modifiées. Il est donc crucial que les clés MAC restent secrètes.

Pour les besoins de DLMS®/COSEM, l'algorithme GMAC tel que spécifié en 5.3.3.7.2 doit être utilisé.

### 5.3.3.6 Enveloppement de clé

NOTE Le texte suivant est issu du document NIST SP 800-21:2005, 3.3.3.

Les algorithmes à clé symétrique peuvent être utilisés pour envelopper (c'est-à-dire crypter) le matériel de chiffrement à l'aide d'une clé d'enveloppement de clé (également appelée clé de cryptage de clé). Le matériel de chiffrement enveloppé peut ensuite être stocké ou transmis de manière sécurisée. Le désenveloppement du matériel de chiffrement exige l'utilisation de la même clé d'enveloppement de clé que celle utilisée pendant le processus d'enveloppement initial.

L'enveloppement de clé diffère du simple cryptage, car le processus d'enveloppement inclut une fonction d'intégrité. Pendant le processus de désenveloppement, cette fonction d'intégrité détecte les modifications accidentelles ou intentionnelles du matériel de chiffrement enveloppé. Pour les besoins de DLMS®/COSEM, l'algorithme d'enveloppement de clé AES doit être utilisé (voir 5.3.3.8).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.3.3.7 Mode Galois/Counter

##### 5.3.3.7.1 Généralités

NOTE Le texte suivant est issu du document NIST SP 800-38D:2007, Article 3.

Le mode Galois/Counter (GCM) est un algorithme de cryptage authentifié avec des données associées. Le GCM est construit à partir d'un chiffrement par blocs par clé symétrique approuvé, avec une taille de bloc de 128 bits, tel que l'algorithme de la norme Advanced Encryption Standard (AES) (voir FIPS PUB 197). Par conséquent, le GCM est un mode de fonctionnement de l'algorithme AES.

Le GCM assure la confidentialité des données à l'aide d'une variation du mode de fonctionnement Counter pour le cryptage.

Le GCM assure l'authenticité des données confidentielles (jusqu'à environ 64 gigaoctets par appel) à l'aide d'une fonction universelle de hachage qui est définie dans un champ de Galois (c'est-à-dire un corps fini) binaire (GHASH). Le GCM peut également offrir une assurance d'authentification pour des données supplémentaires (longueur presque illimitée par appel) qui ne sont pas cryptées.

Si les valeurs d'entrée GCM sont limitées aux données qui ne sont pas cryptées, la spécialisation résultante du GCM, appelée GMAC, est simplement un mode d'authentification appliqué aux données entrantes.

Le GCM offre une assurance d'authentification supérieure à celle d'une somme de contrôle ou d'un code de détection d'erreur (non cryptographique). Le GCM peut en particulier détecter 1) les modifications accidentelles des données et 2) les modifications intentionnelles, non autorisées.

Dans DLMS®/COSEM, il est également possible d'utiliser le GCM pour offrir uniquement la confidentialité. Dans ce cas, les balises d'authentification ne sont simplement pas calculées ni vérifiées.

##### 5.3.3.7.2 Fonctions du GCM

NOTE Le texte suivant est issu du document NIST SP 800-38D:2007, 5.2.

Les deux fonctions qui constituent le GCM sont appelées cryptage authentifié et décryptage authentifié (voir Figure 18).

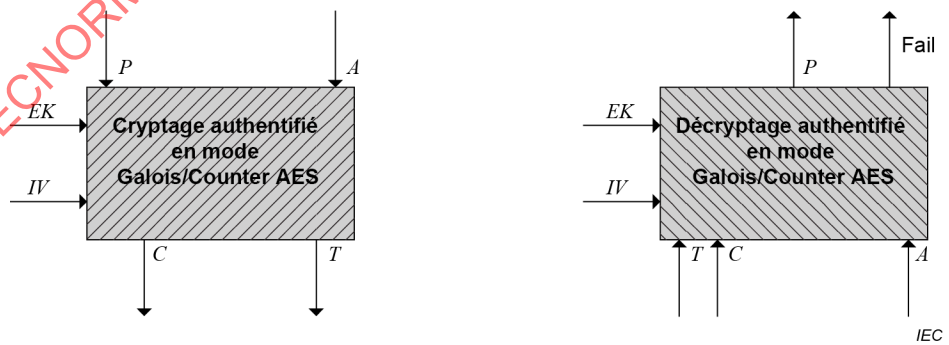



**Figure 18 – Fonctions du GCM**

La fonction de cryptage authentifié crypte les données confidentielles et calcule une balise d'authentification pour les données confidentielles ainsi que pour toutes les données supplémentaires, non confidentielles. La fonction de décryptage authentifié décrypte les données confidentielles en fonction de la vérification de la balise.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Lorsque la valeur d'entrée est limitée à des données non confidentielles, la variante résultant de GCM est appelée GMAC. Pour GMAC, les fonctions de cryptage et de décryptage authentifiés deviennent les fonctions de génération et de vérification d'une balise d'authentification pour les données non confidentielles.

Enfin, si l'authentification n'est pas exigée, la fonction de cryptage authentifié crypte les données confidentielles, mais la balise d'authentification n'est pas calculée. La fonction de décryptage authentifié décrypte les données confidentielles, mais aucune balise d'authentification n'est calculée ni vérifiée.

Dans DLMS®/COSEM, l'utilisation de l'authentification et du cryptage est indiquée par le bit 4 et le bit 5 de l'octet de contrôle de sécurité, spécifié en 5.7.2.4.

a) La fonction de cryptage authentifié (étant donné le choix d'une clé de chiffrement par blocs $EK$) a trois chaînes de saisie:
   - un texte brut, symbolisé par $P$;
   - les données supplémentaires authentifiées (AAD), symbolisées par $A$;
   - un vecteur d'initialisation (IV) symbolisé par $IV$.

Le texte brut et les AAD constituent les deux catégories de données protégées par le GCM. Le GCM protège l'authenticité du texte brut et des AAD. Il protège également la confidentialité du texte brut, alors que les AAD sont conservées en clair.

L'IV est une valeur unique, c'est-à-dire une valeur qui est unique dans le contexte (de sécurité) spécifié, qui détermine un appel de la fonction de cryptage authentifié pour les données entrantes à protéger. Voir 5.3.3.7.3.

Les longueurs en bits des chaînes de saisie vers la fonction de cryptage authentifié doivent satisfaire aux exigences suivantes:

- len($P$) < $2^{39}$-256;
- len($A$) < $2^{64}$-1;
- 1 ≤ len($IV$) ≤ $2^{64}$-1.

Les longueurs en bits de $P$, $A$ et $IV$ doivent être des multiples de 8, de sorte que ces valeurs sont des chaînes d'octets.

Il existe deux valeurs de sortie:

- du texte chiffré, symbolisé par $C$, dont la longueur en bits est la même que celle du texte brut P;
- une balise d'authentification, ou "balise" par souci de brièveté, symbolisée par $T$.

b) La fonction de décryptage authentifié (étant donné le choix d'une clé de chiffrement par blocs $EK$) a quatre chaînes de saisie:
   - le vecteur d'initialisation, symbolisé par $IV$;
   - le texte chiffré, symbolisé par $C$;
   - les données supplémentaires authentifiées (AAD), symbolisées par $A$;
   - la balise d'authentification, symbolisée par $T$.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La valeur de sortie est l'une des suivantes:

- le texte brut $P$ qui correspond au texte chiffré $C$; ou
- un code d'erreur spécial, symbolisé par $FAIL$ dans la présente Norme internationale.

La valeur de sortie $P$ indique que $T$ est la balise d'authentification correcte pour $IV$, $A$ et $C$. Sinon, la valeur de sortie est $FAIL$.

#### 5.3.3.7.3 Vecteur d'initialisation, *IV*

Dans DLMS®/COSEM, pour la construction du vecteur d'initialisation $IV$, la construction déterministe indiquée dans le document NIST SP 800-38D:2007, 8.2.1, doit être utilisée: l'$IV$ est la concaténation de deux champs, appelés le champ fixe et le champ d'appel. Le champ fixe doit identifier le dispositif physique ou, de manière plus générale, le contexte (de sécurité) pour l'instance de la fonction de cryptage authentifié. Le champ d'appel doit identifier les ensembles de valeurs d'entrée dans la fonction de cryptage authentifié pour ce dispositif spécifique.

Pour n'importe quelle clé donnée, deux dispositifs physiques distincts ne doivent pas partager le même champ fixe et deux ensembles distincts de valeurs d'entrée pour un seul dispositif ne doivent pas partager le même champ d'appel.

La longueur de l'$IV$ doit être de 96 bits (12 octets): $\text{len}(IV) = 96$. Dans cette formule:

- les 64 bits (8 octets) de départ (c'est-à-dire les plus à gauche) doivent contenir le champ fixe. Ils doivent contenir le titre système (voir 4.1.3.4);
- les 32 bits de fin (c'est-à-dire les plus à droite) doivent contenir le champ d'appel. Le champ d'appel doit être un compteur d'entiers.

Chaque clé de cryptage *(EK*) a deux compteurs d'invocations (*IC*) associés, l'un pour la fonction de cryptage authentifié et l'autre pour la fonction de décryptage authentifié. L'*EK* est destinée au chiffrement par blocs. Les règles suivantes s'appliquent:

- lorsque la clé est établie, les *IC* correspondants sont remis à 0;
- lorsque la fonction de cryptage authentifié est utilisée, l'*IC* correspondant est utilisé, puis est incrémenté de 1. Si la valeur maximale de l'*IC* a été atteinte, tout appel suivant de la fonction de cryptage authentifié doit retourner une erreur et l'*IC* ne doit pas être incrémenté;
- lorsque la fonction de décryptage authentifié est utilisée, la valeur de l'*IC* est vérifiée. La valeur doit être supérieure ou égale à la valeur acceptable la plus basse.

  Si la valeur vérifiée satisfait à cette exigence, la valeur la plus basse acceptable est réglée à la valeur *IC* vérifiée plus 1 après l'utilisation de la fonction de décryptage authentifié.

  Si la valeur vérifiée est inférieure à la valeur acceptable la plus basse, la vérification échoue, ainsi que la fonction de décryptage authentifié.

  Si la valeur vérifiée est égale à la valeur maximale, la fonction de décryptage authentifié doit retourner une erreur.

  NOTE Le nombre maximal d'appels est de $2^{32}$-1.

La longueur en bits du champ fixe limite à $2^{64}$ le nombre de dispositifs physiques distincts qui peuvent mettre en œuvre la fonction de cryptage authentifié pour la clé donnée. La longueur en bits du champ d'appel limite à $2^{32}$ le nombre d'appels de la fonction de cryptage authentifié, quels que soient les ensembles de valeurs d'entrée, sans violation de l'exigence d'unicité.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.3.3.7.4 Clé de cryptage, *EK*

Le GCM utilise une clé unique, la clé de chiffrement par blocs. Dans DLMS®/COSEM, elle est appelée "clé de cryptage", symbolisée par $EK$. Sa taille dépend de la suite de sécurité (voir 5.3.7) et doit être:

- pour la suite de sécurité 0 et 1, 128 bits (16 octets): $\text{len}(EK) = 128$;
- pour la suite de sécurité 2, 256 bits (32 octets): $\text{len}(EK) = 256$.

La clé doit être générée uniformément et de manière aléatoire, ou presque uniformément et de manière aléatoire, c'est-à-dire que chaque clé possible a (quasiment) autant de chances d'être générée. Par conséquent, la clé est "fraîche", c'est-à-dire qu'il y a de fortes probabilités qu'elle ne soit égale à aucune autre clé. La clé doit être secrète et doit être utilisée exclusivement pour le GCM avec le chiffrement par blocs AES choisi. Les autres exigences relatives à l'établissement et à la gestion de clés sont abordées dans le document NIST SP 800-38D:2007, 8.1.

#### 5.3.3.7.5 Clé d'authentification, *AK*

Dans DLMS®/COSEM, à des fins de sécurité supplémentaire, une clé d'authentification, symbolisée par $AK$, est également spécifiée. Lorsqu'elle est présente, elle doit faire partie des données supplémentaires authentifiées, AAD. Concernant sa longueur et sa génération, les mêmes règles que celles relatives à la clé de cryptage s'appliquent.

#### 5.3.3.7.6 Longueur de la balise d'authentification

La longueur en bits de la balise d'authentification, symbolisée par $t$, est un paramètre de sécurité. Dans les suites de sécurité 0, 1 et 2, sa valeur doit être de 96 bits.

### 5.3.3.8 Emballage AES de clé

Pour l'emballage des données de clé, DLMS®/COSEM a choisi l'algorithme AES d'emballage de clé spécifié dans le document RFC 3394. L'algorithme est conçu pour emballer ou crypter des données de clé. Il fonctionne sur des blocs de 64 bits. Avant d'être emballées, les données de clé sont réparties en *n* blocs de 64 bits. L'algorithme d'emballage de clé limite *n* uniquement dans le sens où *n* est tenu d'être au moins de 2.

L'emballage de clé AES peut être configuré pour utiliser une quelconque taille parmi les trois tailles de clé prises en charge par le code AES: 128, 192, 256.

Les deux algorithmes sont l'emballage de clé et le déballage de clé.

Les valeurs d'entrée vers le processus d'emballage de clé sont la clé de cryptage de clé, $KEK$, et le texte brut à emballer. Le texte brut est constitué de n blocs de 64 bits, comportant les données de clé emballées. La valeur de sortie est le texte chiffré, (n+1) valeurs de 64 bits.

Les valeurs d'entrée vers le processus de déballage sont la $KEK$ et (n+1) blocs de 64 bits de texte chiffré comprenant la clé précédemment emballée. Elles retournent n blocs de texte brut comprenant les n blocs de 64 bits de données de clé décryptées.

Dans DLMS®/COSEM, la taille de la $KEK$ dépend de la suite de sécurité (voir 5.3.7) et doit être:

- pour la suite de sécurité 0 et 1, 128 bits (16 octets): $\text{len}(KEK) = 128$;
- pour la suite de sécurité 2, 256 bits (32 octets): $\text{len}(KEK) = 256$.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.3.4 Algorithmes à clé publique

#### 5.3.4.1 Généralités

En règle générale, les systèmes de cryptographie à clé publique utilisent des problèmes difficiles à résoudre comme base de l'algorithme. L'algorithme RSA est basé sur la factorisation en nombres premiers d'entiers très importants. La cryptographie à courbe elliptique (ECC) est basée sur la difficulté à résoudre le problème du logarithme discret sur le groupe des points d'une courbe elliptique (ECDLP, Elliptic Curve Discrete Logarithm Problem). L'ECC fournit des niveaux de sécurité similaires à ceux de RSA, mais avec des tailles de clé nettement inférieures. L'ECC est particulièrement appropriée pour les dispositifs embarqués, et a par conséquent été choisie pour l'utilisation dans DLMS®/COSEM.

Les algorithmes à clé publique sont utilisés dans DLMS®/COSEM aux fins suivantes:

- authentification des partenaires de communication;
- signature numérique des APDU xDLMS et des données COSEM;
- l'agrément de clé.

NOTE 1 Le texte suivant est issu du document NIST SP 800-21:2005, 3.4.

Les algorithmes à clé asymétrique (souvent appelés algorithmes à clé publique) utilisent deux clés: une clé publique et une clé privée, qui sont mathématiquement liées. La clé publique peut être rendue publique, mais la clé privée doit rester secrète si les données sont tenues de conserver leur protection cryptographique. Bien qu'il existe une relation entre les deux clés, la clé privée ne peut être déterminée à partir de la clé publique. La clé à utiliser pour appliquer, supprimer ou vérifier la protection dépend du service à fournir. Par exemple, une signature numérique est calculée à l'aide d'une clé privée et la signature est vérifiée à l'aide de la clé publique. Pour les algorithmes qui effectuent également le cryptage, ce dernier est effectué à l'aide de la clé publique et le décryptage est effectué avec la clé privée.

NOTE 2 Tous les algorithmes à clé publique ne sont pas capables d'effectuer plusieurs fonctions, par exemple la génération de signatures numériques et le cryptage. Les algorithmes à clé asymétrique ne sont pas utilisés pour le cryptage dans DLMS®/COSEM.

Les algorithmes à clé asymétrique sont principalement utilisés pour les mécanismes d'intégrité des données, d'authentification et de non-répudiation (c'est-à-dire les signatures numériques) et pour l'établissement des clés.

Certains algorithmes à clé asymétrique utilisent des paramètres de domaine, qui sont des valeurs supplémentaires nécessaires au fonctionnement de l'algorithme cryptographique. Ces valeurs sont liées par une relation mathématique. Les paramètres de domaine sont généralement publics et utilisés par une communauté d'utilisateurs pendant un laps de temps conséquent.

L'utilisation sécurisée d'algorithmes à clé asymétrique exige l'obtention par les utilisateurs de certaines assurances:

- l'assurance de la validité du paramètre de domaine assure que ces paramètres sont corrects du point de vue mathématique;
- l'assurance de la validité de la clé publique assure l'adéquation de cette clé;
- l'assurance de la détention d'une clé privée assure que la partie censée être propriétaire de la clé privée la détient réellement.

Certains algorithmes à clé asymétrique peuvent être utilisés pour plusieurs objectifs (par exemple, pour les signatures numériques et l'établissement de clés). Les clés utilisées pour un objectif précis ne doivent pas être utilisées à d'autres fins.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.3.4.2 Cryptographie à courbe elliptique

##### 5.3.4.2.1 Généralités

La cryptographie à courbe elliptique implique des opérations arithmétiques sur une courbe elliptique sur un corps fini. Les courbes elliptiques peuvent être définies sur un champ quelconque de nombres (c'est-à-dire réel, entier, complexe) bien qu'elles soient le plus souvent utilisées sur des corps premiers finis pour des applications cryptographiques.

Une courbe elliptique sur un corps premier comprend l'ensemble de nombres réels $(x, y)$ qui satisfont à l'équation:

$$y^2 = x^3 + ax + b$$

L'ensemble de toutes les solutions à l'équation forme la courbe elliptique. La modification de $a$ et de $b$ modifie la forme de la courbe, et les modifications moindres de ces paramètres peuvent donner lieu à des modifications majeures dans l'ensemble de solutions (x, y).

##### 5.3.4.2.2 Courbes elliptiques recommandées par le NIST

Le document FIPS PUB 186-4:2013 recommande cinq courbes elliptiques à corps premier sur un corps premier $GF(p)$. Parmi ces courbes, les courbes P-256 et P-384 ont été choisies pour DLMS®/COSEM, comme présenté dans le Tableau 3.

**Tableau 3 – Courbes elliptiques dans les suites de sécurité DLMS®/COSEM**

| Suite de sécurité DLMS® | Nom de la courbe dans le document FIPS PUB 186-4:2013 | Identificateur d'objet ASN.1 |
|---|---|---|
| Suite 0 | – | – |
| Suite 1 | Courbe NIST P-256 | 1.2.840.10045.3.1.7 |
| Suite 2 | Courbe NIST P-384 | 1.3.132.0.34 |
| NOTE L'identificateur d'objet de notation de syntaxe abstraite numéro 1 (ASN.1, Abstract Syntax Notation 1) apparaît dans le Certificat sous l'appellation AlgorithmIdentifier: Parameters. Voir 5.6.4.2. | | |

#### 5.3.4.3 Conversions de données

##### 5.3.4.3.1 Vue d'ensemble

Le présent paragraphe décrit les primitives de conversion de données qui doivent être utilisées pour les conversions entre différents types de données utilisés pour spécifier les algorithmes à clé publique: chaînes d'octets (OS), chaînes binaires (BS), entiers (I, Integer), éléments de champ (FE) et points de courbe elliptique (ECP). DLMS®/COSEM utilise les chaînes d'octets pour représenter les éléments des algorithmes à clé publique et utilise les primitives de conversion entre ces types de données à partir de et vers des chaînes d'octets. La chaîne d'octets $M_{d-1}\ M_{d-2}\ ...\ M_0$ de longueur d est codée en: A-XDR OCTET STRING, où l'octet le plus à gauche $M_{d-1}$ correspond au premier octet de la valeur codée d'OCTET STRING.

##### 5.3.4.3.2 Conversion entre chaînes binaires et chaînes d'octets (BS2OS)

La primitive de conversion de données qui convertit une chaîne binaire en chaîne d'octets est appelée la primitive de conversion de chaîne binaire en chaîne d'octets (BS2OS, Bit String to Octet String conversion primitive). Elle prend la chaîne binaire comme valeur d'entrée et obtient comme résultat la chaîne d'octets. La chaîne binaire $_{bl-1}\ b_{l-2}\ ...\ b_0$ de longueur $l$ doit être convertie en une chaîne d'octets $M_{d-1}\ M_{d-2}\ ...\ M_0$ de longueur $d = \lceil l/8 \rceil$.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La conversion apporte suffisamment de zéros sur la gauche pour que le nombre de bits soit un multiple de 8, puis elle casse la chaîne en octets.

Plus précisément, la conversion doit être comme suit:

- pour $0 \le i < d - 1$, laisser l'octet $M_i = b_{8i+7}\, b_{8i+6} \ldots b_{8i}$;
- l'octet le plus à gauche, $M_{d-1}$, doit avoir ses $8d - l$ bits les plus à gauche à zéro;
- ses $8 - (8d - l)$ bits les plus à droite doivent être $b_{l-1}\, b_{l-2} \ldots b_{8d-8}$.

#### 5.3.4.3.3 Conversion entre chaînes d'octets et chaînes binaires (OS2BS)

La primitive de conversion de données qui convertit une chaîne d'octets en chaîne binaire est appelée la primitive de conversion de chaîne d'octets en chaîne binaire (OS2BS, Octet String to Bit String conversion primitive). Elle prend la chaîne d'octets comme valeur d'entrée et obtient comme résultat la chaîne binaire. La chaîne d'octets $M_{d-1}\, M_{d-2} \ldots M_0$ de longueur d doit être convertie en une chaîne binaire $b_{l-1}\, b_{l-2} \ldots b_0$ de longueur souhaitée $l$. Où $d = \lceil l/8 \rceil$ et les bits de $8d\text{-}l$ les plus à gauche de l'octet le plus à gauche sont à zéro.

Plus précisément, la conversion doit être comme suit:

- pour $0 \le i < l - 1$, laisser les bits $b_{8i+7}\, b_{8i+6} \ldots b_{8i} = M_i$;
- ses $(8d - l)$ bits les plus à gauche de l'octet le plus à gauche doivent être à zéro.

#### 5.3.4.3.4 Conversion entre entiers et chaînes d'octets (I2OS)

La primitive de conversion qui convertit un entier en chaîne d'octets est appelée la primitive de conversion d'entier en chaîne d'octets (I2OS, Integer to Octet String conversion primitive). Elle prend un entier non négatif $x$ et la longueur $d$ souhaitée de la chaîne d'octets comme valeur d'entrée. La longueur $d$ est tenue de satisfaire à $256^d > x$, sinon, elle doit obtenir comme résultat "error" (erreur). I2OS obtient comme résultat la chaîne d'octets correspondante.

L'entier $x$ doit être écrit dans sa représentation unique à $I$ chiffres, en base 256:

- $x = x^{d-1} \cdot 256^{d-1} + x^{d-2} \cdot 256^{d-2} + \cdots + x_1 \cdot 256 + x_0$,;
- où $0 \le x_i < 256$ pour $0 \le i \le d\text{-}1$;
- $M_i = x_i$, pour $0 \le i \le d\text{-}1$.

La chaîne d'octets de sortie doit être $M_{d-1}\, M_{d-2} \ldots M_0$.

#### 5.3.4.3.5 Conversion entre chaînes d'octets et entiers (OS2I)

La primitive de conversion de données qui convertit une chaîne d'octets en entier est appelée la primitive de conversion de chaîne d'octets en entier (OS2I, Octet String to Integer conversion primitive). Elle prend la chaîne d'octets $M_{d-1}\, M_{d-2} \ldots M_0$ de longueur $d$ comme valeur d'entrée et obtient comme résultat l'entier $x$ correspondant. En cas de chaîne d'octets de longueur nulle, la conversion obtient comme résultat l'entier 0.

Chaque octet est interprété comme un entier non négatif sur la base 256. Plus précisément, la conversion doit être comme suit:

- $x_i = M_i$, pour $0 \le i \le d\text{-}1$;
- $x = x_{d-1} \cdot 256^{d-1} + x_{d-2} \cdot 256^{d-2} + \cdots + x_1 \cdot 256 + x_0$.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.3.4.3.6 Conversion entre éléments de champ et chaînes d'octets (FE2OS)

La primitive de conversion de données qui convertit un élément de champ en chaîne d'octets est appelée la primitive de conversion d'élément de champ en chaîne d'octets (FE2OS, Field Element to Octet String conversion primitive). Elle prend un élément de champ comme valeur d'entrée et obtient comme résultat la chaîne d'octets correspondante. Un élément de champ $x \in F_p$ est converti en chaîne d'octets $M_{d-1}\ M_{d-2}\ \ldots\ M_0$ de longueur $d = \lceil log256\ p \rceil$ en appliquant la primitive de conversion I2OS avec le paramètre $l$, où

- FE2OS($x$) = I2OS($x$, $l$).

#### 5.3.4.3.7 Conversion entre chaînes d'octets et éléments de champ (OS2FE)

La primitive de conversion de données qui convertit une chaîne d'octets en élément de champ est appelée la primitive de conversion de chaîne d'octets en élément de champ (OS2FE, Octet String to Field Element conversion primitive). Elle prend une chaîne d'octets comme valeur d'entrée et obtient comme résultat l'élément de champ correspondant. Une chaîne d'octets $M_{d-1}\ M_{d-2}\ \ldots\ M_0$ de longueur $d$ est convertie en élément de champ $x \in F_p$ en appliquant la primitive de conversion OS2I, où:

- OS2FE($x$) = OS2I($x$) mod $p$.

### 5.3.4.4 Signature numérique

NOTE Le texte suivant est issu du document NIST SP 800-21:2005, 3.4.1.

Une signature numérique est un équivalent électronique de la signature écrite qui peut être utilisée pour prouver au destinataire ou à une tierce partie que le message a été signé par l'émetteur (une propriété appelée non-répudiation). Les signatures numériques peuvent également être générées pour des données et programmes stockés de sorte que leur intégrité puisse être vérifiée ultérieurement.

Les signatures numériques authentifient l'intégrité des données signées et l'identité du signataire. Dans un ordinateur, la signature numérique est représentée sous forme de chaîne binaire et est calculée à l'aide d'un algorithme de signature numérique qui fournit des fonctions de génération et de vérification de signatures. La génération de signatures utilise une clé privée pour générer une signature numérique. La vérification de signature utilise la clé publique qui correspond à la clé privée (mais n'est pas identique) afin de vérifier la signature. Chaque signataire possède une paire clé publique/clé privée. La génération de signature peut uniquement être effectuée par le détenteur de la clé privée du signataire. Cependant, la vérification de la signature peut être effectuée par toute personne qui a la clé publique du signataire. La sécurité du système de signatures numériques dépend du maintien du caractère secret de la clé privée d'un signataire. Par conséquent, les utilisateurs doivent se protéger contre l'acquisition non autorisée de leurs clés privées.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

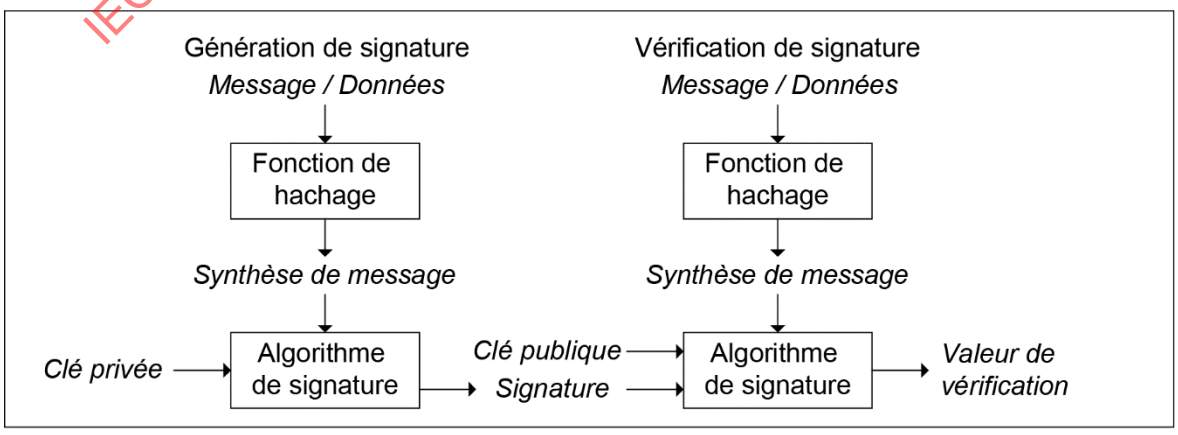



**Figure 19 – Signatures numériques**

La Figure 19 représente le processus de signature numérique. Une fonction de hachage (voir 5.3.2) est utilisée dans le processus de génération de signature pour obtenir une version condensée de données à signer, appelée "synthèse de message" ou "valeur de hachage". La synthèse de message est alors saisie dans l'algorithme de signature numérique pour générer la signature numérique. La signature numérique ainsi que les données signées (souvent appelées "le message") sont envoyées au vérificateur prévu. Le vérificateur du message et de la signature vérifie la signature en utilisant la clé publique du signataire. La même fonction de hachage et le même algorithme de signature numérique doivent également être utilisés lors du processus de vérification. Des procédures similaires peuvent être utilisées pour générer et vérifier les signatures pour les données stockées et transmises.

#### 5.3.4.5 Signature numérique à courbe elliptique (ECDSA)

Pour DLMS®/COSEM, l'algorithme de signature numérique à courbe elliptique (ECDSA) tel que spécifié dans le document FIPS PUB 186-4:2013 a été choisi. Le document NSA1 fournit un guide de mise en œuvre.

Dans DLMS®/COSEM, les courbes elliptiques et les algorithmes utilisés doivent être:

- en cas de suite de sécurité 1, la courbe elliptique P-256 avec l'algorithme de hachage SHA-256;
- en cas de suite de sécurité 2, la courbe elliptique P-384 avec l'algorithme de hachage SHA-384.

Les valeurs d'entrée de la génération de signature numérique ECDSA sont les suivantes:

- le message $M$ à signer;

  NOTE 1 Dans le contexte de DLMS®/COSEM, "Message" peut être une APDU xDLMS (voir 5.7.2), ou des données COSEM (voir 0).

- la clé privée du signataire, $d$.

La valeur de sortie est la signature ECDSA $(r, s)$ sur M.

Dans DLMS®/COSEM, le format brut doit être utilisé: la signature $(r, s)$ est codée comme une chaîne d'octets $R$ II $S$, c'est-à-dire comme une concaténation des chaînes d'octets $R$ = I2OS $(r, l)$ et $S$ = I2OS$(s, l)$ avec $l = \lceil log_{256}\, n \rceil$. Par conséquent, la signature a une longueur fixe de $2\, l$ octets.

NOTE 2 Ici, $n$ est l'ordre du point de base $G$ de la courbe elliptique. I2OS est la primitive de conversion d'entier en chaîne d'octets. Voir 5.3.4.3.

Les valeurs d'entrée vers la vérification de la génération de signature numérique ECDSA sont:

- le message signé $M'$;
- la signature ECDSA reçue $(r', s')$;
- la clé publique authentique du signataire, $Q$.

Le processus de génération et de vérification des signatures doit être tel que spécifié dans le document NSA1, 3.4.

#### 5.3.4.6 Agrément de clé

##### 5.3.4.6.1 Vue d'ensemble

L'agrément de clé permet à deux entités de calculer ensemble un secret partagé et de dériver la clé secrète de chiffrement à partir du secret partagé. Voir également NIST SP 800-56A Rev. 2: 2013.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Pour DLMS®/COSEM, trois mécanismes d'agrément de clé à courbe elliptique ont été tirés du document NIST SP 800-56A Rev. 2: 2013:

- le schéma Ephemeral Unified Model (modèle unifié éphémère) C(2e, 0s, ECC CDH);
- le schéma One-pass Diffie-Hellman C(1e, 1s, ECC CDH);
- le schéma de modèle unifié statique C(0e, 2s, ECC CDH).

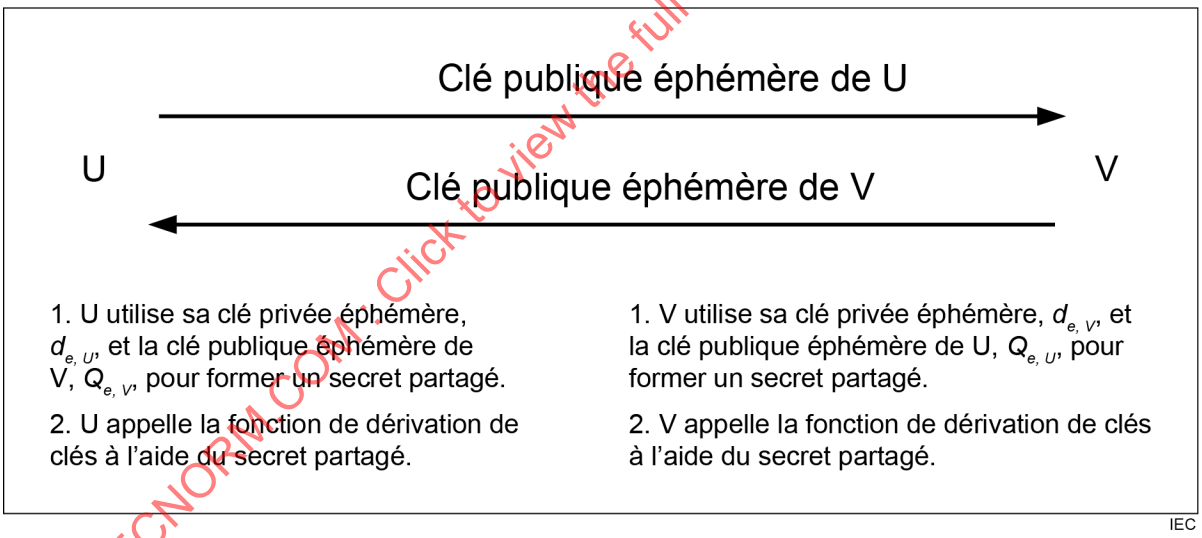
NOTE Les deux premiers schémas ont été utilisés pour la Suite B NSA.

#### 5.3.4.6.2 Schéma Ephemeral Unified Model C(2e, 0s, ECC CDH)

Ce schéma est destiné à être utilisé entre un client DLMS®/COSEM et un serveur pour convenir de la clé principale, sur les clés globales de cryptage et/ou la clé d'authentification. Le client joue le rôle de la partie U et le serveur joue le rôle de la partie V. Le processus est pris en charge par les méthodes de la classe d'interface "Security setup" (voir IEC 62056-6-2:2021, 4.4.7).

Les parties génèrent une paire de clés éphémères à partir des paramètres de domaine $D$. Les parties échangent des clés publiques éphémères, puis calculent le secret partagé $Z$ à l'aide des paramètres de domaine, de leur clé privée éphémère et de la clé publique éphémère de l'autre partie. La clé secrète de chiffrement est dérivée à l'aide de la fonction de dérivation de clé spécifiée en 5.3.4.6.5 à partir du secret partagé $Z$ et des autres valeurs d'entrée.

Le processus est spécifié en détail dans le document NIST SP 800-56A Rev. 2:2013, 6.1.2.2, et dans le document NSA2:2009, 3.1, et il est représenté à la Figure 20 et dans le Tableau 4.




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

NOTE Cette figure reproduit la Figure 1 du document NSA2:2009.

**Figure 20 – Schéma C(2e, 0s): chaque partie apporte uniquement une paire de clés éphémères**

Conditions préalables:

a) chaque partie a une copie authentique du même ensemble de paramètres de domaine, $D$. $D$ doit être choisi à partir de l'un des deux ensembles de paramètres de domaine (voir Annexe G);

b) les parties ont convenu de l'utilisation de la concaténation KDF de NIST (voir 5.3.4.6.5). SHA-256 est la fonction de hachage à utiliser avec les paramètres de domaine pour P-256 et SHA-384 est la fonction de hachage à utiliser avec les paramètres de domaine pour P-384;

c) avant ou pendant le processus d'agrément de clé, les parties obtiennent l'identificateur associé à l'autre partie pendant le mécanisme d'agrément de clé.

NOTE Voir également NIST SP 800-56A Rev. 2:2013 et NSA2 pour avoir des informations supplémentaires concernant l'assurance de la validité des paramètres de domaine, la validité de la clé privée et de la clé publique et l'assurance relative à la possession de la clé privée.

**Tableau 4 – Récapitulatif de mécanisme d'agrément de clé Ephemeral Unified Model**

| | Partie U | Partie V |
|---|---|---|
| Paramètres de domaine | (q, FR, a, b{,SEED}, G, n, h)<br>tel que déterminé par la suite de sécurité | (q, FR, a, b{,SEED}, G, n, h)<br>tel que déterminé par la suite de sécurité |
| Données statiques | s.o. | s.o. |
| Données éphémères | Clé privée éphémère, $d_{e,U}$<br>Clé publique éphémère, $Q_{e,U}$ | Clé privée éphémère, $d_{e,V}$<br>Clé publique éphémère, $Q_{e,V}$ |
| Calcul | Calcul de $Z$ en appelant ECC CDH à l'aide de $d_{e,U}$ et $Q_{e,V}$ | Calcul de $Z$ en appelant ECC CDH à l'aide de $d_{e,V}$ et $Q_{e,U}$ |
| Dérivation de la clé secrète de chiffrement | 1. Calcul de kdf($Z$, $OtherInput$)<br>2. Destruction de $Z$ | 1. Calcul de kdf($Z$, $OtherInput$)<br>2. Destruction de $Z$ |
| NOTE Ce tableau est basé sur les documents NIST SP 800-56A Rev. 2: 2013, Tableau 18, et NSA2:2009, Tableau 1. | | |

Les justifications du choix du schéma C(2e, 0s) sont spécifiées dans le document NIST SP 800-56A Rev. 2:2013, 8.2.

L'utilisation de ce schéma dans DLMS®/COSEM est expliquée plus en détail en I.1.

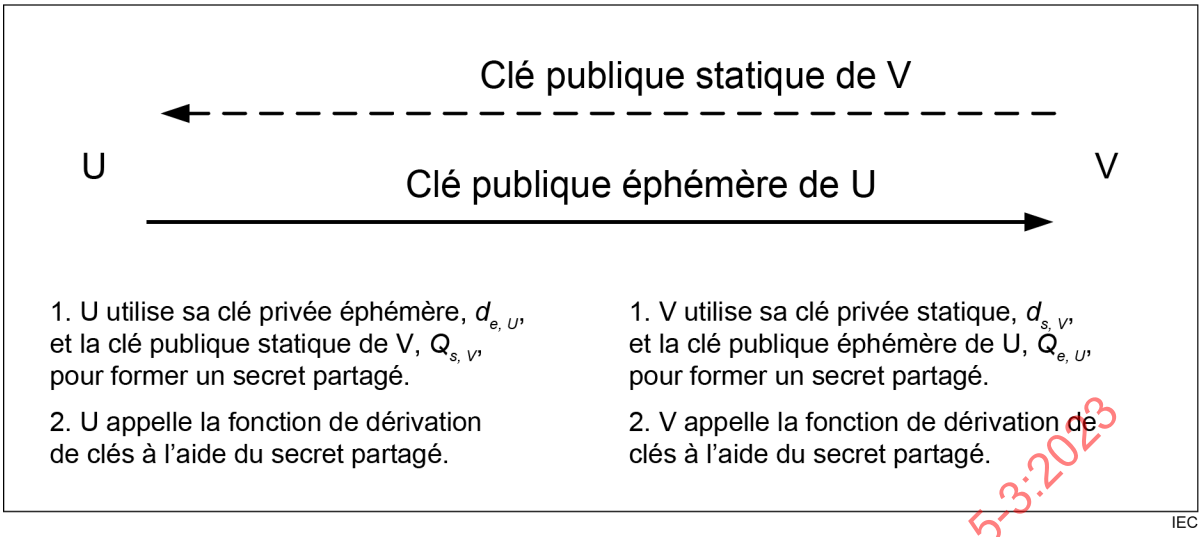
#### 5.3.4.6.3 Schéma One-pass Diffie-Hellman C(1e, 1s, ECC CDH)

Ce schéma est destiné à être utilisé par un serveur DLMS®/COSEM et une autre partie pour convenir d'une clé éphémère de cryptage pour protéger les APDU xDLMS ou les données COSEM. La partie qui envoie le message (l'émetteur) joue le rôle de la partie U et l'autre partie (le destinataire) joue le rôle de la partie V.

NOTE 1 Les termes "émetteur" et "destinataire" font référence aux champs de l'APDU general-ciphering.

Pour ce schéma, la partie U génère une paire de clés éphémères; la partie V a uniquement une paire de clés statiques. La partie U obtient la clé publique statique de la partie V de manière fiable (par exemple, à partir d'un certificat signé par une CA de confiance) et envoie sa clé publique éphémère à la partie V. Chaque partie dérive le secret partagé $Z$ en utilisant sa propre clé privée et la clé publique de l'autre partie. Chaque partie dérive la clé secrète de chiffrement à l'aide de la méthode de dérivation de clé spécifiée en 5.3.4.6.5 à partir du secret partagé $Z$ et des autres valeurs d'entrée.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




NOTE Cette figure reproduit la Figure 2 du document NSA2:2009.

**Figure 21 – Schémas C(1e, 1s): la partie U apporte une paire de clés éphémères, et la partie V apporte une paire de clés statiques**

Le processus est spécifié en détail dans le document NIST SP 800-56A Rev. 2:2013, 6.2.2.2, et dans le document NSA2:2009, 3.2, et il est représenté à la Figure 21 et dans le Tableau 5.

Conditions préalables:

a) chaque partie doit avoir une copie authentique du même ensemble de paramètres de domaine, $D$. $D$ doit être choisi à partir de l'un des deux ensembles de paramètres de domaine (voir Annexe G);

b) la partie V doit avoir été désignée comme étant propriétaire d'une paire de clés statiques générée comme spécifié en 5.6.2 à l'aide de l'ensemble de paramètres de domaine, $D$;

c) les parties ont convenu de l'utilisation de la concaténation KDF de NIST (voir 5.3.4.6.5). SHA-256 est la fonction de hachage à utiliser avec les paramètres de domaine pour P-256 et SHA-384 est la fonction de hachage à utiliser avec les paramètres de domaine pour P-384;

d) avant ou pendant le processus d'agrément de clé, les parties obtiennent l'identificateur associé à l'autre partie pendant le mécanisme d'agrément de clé. La partie U doit obtenir la clé publique statique qui est liée à l'identificateur de la partie V. Cette clé publique statique doit être obtenue de manière fiable (par exemple, à partir d'un certificat signé par une CA de confiance).

NOTE 2 Voir également NIST SP 800-56A Rev. 2:2013 et NSA2 pour avoir des informations supplémentaires concernant l'assurance de la validité des paramètres de domaine, la validité de la clé privée et de la clé publique et l'assurance relative à la possession de la clé privée.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 5 – Récapitulatif de mécanisme d'agrément de clé One-pass Diffie-Hellman**

| | Partie U | Partie V |
|---|---|---|
| Paramètres de domaine | (q, FR, a, b{,SEED}, G, n, h)<br>Tel que déterminé par la suite de sécurité | (q, FR, a, b{,SEED}, G, n, h)<br>Tel que déterminé par la suite de sécurité |
| Données statiques | s.o. | Clé privée statique, $d_{s,\ V}$<br>Clé publique statique, $Q_{s,\ V}$ |
| Données éphémères | Clé privée éphémère, $d_{e,\ U}$<br>Clé publique éphémère, $Q_{e,\ U}$ | s.o. |
| Calcul | Calcul de $Z$ en appelant ECC CDH à l'aide de $d_{e,\ U}$ et $Q_{s,\ V}$ | Calcul de $Z$ en appelant ECC CDH à l'aide de $d_{s,\ V}$ et $Q_{e\ ,U}$ |
| Dérivation de la clé secrète de chiffrement | 1. Calcul de kdf($Z$, $OtherInput$)<br>2. Destruction de $Z$ | 1. Calcul de kdf($Z$, $OtherInput$)<br>2. Destruction de $Z$ |
| NOTE Ce tableau est basé sur les documents NIST SP 800-56A Rev. 2:2013, Tableau 24, et NSA2:2009, Tableau 2. | | |

Les justifications du choix de ce schéma sont spécifiées dans le document NIST SP 800-56A Rev. 2:2013, 8.4.

L'utilisation de ce schéma dans DLMS®/COSEM est expliquée plus en détail en I.2.

#### 5.3.4.6.4 Schéma de modèle unifié statique C(0e, 2s, ECC CDH)

Ce schéma est destiné à être utilisé par un serveur DLMS®/COSEM et une autre partie pour convenir d'une clé éphémère de cryptage pour protéger les APDU xDLMS ou les données COSEM. La partie qui envoie le message (l'émetteur) joue le rôle de la partie U et l'autre partie (le destinataire) joue le rôle de la partie V.

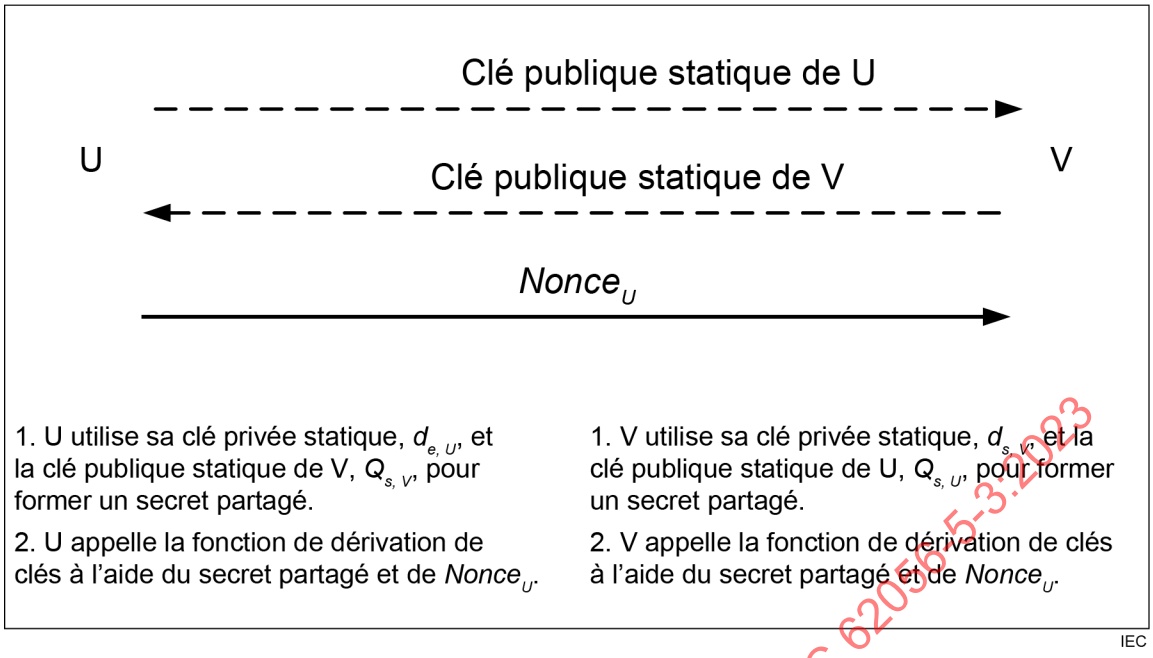
NOTE 1 Les termes "émetteur" et "destinataire" font référence aux champs de l'APDU general-ciphering.

Dans ce cas, les parties utilisent uniquement des paires de clés statiques. Chaque partie obtient la clé publique statique de l'autre partie. Une valeur unique, $Nonce_U$, est envoyée par la partie U à la partie V pour assurer que le matériel de chiffrement dérivé est différent pour chaque transaction d'établissement de clé.

Les parties dérivent le secret partagé $Z$ à l'aide de leur propre clé privée statique et de la clé publique statique de l'autre partie. La clé secrète de chiffrement est dérivée à l'aide de la fonction de dérivation de clé spécifiée en 5.3.4.6.5, du secret partagé $Z$, de l'identificateur de U et de V et de la valeur unique.

Le processus est représenté à la Figure 22 et présenté dans le Tableau 6.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




NOTE Cette figure est basée sur le document NIST SP 800-56A Rev. 2:2013, Figure 15.

**Figure 22 – Schéma C(0e, 2s): chaque partie apporte uniquement une paire de clés statiques**

Conditions préalables:

a) chaque partie doit avoir une copie authentique du même ensemble de paramètres de domaine, $D$. $D$ doit être choisi à partir de l'un des deux ensembles de paramètres de domaine (voir Annexe G);

b) chaque partie a été désignée comme propriétaire d'une paire de clés statiques générée comme spécifié en 5.6.2 à l'aide de l'ensemble de paramètres de domaine, $D$;

c) les parties ont convenu de l'utilisation de la concaténation KDF de NIST (voir 5.3.4.6.5). SHA-256 est la fonction de hachage à utiliser avec les paramètres de domaine pour P-256 et SHA-384 est la fonction de hachage à utiliser avec les paramètres de domaine pour P- 384;

d) avant ou pendant le processus d'agrément de clé, chaque partie doit obtenir l'identificateur associé à l'autre partie pendant le mécanisme d'agrément de clé;

e) les deux parties doivent obtenir la clé publique statique de l'autre partie qui est liée à l'identificateur. Ces clés publiques statiques doivent être obtenues de manière fiable (par exemple, à partir d'un certificat signé par une CA de confiance).

NOTE 2 Voir également NIST SP 800-56A Rev. 2:2013 pour avoir des informations supplémentaires concernant l'assurance de la validité des paramètres de domaine, la validité de la clé privée et de la clé publique et l'assurance relative à la possession de la clé privée.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 6 – Récapitulatif du mécanisme d'agrément de clé du modèle unifié statique**

| | Partie U | Partie V |
|---|---|---|
| **Paramètres de domaine** | (q, FR, a, b{,SEED}, G, n, h)<br>Tel que déterminé par la suite de sécurité | (q, FR, a, b{,SEED}, G, n, h)<br>Tel que déterminé par la suite de sécurité |
| **Données statiques** | Clé privée statique, $d_{s,\,U}$<br>Clé publique statique, $Q_{s,\,U}$ | Clé privée statique, $d_{s,\,V}$<br>Clé publique statique, $Q_{s,\,V}$ |
| **Données éphémères** | $\text{Nonce}_U$ | |
| **Calcul** | Calcul de $Z$ en appelant ECC CDH à l'aide de $d_{s,\,U}$ et $Q_{s,\,V}$ | Calcul de $Z$ en appelant ECC CDH à l'aide de $d_{s,\,V}$ et $Q_{s\,,U}$ |
| **Dérivation de la clé secrète de chiffrement** | 1. Calculer DerivedKeyingMaterial à l'aide de $\text{Nonce}_U$<br>2. Destruction de $Z$ | 1. Calculer DerivedKeyingMaterial à l'aide de $\text{Nonce}_U$<br>2. Destruction de $Z$ |
| NOTE 1 Ce tableau est basé sur le document NIST SP 800-56A Rev. 2:2013, Tableau 26.<br><br>NOTE 2 La valeur de $Z$ est la même dans toutes les transactions d'établissement de clé C(0e, 2s) entre deux mêmes parties. Par conséquent, si elle vient à être compromise, tous les matériels de chiffrement dérivés au cours des transactions d'établissement de clé C(0e, 2s) passées, actuelles et futures entre ces deux mêmes entités qui utilisent ces mêmes paires de clés statiques peuvent également être compromis. Tout secret partagé $Z$ qui n'est pas "remis à zéro" doit être stocké et utilisé avec les mêmes protections de sécurité que les clés privées. | | |

Les justifications du choix de ce schéma sont spécifiées dans le document NIST SP 800-56A Rev. 2:2013, 8.5.

L'utilisation de ce schéma dans DLMS®/COSEM est expliquée plus en détail en I.3.

#### 5.3.4.6.5 Fonction de dérivation de clé — La concaténation KDF de NIST

La concaténation KDF de NIST, telle que spécifiée dans le document NIST SP 800-56A Rev. 2:2013, 5.8.1.1, et le document NSA2, Article 5, doit être utilisée. Elle se déroule comme suit:

**Appel de fonction**: kdf(*Z*, *OtherInput*)

où *OtherInput* est constituée de *keydatalen* et de *OtherInfo*.

**Paramètres dépendant de l'implémentation:**

a) *hashlen*: entier qui indique la longueur (en bits) de la valeur de sortie de la fonction de hachage, *hash*, utilisé pour dériver des blocs de clés secrètes de chiffrement. Il est de 256 (pour SHA-256) en cas de suite de sécurité 1 et de 384 (pour SHA-384) en cas de suite de sécurité 2.

b) *max_hash_inputlen*: entier qui indique la longueur maximale (en bits) de la valeur d'entrée de la ou des chaînes binaires vers la fonction de hachage.

La longueur doit être inférieure à 264 bits pour SHA-256 et inférieure à 2 128 bits pour SHA-384.

**Fonction:** H: fonction de hachage: SHA-256 en cas de suite de sécurité 1 et SHA-384 en cas de suite de sécurité 2.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Valeur d'entrée:**

c) *Z*: chaîne d'octets qui représente le secret partagé *z*;

d) *keydatalen*: entier qui indique la longueur (en bits) de la clé secrète de chiffrement à générer: 128 bits pour la suite de sécurité 1 et 256 bits pour la suite de sécurité 2;

e) *OtherInfo:* chaîne binaire égale à la concaténation suivante:

*AlgorithmID* || *PartyUInfo* || *PartyVInfo* {||*SuppPubInfo*}{||*SuppPrivInfo*}

où les sous-champs sont définis comme suit:

f) *AlgorithmID*: Chaîne binaire qui indique la manière dont la clé secrète de chiffrement est analysée et pour quel ou quels algorithmes la clé secrète de chiffrement dérivée est utilisée. Voir Tableau 8;

g) *PartyUInfo*: Chaîne binaire qui contient des informations publiques exigées par l'application utilisant cette KDF que la Partie U est tenue d'apporter au processus de dérivation de clés;

h) *PartyVInfo*: Chaîne binaire qui contient des informations publiques exigées par l'application utilisant cette KDF que la Partie V est tenue d'apporter au processus de dérivation de clés;

i) (Facultatif) *SuppPubInfo*: Chaîne binaire qui contient des informations publiques supplémentaires mutuellement connues. Non utilisé dans DLMS®/COSEM;

j) (Facultatif) *SuppPrivInfo*: Chaîne binaire qui contient des informations privées supplémentaires mutuellement connues (par exemple, une clé symétrique secrète partagée qui a été communiquée par le biais d'un canal séparé). Non utilisé dans DLMS®/COSEM.

Le format et le contenu de chaque sous-champ et sous-chaîne sont spécifiés dans le Tableau 7.

**Tableau 7 – Sous-champs et sous-chaînes *OtherInfo***

| Sous-champ / Sous-chaîne | Mécanisme d'agrément de clé | | | Longueur en octets | Valeur |
|---|---|---|---|---|---|
| | C(2e, 0s) | C(1e, 1s) | C(0e, 2s) | | |
| | Format | | | | |
| AlgorithmID | Fixe | Fixe | Fixe | 7 | Voir Tableau 8 |
| PartyUinfo | Fixe | Fixe | Variable | 8+n | – |
| $ID_U$ | Fixe | Fixe | Fixe | 8 | originator-system-title |
| NonceU | – | – | Variable | n | *Datalen* = longueur de transaction-id, doit être de 1 octet<br>*Data* = valeur de transaction-id |
| PartyVInfo | Fixe | Fixe | Fixe | 8 | – |
| $ID_V$ | Fixe | Fixe | Fixe | 8 | recipient-system-title |
| "originator-system-title", "transaction-id" et "recipient-system-title" sont les champs de l'APDU general-ciphering. | | | | | |

Dans DLMS®/COSEM, la dérivation de clé fournit une clé unique pour un objectif donné. La longueur est telle que déterminée par la suite de sécurité. *AlgorithmId* doit être tel que spécifié dans le Tableau 8. Voir aussi 7.2.2.4.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 8 – ID d'algorithmes de sécurité**

| Algorithme | ID d'algorithme cryptographique COSEM | Valeur codée |
|---|---|---|
| AES-GCM-128 | 2.16.756.5.8.3.0 | 60 85 74 05 08 03 00 |
| AES-GCM-256 | 2.16.756.5.8.3.1 | 60 85 74 05 08 03 01 |
| AES-WRAP-128 | 2.16.756.5.8.3.2 | 60 85 74 05 08 03 02 |
| AES-WRAP-256 | 2.16.756.5.8.3.3 | 60 85 74 05 08 03 03 |

### 5.3.5 Génération de nombres aléatoires

Un générateur de nombres aléatoires (RNG) fort doit être fourni pour générer les nombres aléatoires exigés pour les différents algorithmes utilisés dans DLMS®/COSEM. Le RNG doit, de préférence, être non déterministe. Si aucun RNG non déterministe n'est disponible, le système doit utiliser une entropie suffisante pour créer une valeur de départ de bonne qualité pour un RNG déterministe.

### 5.3.6 Compression

NOTE La compression n'implique pas la cryptographie, mais il s'agit d'une transformation de l'APDU xDLMS. En plus d'être contrôlée avec le chiffrement à clé symétrique, c'est la raison pour laquelle elle est spécifiée dans la présente norme.

La compression peut être appliquée aux données COSEM ou aux APDU xDLMS. Ce processus peut être combiné au chiffrement à clé symétrique. Voir 5.7.2.4.7 et 5.7.2.4.8. L'algorithme de compression doit être tel que spécifié dans l'UIT-T V.44: 2000. Cet algorithme a été choisi pour satisfaire aux exigences suivantes:

- charge de traitement basse;
- exigences de mémoires basses;
- faible latence.

L'utilisation de la compression est indiquée par le bit 7 de l'octet de contrôle de sécurité, spécifié en 5.7.2.4.

### 5.3.7 Suite de sécurité

Une suite de sécurité détermine l'ensemble d'algorithmes cryptographiques disponibles pour les différentes primitives cryptographiques et les tailles de clé.

Les suites de sécurité DLMS®/COSEM (voir Tableau 9) sont basées sur la Suite B NSA et comprennent des algorithmes cryptographiques pour l'authentification, le cryptage, l'agrément de clé, la signature numérique et le hachage, en particulier:

- authentification et cryptage: la norme Advanced Encryption Standard (AES) doit être utilisée tel que spécifié dans le document FIPS PUB 197, avec des tailles de clé de 128 bits et 256 bits. L'AES doit être utilisée avec le mode Galois/Counter (GCM) de fonctionnement spécifié dans le document NIST SP 800-38D:2007;
- signature numérique: l'algorithme de signature numérique à courbe elliptique (ECDSA) doit être utilisé tel que spécifié dans les documents FIPS PUB 186-4:2013 et NSA1, à l'aide des courbes P-256 ou P-384;
- agrément de clé:
  - schéma Ephemeral Unified Model C(2e, 0s, ECC CDH) (voir 5.3.4.6.2);
  - schéma One-pass Diffie-Hellman C(1e, 1s, ECC CDH) (voir 5.3.4.6.3);
  - schéma de modèle unifié statique C(0e, 2s, ECC CDH) (voir 5.3.4.6.4);

  doit être utilisé à l'aide des courbes elliptiques P-256 ou P-384;

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- hachage: les algorithmes de hachage sécurisé (SHA) SHA-256 et SHA-384 doivent être utilisés tel que spécifié dans le document FIPS PUB 180-4:2012.

En outre, un enveloppement de clé et un algorithme de compression sont disponibles.

**Tableau 9 – Suites de sécurité DLMS®/COSEM**

| ID de la suite de sécurité | Nom de la suite de sécurité | Cryptage authentifié | Signature numérique | Agrément de clé | Hachage | Transport de clé | Compression |
|---|---|---|---|---|---|---|---|
| 0 | AES-GCM-128 | AES-GCM-128 | – | – | – | enveloppement de clé AES-128 | – |
| 1 | ECDH-ECDSA-AES-GCM-128-SHA-256 | AES-GCM-128 | ECDSA avec P-256 | ECDH avec P-256 | SHA-256 | enveloppement de clé AES-128 | V.44 |
| 2 | ECDH-ECDSA-AES-GCM-256-SHA-384 | AES-GCM-256 | ECDSA avec P-384 | ECDH avec P-384 | SHA-384 | AES-256 enveloppement de clé | V.44 |
| Tous les autres sont réservés | – | – | – | – | – | – | – |

## 5.4 Clés cryptographiques — Vue d'ensemble

Une clé cryptographique est un paramètre utilisé conjointement avec un algorithme cryptographique qui détermine son fonctionnement de sorte qu'une entité ayant des connaissances sur la clé puisse reproduire ou inverser le fonctionnement, tandis qu'une entité sans connaissance sur la clé ne peut pas le faire. Dans DLMS®/COSEM, les exemples de fonctionnements comprennent:

- la transformation de texte brut en texte chiffré;
- la transformation de texte chiffré en texte brut;
- le calcul et la vérification d'un code d'authentification (MAC);
- l'enveloppement de clé;
- l'application et la vérification d'une signature numérique;
- l'agrément de clé.

Les clés utilisées avec des algorithmes à clé symétrique sont spécifiées en 5.5.

Les clés utilisées avec des algorithmes à clé publique sont spécifiées en 5.6.

## 5.5 Clés utilisées avec des algorithmes à clé symétrique

### 5.5.1 Types de clés symétriques

Les clés symétriques sont classées selon:

a) leur objectif:

1) une clé de cryptage de clé (KEK) est utilisée pour crypter/décrypter d'autres clés symétriques (voir 5.5.4). Dans DLMS®/COSEM, il s'agit de la clé principale;

2) une clé de cryptage est utilisée comme clé de chiffrement par blocs de l'algorithme AES-GCM (voir aussi 5.3.3.7.4);

3) une clé d'authentification est utilisée à titre de données supplémentaires authentifiées (AAD) dans l'algorithme AES-GCM (voir aussi 5.3.3.7.5);

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

b) leur durée de vie:

1) clés statiques qui sont destinées à être utilisées pour un laps de temps relativement long. Dans DLMS®/COSEM, ces clés peuvent être:

- une clé globale qui peut être utilisée sur plusieurs AA établies de manière répétée entre les mêmes partenaires. Une clé globale peut être une clé de cryptage de monodiffusion (GUEK), une clé de cryptage de diffusion (GBEK) ou une clé d'authentification (GAK);
- une clé dédiée pouvant être utilisée de manière répétée au cours de l'établissement d'une AA unique entre deux partenaires. Par conséquent, sa durée de vie est identique à celle de l'AA. Une clé dédiée peut uniquement être une clé de cryptage de monodiffusion;

2) les clés éphémères utilisées généralement pour un échange unique dans une AA.

Pour la génération et la distribution des clés symétriques, voir NIST SP 800-57:2012, 8.1.5.2.

Une clé principale et les clés globales sont établies entre chaque paire DLMS®/COSEM client-serveur utilisant l'une des méthodes présentées dans le Tableau 10. Il convient de renouveler ces clés pendant les intervalles appropriés (voir 5.3.3.7.3 et 5.5.6).

Les clés dédiées sont générées par le client DLMS®/COSEM et transportées jusqu'au serveur dans le champ dédié à la clé de l'APDU xDLMS InitiateRequest, contenue dans le champ user-information de l'APDU AARQ. Lorsque la clé dédiée est présente, l'APDU xDLMS InitiateRequest doit être authentifiée et cryptée à l'aide de l'algorithme AES-GCM-128 / 256, de la clé globale de cryptage de monodiffusion, et (le cas échéant, voir 5.3.3.7.5) de la clé d'authentification. L'APDU xDLMS InitiateResponse, contenue dans le champ user-information de l'APDU AARE, doit également être cryptée et authentifiée de la même manière. Lorsque la clé dédiée est utilisée, le bit "key-set" de l'octet de contrôle de sécurité (voir Tableau 27) n'est pas pertinent et doit être défini sur zéro.

NOTE Les APDU AARQ et AARE ne sont pas protégées.

Le Tableau 10 récapitule les types de clés symétriques, leur objectif, les méthodes permettant de les établir et leur utilisation avec les différentes APDU et entre les différentes entités.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 10 – Types de clés symétriques**

| Type de clé | Objectif | Établissement de clé | Utilisation |
|---|---|---|---|
| Clé principale, KEK [1] | Clé de cryptage de clé (KEK) pour:<br>– la (nouvelle) clé principale;<br>– les clés globales de cryptage ou d'authentification;<br>– les clés éphémères de cryptage. | Hors bande<br>Enveloppement de clé<br>Agrément de clé [3] | Peut être identifiée comme KEK dans les APDU general-ciphering entre client-serveur (voir Tableau 11) |
| Clé globale de cryptage de monodiffusion, GUEK [2] | Clé de chiffrement par blocs pour la monodiffusion<br>– APDU xDLMS et/ou<br>– données COSEM | Enveloppement de clé<br>Agrément de clé [3] | – APDU de chiffrement global spécifique au service entre client-serveur<br>– APDU general-glo-ciphering entre client-serveur<br>– APDU general-ciphering entre client-serveur<br>– paramètres de protection d'objet "Data protection" (protection des données) |
| Clé globale de cryptage de diffusion, GBEK [2] | Clé de chiffrement par blocs pour la diffusion<br>– APDU xDLMS et/ou<br>– données COSEM | Enveloppement de clé | |
| Clé (globale) d'authentification, GAK [2] | Partie des AAD du processus de chiffrement des APDU xDLMS et/ou des données COSEM | Enveloppement de clé<br>Agrément de clé [3] | Toutes les APDU entre client-serveur et tierce partie-serveur |
| Clé dédiée (monodiffusion) | Clé de chiffrement par blocs des APDU xDLMS de monodiffusion, dans une AA établie | Transport de clé dans l'APDU xDLMS Initiate.request | – APDU de chiffrement dédié spécifique au service entre client-serveur<br>– APDU general-ded-ciphering entre client-serveur<br>pendant la durée de vie d'une AA |
| Clé éphémère de cryptage | Clé de chiffrement par blocs pour:<br>– APDU xDLMS et/ou<br>– données COSEM | Enveloppement de clé | – APDU general-ciphering entre client-serveur<br>– paramètres de protection d'objet "Data protection" (protection des données) |
| | Clé de chiffrement par blocs pour:<br>– APDU xDLMS et/ou<br>– données COSEM | Agrément de clé [4] | – APDU general-ciphering entre client-serveur ou entre tierce partie-serveur<br>– paramètres de protection d'objet "Data protection" (protection des données) |

[1] Contenue par un objet "Security setup". Différentes AA peuvent utiliser les mêmes objets "Security setup" ou des objets "Security setup" différents.

[2] Contenue par un objet "Security setup". Différentes AA peuvent utiliser les mêmes objets "Security setup" ou des objets "Security setup" différents. L'utilisation de la GUEK ou de la GBEK peut être identifiée par:

– le bit key-set de l'octet de contrôle de sécurité (voir Tableau 27);

– le paramètre key-id de l'APDU general-ciphering (voir Tableau 11);

– les paramètres de protection de l'IC "Data protection" (voir IEC 62056-6-2:2021, 4.4.9).

[3] Établi à l'aide du schéma Ephemeral Unified Model C(2e, 0s, ECC CDH) (voir 5.3.4.6.2).

[4] Établi à l'aide du schéma One-pass Diffie-Hellman C(1e, 1s, ECC CDH) ou le schéma de modèle unifié statique C(0e, 2s, ECC CDH). Voir 5.3.4.6.3 et 5.3.4.6.4.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.5.2 Informations relatives aux clés avec APDU general-ciphering et protection des données

Lorsque l'APDU general-ciphering est utilisée pour protéger des APDU xDLMS ou lorsque les données COSEM sont protégées, l'expéditeur envoie les informations nécessaires concernant la clé qui a été utilisée/doit être utilisée pour chiffrer/déchiffrer l'APDU xDLMS/les données COSEM, ainsi que l'APDU xDLMS/les données COSEM chiffrées.

Les informations exigées concernant les clés sont récapitulées dans le Tableau 11 et spécifiées en détail en 5.5.3, 5.5.4 et 5.5.5.

**Tableau 11 – Informations relatives aux clés avec APDU general-ciphering et protection des données**

| Choix des informations relatives aux clés | | Commentaire |
|---|---|---|
| key-Info | | |
| **identified-key** | S | L'EK est identifiée |
| key-id | O | |
| global-unicast-encryption-key | S | GUEK |
| global-broadcast-encryption-key | S | GBEK |
| **wrapped-key** | S | L'EK est transportée à l'aide de l'enveloppement de clé |
| kek-id | O | |
| master-key | O | Identifie la clé utilisée pour envelopper key-ciphered-data.<br>0 = clé principale (KEK) |
| key-ciphered-data | O | Clé générée de manière aléatoire enveloppée avec KEK |
| **agreed-key** | S | La clé fait l'objet d'un accord entre les parties en utilisant:<br>– le schéma One-pass Diffie-Hellman C(1e, 1s, ECC CDH) (voir 5.3.4.6.3);<br>– le schéma de modèle unifié statique C(0e, 2s, ECC CDH) (voir 5.3.4.6.4). |
| key-parameters | O | Identificateur du mécanisme d'agrément de clé:<br>0x01: C(1e, 1s ECC CDH)<br>0x02: C(0e, 2s ECC CDH)<br>Tous les autres sont réservés. |
| key-ciphered-data | O | – En cas de schéma C(1e, 1s, ECC CDH): la clé publique $Q_{e,U}$, de la paire de clés d'agrément de clé éphémère de la partie U, signée avec la clé de signature numérique privée de la partie U.<br>– En cas de schéma C(0e, 2s, ECC CDH): chaîne d'octets de longueur nulle. Dans ce cas, la partie U est tenue de fournir une valeur unique, $Nonce_U$. Voir 5.3.4.6.4 et 5.3.4.6.5. |
| O: Obligatoire (partie d'une SEQUENCE)<br>S: Sélectionnable (partie d'un CHOICE)<br>Pour la spécification ASN.1, voir Article 8. | | |
| NOTE L'utilisation de l'identification de clé limite l'échange d'APDU xDLMS/de données COSEM protégées entre un client et un serveur, car la GUEK et la GBEK ne doivent pas être divulguées à une quelconque partie autre que le client et le serveur. | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.5.3 Identification de clé

La clé identifiée peut être la clé globale de cryptage de monodiffusion (GUEK) ou la clé globale de cryptage de diffusion (GBEK). Dans ce cas, le bit "key-set" de l'octet de contrôle de sécurité (voir Tableau 27) n'est pas pertinent et doit être défini sur zéro.

### 5.5.4 Enveloppement de clé

L'enveloppement de clé peut être utilisé pour établir des clés symétriques statiques ou éphémères.

L'algorithme est l'algorithme AES d'enveloppement de clé spécifié en 5.3.3.8. La KEK est la clé principale. Par conséquent, cette méthode peut être utilisée uniquement entre des parties partageant la clé principale, c'est-à-dire entre un client et un serveur.

Les clés statiques pouvant être établies à l'aide de l'enveloppement de clé peuvent être:

- la clé principale, KEK;
- la clé globale de cryptage de monodiffusion, GUEK;
- la clé globale de cryptage de diffusion, GBEK;
- la clé (globale) d'authentification, GAK.

Pour établir ces clés statiques à l'aide de l'enveloppement, la clé doit tout d'abord être générée par le client, puis elle doit être transférée au serveur en appelant la méthode *key_transfer* de l'objet "Security setup" (voir IEC 62056-6-2:2021, 4.4.7). Le paramètre d'appel de méthode doit comprendre le ou les key_id et la ou les clés enveloppées. L'APDU comprenant le service qui appelle la méthode et les paramètres d'appel de méthode doit être protégée comme exigé par la politique de sécurité et les droits d'accès.

NOTE Le niveau exigé de protection peut être stipulé dans les spécifications auxiliaires spécifiques à un projet donné.

Pour établir une clé éphémère à l'aide d'un enveloppement de clé, l'émetteur de l'APDU xDLMS ou des données COSEM génère de manière aléatoire une clé éphémère. Cette clé doit être enveloppée à l'aide de l'algorithme AES d'enveloppement de clé et de la KEK, et doit être envoyée au destinataire avec l'APDU xDLMS ou les données COSEM qui ont été chiffrées à l'aide de la clé éphémère. Le destinataire doit désenvelopper la clé puis doit l'utiliser pour déchiffrer l'APDU xDLMS/les données COSEM reçue(s).

### 5.5.5 Agrément de clé

L'agrément de clé peut être utilisé pour établir des clés statiques entre un serveur et un client, ou bien des clés éphémères entre un serveur et un client ou une tierce partie. Différents mécanismes d'agrément de clé sont disponibles pour établir différentes clés.

Le schéma Ephemeral Unified Model C(2e, 0s, ECC CDH) peut être utilisé par le client et le serveur pour convenir de:

- la clé principale, KEK;
- la clé globale de cryptage de monodiffusion, GUEK;
- la clé globale de cryptage de diffusion, GBEK;
- la clé (globale) d'authentification, GAK.

Ce schéma est pris en charge par la méthode *key_agreement* de la classe d'interface "Security setup" (voir IEC 62056-6-2:2021, 4.4.7). Les paramètres d'appel de méthode transportent les paramètres nécessaires, comme spécifié en 5.3.4.6.2. Les APDU contenant le service qui appelle la méthode ainsi que les paramètres d'appel de méthode doivent être protégés comme exigé par la politique de sécurité et les droits d'accès. Voir aussi Annexe I.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

NOTE Le niveau exigé de protection peut être stipulé dans les spécifications auxiliaires spécifiques à un projet donné.

Pour établir une clé éphémère de cryptage (utilisée comme clé de chiffrement par blocs), deux schémas sont disponibles:

- le schéma One-pass Diffie-Hellman C(1e, 1s, ECC CDH) spécifié en 5.3.4.6.3.

  Sauf indication contraire dans une spécification auxiliaire spécifique au projet donné, le schéma C(1e, 1s ECC CDH) doit être utilisé;
- le schéma de modèle unifié statique C(0e, 2s, ECC CDH) spécifié en 5.3.4.6.4.

### 5.5.6 Périodes cryptographiques à clé symétrique

Il convient de déterminer les périodes cryptographiques à clé symétrique dans des spécifications auxiliaires spécifiques au projet donné. Des recommandations sont données dans le document NIST SP 800-57:2012, Partie 1, 5.3.5, *Symmetric Key Usage Periods and Cryptoperiods*, et 5.3.6, *Cryptoperiod Recommendations for Specific Key Types*.

## 5.6 Clés utilisées avec des algorithmes à clé publique

### 5.6.1 Vue d'ensemble

Les clés asymétriques (voir Tableau 12) sont classées selon:

- leur objectif: clé de signature numérique ou clé d'agrément de clé;
- leur durée de vie: clés statiques ou clés éphémères.

**Tableau 12 – Types de clés asymétriques et leur utilisation**

| Signature numérique | | | |
|---|---|---|---|
| **Type de clé** | **Signataire** | **Vérificateur** | **Utilisation** |
| Paire de clés de signature numérique | Clé privée | Clé publique | Le signataire utilise la clé privée pour calculer la signature numérique:<br>– sur une APDU xDLMS;<br>– sur des données COSEM;<br>– sur une clé d'agrément de clé éphémère publique.<br>Le vérificateur utilise la clé publique pour vérifier la signature numérique<br>– sur une APDU xDLMS;<br>– sur des données COSEM;<br>– sur une clé d'agrément de clé éphémère publique reçue. |
| **Agrément de clé** | | | |
| **Type de clé** | **Partie U** | **Partie V** | **Utilisation** |
| Paire de clés éphémères d'agrément de clé | Clé privée<br>Clé publique | Clé privée<br>Clé publique | En cas de schéma Ephemeral Unified Model C(2e, 0s, ECC CDH), les deux parties ont une paire de clés éphémères. Voir 5.3.4.6.2.<br>En cas de schéma One-pass Diffie-Hellman C(1e,1s, ECC CDH), seule la partie U a une paire de clés éphémères. Voir 5.3.4.6.3. |
| Paire de clés statiques d'agrément de clé | Clé privée<br>Clé publique | Clé privée<br>Clé publique | En cas de schéma One-pass Diffie-Hellman C(1e,1s, ECC CDH), seule la partie V a une paire de clés statiques. Voir 5.3.4.6.3.<br>En cas de schéma de modèle unifié statique, C(0e, 2s, ECC CDH), la partie U et la partie V ont toutes les deux une paire de clés statiques. Voir 5.3.4.6.4. |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.6.2 Génération de paires de clés

Une paire de clés ECC $d$ et $Q$ est générée pour un ensemble de paramètres de domaine $(q, FR, a, b \{, domain_parameter_seed\}, G, n, h)$. Deux méthodes sont fournies pour la génération de la clé privée ECC $d$ et de la clé publique ECC $Q$; l'une de ces deux méthodes doit être utilisée pour générer $d$ et $Q$.

Avant de générer les paires de clés ECDSA, l'assurance de la validité des paramètres de domaine $(q, FR, a, b \{, domain_parameter_seed\}, G, n, h)$ doit avoir été obtenue.

Pour plus de détails, voir FIPS PUB 186-4:2013, Annexe B.4.

### 5.6.3 Certificats de clé publique et infrastructure à clé publique

#### 5.6.3.1 Vue d'ensemble

Le présent paragraphe 5.6.3 décrit les certificats de clé publique pour les besoins de DLMS®/COSEM et donne un exemple d'infrastructure PKI pour la gestion de ces certificats. Il se base sur les documents suivants:

- NIST SP 800-21:2005 et NIST SP 800-32:2001, donnant une description générale de la cryptographie à clé publique et des infrastructures à clé publique;
- UIT-T X.509:2008, spécifiant les cadres des certificats de clé publique et d'attribut;
- RFC 5280, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile;
- NSA3, spécifiant le certificat de base de la Suite B NSA et le profil CRL.

Le modèle de confiance est décrit en 5.6.3.2.

Une architecture PKI (par exemple) est décrite en 5.6.3.3.

Le certificat et le profil d'extension de certificat sont spécifiés en 5.6.4.

Les certificats de clé publique que les serveurs DLMS®/COSEM sont tenus de contenir sont spécifiés en 5.6.5.

La gestion des certificats est spécifiée en 5.6.6.

#### 5.6.3.2 Modèle de confiance

Pour les systèmes DLMS®/COSEM d'échange de données à compteur qui utilisent la cryptographie à clé publique, il convient de mettre en place plusieurs paires de clés publique-privée et certificats de clé publique.

Un certificat de clé publique lie une clé publique à une identité: le sujet. Un certificat est signé numériquement par une autorité de certification.

Pour fournir et gérer les certificats, une certaine forme d'infrastructure à clé publique est exigée. Une PKI est constituée d'autorités de certification qui émettent des certificats et des entités finales à l'aide de ces certificats. Pour voir un exemple de PKI, voir 5.6.3.3.

Dans sa forme la plus simple, une hiérarchie de certification est constituée d'une CA unique. Cependant, la hiérarchie contient généralement plusieurs CA qui ont des relations parent-enfant clairement définies. Il est également possible de déployer plusieurs hiérarchies.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La PKI nécessite une ancre de confiance qui est utilisée pour valider le premier certificat d'une séquence de certificats. L'ancre de confiance peut être un certificat Root-CA, un certificat Sub-CA ou une clé de confiance directe.

NOTE 1 Les ancres de confiance sont des certificats ou des clés de confiance directe marquées comme telles. Cependant, ce marquage ne relève pas du domaine d'application du présent document.

Les serveurs DLMS®/COSEM doivent être équipés d'une ou plusieurs ancres de confiance pendant la fabrication à l'aide d'un processus de confiance hors bande (OOB).

NOTE 2 L'approvisionnement d'ancres de confiances aux clients et tierces parties ne relève pas du domaine d'application du présent document.

Les serveurs DLMS®/COSEM peuvent également être équipés de leurs propres certificats et des certificats des CA, des clients DLMS®/COSEM et des tierces parties. Ceci peut également se produire à l'aide d'un processus OOB de confiance ou par l'intermédiaire d'un objet "Security setup".

La classe d'interface "Security setup" (voir IEC 62056-6-2:2021, 4.4.7) comprend:

- un attribut qui fournit des informations sur les certificats stockés sur le serveur;
- une méthode permettant de générer des paires de clés serveur et une méthode permettant de générer des informations sur la demande de signature de certificat (CSR) sur le serveur que le client est tenu d'envoyer à une CA;
- des méthodes permettant d'importer, d'exporter et de supprimer des certificats.

Ces attributs/méthodes peuvent être utilisés pour faire gérer les certificats par le client ou par une tierce partie par le biais du client agissant comme courtier. Les messages accédant aux attributs et aux méthodes d'objets "Security setup" ainsi que les données incluses doivent être protégés de manière appropriée.

Les certificats ont généralement une période de validité. Cependant, les certificats émis sur des serveurs DLMS®/COSEM peuvent être valides indéfiniment. Les certificats peuvent être remplacés lorsqu'ils expirent.

Avant qu'un serveur utilise un certificat, ce certificat est tenu d'être vérifié. La vérification comprend:

- la vérification de la validité syntaxique du certificat;
- la vérification des attributs inclus dans le certificat;
- la vérification que la période de validité du certificat n'a pas expiré;
- la vérification du chemin de certification jusqu'à l'ancre de confiance;
- la vérification de la signature de l'émetteur du certificat.

Il est pris pour hypothèse que l'ancre de confiance, les autres certificats de la CA, ainsi que les certificats des clients DLMS®/COSEM et des tierces parties contenus par le serveur sont tous valides. Il incombe au système de remplacer/supprimer tout certificat dont la validité a expiré ou ayant été révoqué.

Les clients et les tierces parties sont également tenus de vérifier les certificats du serveur avant de les utiliser. Ils peuvent avoir la capacité de vérifier le statut des certificats qu'ils utilisent. Cependant, cela ne relève pas du domaine d'application de la présente Norme internationale.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.6.3.3 Architecture PKI – informative

#### 5.6.3.3.1 Généralités

NOTE 1 Le texte introductif est tiré du document NIST SP 800-21:2005, 3.7.

Une PKI est une infrastructure de sécurité qui crée et gère des certificats de clé publique pour faciliter l'utilisation de la cryptographie à clé publique (c'est-à-dire clé asymétrique). Pour atteindre cet objectif, une PKI nécessite d'effectuer deux tâches fondamentales:

a) générer et distribuer des certificats de clé publique pour lier des clés publiques à d'autres informations après validation de l'exactitude de la liaison;

b) conserver et distribuer les informations relatives au statut des certificats non expirés.

Pour DLMS®/COSEM, une PKI hiérarchique est prévue, comprenant les composants suivants, comme représenté à la Figure 23.

NOTE 2 La structure réelle de la PKI relève des spécifications auxiliaires spécifiques à un projet donné afin de satisfaire aux besoins des opérateurs.

- Autorité de certification racine (Root-CA): voir 5.6.3.3.2;
- Autorité de certification / Autorité subordonnée (Sub-CA): voir 5.6.3.3.3;
- Entités finales: entités qui ne délivrent pas de certificats: clients DLMS®/COSEM, serveurs DLMS®/COSEM et tierces parties: voir 5.6.3.3.4.

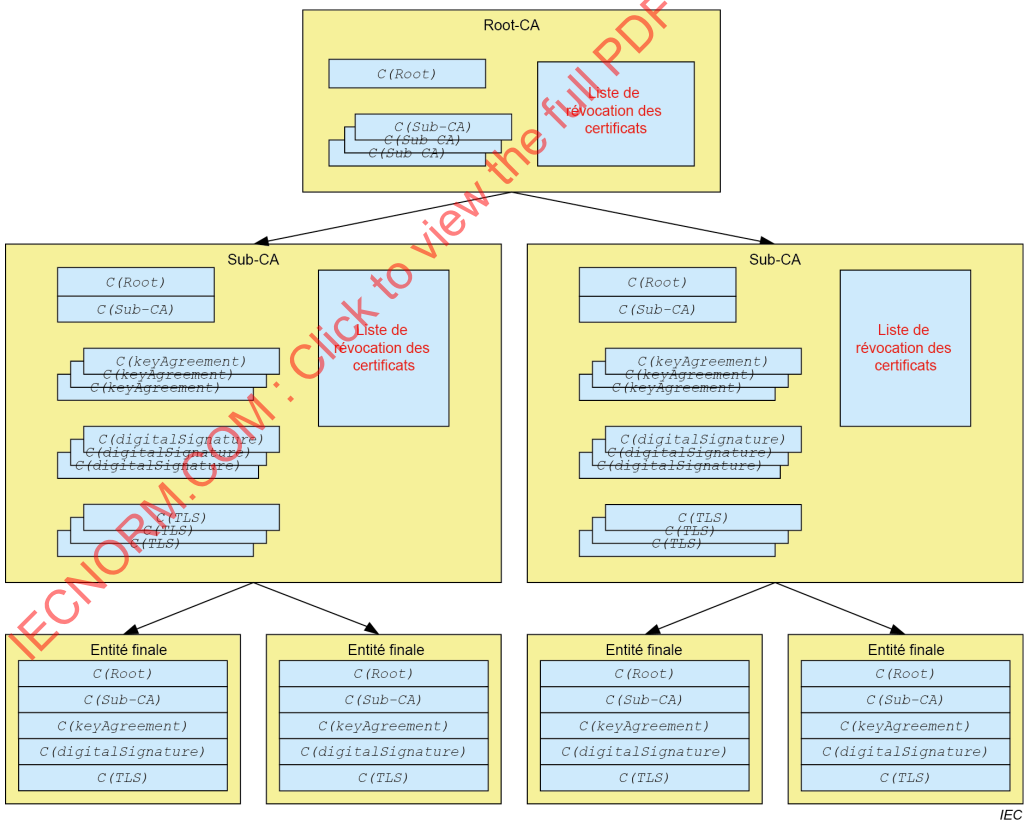



**Figure 23 – Architecture d'une infrastructure à clé publique (exemple)**

#### 5.6.3.3.2 Root-CA

La Root-CA fournit l'ancre de confiance de la PKI. Elle délivre des certificats pour les Sub-CA et conserve une liste de révocation des certificats (CRL). La politique de certification de la Root-CA définit les règles de traitement de l'émission de certificats.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La Root-CA possède le certificat Root (racine) `C(Root)`. Le certificat de la Root-CA est autosigné avec la clé privée de la Root-CA. Les certificats des Sub-CA sont également signés avec la clé privée de la Root-CA.

#### 5.6.3.3.3 Sub-CA

Une Sub-CA est un organisme qui émet des certificats pour les entités finales. La Root-CA autorise chaque Sub-CA à émettre ces certificats.

NOTE Les Sub-CA peuvent être des organismes indépendants, ou peuvent être des acteurs du marché des compteurs, des opérateurs de compteurs, des fabricants.

Chaque Sub-CA doit gérer une politique de certification, qui doit être conforme à la politique de certification de la Root-CA.

Les Sub-CA maintiennent également la liste des certificats émis aux entités finales et une liste de révocation des certificats.

Une Sub-CA possède un certificat *C(Sub-CA)*. Ce certificat est émis par la Root-CA. La clé privée de la Sub-CA est utilisée pour signer des certificats d'entités finales.

#### 5.6.3.3.4 Entités finales

Dans l'infrastructure PKI, une entité finale est un utilisateur de certificats PKI et/ou un système utilisateur final qui est soumis à un certificat. Les certificats suivants d'entités finales sont définis:

- certificat de clé de signature numérique *C(digitalSignature)*, utilisé pour la signature numérique;
- certificat de clé d'agrément de clé statique *C(keyAgreement)*, utilisé pour l'agrément de clé;
- [facultatif] certificat pour la sécurité de la couche transport (TLS, Transport Layer Security) *C(TLS)*, utilisé pour effectuer une authentification entre un client DLMS®/COSEM et un serveur DLMS®/COSEM avant l'établissement d'un canal TLS sécurisé.

Voir aussi 5.6.5.

### 5.6.4 Certificat et profil d'extension de certificat

#### 5.6.4.1 Généralités

Le présent paragraphe 5.6.4 spécifie le certificat et le profil d'extension de certificat exigés pour les systèmes DLMS®/COSEM utilisant la cryptographie à clé publique.

Tous les certificats doivent avoir la structure spécifiée pour les certificats X.509 version 3.

Pour les tableaux présentant les champs du certificat et des extensions de certificats, la notation suivante s'applique:

- o (obligatoire): le champ doit être rempli;
- f (facultatif): le champ est facultatif;
- x (ne pas utiliser): le champ ne doit pas être utilisé.

Chaque extension de certificat est désignée comme étant critique ou non critique. Un système utilisant un certificat doit rejeter le certificat s'il trouve une extension critique qu'il ne reconnaît pas ou une extension critique qui contient des informations qu'il ne peut pas traiter. Une extension non critique peut être ignorée si elle n'est pas reconnue, mais elle doit être traitée si elle est reconnue.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le présent document spécifie les exigences minimales. Les spécifications auxiliaires spécifiques à un projet donné peuvent définir des exigences plus restrictives, par exemple un champ spécifié dans le présent document comme étant facultatif peut être rendu obligatoire ou une extension désignée comme étant non critique peut être désignée comme étant critique.

#### 5.6.4.2 Certificat X.509 v3

Un certificat X.509 v3 est une SEQUENCE de trois champs exigés, comme présenté dans le Tableau 13.

**Tableau 13 – Structure de certificat X.509 v3**

| Champ de certificat | Référence dans le document RFC 5280 | o/x/f | Valeur |
|---|---|---|---|
| `Certificate` | 4.1.1 | | |
| `tbsCertificate` | 4.1.1.1 | o | Voir ci-dessous |
| `signatureAlgorithm` | 4.1.1.2 | o | Voir ci-dessous |
| `signatureValue` | 4.1.1.3 | o | Voir ci-dessous |

Le champ `tbsCertificate` contient les noms de l'objet et de l'émetteur, une clé publique associée à l'objet, une période de validité, et d'autres informations associées. Les champs du `tbsCertificate` sont récapitulés dans le Tableau 14. Le `tbsCertificate` comprend généralement des extensions; ces extensions sont décrites en 5.6.4.4.

Le champ `signatureAlgorithm` contient l'identificateur pour l'algorithme cryptographique utilisé par la CA pour signer ce certificat.

```
AlgorithmIdentifier::= SEQUENCE {
        algorithme          OBJECT IDENTIFIER (IDENTIFICATEUR D'OBJET)
        paramètres          ANY DEFINED BY algorithm OPTIONAL
                            (TOUT PARAMÈTRE DÉFINI PAR l'algorithme
                            FACULTATIF) }
```

Les deux identificateurs d'algorithmes utilisés dans DLMS®/COSEM sont:

- `ecdsa-with-SHA256`, OID 1.2.840.10045.4.3.2 dans le cas de la suite de sécurité 1;
- `ecdsa-with-SHA384`, OID 1.2.840.10045.4.3.3 dans le cas de la suite de sécurité 2.

La `signatureValue` contient une signature numérique calculée d'après le `tbsCertificate` ASN.1 codé en DER. Le tbsCertificate ASN.1 codé en DER est utilisé comme valeur d'entrée de la fonction de signature.

En générant cette signature, une CA certifie la validité des informations dans le champ `tbsCertificate`. En particulier, la CA certifie la liaison entre le matériel de la clé publique et l'objet du certificat.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.6.4.3 tbsCertificate

##### 5.6.4.3.1 Vue d'ensemble

Les champs du `tbsCertificate` sont présentés dans le Tableau 14.

**Tableau 14 – Champs du tbsCertificate X.509 v3**

| Champ de certificat | Référence dans le document RFC 5280 | Article/Paragraphe | o/x/f | Commentaire |
|---|---|---|---|---|
| `tbsCertificate` | 4.1.2 | 5.6.4.2 | | |
| `Version` | 4.1.2.1 | – | o | 'v3' (la valeur est de 2) |
| `Serial Number` | 4.1.2.2 | 5.6.4.3.2 | o | Numéro de série de certificat assigné par la CA (pas plus de 20 octets) |
| `Signature` | 4.1.2.3 | – | o | Même identificateur d'algorithme que celui de `signatureAlgorithm` dans le certificat |
| `Issuer` | 4.1.2.4 | 5.6.4.3.3 | o | Nom distinctif (DN, Distinguished Name) de l'émetteur du certificat. |
| `Validity` | 4.1.2.5 | 5.6.4.3.4 | o | Validité du certificat. |
| `Subject` | 4.1.2.6 | 5.6.4.3.3 | o | Nom distinctif (DN) de l'objet du certificat. |
| `Subject Public Key Info` | 4.1.2.7 | 5.6.4.3.5 | o | |
| `Issuer Unique ID` | 4.1.2.8 | | x | Non applicable |
| `Subject Unique ID` | 4.1.2.8 | 5.6.4.3.6 | f | |
| `Extensions` | 4.1.2.9 | 5.6.4.4 | o | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

##### 5.6.4.3.2 Numéro de série

Tel que spécifié en 4.1.2.2 du document RFC 5280:2008, le numéro de série doit être un entier positif assigné par la CA à chaque certificat. Il doit être unique pour chaque certificat émis par une CA donnée (c'est-à-dire que le nom de l'émetteur et le numéro de série identifient un certificat unique).

Les utilisateurs de certificats doivent être en mesure de gérer les valeurs `serialNumber` jusqu'à 20 octets. Les CA conformes ne doivent pas utiliser des valeurs `serialNumber` supérieures à 20 octets.

##### 5.6.4.3.3 Emetteur et objet

Le champ `Issuer` identifie l'entité qui a signé et émis le certificat.

Le champ `Subject` identifie l'entité associée à la clé publique stockée dans le champ "subject public key". Le nom d'objet PEUT être contenu dans le champ `Subject` et/ou l'extension `subjectAltName`. Si les informations relatives au nommage d'objet sont présentes uniquement dans l'extension `subjectAltName`, alors le nom d'objet doit être une séquence vide et l'extension `subjectAltName` doit être critique. Voir 5.6.4.4.6.

Le schéma de nommage des différentes entités de la PKI est représenté dans les tableaux suivants. Les noms doivent être insérés dans le champ `Issuer` ou `Subject` du `tbsCertificate`, suivant le cas. Voir Tableau 15, Tableau 16 et Tableau 17.

**Tableau 15 – Schéma de nommage pour l'instance de la Root-CA (informatif)**

| Attribut | Abréviation | o/x/f | Nom | Commentaire |
|---|---|---|---|---|
| `Common Name` | CN | o | <Root-CA> | Nom de la Root-CA |
| `Organization` | O | f | <PKI-Name> | Nom de la PKI |
| `Organizational Unit` | OU | f | | Nom de l'unité organisationnelle |
| `Country` | C | f | | Code pays ISO 3166 |
| NOTE Les valeurs comprises entre les signes "inférieur à" et "supérieur à", "< >", sont à assigner par la PKI ou la CA, suivant le cas. | | | | |

**Tableau 16 – Schéma de nommage pour l'instance de la Sub-CA (informatif)**

| Attribut | Abréviation | o/x/f | Nom | Commentaire |
|---|---|---|---|---|
| `Common Name` | CN | o | <XXX-CA> | Nom de la Sub-CA<br>Le CN doit finir par "CA" de sorte que la fonction de la CA soit reconnue. |
| `Organization` | O | f | <PKI-Name> | Nom de la PKI |
| `Organizational Unit` | OU | f | | Nom de l'unité organisationnelle |
| `Country` | C | f | | Code pays ISO 3166 |
| `Locality` | L | f | <Locality> | Emplacement dans lequel la Sub-CA est localisée. |
| `State` | ST | f | <State> | |
| NOTE Les valeurs comprises entre les signes "inférieur à" et "supérieur à", "< >", sont à assigner par la PKI ou la CA, suivant le cas. | | | | |

Le format des éléments du schéma de nommage des instances de la Root-CA et de la Sub-CA dépend des spécifications auxiliaires spécifiques à un projet donné.

**Tableau 17 – Schéma de nommage pour l'instance de l'entité finale**

| Attribut | Abréviation | o/x/f | Nom | Commentaire |
|---|---|---|---|---|
| Common Name | CN | o | <System-title> | Titre système DLMS®/COSEM:<br>8 octets représentés par<br>16 caractères:<br>Exemple: "4D4D4D0000BC614E" |
| Organization | O | f | <PKI-Name> | Nom de la PKI |
| Organizational Unit | OU | f | | Nom de l'unité organisationnelle |
| Country | C | f | | Code pays ISO 3166 |
| Locality | L | x | <Locality> | Emplacement dans lequel l'entité est localisée. |
| State | S | x | <State> | |
| NOTE Les valeurs comprises entre les signes "inférieur à" et "supérieur à", "< >", sont à assigner par la PKI ou la CA, suivant le cas. | | | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.6.4.3.4 Période de validité

La période de validité d'un certificat est l'intervalle de temps au cours duquel la CA assure qu'elle conserve des informations concernant le statut du certificat. Le champ est représenté comme une SEQUENCE de deux dates:

- la date à laquelle la période de validité de certificat commence (`notBefore`);
- la date à laquelle la période de validité de certificat se termine (`notAfter`).

En cas de certificats de la CA, certificats des Sub-CA et entités finales autres que les serveurs DLMS®/COSEM, `notBefore` et `notAfter` doivent être des dates bien définies.

Les serveurs DLMS®/COSEM peuvent se voir octroyer des certificats pour lesquels aucune bonne date d'expiration ne peut être assignée; un tel certificat est destiné à être utilisé pendant la totalité de la durée de vie du dispositif.

Pour indiquer qu'un certificat n'a pas de date d'expiration bien définie, il convient d'assigner `notAfter` à la valeur GeneralizedTime de 99991231235959 Z.

Pour plus de détails, voir RFC 5280:2008, 4.1.2.5.

#### 5.6.4.3.5 SubjectPublicKeyInfo

Le champ `SubjectPublicKeyInfo` doit avoir la structure suivante:

```
SubjectPublicKeyInfo::= SEQUENCE {
      Algorithm                                AlgorithmIdentifier,
      subjectPublicKey                  BIT STRING }
```

Un identificateur d'algorithme est défini par la structure ASN.1 suivante:

```
AlgorithmIdentifier::= SEQUENCE {
      algorithme               OBJECT IDENTIFIER (IDENTIFICATEUR D'OBJET)
      paramètres               ANY DEFINED BY algorithm OPTIONAL (TOUT PARAMÈTRE
DÉFINI PAR l'algorithme FACULTATIF) }
```

L'identificateur d'algorithme est utilisé pour identifier un algorithme cryptographique. Le composant OBJECT IDENTIFIER identifie l'algorithme (tel que DSA avec SHA-1). Le contenu du champ "optional parameters" (paramètres facultatifs) varie selon l'algorithme identifié. Ce champ doit contenir le même identificateur d'algorithme que le champ "signature" dans la séquence `tbsCertificate` (voir 5.6.4.2).

L'algorithme `OBJECT IDENTIFIER` doit contenir les valeurs suivantes:

- valeur OID: 1.2.840.10045.2.1,
- description OID: ECDSA et clé publique ECDH.

La valeur de `parameter` doit être de 1.2.840.10045.3.1.7 pour la courbe NIST P-256 et de 1.3.132.0.34 pour la courbe NIST P-384.

La subjectPublicKey de SubjectPublicKeyInfo est la clé publique ECC. Les clés publiques ECC ont la syntaxe suivante:

ECPoint::= OCTET STRING (chaîne d'octets)

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Les implémentations de la cryptographie à courbe elliptique conformément au présent document doivent prendre en charge la forme non compressée et peuvent prendre en charge la forme compressée de la clé publique ECC. La forme hybride de la clé publique ECC de [X9.62] ne doit pas être utilisée.

Comme spécifié dans le document SEC1:2009:

- la clé publique à courbe elliptique (une valeur de type ECPoint qui est une OCTET STRING) est mappée sur subjectPublicKey (une valeur de type BIT STRING) comme suit: le bit de poids fort de la valeur OCTET STRING devient le bit de poids fort de la valeur BIT STRING, etc.; le bit de poids faible de l'OCTET STRING devient le bit de poids faible de BIT STRING. Les routines de conversion peuvent être consultées en 2.3.1 et 2.3.2 du document SEC1:2009;
- le premier octet de l'OCTET STRING indique si la clé est compressée ou non compressée. La forme non compressée est indiquée par 0x04 et la forme compressée est indiquée par 0x02 ou 0x03 (voir 2.3.3 de SEC1:2009). La clé publique doit être rejetée si le premier octet comprend une quelconque autre valeur.

#### 5.6.4.3.6 ID d'objet unique

Les ID d'objet unique peuvent être utilisés de manière facultative dans les certificats de dispositifs finaux autres que les certificats de serveur. L'utilisation de ce champ dépend des spécifications auxiliaires spécifiques à un projet donné.

### 5.6.4.4 Extensions de certificat

#### 5.6.4.4.1 Vue d'ensemble

Les extensions définies pour les certificats X.509 v3 donnent des méthodes permettant d'associer des attributs supplémentaires avec des utilisateurs ou des clés publiques et permettant de gérer les relations entre les CA. Chaque extension dans un certificat est désignée comme étant critique (TRUE) ou non critique (FALSE).

Les champs d'extension sont à renseigner conformément au type de certificat utilisé, comme spécifié dans le Tableau 18.

**Tableau 18 – Extensions de certificat X.509 v3**

| | Attributs | Référence dans le document RFC 5280:2008 | Article/Paragraphe | CA | | Entités finales | | |
|---|---|---|---|---|---|---|---|---|
| | | | | C(Root) | C(Sub-CA) | C(TLS) | C(Key Agree) | C(Data Sign) |
| 1 | AuthorityKeyIdentifier | 4.2.1.1 | 5.6.4.4.2 | f | o | o | o | o |
| 2 | SubjectKeyIdentifier | 4.2.1.2 | 5.6.4.4.3 | o | o | f | f | f |
| 3 | KeyUsage | 4.2.1.3 | 5.6.4.4.4 | o | o | o | o | o |
| 4 | CertificatePolicies | 4.2.1.4 | 5.6.4.4.5 | f | o | o | f | f |
| 5 | SubjectAltNames | 4.2.1.6 | 5.6.4.4.6 | f | f | f | f | f |
| 6 | IssuerAltNames | 4.2.1.7 | 5.6.4.4.7 | f | f | x | x | x |
| 7 | BasicConstraints | 4.2.1.9 | 5.6.4.4.8 | o | o | x | x | x |
| 8 | ExtendedKeyUsage | 4.2.1.12 | 5.6.4.4.9 | x | x | o | x | x |
| 9 | cRLDistributionPoints | 4.2.1.13 | 5.6.4.4.10 | f | f | x | x | x |

C(Root): Certificat de la Root-CA

C(Sub-CA): Certificat d'une Sub-CA

C(TLS): Certificat pour la sécurité de la couche transport (TLS)

C(KeyAgree): Certificat d'une clé d'établissement de clé prenant en charge ECDH

C(DataSign): Certificat d'une clé de signature prenant en charge ECDSA

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.6.4.4.2 Identificateur de clé d'autorité

- ID d'extension (OID): 2.5.29.35.
- Critique: FALSE.
- Description: l'extension AuthorityKeyIdentifier fournit un moyen permettant d'identifier la clé publique correspondant à la clé privée utilisée pour signer un certificat.
- Valeur: l'extension `AuthorityKeyIdentifier` doit comprendre le champ `keyIdentifier`.

La valeur du champ `keyIdentifier` nécessite d'être calculée:

- avec la méthode 1 définie en 4.2.1.2 du document RFC 5280:2008, c'est-à-dire que le `keyIdentifier` est constitué d'un hachage SHA-1 160 bits de la valeur de la BIT STRING `SubjectPublicKey` (excluant la balise, la longueur et le nombre de bits non utilisés);
- avec la méthode 2 définie en 4.2.1.2 du document RFC 5280:2008, c'est-à-dire que le keyIdentifier est constitué d'un champ de type à 4 bits avec la valeur 0100 suivie des 60 bits de poids faible du hachage SHA-1 de la valeur de la BIT STRING `subjectPublicKey` (excluant la balise, la longueur et le nombre de bits non utilisés).

NOTE Le choix de la méthode dépend des spécifications auxiliaires spécifiques à un projet donné.

#### 5.6.4.4.3 SubjectKeyIdentifier

- ID d'extension (OID): 2.5.29.14.
- Critique: FALSE.
- Description: l'extension `SubjectKeyIdentifier` fournit un moyen d'identifier des certificats qui contiennent une clé publique particulière.
- Valeur: l'extension `SubjectKeyIdentifier` doit comprendre le champ `keyIdentifier`.

Pour la méthode de calcul du `keyIdentifier`, voir 5.6.4.4.2.

#### 5.6.4.4.4 KeyUsage

- ID d'extension (OID): 2.5.29.15.
- Critique: TRUE.
- Description: l'extension KeyUsage définit l'objectif de la clé contenue dans le certificat.
- Valeur: Les bits qui doivent être définis sont présentés dans le Tableau 19.

**Tableau 19 – Extensions KeyUsage**

| Certificat | C(Root) | C(Sub-CA) | C(TLS) | C(KeyAgree) | C(DataSign) |
|---|---|---|---|---|---|
| Bits à définir | keyCertSign, cRLSign | keyCertSign, cRLSign | digitalSignature keyAgreement | keyAgreement | digitalSignature |

Pour plus de détails, voir RFC 5280:2008, 4.2.1.3, et NSA3.

#### 5.6.4.4.5 CertificatePolicies

- ID d'extension (OID): 2.5.29.32.
- Critique: FALSE.
- Description: L'extension de politiques de certification (CertificatePolicies) contient une séquence d'un ou plusieurs termes d'informations relatifs à la politique, dont chacun est constitué d'un identificateur d'objet (OID) et d'un qualificateur facultatif. Pour plus de détails, voir RFC 5280, 4.2.1.4;
- Valeur: contient l'OID pour la politique de certification applicable.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.6.4.4.6 SubjectAltNames

- ID d'extension (OID): 2.5.29.17.
- Critique: TRUE si le champ "subject" du certificat est vide (séquence vide), sinon FALSE.

Description: cette extension permet de lier des identités à l'objet du certificat. Ces identités peuvent être comprises en plus de ou à la place de l'identité du champ "subject" du certificat. Si le nom d'objet est une séquence vide, l'extension `subjectAltName` doit être ajoutée dans la signature de l'entité finale (End Entity Signature) et dans les certificats d'établissement de clé (Key Establishment Certificates) et doit être marquée comme étant critique. Si ce n'est pas le cas, l'extension `subjectAltName` est FACULTATIVE et si elle est incluse, elle doit être marquée comme étant critique.

L'extension `SubjectAltName`, lorsqu'elle est utilisée, doit contenir un GeneralName unique de type OtherName qui fait partie du sous-type HardwareModuleName (id-on-HardwareModuleName), tel que défini dans le document RFC 4108. Le champ hwSerialNum doit être défini sur le titre système.

- Valeur: Voir Tableau 20.

**Tableau 20 – Valeurs Subject Alternative Name (nom alternatif d'objet)**

| Certificat | C(Root) | C(Sub-CA) | C(TLS) | C(KeyAgree) | C(DataSign) |
|---|---|---|---|---|---|
| rfc822Name | <E-Mail Address> | <E-Mail Address> | - | – | – |
| uRI | <Web site> | <Web site> | - | – | – |
| otherName | - | - | - | <otherName> | <otherName> |
| NOTE Les valeurs comprises entre les signes "inférieur à" et "supérieur à", "< >", sont à assigner par la PKI ou la CA, suivant le cas. | | | | | |

#### 5.6.4.4.7 IssuerAltName

- ID d'extension (OID): 2.5.29.18.
- Critique: FALSE.
- Description: cette extension est utilisée pour associer les identités de style Internet avec l'émetteur du certificat. Pour plus de détails, voir RFC 5280:2008, 4.2.1.7;
- Valeur: Voir Tableau 21.

**Tableau 21 – Valeurs Issuer Alternative Name (nom alternatif de l'émetteur)**

| Certificat | C(Root) | C(Sub-CA) | C(TLS) | C(KeyAgree) | C(dataSign) |
|---|---|---|---|---|---|
| rfc822Name | <E-Mail Address> | <E-Mail Address> | – | – | – |
| URI | <Web site> | <Web site> | – | – | – |
| NOTE Les valeurs comprises entre les signes "inférieur à" et "supérieur à", "< >", sont à assigner par la PKI ou la CA, suivant le cas. | | | | | |

#### 5.6.4.4.8 Basic Constraints

- ID d'extension (OID): 2.5.29.19.
- Critique: TRUE.
- Description: L'extension Basic Constraints (contraintes de base) identifie si l'objet du certificat est une CA et la profondeur maximale des chemins valides de certification qui comprennent ce certificat. Voir RFC 5280:2008, 4.2.1.9;
- Valeur: Voir Tableau 22.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 22 – Valeurs de l'extension Basic Constraints**

| Certificat | C(Root) | C(Sub-CA) | C(TLS) | C(KeyAgree) | C(dataSign) |
|---|---|---|---|---|---|
| cA | TRUE | TRUE | – | – | – |
| pathLenConstraint | Voir Note 1 | Voir Note 1 | – | – | – |
| NOTE La valeur de pathLenConstraint (facultatif) dépend de la structure de la PKI. | | | | | |

#### 5.6.4.4.9 Extended Key Usage

- ID d'extension (OID): 2.5.29.37.
- Critique: FALSE.
- Description: Indique qu'un certificat peut être utilisé comme certificat du serveur TLS;
  - OID d'authentification du serveur TLS: 1.3.6.1.5.5.7.3.1;
  - OID d'authentification du client TLS: 1.3.6.1.5.5.7.3.2.

#### 5.6.4.4.10 cRLDistributionPoints

- ID d'extension (OID): 2.5.29.31.
- Critique: FALSE.
- Description: L'extension cRLDistributionPoints (points de distribution CRL) identifie la manière dont les informations CRL sont obtenues.
- Cette extension n'est pas utilisée dans les certificats de serveurs DLMS®/COSEM.

#### 5.6.4.4.11 Autres extensions

Il convient de considérer toutes les autres extensions non décrites dans ce profil comme FACULTATIVES; leur inclusion ou exclusion et leurs valeurs dépendent de l'application particulière ou du profil de la PKI.

### 5.6.5 Types de certificats d'entités finales de la Suite B à prendre en charge par les serveurs DLMS®/COSEM

Chaque serveur DLMS®/COSEM doit utiliser le format X.509 v3 et contenir:

- une clé de signature P-256 ou P-384 prenant en charge l'ECDSA;
- une clé d'agrément de clé P-256 ou P-384 prenant en charge ECDH.

Chaque certificat doit être signé à l'aide de l'ECDSA. La clé de la CA signataire doit être P-256 ou P-384 si le certificat contient une clé sur P-256. La clé de la CA signataire doit être P-384 si le certificat contient une clé sur P-384.

Selon la politique de sécurité, les certificats X.509 v3 doivent être traités par des entités finales DLMS®/COSEM (voir Tableau 23).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 23 – Certificats traités par des entités finales DLMS®/COSEM**

| Suite de sécurité 1 | Suite de sécurité 2 | Rôle |
|---|---|---|
| Certificat autosigné d'autorité racine de certification (Root-CA) utilisant P-256 signé avec P-256 | Certificat autosigné d'autorité racine de certification (Root-CA) utilisant P-384 signé avec P-384 | Ancre de confiance; il peut y en avoir plusieurs. |
| Certificat d'autorité subordonnée de certification (Sub-CA) utilisant P-256 signé avec P-256 | Certificat d'autorité subordonnée de certification (Sub-CA) utilisant P-384 signé avec P-384 | Certificat d'une CA émettrice.<br>Les CA subordonnées peuvent également être utilisées comme ancres de confiance. |
| – | Certificat d'autorité subordonnée de certification (Sub-CA) utilisant P-256 signé avec P-384 | |
| Certificat de signature d'entité finale utilisant P-256 signé avec P-256 | Certificat de signature d'entité finale utilisant P-384 signé avec P-384 | Clé publique pour la génération et la vérification de signature ECDSA |
| – | Certificat de signature d'entité finale utilisant P-256 signé avec P-384 | |
| Certificat d'établissement de clé d'entité finale utilisant P-256 signé avec P-256 | Certificat d'établissement de clé d'entité finale utilisant P-384 signé avec P-384 | Utilisé avec le schéma One-pass Diffie-Hellman C(1e, 1s) ou avec le schéma de modèle unifié statique C(0e, 2s, ECC CDH) |
| – | Certificat d'établissement de clé d'entité finale utilisant P-256 signé avec P-384 | |
| Certificat TLS d'entité finale utilisant P-256 signé avec P-256 | Certificat TLS d'entité finale utilisant P-384 signé avec P-384 | |
| – | Certificat TLS d'entité finale utilisant P-256 signé avec P-384 | |

Un exemple de certificat est donné à l'Annexe H.

### 5.6.6 Gestion des certificats

#### 5.6.6.1 Vue d'ensemble

Le présent paragraphe 5.6.6 s'applique uniquement à la gestion des certificats de clé publique dans les serveurs DLMS®/COSEM, y compris:

- l'approvisionnement du serveur en ancres de confiance (voir 5.6.6.2);
- l'approvisionnement du serveur en certificats supplémentaires de la CA (voir 5.6.6.3);
- la personnalisation de sécurité du serveur (voir 5.6.6.4);
- l'approvisionnement du serveur en certificats des clients et des tierces parties (voir 5.6.6.5);
- l'approvisionnement des clients et des tierces parties en certificats de serveurs (voir 5.6.6.6);
- la suppression de certificats (voir 5.6.6.7).

NOTE La gestion des certificats de clé publique dans les systèmes DLMS®/COSEM de clients et de tierces parties ne relève pas du domaine d'application du présent document.

#### 5.6.6.2 Approvisionnement des serveurs en ancres de confiance

Avant le début des opérations en schéma établi, les serveurs doivent être approvisionnés en ancres de confiance utilisées pour valider les certificats. Les ancres de confiance peuvent être des certificats de la Root-CA (c'est-à-dire autosignés), des certificats d'une Sub-CA ou des clés de CA de confiance directe. Un serveur peut être approvisionné avec une ou plusieurs ancres de confiance.

Les ancres de confiance doivent être placées dans le serveur hors bande (OOB).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Les certificats d'ancres de confiance sont stockés avec d'autres certificats.

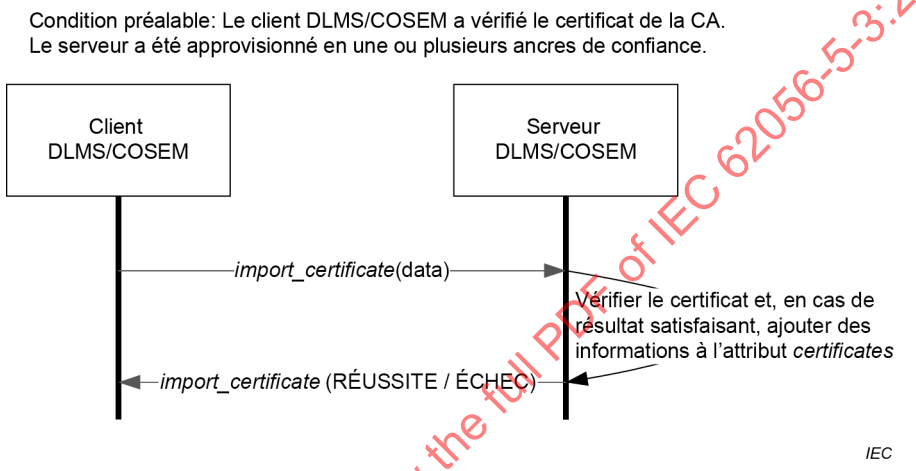
Ils peuvent être exportés, mais ils ne peuvent pas être importés ou supprimés.

Les clés de CA de confiance directe ne peuvent pas être exportées.

#### 5.6.6.3 Approvisionnement du serveur en certificats supplémentaires de la CA

Le serveur peut être approvisionné à l'aide de certificats supplémentaires de la CA utilisés pour vérifier les signatures numériques sur les certificats de dispositifs finaux.

À cet effet, la méthode *import_certificate* de l'objet "Security setup" est disponible. Le processus est représenté à la Figure 24.




NOTE Lorsqu'une tierce partie est responsable de la gestion des certificats de la CA, la méthode *import_certificate* peut être appelée par cette tierce partie par l'intermédiaire du client agissant comme courtier.

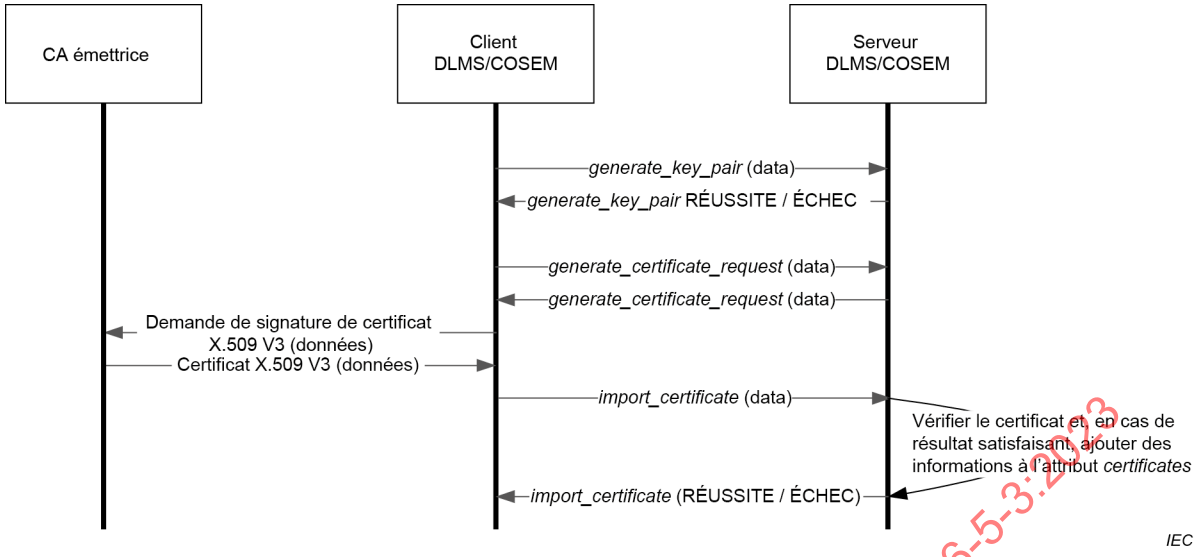
**Figure 24 – MSC pour l'approvisionnement du serveur en certificats de la CA**

#### 5.6.6.4 Personnalisation de sécurité du serveur

La personnalisation de sécurité signifie l'approvisionnement du serveur en paires de clés asymétriques avec les certificats de clé publique correspondants. Ceci peut se produire:

- à l'aide des primitives de sécurité fournies par le fabricant pour injecter la clé privée et les certificats de clé publique. Les clés privées sont à stocker de manière sécurisée dans le serveur et ne doivent jamais être exposées;
- à l'aide des méthodes appropriées d'un objet "Security setup". Ce processus peut être utilisé pendant le processus de fabrication et lorsqu'une nouvelle paire de clés et le certificat de clé publique lié nécessitent d'être générés. Ce processus est représenté à la Figure 25 et décrit ci-dessous.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




NOTE La personnalisation de sécurité peut être effectuée par une tierce partie, en lieu et place du client. Dans ce cas, les méthodes de l'objet "Security setup" peuvent être appelées par cette tierce partie par le biais du client agissant comme courtier.

**Figure 25 – MSC pour la personnalisation de sécurité du serveur**

- Étape 1: le client appelle la méthode *generate_key_pair*. Les paramètres d'appel de méthode spécifient la paire de clés à générer: signature numérique, agrément de clé ou paire de clés TLS.

  NOTE 1 La nouvelle paire de clés peut être utilisée dans les transactions une fois que son certificat a été importé et vérifié avec succès.

- Étape 2: le client appelle la méthode *generate_certificate_request*. Les paramètres d'appel de méthode identifient la paire de clés pour laquelle la demande de signature de certificat (CSR, Certificate Signing Request) est générée. Les paramètres de retour comprennent la CSR du certificat X.509 v3, signée par la clé privée de la paire de clés nouvellement générée.
- Étape 3: le client envoie la CSR du certificat X.509 v3 à la CA. Ce message doit encapsuler les paramètres de retour résultant de l'appel de la méthode *generate_certificate_request*. La CA (dans la mesure où les conditions nécessaires sont remplies) émet le certificat et l'envoie au client.

  NOTE 2 Le format des messages entre le client et la CA émettrice ne relève pas du domaine d'application du présent document.

- Étape 4: le client appelle la méthode *import_certificate*. Les paramètres d'appel de méthode contiennent le certificat. Le serveur vérifie le certificat et, en cas de résultat satisfaisant, il ajoute des informations sur le certificat à l'attribut *certificates*. Si la vérification échoue, le certificat doit être supprimé.

Il ne peut y avoir qu'une seule paire de clés et un seul certificat présents pour le même objectif (signature numérique, agrément de clé, TLS). Par conséquent, lorsque le nouveau certificat est importé avec succès, l'ancien certificat est supprimé. À partir de là, la nouvelle paire de clés peut être utilisée pour les transactions.

Pour avoir des détails concernant l'appel de méthode et les paramètres de retour, voir IEC 62056-6-2:2021, 4.4.7.

Les parties utilisant les certificats du serveur peuvent obtenir ces certificats:

- hors bande;
- à l'aide de la méthode *export_certificate* des objets "Security setup" (voir 5.6.6.6);
- dans le cadre de l'AARE (pendant l'authentification HLS) (voir 5.7.4).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

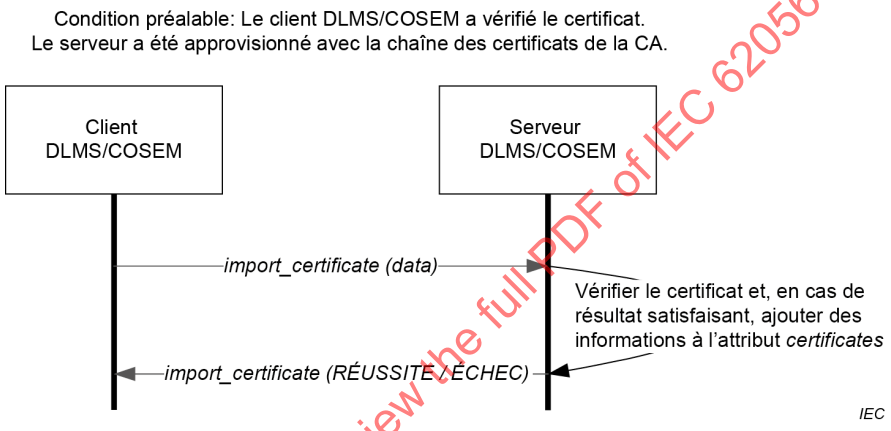
#### 5.6.6.5 Approvisionnement de serveurs en certificats de clients et de tierces parties

Pour vérifier les signatures numériques, pour effectuer l'agrément de clé en utilisant un schéma qui utilise des clés statiques d'agrément de clé, ou pour établir une connexion TLS, le serveur nécessite d'avoir les certificats appropriés de clé publique de l'autre partie.

Si, au moment de la fabrication, le client et/ou les tierces parties sont déjà connus, leurs certificats de clé publique peuvent être injectés dans le serveur par le fabricant.

NOTE La distribution de certificats de clé publique au fabricant ne relève pas du domaine d'application du présent document.

Sinon, les serveurs peuvent être approvisionnés en certificats de clients et tierces parties à l'aide de la méthode *import_certificate* de l'objet "Security setup". Le paramètre d'appel de méthode est le certificat à placer dans le serveur. Pour avoir des détails concernant l'appel de méthode et les paramètres de retour, voir IEC 62056-6-2:2021, 4.4.7. Le processus est représenté à la Figure 26.




NOTE La méthode *import_certificate* (données) peut être appelée par une tierce partie, en utilisant le client comme courtier.

**Figure 26 – Approvisionnement du serveur en certificat du client**

En cas de mécanisme d'authentification HLS utilisant ECDSA, le certificat de clé publique de la clé de signature numérique des clients peut être contenu par le champ calling-AE-qualifier de l'AARQ. Voir 5.7.4.

#### 5.6.6.6 Approvisionnement des clients et des tierces parties en certificats de serveurs

Pour vérifier les signatures numériques appliquées par le serveur, pour effectuer l'agrément de clé qui implique une clé statique d'agrément de clé, ou pour établir une connexion TLS, le client ou la tierce partie nécessite d'avoir le certificat approprié de clé publique du serveur.

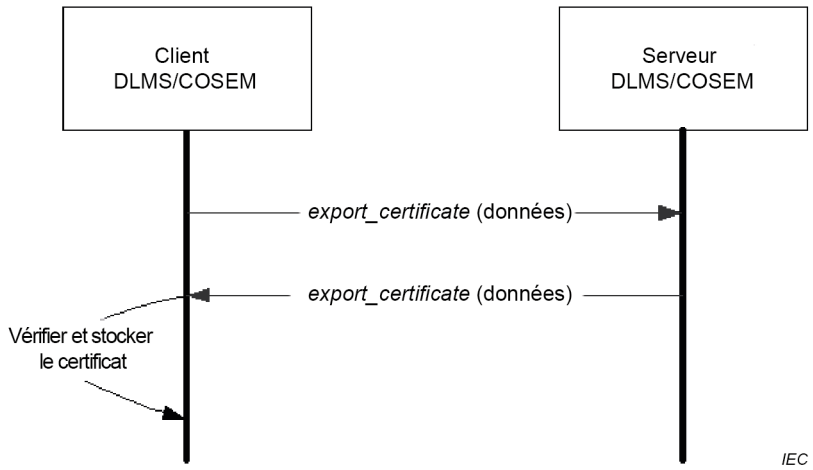
Le certificat peut être fourni avec le serveur et inséré dans les clients/tierces parties OOB.

En variante, le client ou la tierce partie peut demander le certificat à partir du serveur en utilisant la méthode *export_certificate* de l'objet "Security setup". Le paramètre d'appel de méthode identifie le certificat demandé.

NOTE En cas d'authentification HLS utilisant ECDSA (il s'agit de authentication_mechanism 7), le certificat de clé publique du serveur pour la signature numérique est transporté dans l'AARE.

Les paramètres de retour (en cas de réussite) comprennent le certificat. Pour avoir des détails concernant l'appel de méthode et les paramètres de retour, voir IEC 62056-6-2:2021, 4.4.7. Le processus est représenté à la Figure 27.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




NOTE La méthode *export_certificate* (données) peut également être appelée par une tierce partie, en utilisant le client comme courtier.

**Figure 27 – Approvisionnement du client/de la tierce partie en certificat du serveur**

### 5.6.6.7 Suppression de certificat du serveur

Il est parfois nécessaire de supprimer un certificat de clé publique stocké par le serveur.

NOTE 1 Ceci peut être lié aux certificats qui appartiennent au serveur ou aux certificats qui appartiennent à un client ou une tierce partie.

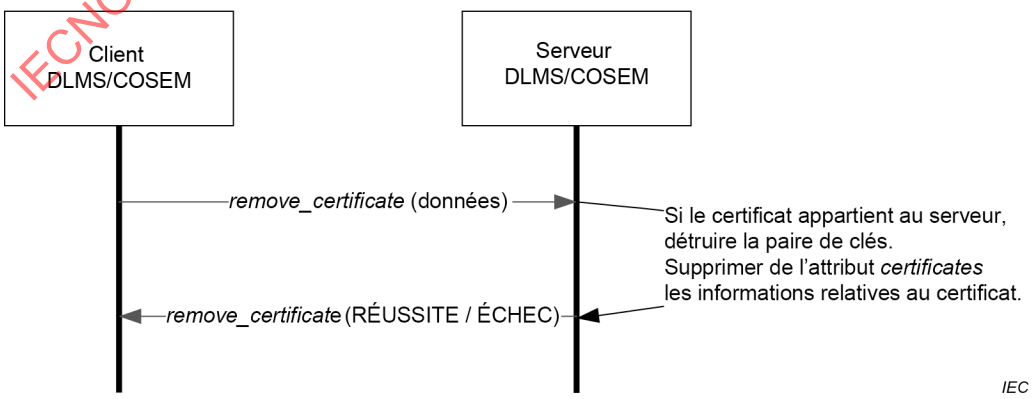
NOTE 2 Les conditions dans lesquelles la suppression d'un certificat de clé publique est nécessaire ne relèvent pas du domaine d'application du présent document.

Lorsqu'un certificat qui appartient au serveur est supprimé, la clé privée associée à la clé publique doit être détruite.

Les informations relatives au certificat supprimé doivent également être supprimées de l'attribut *certificates* de l'objet "Security setup".

La paire de clés dont le certificat de clé publique a été supprimé ne peut plus être utilisée pour des transactions.

Le processus est représenté à la Figure 28.




NOTE La méthode *remove_certificate* (données) peut également être appelée par une tierce partie, en utilisant le client comme courtier.

**Figure 28 – Suppression de certificat du serveur**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 5.7 Application de la protection cryptographique

### 5.7.1 Vue d'ensemble

Les algorithmes cryptographiques spécifiés en 5.3 peuvent être appliqués:

- pour protéger les APDU xDLMS (voir 5.7.2);
- pour traiter les épreuves pendant l'authentification HLS (voir 5.7.4);
- pour protéger les données COSEM (voir 0).

### 5.7.2 Protection des APDU xDLMS

#### 5.7.2.1 Vue d'ensemble

Le présent paragraphe 5.7.2 spécifie la manière dont les algorithmes cryptographiques spécifiés en 5.3.3 et 5.3.4 peuvent être utilisés pour protéger les APDU xDLMS:

- 5.7.2.2 spécifie les valeurs possibles de la politique de sécurité et des droits d'accès;
- 5.7.2.3 présente les types d'APDU chiffrées;
- 5.7.2.4 spécifie l'utilisation de l'algorithme AES-GCM pour l'authentification et le cryptage;
- 5.7.2.5 spécifie l'utilisation de l'algorithme ECDSA pour la signature numérique.

Lorsqu'une primitive de service xDLMS relative un attribut ou une méthode d'objet COSEM est appelée par l'AP, les paramètres de service comprennent le paramètre Security_Options. Ce paramètre informe l'AL sur le type d'APDU chiffrée à utiliser, sur la protection à appliquer et comprend le matériau de sécurité nécessaire. L'AL construit d'abord l'APDU correspondant à la primitive de service et ensuite construit l'APDU chiffrée.

Lorsque l'AL reçoit une APDU chiffrée de la part d'un partenaire distant, elle la déchiffre et restaure l'APDU d'origine, non chiffrée. Lorsque ceci est réalisé avec succès, elle appelle la primitive de service appropriée. Les paramètres de service supplémentaires incluent les paramètres Security_Status et Protection_Element, qui informent l'AP sur le type d'APDU chiffrée utilisée, sur la protection qui a été vérifiée et supprimée, et peut inclure le matériau de sécurité. Voir aussi 6.5.

#### 5.7.2.2 Valeurs de la politique de sécurité et des droits d'accès

En cas de version 1 de "Security setup" (voir IEC 62056-6-2:2021, 4.4.7), le type enum doit être interprété comme un type 8 non signé (unsigned8). La signification de chaque bit est telle que présentée dans le Tableau 24.

**Tableau 24 – Valeurs de la politique de sécurité ("Security setup" version 1)**

| Bit | Politique de sécurité |
|---|---|
| 0 | non utilisée, doit être définie sur 0 |
| 1 | non utilisée, doit être définie sur 0 |
| 2 | demande authentifiée |
| 3 | demande cryptée |
| 4 | demande signée numériquement |
| 5 | réponse authentifiée |
| 6 | réponse cryptée |
| 7 | réponse signée numériquement |
| NOTE En cas de version 0 de "Security policy", les valeurs possibles de la politique de sécurité sont spécifiées en 5.3.8 de l'IEC 62056-6-2:2021. Pour la version 0 et la version 1 de "Security policy", la valeur (0) signifie qu'aucune protection cryptographique n'est exigée. | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Les droits d'accès sont contenus par l'attribut *object_list* de "Association LN" ou *access_rights_list* des objets "Association SN". L'élément access_mode des access_rights détermine le type d'accès et spécifie la protection cryptographique. Il s'agit d'un type de données *enum*.

En cas de version 3 de "Association LN" et de version 4 de "Association SN" (voir IEC 62056-6-2:2021, 4.4.4 et 4.4.3), la valeur *enum* est interprétée comme unsigned8: La signification de chaque bit est telle que présentée dans le Tableau 25.

Pour les versions antérieures, voir leur spécification dans l'IEC 62056-6-2:2021.

**Tableau 25 – Valeurs des droits d'accès ("Association LN" ver 3, "Association SN" ver 4)**

| Bit | Accès à l'attribut | Accès à la méthode |
|---|---|---|
| 0 | accès en lecture | accès |
| 1 | accès en écriture | non utilisé |
| 2 | demande authentifiée | demande authentifiée |
| 3 | demande cryptée | demande cryptée |
| 4 | demande signée numériquement | demande signée numériquement |
| 5 | réponse authentifiée | réponse authentifiée |
| 6 | réponse cryptée | réponse cryptée |
| 7 | réponse signée numériquement | réponse signée numériquement |
| Exemples | enum (3): lecture-écriture<br>enum (6): accès en écriture avec demande authentifiée<br>enum (255): accès en lecture-écriture avec demande et réponse authentifiées, cryptées et signées numériquement | enum (1): accès<br>enum (2): non utilisé<br>enum (5): accès avec demande authentifiée<br>enum (253): accès avec demande et réponse authentifiées, cryptées et signées numériquement |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Les droits d'accès aux attributs et/ou méthodes d'objet COSEM peuvent exiger des APDU xDLMS authentifiées, cryptées et/ou signées. Par conséquent, les APDU ayant une protection supérieure à celle exigée par la politique de sécurité sont toujours autorisées. Les APDU avec une protection moindre par rapport à celle exigée par la politique de sécurité et les droits d'accès doivent être rejetées.

Dans ce contexte, une protection supérieure signifie qu'un nombre plus important de types de protections est appliqué à l'APDU xDLMS par rapport à ce qui est demandé par la politique de sécurité: par exemple, la politique de sécurité exige que toutes les APDU soient authentifiées, mais les droits d'accès exigent que l'APDU soit cryptée et authentifiée, c'est-à-dire qu'elle ait une protection plus élevée.

#### 5.7.2.3 APDU xDLMS chiffrées

Les différents types d'APDU xDLMS chiffrées sont présentés dans le Tableau 26. Voir aussi 6.5. Les APDU xDLMS chiffrées peuvent être utilisées uniquement dans un contexte chiffré d'application. D'autre part, dans un contexte chiffré d'application, des APDU chiffrées et non chiffrées peuvent être utilisées.

**Tableau 26 – APDU xDLMS chiffrées**

| Type d'APDU | Parties | Type de chif-frement | Services de sécurité | Clé utilisée | Compres-sion |
|---|---|---|---|---|---|
| glo-ciphering ou ded-ciphering spécifique au service | Client – Serveur | Clé symétrique | Authentification<br>Cryptage | Clé de chiffrement par blocs:<br>– clé dédiée [1];<br>– clé globale de monodiffusion / diffusion *établie* [2] en dehors de l'*échange* [3], *identifiée* par l'octet de contrôle de sécurité.<br>Clé d'authentification: globale, *établie* [2] en dehors de l'échange [3] | – |
| general-glo-ciphering<br>general-ded-ciphering | | | | | Oui [5] |
| general-ciphering | Tierce partie ou Client – Serveur | Clé symétrique | Authentification<br>Cryptage | Clé de chiffrement par blocs:<br>– clé globale de monodiffusion / diffusion établie [2] en dehors de l'échange [3], identifiée dans le cadre de l'échange;<br>– établie [2] dans le cadre de l'échange [4].<br>Clé d'authentification: globale, établie [2] en dehors de l'échange [3] | Oui [5] |
| general-signing | | Clé asymétrique | Signature numérique | Clé de signature | Non |

[1] Transportée par l'AARQ.

[2] L'établissement de clé peut être un enveloppement de clé ou un agrément de clé.

[3] Dans le serveur, ces clés sont conservées par les objets "Security setup".

[4] Les données de clé sont transportées dans l'APDU protégée.

[5] L'utilisation de la compression est contrôlée par l'octet de contrôle de sécurité.

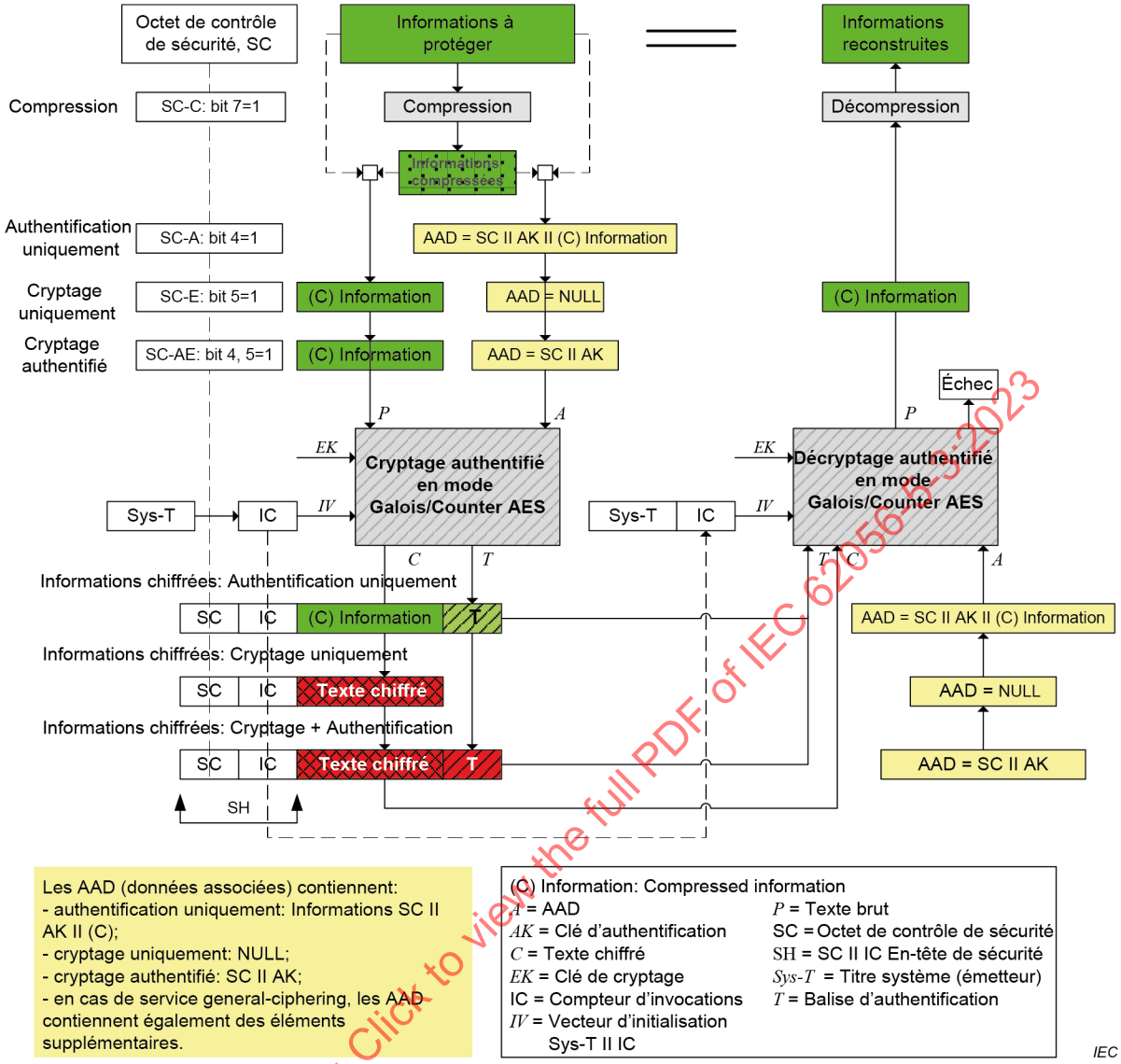
IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.7.2.4 Cryptage, authentification et compression

##### 5.7.2.4.1 Vue d'ensemble

Le cryptage et l'authentification visant à protéger les informations à l'aide de l'algorithme AES-GCM sont représentés à la Figure 29. Voir aussi 5.3.3.7. Cet algorithme peut être associé à la compression.

En cas de protection de message, les informations à protéger consistent en une APDU xDLMS. En cas de protection de données COSEM, les informations à protéger sont les données COSEM, c'est-à-dire la ou les valeurs de l'attribut COSEM ou le ou les paramètres d'appel de méthode ou de retour.




NOTE En cas de general-ciphering, les AAD comprennent également des champs supplémentaires (voir Tableau 28).

**Figure 29 – Protection cryptographique des informations à l'aide d'AES-GCM**

Le matériau de sécurité exigé est spécifié de 5.7.2.4.2 à 5.7.2.4.5.

#### 5.7.2.4.2 En-tête de sécurité

L'en-tête de sécurité SH comprend l'octet de contrôle de sécurité concaténé avec le compteur d'invocations: SH = SC II IC. L'octet de contrôle de sécurité est présenté dans le Tableau 27, où:

- Bit 3…0: Security_Suite_Id (voir 5.3.7);
- Bit 4: sous-champ "A": indique que l'authentification est appliquée;
- Bit 5: sous-champ "E": indique que le cryptage est appliqué;
- Bit 6: Sous-champ Key_Set: 0 = Monodiffusion;
  1 = Diffusion;
- Bit 7: Indique l'utilisation de la compression.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 27 – Octet de contrôle de sécurité**

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3..0 |
|---|---|---|---|---|
| Compression | Key_Set | E | A | Security_Suite_Id |
| Le bit Key_Set n'est pas pertinent et doit être défini sur 0 lorsque le chiffrement dédié spécifique au service ou les APDU general-ded-ciphering ou general-ciphering sont utilisés. | | | | |

#### 5.7.2.4.3 Texte brut et données supplémentaires authentifiées

Le texte brut, symbolisé par $P$, et les données supplémentaires authentifiées, symbolisées par $A$, dépendent du type de protection. Ils sont présentés dans le Tableau 28, où $SC$ est l'octet de contrôle de sécurité, $AK$ est la clé d'authentification et $I$ représente les informations, c'est-à-dire l'APDU xDLMS ou les données COSEM à protéger.

**Tableau 28 – Texte brut et données supplémentaires authentifiées**

| Contrôle de sécurité, $SC$ | | Protection | P | A, données supplémentaires authentifiées | |
|---|---|---|---|---|---|
| Champ E | Champ A | | | Glo-ciphering spécifique au service<br>Ded-ciphering spécifique au service<br>general-glo-ciphering<br>general-ded-ciphering | general-ciphering |
| 0 | 0 | Aucune | – | – | – |
| 0 | 1 | Authentification uniquement | – | SC II AK II (C) I | $SC$ II $AK$ II<br>transaction-id II [1]<br>originator-system-title II [1]<br>recipient-system-title II [1]<br>date-time II [1]<br>other-information II [1]<br>(C) I |
| 1 | 0 | Cryptage uniquement | (C) I | – | – |
| 1 | 1 | Cryptage et authentification | (C) I | $SC$ II $AK$ | $SC$ II $AK$ II<br>transaction-id II [1]<br>originator-system-title II [1]<br>recipient-system-title II [1]<br>date-time II [1]<br>other-information [1] |
| [1] Les champs transaction-id ….other-information sont des CHAÎNES D'OCTETS codées en A-XDR. La longueur et la valeur de chaque champ sont incluses dans les AAD. | | | | | |

#### 5.7.2.4.4 Clé de cryptage et clé d'authentification

Ces clés, utilisées par AES-GCM, sont spécifiées en 5.3.3.7.4 et 5.3.3.7.5, respectivement. Les différentes clés utilisées dans DLMS®/COSEM et leur établissement sont spécifiés en 5.5.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.7.2.4.5 Vecteur d'initialisation

Voir 5.3.3.7.3.

#### 5.7.2.4.6 APDU xDLMS de chiffrement spécifique au service

Pour certaines APDU xDLMS (voir 4.2.4.4.6 et 0), une variante chiffrée utilisant la clé globale et une utilisant la clé dédiée sont disponibles. Ces APDU chiffrées peuvent être utilisées entre un client et un serveur. La structure des APDU de chiffrement spécifique au service est représentée à la Figure 30. Voir aussi Tableau 28 et Tableau 29.

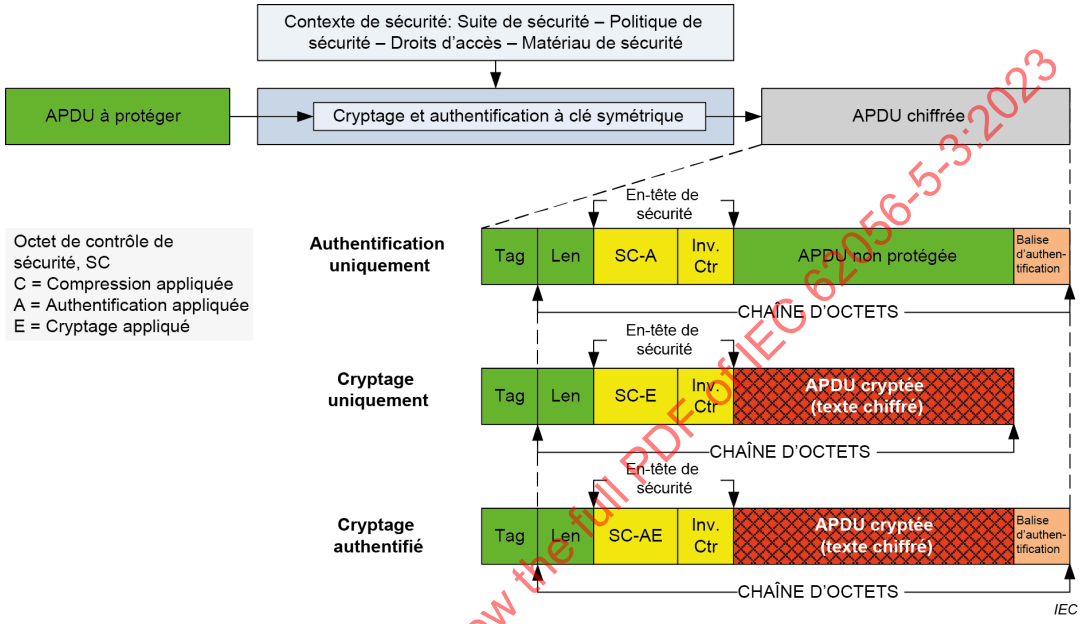



**Figure 30 – Structure des APDU xDLMS de chiffrement global spécifique au service / de chiffrement dédié spécifique au service**

#### 5.7.2.4.7 APDU xDLMS general-glo-ciphering et general-ded-ciphering

Ces APDU peuvent être utilisées pour chiffrer d'autres APDU xDLMS à l'aide de la clé globale ou de la clé dédiée. Elles peuvent être utilisées entre un client et un serveur. Leur structure est représentée à la Figure 31. Voir aussi Tableau 28 et Tableau 29.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

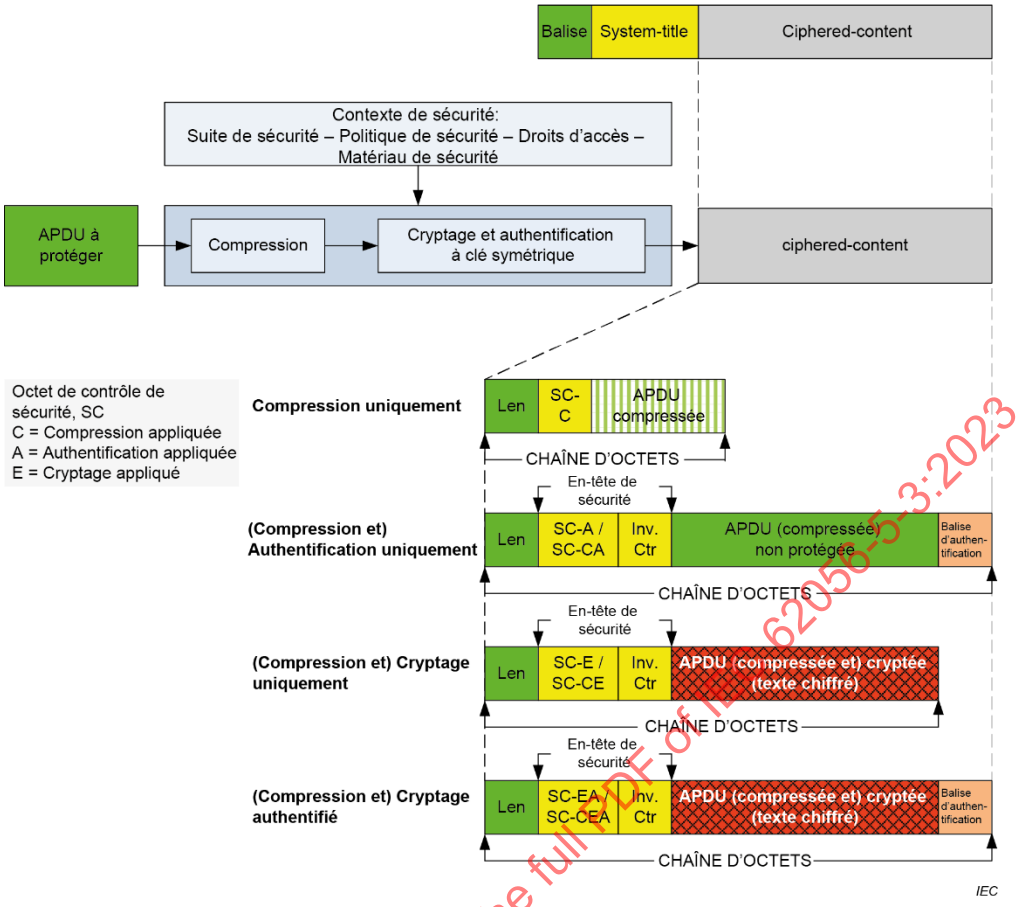



**Figure 31 – Structure des APDU xDLMS general-glo-ciphering et general-ded-ciphering**

#### 5.7.2.4.8 APDU general-ciphering

L'APDU general-ciphering peut être utilisée entre un client et un serveur ou entre une tierce partie et le serveur. Ces APDU contiennent également les informations nécessaires relatives à la clé à utiliser. La structure de l'APDU general-ciphering est représentée à la Figure 32. Voir aussi Tableau 28 et Tableau 29.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

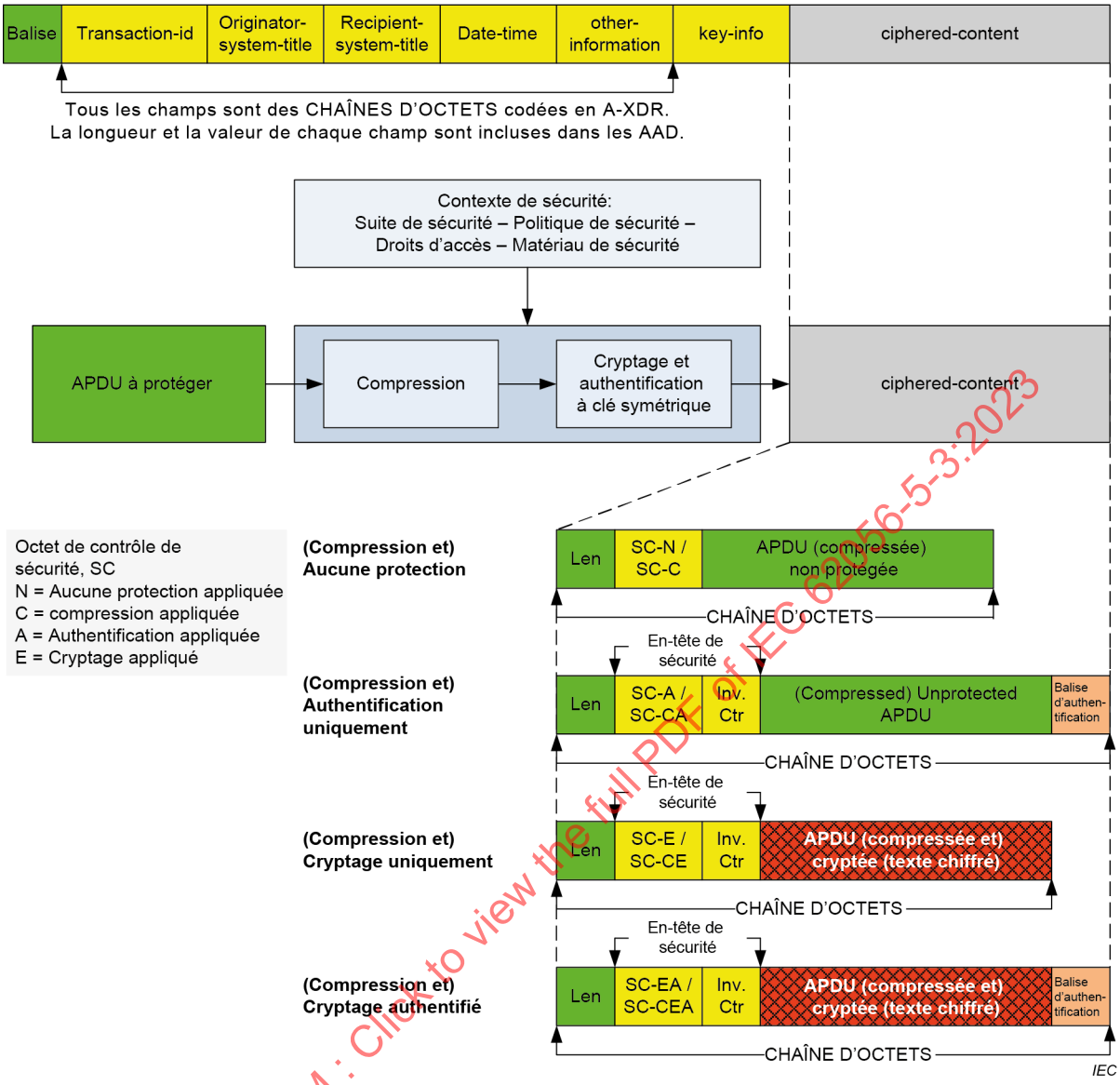



**Figure 32 – Structure des APDU xDLMS general-ciphering**

#### 5.7.2.4.9 Utilisation des champs des APDU xDLMS de chiffrement

L'utilisation des champs des APDU xDLMS de chiffrement est récapitulée dans le Tableau 29.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 29 – Utilisation des champs des APDU xDLMS de chiffrement**

| Champ d'APDU | Chiffrement global / dédié spécifique au service | general-glo- /general-ded- ciphering | general-ciphering | Signification |
|---|---|---|---|---|
| tag | Spécifique au service | [219] [220] | [221] | Balise de l'APDU de chiffrement (voir 0) |
| system-title | – | + | – | Voir 6.5. |
| transaction-id | – | – | + | |
| originator-system-title | – | – | + | |
| recipient-system-title | – | – | + | |
| date-time | – | – | + | |
| other-information | – | – | + | |
| key-info | – | – | + | |
| octet de contrôle de sécurité | + | + | + | Fournit des informations sur la protection appliquée, le jeu de clés et la suite de sécurité appliquée. Voir Tableau 27. [1)] |
| Compteur d'invocations | + | + | + | Champ d'appel du vecteur d'initialisation. Il s'agit d'un compteur d'entiers qui augmente à chaque appel de la fonction de cryptage authentifié à l'aide de la même clé.<br>Lorsqu'une nouvelle clé est établie, le compteur d'invocations lié doit être redéfini sur 0. |
| APDU non protégée | + | + | + | APDU non protégée (idem que pour APDU à protéger). |
| APDU cryptée | + | + | + | APDU cryptée, c'est-à-dire texte chiffré. |
| balise d'authentification | + | + | + | Calculé par l'algorithme AES-GCM (voir 5.3.3.7). |

[1)] En cas d'APDU general-ciphering, le bit "key-set" de l'octet de contrôle de sécurité n'est pas pertinent et doit être défini sur 0.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 5.7.2.4.10 Exemple de codage: APDU xDLMS global-get-request

Le Tableau 30 présente un exemple de codage d'une APDU xDLMS de chiffrement global spécifique au service: glo-get-request.

**Tableau 30 – Exemple: APDU xDLMS glo-get-request**

| | X | Contenu | | | LEN (X) octets | Len (X) bits |
|---|---|---|---|---|---|---|
| Matériau de sécurité | | | | | | |
| Suite de sécurité | | GCM-AES-128 | | | | |
| Titre système | *Sys-T* | **4D4D4D**0000BC614E (ici, les cinq derniers octets contiennent le numéro de fabrication au format hexadécimal) | | | 8 | 64 |
| Compteur d'invocations | *IC* | 01234567 | | | 4 | 32 |
| Vecteur d'initialisation | *IV* | *Sys-T* II IC | | | 12 | 96 |
| | | 4D4D4D0000BC614E01234567 | | | | |
| Clé de chiffrement par blocs (globale) | *EK* | 000102030405060708090A0B0C0D0E0F | | | 16 | 128 |
| Clé d'authentification | *AK* | D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF | | | 16 | 128 |
| **Sécurité appliquée** | | **Authentification** | **Cryptage** | **Cryptage authentifié** | | |
| Octet de contrôle de sécurité (avec clé de monodiffusion) | *SC* | *SC-A* | *SC-E* | SC-AE | 1 | 8 |
| | | 10 | 20 | 30 | | |
| En-tête de sécurité | *SH* | *SH* = *SC-A* II IC | *SH* = *SC-E* II IC | *SH* = *SC-AE* II IC | | |
| | | 1001234567 | 2001234567 | 3001234567 | 5 | 40 |
| **Valeurs d'entrée** | | **Authentification** | **Cryptage** | **Cryptage authentifié** | | |
| APDU xDLMS à protéger | *APDU* | C0010000080000010000FF0200 (Get-request, attribut 2 de l'objet Clock) | | | 13 | 104 |
| Texte brut | *P* | NULL | C0010000080000010000FF0200 | C0010000080000010000FF0200 | 13 | 104 |
| Données associées | *A* | SC II AK II APDU | – | SC II AK | | |
| Données associées – Authentification | *A-A* | 10D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDFC0010000080000010000FF0200 | – | – | 30 | 240 |
| Données associées – Cryptage | *A-E* | – | – | – | 0 | 0 |
| Données associées – Cryptage authentifié | *A-AE* | – | – | 30D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF | 17 | 136 |
| **Valeurs de sortie** | | **Authentification** | **Cryptage** | **Cryptage authentifié** | | |
| Texte chiffré | *C* | NULL | 411312FF935A47566827C467BC | 411312FF935A47566827C467BC | 13 | 104 |
| Balise d'authentification | *T* | 06725D910F9221D263877516 | – | 7D825C3BE4A77C3FCC056B6B | 12 | 96 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| | X | Contenu | | | LEN (X) octets | Len (X) bits |
|---|---|---|---|---|---|---|
| APDU chiffrée complète | | TAG II LEN II SH II APDU II T | TAG II LEN II SH II C | TAG II LEN II SH II C II T | – | – |
| APDU authentifiée | | C81E1001234567C0 0100000800000100 00FF020006725D91 0F9221D263877516 | – | – | 32 | 256 |
| APDU cryptée | | – | C812200123456741 1312FF935A475668 27C467BC | – | 20 | 160 |
| APDU authentifiée et cryptée | | – | – | C81E300123456741 1312FF935A475668 27C467BC7D825C3B E4A77C3FCC056B6B | 32 | 256 |
| NOTE Dans cet exemple, la valeur du compteur d'invocations est de 01234567. En réalité, la valeur doit être incrémentée à chaque appel de l'algorithme AES-GCM. | | | | | | |

Le Tableau 31 présente un exemple dans lequel les APDU ACCESS.request et ACCESS.response présentées dans le Tableau F.10 sont protégées à l'aide du cryptage authentifié. L'APDU general-ciphering spécifiée en 5.7.2.4.8 est utilisée. La clé de cryptage est convenue à l'aide du mécanisme d'agrément de clé One-pass Diffie-Hellman C(1e, 1s, ECC CDH) (voir 5.3.4.6.3). La clé d'authentification est la même que celle du Tableau 30.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 31 – Service ACCESS avec le mécanisme d'agrément de clé One-pass Diffie-Hellman C(1e, 1s, ECC CDH) et general-ciphering**

| Éléments de messages | Contenu | LEN (octets) |
|---|---|---|
| **General-Ciphering** | DD | 1 |
| **transaction-id** | | 0 |
| **length** | 08 | 1 |
| **value** | 0102030405060708 | 8 |
| **originator-system-title** | | 0 |
| **length** | 08 | 1 |
| **value** | 4D4D4D0000BC614E | 8 |
| **recipient-system-title** | | 0 |
| **length** | 08 | 1 |
| **value** | 4D4D4D0000000001 | 8 |
| **date-time** | | 0 |
| **length** | 00 | 1 |
| **value** | | 0 |
| **other-information** | | 0 |
| **length** | 00 | 1 |
| **value** | | 0 |
| **key-info OPTIONAL** | 01 | 1 |
| **agreed-key CHOICE** | 02 | 1 |
| **key-parameters** | | 0 |
| **length** | 01 | 1 |
| **value** | 01 | 1 |
| **key-ciphered-data** | | 0 |
| **length** | 8180 | 2 |
| **value** | C323C2BD45711DE4688637D919F92E9D<br>B8FB2DFC213A88D21C9DC8DCBA917D81<br>7051DE1BADB360D50058F794B0960AE<br>11FA28D392CFF907A62D13E3357B1DC0<br>B51BE089D0B682863B2217201E73A1A9<br>031968A9B4121DCBC3281A69739AF874<br>29F5B3AC5471E7B6A04A2C0F2F8A25FD<br>772A317DF97FC5463FEAC248EB8AB8BE | 128 |
| **ciphered-content** | | 0 |
| **length** | 81EB | 2 |
| **value**<br><br>ACCESS.request avec cryptage authentifié<br><br>SC II IC II texte chiffré II balise d'authentification | 3100000000F435069679270C5BF4425E<br>E5777402A6C8D51C620EED52DBB18837<br>8B836E2857D5C053E6DDF27FA87409AE<br>F502CD9618AE47017C010224FD109CC0<br>BEB21E742D44AB40CD11908743EC90EC<br>8C40E221D517F72228E1A26E827F43DC<br>18ED27B5F458D66508B05A2A4CC6FED1<br>78C881AFC3BC67064689BE8BB41C80AB<br>B3C114A31F4CB03B8B64C7E0B4CE77B2<br>399C93347858888F92239713B38DF01C<br>4858245827A92EF334172EA636B31CBB<br>DF2A96AD5D035F66AA38F1A2D97D4BBA<br>99622E6B5F18789CECB2DFB3937D9F3E<br>17F8B472098E6563238F375283748098<br>36002AEA6E7012D2ADFAA7 | 235 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Éléments de messages | Contenu | LEN (octets) |
|---|---|---|
| **general-ciphering (codé)** | DD080102030405060708084D4D4D0000<br>BC614E084D4D4D000000000100000102<br>01018180C323C2BD45711DE4688637D9<br>19F92E9DB8FB2DFC213A88D21C9DC8DC<br>BA917D8170511DE1BADB360D50058F79<br>4B0960AE11FA28D392CFF907A62D13E3<br>357B1DC0B51BE089D0B682863B221720<br>1E73A1A9031968A9B4121DCBC3281A69<br>739AF87429F5B3AC5471E7B6A04A2C0F<br>2F8A25FD772A317DF97FC5463FEAC248<br>EB8AB8BE81EB3100000000F435069679<br>270C5BF4425EE5777402A6C8D51C620E<br>ED52DBB188378B836E2857D5C053E6DD<br>F27FA87409AEF502CD9618AE47017C01<br>0224FD109CC0BEB21E742D44AB40CD11<br>908743EC90EC8C40E221D517F72228E1<br>A26E827F43DC18ED27B5F458D66508B0<br>5A2A4CC6FED178C881AFC3BC67064689<br>BE8BB41C80ABB3C114A31F4CB03B8B64<br>C7E0B4CE77B2399C93347858888F9223<br>9713B38DF01C4858245827A92EF33417<br>2EA636B31CBBDF2A96AD5D035F66AA38<br>F1A2D97D4BBA99622E6B5F18789CEEB2<br>DFB3937D9F3E17F8B472098E6563238F<br>375283748098360O2AEA6E7012D2ADFA<br>A7 | 401 |
| **General-Ciphering** | DD | 1 |
| **transaction-id** | | 0 |
| **length** | 08 | 1 |
| **value** | 0123456789012345 | 8 |
| **originator-system-title** | | 0 |
| **length** | 08 | 1 |
| **value** | 4D4D4D0000000001 | 8 |
| **recipient-system-title** | | 0 |
| **length** | 08 | 1 |
| **value** | 4D4D4D0000BC614E | 8 |
| **date-time OPTIONAL** | | 0 |
| **length** | 00 | 1 |
| **value** | | 0 |
| **other-information** | | 0 |
| **length** | 00 | 1 |
| **value** | | 0 |
| **key-info OPTIONAL** | 01 | 1 |
| **agreed-key CHOICE** | 02 | 1 |
| **key-parameters** | | 0 |
| **length** | 01 | 1 |
| **value** | 01 | 1 |
| **key-ciphered-data** | | 0 |
| **length** | 8180 | 2 |
| **value** | 6439724714B47CD9CB988897D8424AB9<br>46DCD083D37A954637616011B9C23787<br>73295F0F850D8DAFD1BBE9FE666E53E4<br>F097CD10B38B69622152724A90987444<br>E1FF47974A1F6931A6502F58147463F0<br>E8CC517D47F55B0AC56DD8AC5C9D0E48<br>1934F2D90F9893016BD82B6E3FFE21FF<br>1588F3278B4E9D98EB4FB62ADD64B380 | 128 |
| **ciphered-content** | | 0 |
| **length** | 3D | 1 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Éléments de messages | Contenu | LEN (octets) |
|---|---|---|
| **value**<br><br>ACCESS.response avec cryptage authentifié<br><br>SC II IC II texte chiffré II balise d'authentification | 3100000000B3FFCAA594642D8319CEC6<br>B2A233E2BF4621D6991B97E4565B986E<br>8CCBE9A299D8E7869723638FF6BB20E6<br>6E175E6F2D762CFD26B3D58733 | 61 |
| **general-ciphering (codé)** | DD0801234567890123450 84D4D4D0000<br>000001084D4D4D0000BC614E00000102<br>0101818064397247 14B47CD9CB988897<br>D8424AB946DCD083D37A954637616011<br>B9C2378773295F0F850D8DAFD1BBE9FE<br>666E53E4F097CD10B38B69622152724A<br>90987444E1FF47974A1F6931A6502F58<br>147463F0E8CC517D47F55B0AC56DD8AC<br>5C9D0E481934F2D90F9893016BD82B6E<br>3FFE21FF1588F3278B4E9D98EB4FB62A<br>DD64B3803D3100000000B3FFCAA59464<br>2D8319CEC6B2A233E2BF4621D6991B97<br>E4565B986E8CCBE9A299D8E786972363<br>8FF6BB20E66E175E6F2D762CFD26B3D5<br>8733 | 226 |

### 5.7.2.5 Signature numérique

L'algorithme est l'algorithme de signature numérique à courbe elliptique (ECDSA) tel que spécifié en 5.3.4.5.

La structure de l'APDU general-signing est représentée à la Figure 33. Pour les champs supplémentaires, voir Tableau 29 et 6.5.

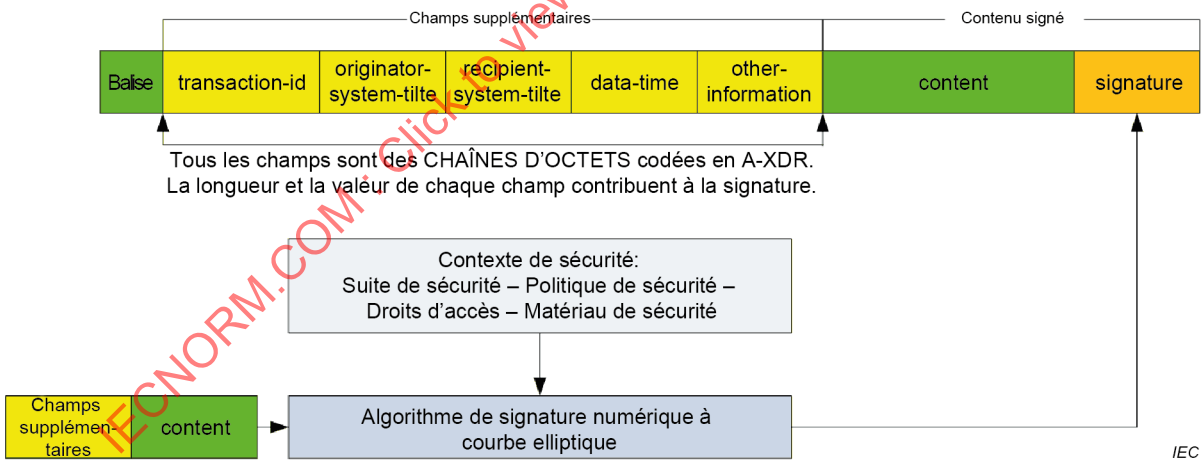



**Figure 33 – Structure des APDU general-signing**

### 5.7.3 Protection multicouche par plusieurs parties

La protection cryptographique peut être appliquée par plusieurs parties. En règle générale, les parties sont:

- un serveur;
- un client;
- une tierce partie.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Chaque partie peut appliquer une ou plusieurs couches de protection:

- pour appliquer le cryptage, l'authentification ou le cryptage authentifié, les APDU de chiffrement sont utilisées. Une tierce partie doit utiliser l'APDU general-ciphering. Le client peut utiliser une quelconque des APDU de chiffrement. Le cryptage authentifié est considéré comme une couche unique de protection;
- pour appliquer une signature numérique, l'APDU general-signing est utilisée.

Si le chiffrement et la signature numérique sont appliqués par la même partie pour la même partie, alors la signature numérique est généralement appliquée en premier.

L'APDU general-ciphering et l'APDU general-signing comprennent l'Originator_System_Title et le Recipient_System_Title, identifiant la partie qui a appliqué la protection et la partie qui doit vérifier/supprimer la protection.

La protection à appliquer sur la réponse dépend de la politique de sécurité et des droits d'accès sur la réponse, et de la protection appliquée sur la demande. Si un type de protection a été appliqué sur la demande par une partie, le même type de protection est appliqué pour la même partie dans la réponse. Cependant, si un type de protection qui a été appliqué sur la demande n'est pas exigé sur la réponse, aucune protection n'est appliquée sur la réponse pour cette partie.

Exemple 1 Si la demande a été signée numériquement par la tierce partie et authentifiée par le client, et si la protection exigée sur la réponse comprend l'authentification et la signature numérique, la réponse est authentifiée pour le client et signée numériquement pour la tierce partie.

Exemple 2 Si la demande a été signée numériquement par la tierce partie et authentifiée par le client, et si la protection exigée sur la réponse comprend l'authentification uniquement, la réponse est authentifiée pour le client et aucune protection n'est appliquée à la tierce partie. (La TP reçoit une APDU general-ciphering sans protection appliquée.)

Si une protection qui n'était pas appliquée sur la demande est exigée sur la réponse, le serveur ne peut pas déterminer la partie comprenant la réponse à protéger à partir de la demande. Par conséquent, il doit appliquer la protection pour toutes les parties.

Voir aussi Annexe J.

### 5.7.4 Mécanismes d'authentification HLS

L'authentification HLS exige le traitement cryptographique des épreuves échangées par le client et le serveur. Les mécanismes d'authentification HLS, les informations échangées et les formules permettant de traiter les épreuves sont présentés dans le Tableau 32.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 32 – Mécanismes d'authentification HLS DLMS®/COSEM**

| Mécanisme d'authentification | Passe 1: C →S | Passe 2: S →C | Passe 3: C →S f(StoC) | Passe 4: S→C f(CtoS) |
|---|---|---|---|---|
| | Contenue par | | | |
| | AARQ | AARE | XX.request<br>Authentification reply_to_HLS | XX.response<br>Authentification reply_to_HLS |
| mechanism_id(2)<br>Spéc. manuelle de HLS | *CtoS*: chaîne aléatoire de 8 octets à 64 octets | *StoC*: chaîne aléatoire de 8 octets à 64 octets | Spéc. manuelle | Spéc. manuelle |
| mechanism_id(3)<br>HLS MD5 [1)] | | | **MD5**(StoC \|\| Secret HLS) | **MD5**(CtoS \|\| Secret HLS) |
| mechanism_id(4)<br>HLS SHA-1 [1] | | | **SHA-1**(StoC \|\| Secret HLS) | **SHA-1**(CtoS \|\| Secret HLS) |
| mechanism_id(5)<br>HLS GMAC | *CtoS*: chaîne aléatoire de 8 octets à 64 octets<br>Facultatif:<br>System-Title-C dans calling-AP-title | *StoC*: chaîne aléatoire de 8 octets à 64 octets<br>Facultatif:<br>System-Title-S dans responding-AP-title | SC II IC II **GMAC** (SC \|\| AK \|\| StoC) | SC II IC II **GMAC** (SC \|\| AK \|\| CtoS) |
| mechanism_id(6)<br>HLS SHA-256 | | | **SHA-256** (HLS_Secret \|\| SystemTitle-C \|\| SystemTitle-S \|\| StoC II CtoS) | **SHA-256** (HLS_Secret \|\| SystemTitle-S \|\| SystemTitle-C \|\| CtoS \|\| StoC) |
| mechanism_id(7)<br>HLS ECDSA | *CtoS*: chaîne aléatoire de 32 octets à 64 octets<br>Facultatif:<br>System-Title-C dans calling-AP-title,<br>Cert-Sign-Client dans calling-AE-qualifier | *StoC*: chaîne aléatoire de 32 octets à 64 octets<br>Facultatif:<br>System-Title-S dans responding-AP-title,<br>Cert-Sign-Server dans responding-AE-qualifier | **ECDSA**( SystemTitle-C \|\| SystemTitle-S \|\| StoC II CtoS) | **ECDSA**( SystemTitle-S \|\| SystemTitle-C \|\| CtoS II StoC) |

Légende:

– C: Client, S: Serveur, CtoS: Épreuve client à serveur, StoC: Épreuve serveur à client;

– IC: Compteur d'invocations;

– xx.request/.response: primitives de service xDLMS utilisées pour accéder à la méthode *reply_to_HLS authentication* de l'objet "Association SN / LN".

[1] L'utilisation des mécanismes d'authentification 3 et 4 n'est pas recommandée pour de nouvelles implémentations.

NOTE Les titres système et les certificats sont à envoyer uniquement s'ils ne sont pas déjà connus par l'autre partie.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Lorsque les titres système et les certificats pour la clé de signature numérique sont également nécessaires, ils peuvent être transportés dans les APDU AARQ/AARE, contenant le service COSEM-OPEN .request/.response (voir Figure 12). System_Title et Cert-Sign peuvent être déjà connus; dans ce cas, ils ne sont pas à transporter. Si ces éléments ne sont pas disponibles, le résultat du traitement de l'épreuve échoue et l'AA ne doit pas être établie.

Le Tableau 33 fournit un vecteur d'essai pour le mécanisme d'authentification HLS 5 avec GMAC.

**Tableau 33 – Exemple de HLS utilisant le mécanisme d'authentification 5 avec GMAC**

| Matériau de sécurité | X | Contenu | | LEN(X) octets | len(X) bits |
|---|---|---|---|---|---|
| Suite de sécurité | | GCM-AES-128 | | | |
| Titre système | *Sys-T* | Client | Serveur | | |
| | | **4D4D4D**0000000001 | **4D4D4D**0000BC614E | | |
| | | (ici, les cinq derniers octets contiennent le numéro de fabrication au format hexadécimal) | | 8 | 64 |
| Compteur d'invocations | *IC* | 00000001 | 01234567 | 4 | 32 |
| Vecteur d'initialisation | *IV* | Sys-T II IC | | 12 | 96 |
| | | Client | Serveur | | |
| | | 4D4D4D0000000001 00000001 | 4D4D4D0000BC614E 01234567 | | |
| Clé de chiffrement par blocs (globale) | *EK* | 000102030405060708090A0B0C0D0E0F | | 16 | 128 |
| Clé d'authentification | *AK* | D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF | | 16 | 128 |
| Octet de contrôle de sécurité | SC | 10 | | 1 | 8 |
| | | **Passe 1: Le client envoie une épreuve au serveur** | | | |
| CtoS | | 4B35366956616759 "K56iVagY" | | 8 | 64 |
| | | **Passe 2: Le serveur envoie une épreuve au client** | | | |
| StoC | | 503677524A323146 "P6wRJ21F" | | 8 | 64 |
| | | **Passe 3: Le client traite StoC** | | | |
| SC II AK II StoC | | 10D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF503677524A323146 | | | |
| T = **GMAC** (SC II AK II StoC) | | 1A52FE7DD3E72748973C1E28 | | 12 | 96 |
| f(StoC) = SC II IC II T | | 10000000011A52FE7DD3E72748973C1E28 | | 17 | 136 |
| | | **Passe 4: Le serveur traite CtoS** | | | |
| SC II AK II CtoS | | 10D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF4B35366956616759 | | | |
| T (SC II AK II CtoS) | | FE1466AFB3DBCD4F9389E2B7 | | 12 | 96 |
| f(CtoS) = SC II IC II T | | 1001234567FE1466AFB3DBCD4F9389E2B7 | | 17 | 136 |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le Tableau 34 fournit un vecteur d'essai pour le mécanisme d'authentification HLS 7 avec ECDSA.

**Tableau 34 – Exemple de HLS utilisant le mécanisme d'authentification 7 avec ECDSA**

| Matériau de sécurité | X | Contenu | LEN (octets) |
|---|---|---|---|
| Suite de sécurité | | ECDH-ECDSA-AES-128-GCM | |
| Courbe | | P-256 | |
| Paramètres de domaine | D | Voir Tableau G.1. | |
| Titre système client | Sys-TC | 4D4D4D0000BC614E | 8 |
| Titre système serveur | Sys-TS | 4D4D4D0000000001 | 8 |
| Clé privée Client | Pri-KC | E9A045346B2057F1820318AB125493E9AB36CE590011C0FF30090858A118DD2E | 32 |
| Clé privée Serveur | Pri-KS | B582D8C910018302BA3131BAB9BB6838108BB9408C30B2E49285985256A59038 | 32 |
| Clé publique Client | Pub-KC | 917DBFECA43307375247989F07CC23F53D4B963AF8026C749DB33852011056DFDBE8327BD69CC149F018A8E446DDA6C55BCD78E596A56D403236233F93CC89B3 | 64 |
| Clé publique Serveur | Pub-KS | E4D07CEB0A5A6DA9D2228B054A1F5E295E1747A963974AF75091A0B0BC2FB92DA7D2ABD9FDD41579F36A1C8171A0CB6382211F1949FD95C8FAE148896920450D | 64 |
| Épreuve Client à Serveur | CtoS | 2CA1FC2DE9CD03B5E8E234CEA16F2853F6DC5F54526F4F4995772A50FB7E63B3 | 32 |
| Épreuve Serveur à Client | StoC | 18E95FFE3AD0DCABDC5D0D141DC987E270CB0A395948D4231B09DE6579883657 | 32 |
| ECDSA(SystemTitle-C \|\| SystemTitle-S \|\|StoC \|\| CtoS)<br>(calculé avec la Pri-KC) | f(StoC) | C5C6D6620BDB1A39FCE50F4D64F0DB712D6FB57A64030B0C297E1250DC859660D3B1FA334AD8041180 7369F5DD3BC17B59894C9E9C11C59376580D15A2646D16 | 64 |
| ECDSA(SystemTitle-S \|\| SystemTitle-C \|\| CtoS \|\| StoC)<br>(calculé avec la Pri-KS) | f(CtoS) | 946C2E3E4F18291571F4A45ACB708610057469 4A3BAF67D2D147FE8F92481A5AB2186C5CBC3F80E94482D9388B85C6A73E5FD687F09773C1F615AA2A905ED057 | 64 |
| NOTE Les valeurs des clés publiques sont représentées comme suit: FE2OS($x_p$)II FE2OS($y_p$). | | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 5.7.5 Protection des données COSEM

Les algorithmes cryptographiques appliqués aux APDU xDLMS peuvent également être appliqués aux données COSEM, c'est-à-dire aux valeurs d'attribut et aux paramètres d'appel de méthode/de retour. Ceci est effectué par l'accès aux attributs et/ou méthodes des autres objets COSEM de manière indirecte par les instances de la classe d'interface "Data protection", voir IEC 62056-6-2:2021, 4.4.9.

La liste des données à protéger, la protection exigée et les paramètres de protection sont déterminés par les objets "Data protection".

Les objets "Data protection" permettent l'application ou la suppression de la protection lors de la lecture ou de l'écriture d'une liste d'attributs, ou lors de l'appel de méthodes des objets COSEM. La protection à appliquer/supprimer peut comprendre toute combinaison de l'authentification, du cryptage et de la signature numérique.

Les APDU contenant les appels de service permettant d'accéder aux attributs et méthodes des objets "Data protection" sont protégées comme exigé par la politique de sécurité qui prévaut et par les droits d'accès de l'objet "Data protection".

# 6 Spécification de service de la couche application DLMS®/COSEM

## 6.1 Primitives de service et paramètres

En général, les services d'une couche (ou d'une sous-couche) sont des capacités que cette couche offre à un utilisateur dans la couche (ou sous-couche) immédiatement supérieure. Afin de fournir ses services, une couche base ses fonctions sur les services qu'elle exige de la couche immédiatement inférieure. La Figure 34 représente cette notion de hiérarchie de service et représente la relation entre deux utilisateurs N correspondants et leurs entités de protocole homologue de couche N.

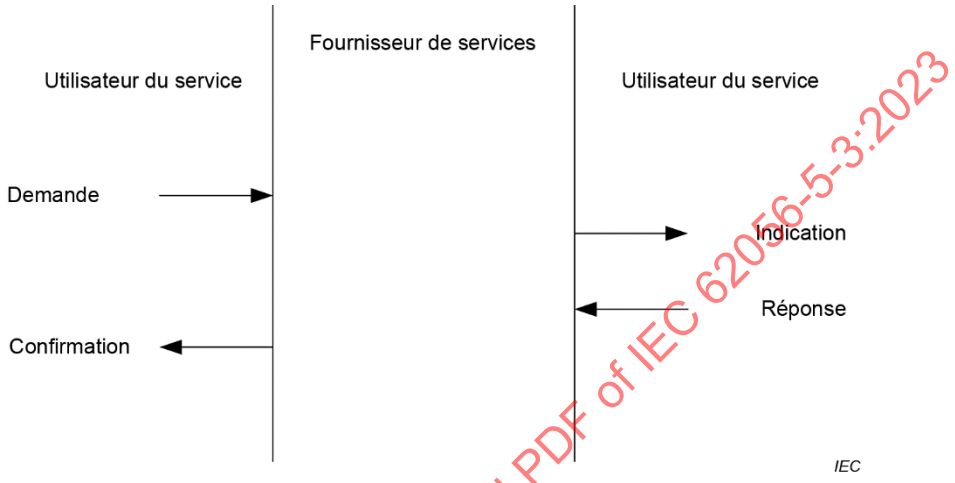



**Figure 34 – Primitives de service**

Les services sont indiqués en décrivant le flux d'informations entre l'utilisateur N et la couche N. Ce flux d'informations est modélisé par des événements discrets et instantanés, qui caractérisent la fourniture d'un service. Chaque événement consiste en la transmission d'une primitive de service d'une couche à l'autre par l'intermédiaire d'un point d'accès aux services de la couche N associé à un utilisateur N. Les primitives de service transportent les informations exigées en fournissant un service spécifique. Ces primitives de service sont abstraites du fait qu'elles ne spécifient que le service fourni et non les moyens par lesquels le service est fourni. Cette définition de service est indépendante de toute implémentation spécifique d'interface.

Les services sont spécifiés en décrivant les primitives de service et les paramètres qui caractérisent chaque service. Un service peut avoir une ou plusieurs primitives associées, qui constituent l'activité associée au service spécifique. Chaque primitive de service peut comprendre zéro paramètre ou plus, qui transportent les informations exigées pour fournir le service. Les primitives sont de quatre types génériques:

- DEMANDE: la primitive de demande est transmise de l'utilisateur N à la couche N pour demander l'initiation d'un service;
- INDICATION: la primitive indication est transmise de la couche N à l'utilisateur N pour indiquer un événement de couche N interne qui est significatif pour l'utilisateur N. Cet événement peut être logiquement lié à une demande de service distant, ou peut être causé par un événement interne à la couche N;
- RÉPONSE: la primitive de réponse est transmise de l'utilisateur N à la couche N pour exécuter une procédure précédemment appelée par une primitive d'indication;
- CONFIRMATION: la primitive de confirmation est transmise de la couche N à l'utilisateur N pour transporter les résultats d'une ou de plusieurs demandes de service précédentes associées.

Les relations possibles entre les types de primitives sont représentées par les diagrammes de séquences temporelles à la Figure 35. La figure indique également la relation logique des types de primitives. Les types de primitives qui se produisent plus tôt dans le temps et sont connectés

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

par des lignes pointillées dans les diagrammes sont les antécédents logiques des types de primitives ultérieures.

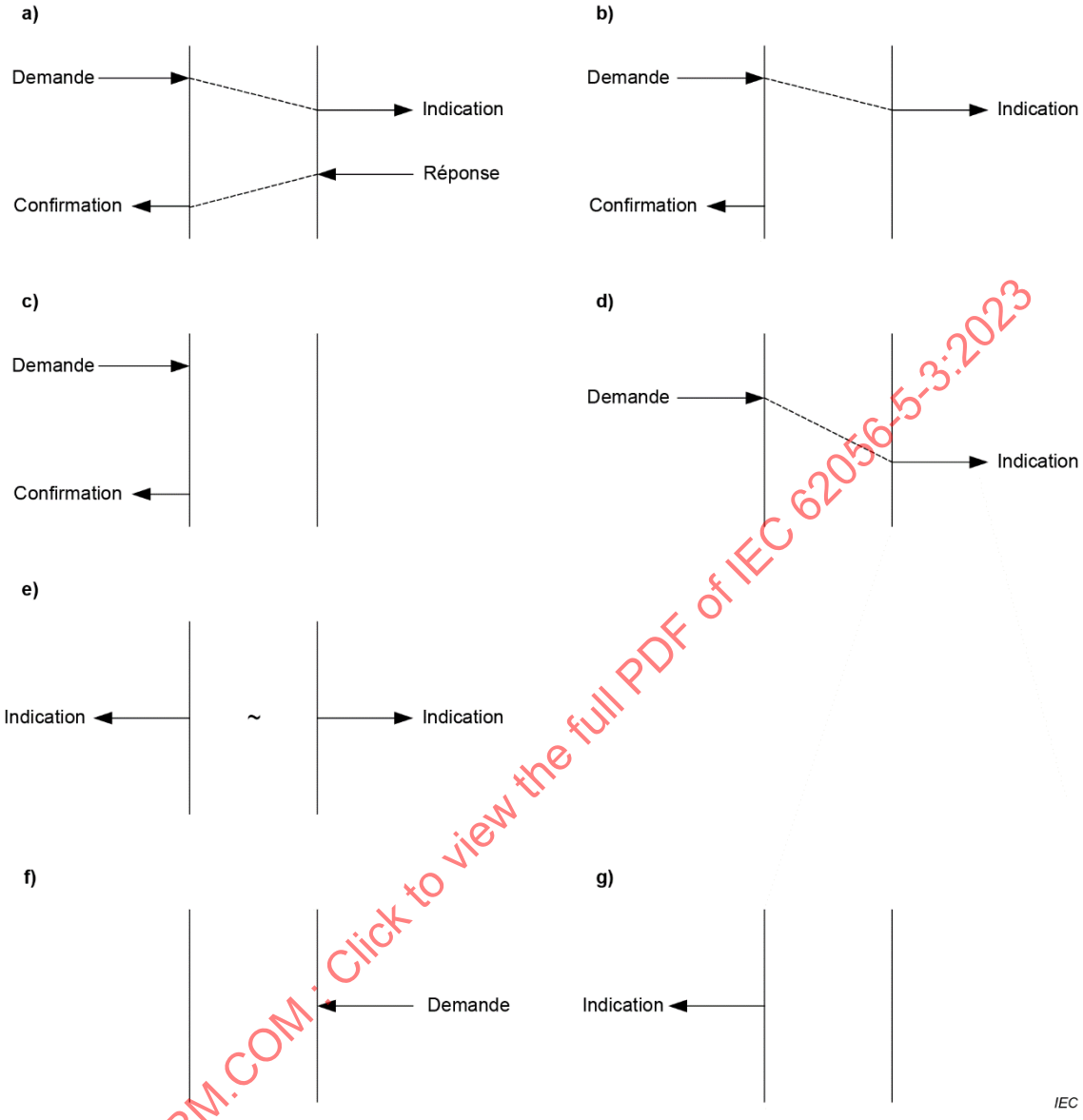



**Figure 35 – Diagrammes de séquences temporelles**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Les paramètres de service des primitives de service de l'AL COSEM sont présentés sous forme de tableau. Chaque tableau comprend de deux à cinq colonnes qui décrivent les primitives de service et leurs paramètres. Dans chaque tableau, un paramètre (ou une partie de celui-ci) est répertorié sur chaque ligne. Dans les colonnes des primitives de service appropriées, un code est utilisé pour spécifier le type d'utilisation du paramètre. Les codes utilisés sont répertoriés dans le Tableau 35.

Certains paramètres peuvent contenir des sous-paramètres. Ceux-ci sont indiqués par l'intermédiaire d'une étiquette (O, U, S ou C) apposée sur les paramètres. Les sous-paramètres sont indentés sous chaque paramètre. La présence de sous-paramètres dépend toujours de celle du paramètre sous lequel ils apparaissent. Par exemple, un paramètre facultatif peut comprendre des sous-paramètres. Si le paramètre n'est pas indiqué, aucun sous-paramètre ne peut être indiqué.

**Tableau 35 – Codes des paramètres de service de l'AL**

| | |
|---|---|
| O | Paramètre obligatoire pour la primitive. |
| U | Option de l'utilisateur; ce paramètre peut ou peut ne pas être indiqué, en fonction de l'utilisation dynamique par l'utilisateur de l'ASE. |
| S | Paramètre sélectionné parmi les autres paramètres S en tant que réponse interne de l'environnement ASE du serveur. |
| C | Paramètre dépendant d'autres paramètres ou de l'environnement de l'utilisateur de l'ASE. |
| (–) | Paramètre jamais spécifié. |
| = | Le code "(=)" suivant le code O, U, S ou C indique que le paramètre équivaut sur le plan sémantique au paramètre de la primitive de service se trouvant immédiatement à sa gauche dans le tableau. Par exemple, un code "O (=)" dans la colonne de la primitive de service .indication et un "O" dans la colonne de la primitive de service .request indiquent que le paramètre dans la primitive .indication équivaut sur le plan sémantique à celui dans la primitive .request. |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Dans la présente norme, les règles suivantes sont observées en ce qui concerne la mise en forme de termes:

- le nom des services ACSE et des services de transfert de données utilisant le référencement LN est écrit en majuscules. Exemples: COSEM-OPEN, GET;
- le nom des services de transfert de données utilisant le référencement SN est écrit en initiales majuscules. Exemples: Read, Write;
- la notation camel est utilisée dans les cas suivants: DataNotification, EventNotification, TriggerEventNotificationSending, UnconfirmedWrite, InformationReport;
- les types de primitives de service LN peuvent être mentionnés sous deux formes alternatives. Exemples: "primitive de service GET.request de type Request_Type == NORMAL" ou "primitive de service GET-REQUEST-NORMAL";
- les éléments de nom de paramètre de service sont en majuscules et sont associés par un trait de soulignement afin de ne constituer qu'une seule entité. Exemples: Protocol_Connection_Parameters, COSEM_Attribute_Descriptor;
- lorsque le même paramètre peut revenir plusieurs fois, le paramètre est répété entre accolades. Exemple: Data { Data };
- dans les spécifications de service de transfert de données, les paramètres utilisés avec le transfert de bloc uniquement sont affichés en gras. Exemple: **DataBlock_G**;
- la référence directe à un paramètre de service utilise les majuscules, alors que la référence indirecte (non spécifique) utilise le texte normal sans trait de soulignement. Exemple de référence directe: "Le paramètre COSEM_Attribute_Descriptor se rapporte à un attribut d'objet COSEM." Exemple de référence indirecte (non spécifique): "Une primitive de service GET-REQUEST-NORMAL contient un descripteur d'attribut COSEM unique.";
- les noms des APDU de transfert de données COSEM utilisant le référencement LN sont écrits en majuscules et associés par un tiret afin de ne constituer qu'une seule entité. Exemple: Get-Request-Normal;
- les noms des APDU de transfert de données COSEM utilisant le référencement SN sont écrits en notation camel. Exemple: ReadRequest.

## 6.2 Service COSEM-OPEN

*Fonction*

Le service COSEM-OPEN a pour fonction d'établir une AA entre des AP COSEM homologues. Il utilise le service A-ASSOCIATE de l'ACSE. Le service COSEM-OPEN fournit uniquement l'infrastructure qui permet de *transporter* ces informations. L'AP COSEM approprié est chargé de fournir et de vérifier ces informations.

*Sémantique des primitives de service*

Les primitives de service COSEM-OPEN doivent fournir les paramètres indiqués dans le Tableau 36.

**Tableau 36 – Paramètres de service des primitives de service COSEM-OPEN**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Protocol_Connection_Parameters | O | O (=) | O | O (=) |
| ACSE_Protocol_Version | U | U (=) | U | U (=) |
| Application_Context_Name | O | O (=) | O | O (=) |
| Called_AP_Title | U | U (=) | – | – |
| Called_AE_Qualifier | U | U(=) | – | – |
| Called_AP_Invocation_Identifier | U | U (=) | – | – |
| Called_AE_Invocation_Identifier | U | U (=) | – | – |
| Calling_AP_Title | C | C (=) | – | – |
| Calling_AE_Qualifier | U | U (=) | – | – |
| Calling_AP_Invocation_Identifier | U | U (=) | – | – |
| Calling_AE_Invocation_Identifier | U | U (=) | – | – |
| Local_Or_Remote | – | – | – | O |
| Result | – | – | O | O |
| Failure_Type | – | – | O | O |
| Responding_AP_Title | – | – | C | C (=) |
| Responding_AE_Qualifier | – | – | U | U (=) |
| Responding_AP_Invocation_Identifier | – | – | U | U (=) |
| Responding_AE_Invocation_Identifier | – | – | U | U (=) |
| ACSE_Requirements | U | U (=) | U | U (=) |
| Security_Mechanism_Name | C | C (=) | C | C (=) |
| Calling_Authentication_Value | C | C (=) | – | – |
| Responding_Authentication_Value | – | – | C | C (=) |
| Implementation_Information | U | U (=) | U | U (=) |
| Proposed_xDLMS_Context | O | O (=) | – | – |
| Dedicated_Key | C | C (=) | – | – |
| Response_Allowed | C | C (=) | | |
| Proposed_DLMS_Version_Number | O | O (=) | – | – |
| Proposed_DLMS_Conformance | O | O (=) | – | – |
| Client_Max_Receive_PDU_Size | O | O (=) | – | – |
| Negotiated_xDLMS_Context | – | – | S | S (=) |
| Negotiated_DLMS_Version_Number | – | – | O | O (=) |
| Negotiated_DLMS_Conformance | – | – | O | O (=) |
| Server_Max_Receive_PDU_Size | – | – | O | O (=) |
| VAA_Name | | | O | O (=) |
| xDLMS_Initiate_Error | | | S | S (=) |
| User_Information | U | C (=) | – | – |
| Service_Class | O | O (=) | – | – |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Les paramètres de service de la primitive de service COSEM-OPEN.request, à l'exception des paramètres Protocol_Connection_Parameters, User_Information et, en fonction du profil de communications, Service_Class, sont contenus dans les champs de l'APDU AARQ envoyée par le client.

Les paramètres de service de la primitive de service COSEM-OPEN.response, à l'exception du paramètre Protocol_Connection_Parameters, sont contenus dans les champs de l'APDU AARE envoyée par le serveur.

Le service A-ASSOCIATE et les APDU AARQ et AARE sont spécifiés en 7.2. Des exemples de codage sont fournis à l'Annexe D et l'Annexe E.

Le paramètre Protocol_Connection_Parameters est obligatoire. Il contient toutes les informations nécessaires pour utiliser les couches du profil de communication, y compris l'identifiant (de protocole) du profil de communication et les adresses exigées. Il identifie les participants à l'AA. Les éléments de ce paramètre sont transmis aux entités qui gèrent les connexions aux couches inférieures, ainsi qu'aux couches inférieures, suivant le cas.

Le paramètre ACSE_Protocol_Version est facultatif. S'il est présent, la valeur par défaut doit être utilisée.

Le paramètre Application_Context_Name est obligatoire. Dans la primitive de demande, il contient la valeur proposée par le client. Dans la primitive de réponse, il contient la même valeur ou celle prise en charge par le serveur.

L'utilisation des paramètres Called_AP_Title, Called_AE_Qualifier, Called_AP_Invocation_Identifier et Called_AE_Invocation_Identifier est facultative. Leur utilisation n'est pas spécifiée dans la présente Norme internationale.

L'utilisation du paramètre Calling_AP_Title est conditionnelle. Lorsque le paramètre Application_Context_Name indique un contexte d'application utilisant le chiffrement, il peut contenir le titre système du client spécifié en 4.1.3.4.

L'utilisation du paramètre Calling_AE_Qualifier est conditionnelle. Lorsque le paramètre Application_Context_Name indique un contexte d'application utilisant le chiffrement, il peut contenir le certificat de clé publique de signature numérique du client.

L'utilisation du paramètre Calling_AP_Invocation_Identifier est facultative. Son utilisation n'est pas spécifiée dans le présent document.

L'utilisation du paramètre Calling_AE_Invocation_Identifier est facultative. Lorsqu'il est présent, il contient l'identificateur de l'utilisateur côté client de l'AA.

NOTE 1 Le mécanisme d'identification des utilisateurs du client est spécifié en 4.4.2 de l'IEC 62056-6-2:2021.

Le paramètre Local_or_Remote est obligatoire. Il indique l'origine de la primitive COSEM-OPEN.confirm. Il est défini sur Remote si la primitive a été générée après réception d'une APDU AARE du serveur. Il est défini sur Local si la primitive a été générée localement.

Le paramètre Result est obligatoire. Dans le cas de la confirmation distante, il indique si le serveur a accepté l'AA proposée ou non. Dans le cas de la confirmation locale, il indique si la pile de protocoles du côté client a accepté la demande ou non.

Le paramètre Failure_type est obligatoire. Dans le cas de la confirmation distante, il contient les informations fournies par le serveur. Dans le cas de la confirmation locale et négative, il indique la raison de l'échec.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

L'utilisation du paramètre Responding_AP_Title est conditionnelle. Lorsque le paramètre Application_Context_Name indique un contexte d'application utilisant le chiffrement, il peut contenir le titre système du serveur indiqué en 4.1.3.4.

L'utilisation du paramètre Responding_AE_Qualifier est conditionnelle. Lorsque le paramètre Application_Context_Name indique un contexte d'application utilisant le chiffrement, il peut contenir le certificat de clé publique de signature numérique du serveur.

L'utilisation des paramètres Responding_AP_Invocation_Identifier et Responding_AE_Invocation_Identifier est facultative. Leur utilisation n'est pas spécifiée dans la présente Norme internationale.

Le paramètre ACSE_Requirements est facultatif. Il permet de sélectionner l'unité (facultative) fonctionnelle d'authentification du service A-Associate pour l'association (voir 7.2.1).

La présence du paramètre ACSE_Requirements dépend du mécanisme d'authentification utilisé:

- dans le cas d'une authentification de niveau de sécurité le plus faible, il ne doit pas être présent; seule l'unité fonctionnelle Kernel est utilisée;
- dans le cas d'une authentification de niveau de sécurité faible (LLS), il doit être présent dans la primitive .request et peut être présent dans la primitive de service .response et doit indiquer l'authentification (bit 0 défini);
- dans le cas d'une authentification de niveau de sécurité élevé (HLS), il doit être présent dans les primitives de service .request et .response et doit indiquer l'authentification (bit 0 défini).

Le paramètre Security_Mechanism_Name est conditionnel. Il est présent uniquement si l'unité fonctionnelle d'authentification a été sélectionnée. S'il est présent, la primitive .request contient la valeur proposée par le client et la primitive .response contient la valeur exigée par le serveur, c'est-à-dire celle à utiliser par le client.

Le paramètre Calling_Authentication_Value et les paramètres Responding_Authentication_Value sont conditionnels. Ils sont présents uniquement si l'unité fonctionnelle d'authentification a été sélectionnée. Ils contiennent respectivement la valeur d'authentification du client/du serveur, appropriée au paramètre Security_Mechanism_Name.

Le paramètre Implementation_Information est facultatif. Son utilisation n'est pas spécifiée dans la présente Norme internationale.

Le paramètre Proposed_xDLMS_Context contient les éléments du contexte xDLMS proposé, qui est transporté par l'APDU xDLMS InitiateRequest, placé dans le champ des informations sur l'utilisateur de l'APDU AARQ.

L'élément Dedicated_Key est conditionnel. Il peut être présent uniquement lorsque le paramètre Application_Context_Name indique un contexte d'application utilisant le chiffrement. La clé dédiée est utilisée pour le chiffrement dédié des APDU xDLMS échangées dans l'AA établie.

Lorsque la clé dédiée est présente, l'APDU xDLMS InitateRequest doit être authentifiée et cryptée à l'aide de l'algorithme AES-GCM, de la clé globale de cryptage de monodiffusion et de la clé d'authentification (le cas échéant). En outre, elle doit être signée numérique si cela est exigé par la politique de sécurité.

L'APDU xDLMS InitiateRequest doit être protégée comme décrit ci-dessus, lorsque la clé dédiée n'est pas présente, mais il est nécessaire de protéger l'APDU RLRQ en incluant le xDLMS InitiateRequest protégé dans son champ d'informations sur l'utilisateur (user-information). Voir 6.3.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

L'utilisation de l'élément Response_Allowed est conditionnelle. Il indique si le serveur est autorisé à répondre avec une APDU AARE, c'est-à-dire si l'AA à établir est confirmée (Response_Allowed == TRUE) ou non confirmée (Response_Allowed == FALSE).

L'élément Proposed_DLMS_Version_Number contient le numéro de version DLMS® proposé (voir 4.2.4).

L'élément Proposed_DLMS_Conformance contient le bloc de conformité proposé (voir 7.3.1).

L'élément Client_Max_Receive_PDU_Size contient la longueur maximale des APDU xDLMS que le client peut recevoir (voir Tableau 2).

Si le contexte xDLMS proposé par le client est acceptable pour le serveur, la primitive de service de réponse doit contenir le paramètre Negotiated_xDLMS_Context. Il contient les éléments du contexte xDLMS négocié, contenus dans l'APDU xDLMS InitiateResponse et placés dans le champ des informations sur l'utilisateur de l'APDU AARE. Si l'APDU xDLMS InitiateRequest a été chiffrée, l'APDU xDLMS InitiateResponse doit également être chiffrée de la même manière.

L'élément Negotiated_DLMS_Version_Number contient le numéro de version DLMS® négocié. Voir 4.2.4.

L'élément Negotiated_DLMS_Conformance contient le bloc de conformité négocié. Voir 7.3.1.

L'élément Server_Max_Receive_PDU_Size contient la longueur maximale des APDU xDLMS que le serveur peut recevoir (voir Tableau 2).

L'élément VAA_name contient la valeur fictive de 0x0007 dans le cas du référencement LN et l'attribut base_name de l'objet Association actuel, 0xFA00, dans le cas du référencement SN.

Si le contexte xDLMS proposé par le client n'est pas acceptable pour le serveur, la primitive de service de réponse doit contenir le paramètre xDLMS_Initiate_Error. Il est contenu dans l'APDU ConfirmedServiceError, avec les éléments de diagnostic appropriés, et placé dans le champ d'informations sur l'utilisateur de l'APDU AARE.

Le paramètre User_Information est facultatif. S'il est présent, il doit être transmis à la couche de support, à condition qu'elle soit capable de le supporter. La primitive d'indication doit alors contenir les informations spécifiques à l'utilisateur, contenues dans la ou les couches inférieures de protocole de support (voir Annexe A).

NOTE 2 Le paramètre User_Information du service COSEM-OPEN est à ne pas confondre avec le champ d'informations sur l'utilisateur des APDU AARQ/AARE.

Le paramètre Service_Class est obligatoire. Il indique si le service doit être appelé de manière confirmée ou non confirmée. La gestion de ce paramètre peut dépendre du profil de communication (voir Annexe A).

*Utilisation*

Les séquences logiques possibles des primitives de service COSEM-OPEN sont représentées à la Figure 35:

- pour l'établissement d'une AA confirmée, avec succès ou non, élément a);
- pour l'établissement d'une AA non confirmée, élément b);
- dans le cas d'une AA préétablie ou d'une tentative infructueuse en raison d'une erreur locale, élément c).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La primitive .request est appelée par l'AP client pour demander l'établissement d'une AA confirmée ou non confirmée avec un AP serveur.

Avant l'appel de la primitive COSEM-OPEN.request, les couches physiques doivent être connectées. En fonction du profil de communication, l'appel de cette primitive peut également impliquer la connexion d'autres couches inférieures.

À réception de l'appel de demande, l'AL construit et envoie une APDU AARQ au serveur.

La primitive .indication est générée par l'AL du serveur à réception d'une APDU AARQ correctement formatée.

La primitive .response est appelée par l'AP serveur pour indiquer à l'AL si l'AA proposée est acceptée ou non. Elle est appelée uniquement si l'AA proposée est confirmée. L'AL construit ensuite une APDU AARE et l'envoie à son homologue, en incluant les paramètres de service reçus du AP.

La primitive .confirm est générée par l'AL du client pour indiquer à l'AP client si l'AA précédemment demandée est acceptée ou non:

- à distance, à réception d'une APDU AARE;
- localement, si l'AA demandée existe déjà. Cela inclut les AA préétablies;
- localement, si la primitive .request correspondante a été appelée avec Service_Class == Unconfirmed;
- localement, si l'AA demandée n'est pas autorisée;
- localement, si une erreur est détectée: paramètres manquants ou incorrects, échecs durant l'établissement des connexions aux couches inférieures demandées, connexion physique manquante, etc.

Le protocole d'établissement d'une AA est indiqué en 7.2.4. Les règles spécifiques au profil de communication sont indiquées à l'Annexe A.

### 6.3 Service COSEM-RELEASE

*Fonction*

La fonction du service COSEM-RELEASE consiste à libérer sans perte de données une AA existante. En fonction de la méthode d'appel de ce service, il utilise le service A-RELEASE de l'ACSE ou non.

*Sémantique des primitives de service*

Les primitives du service COSEM-RELEASE doivent spécifier des paramètres, comme indiqué dans le Tableau 37.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 37 – Paramètres de service des primitives de service COSEM-RELEASE**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Use_RLRQ_RLRE | U | C(=) | C(=) | - |
| Reason | U | U (=) | U | U (=) |
| Proposed_xDLMS_Context | C | C (=) | – | – |
| Negotiated_xDLMS_Context | – | – | C | C (=) |
| Local_Or_Remote | – | – | – | O |
| Result | – | – | O | O |
| Failure_Type | – | – | – | C |
| User_Information | U | C (=) | U | C (=) |

Le paramètre Use_RLRQ_RLRE de la primitive .request est facultatif. S'il est présent, sa valeur peut être FALSE (valeur par défaut) ou TRUE. Il indique s'il convient d'utiliser ou non le service A-RELEASE de l'ACSE, qui implique un échange d'APDU RLRQ/RLRE. Le service A-RELEASE et les APDU RLRQ/RLRE sont indiqués en 7.2. Le paramètre Use_RLRQ_RLRE de la primitive .response est conditionnel. S'il était présent dans la primitive .indication et si sa valeur était TRUE, il doit également être présent et sa valeur doit être TRUE. Sinon, il ne doit pas être présent ou sa valeur doit être FALSE.

Si la valeur du paramètre Use_RLRQ_RLRE est FALSE, l'AA peut être libérée en déconnectant la couche de support de l'AL.

Le paramètre Reason est facultatif. Il peut être présent uniquement si la valeur de Use_RLRQ_RLRE est TRUE. Il est contenu dans le champ Reason de l'APDU RLRQ/RLRE respectivement.

Lorsqu'il est utilisé dans la primitive .request, ce paramètre identifie le niveau général d'urgence de la demande. Il est défini sur l'une des valeurs symboliques suivantes:

- normal;
- urgent (non disponible dans DLMS®/COSEM);
- user defined.

Lorsqu'il est utilisé dans la primitive .response, ce paramètre identifie des informations sur le motif d'acceptation ou de rejet par l'accepteur de la demande de libération. À noter que dans DLMS®/COSEM le serveur ne peut pas rejeter la demande de libération. Il est défini sur l'une des valeurs symboliques suivantes:

- normal;
- not finished;
- user defined.

NOTE 1 La valeur "not finished" est utilisée dans la primitive .response lorsque l'accepteur est forcé de libérer l'association, mais souhaite donner un avertissement indiquant qu'il a des informations supplémentaires à envoyer ou à recevoir.

Le paramètre Proposed_xDLMS_Context est conditionnel. Il est présent uniquement si la valeur du paramètre Use_RLRQ_RLRE est TRUE et si l'AA à libérer a été établie avec un contexte d'application utilisant le chiffrement. Cette option permet de sécuriser le service COSEM-RELEASE et d'éviter ainsi une attaque par déni de service qui peut être exécutée par une libération non autorisée de l'AA.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Dans la primitive .request, le paramètre Proposed_xDLMS_Context doit être le même que dans la primitive de service COSEM-OPEN.request, une fois l'AA à libérer établie. Il est contenu dans l'APDU xDLMS InitiateRequest, protégé de la même manière que dans l'AARQ et placé dans le champ des informations sur l'utilisateur de l'APDU RLRQ.

Si l'APDU xDLMS InitiateRequest peut être déchiffrée avec succès, la primitive .response doit comporter le même paramètre Negotiated_xDLMS_Context que dans la primitive COSEM-OPEN.response. Il est contenu dans l'APDU xDLMS InitiateResponse, protégé de la même manière que dans l'AARE et placé dans le champ des informations sur l'utilisateur de l'APDU RLRE.

Sinon, l'APDU RLRQ est supprimée de manière silencieuse.

Le paramètre Local_or_Remote est obligatoire. Il indique l'origine de la primitive COSEM-RELEASE.confirm.

Il est défini sur Remote si:

- une APDU RLRE a été reçue de la part du serveur;
- une primitive de service de confirmation de la déconnexion a été reçue.

Il est défini sur Local si la primitive a été générée localement.

Le paramètre Result est obligatoire. Dans la primitive .response, il indique si l'AP serveur peut accepter la demande de libération de l'AA ou non. Étant donné que les serveurs ne peuvent pas refuser de telles demandes, il convient que sa valeur soit normalement SUCCESS, à moins que l'AA référencée n'existe pas.

Le paramètre Failure_Type est conditionnel. Il est présent si Result == ERROR. Dans ce cas, il indique la raison de l'échec. Il s'agit d'un paramètre généré localement du côté client.

Le paramètre User_Information de la primitive .request est facultatif. S'il est présent, il est transmis à la couche de support, à condition qu'elle soit capable de le supporter. La primitive .indication contient alors les informations spécifiques à l'utilisateur, contenues dans la ou les couches inférieures de protocole de support. De la même manière, il est facultatif dans la primitive .response. S'il est présent, il est transmis à la couche de support. Dans la primitive .confirm, il peut être présent uniquement lorsque le service est confirmé à distance. Il contient alors les informations spécifiques à l'utilisateur, transportées par la ou les couches inférieures de protocole de support.

NOTE 2 Le paramètre User_Information du service COSEM-RELEASE est à ne pas confondre avec le champ des informations sur l'utilisateur des APDU RLRQ/RLRE.

La spécification du contenu du paramètre User_Information ne relève pas du domaine d'application de la présente Norme internationale. Voir aussi Annexe A.

*Utilisation*

Des exemples de séquences logiques des primitives de service COSEM-RELEASE sont représentés à la Figure 35:

- pour une libération réussie d'une AA confirmée, élément a);
- pour une libération d'une AA non confirmée; élément b), et
- pour une tentative échouée en raison d'une erreur locale, élément c).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

L'utilisation du service COSEM-RELEASE dépend de la valeur du paramètre Use_RLRQ_RLRE. Lorsqu'il est appelé avec Use_RLRQ_RLRE == TRUE, le service est basé sur le service A-RELEASE de l'ACSE. Sinon, l'appel du service entraîne la déconnexion de la couche de support.

La primitive .request est appelée par l'AP client pour demander la libération d'une AA confirmée ou non confirmée avec un AP serveur. À réception de l'appel de la demande avec Use_RLRQ_RLRE == TRUE, l'AL construit et envoie une APDU RLRQ au serveur. Sinon, il envoie une primitive XX-DISCONNECT.request (où XX désigne la couche inférieure de protocole de support).

La primitive .indication est générée par l'AL du serveur si:

- une APDU RLRQ est reçue. Si une erreur de déchiffrement se produit, l'APDU RLRQ est supprimée de manière silencieuse (aucune primitive .indication n'est générée);
- une primitive XX-DISCONNECT.request est reçue.

La primitive .response est appelée par l'AP serveur, mais uniquement si l'AA à libérer est confirmée. À noter que l'AP serveur ne peut pas refuser cette demande. À réception de la primitive de service .response, l'AL du serveur:

- envoie une APDU RLRE si le paramètre Use_RLRQ_RLRE est TRUE;
- envoie une primitive XX-DISCONNECT.response.

La primitive .confirm est générée par l'AL du client pour indiquer à l'AP client si la libération demandée de l'AA est acceptée ou non:

- à distance, à réception d'une primitive XX-DISCONNECT.cnf. La couche de support est déconnectée;
- à distance, à réception d'une APDU RLRE. La couche de support n'est pas déconnectée;
- localement, à expiration d'un délai d'attente pour une APDU RLRE;
- localement, lors de l'envoi d'une APDU RLRQ pour libérer une AA non confirmée;
- localement, lors de la détection d'une erreur locale: paramètres manquants ou incorrects, échec de communication au niveau de la couche de protocole inférieure, etc.

Si l'APDU RLRE reçue contient une APDU xDLMS InitiateResponse chiffrée, mais ne peut pas être déchiffrée, l'APDU RLRE doit être supprimée. C'est au client de gérer la situation.

Le protocole de libération d'une AA est indiqué en 7.2.5. Les règles spécifiques au profil de communication sont indiquées à l'Annexe A. Voir aussi 7.2.1.

### 6.4 Service COSEM-ABORT

*Fonction*

La fonction du service COSEM-ABORT est d'indiquer une déconnexion non sollicitée de la couche de support.

*Sémantique*

Les primitives du service COSEM-ABORT doivent spécifier les paramètres indiqués dans le Tableau 38.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 38 – Paramètres de service des primitives de service COSEM-ABORT**

| | .indication |
|---|---|
| Diagnostics | U |

Le paramètre Diagnostics est facultatif. Il doit indiquer la raison possible de la déconnexion et peut également contenir des informations dépendant de la couche de protocole inférieure. La spécification du contenu de ce paramètre ne relève pas du domaine d'application du présent document.

*Utilisation*

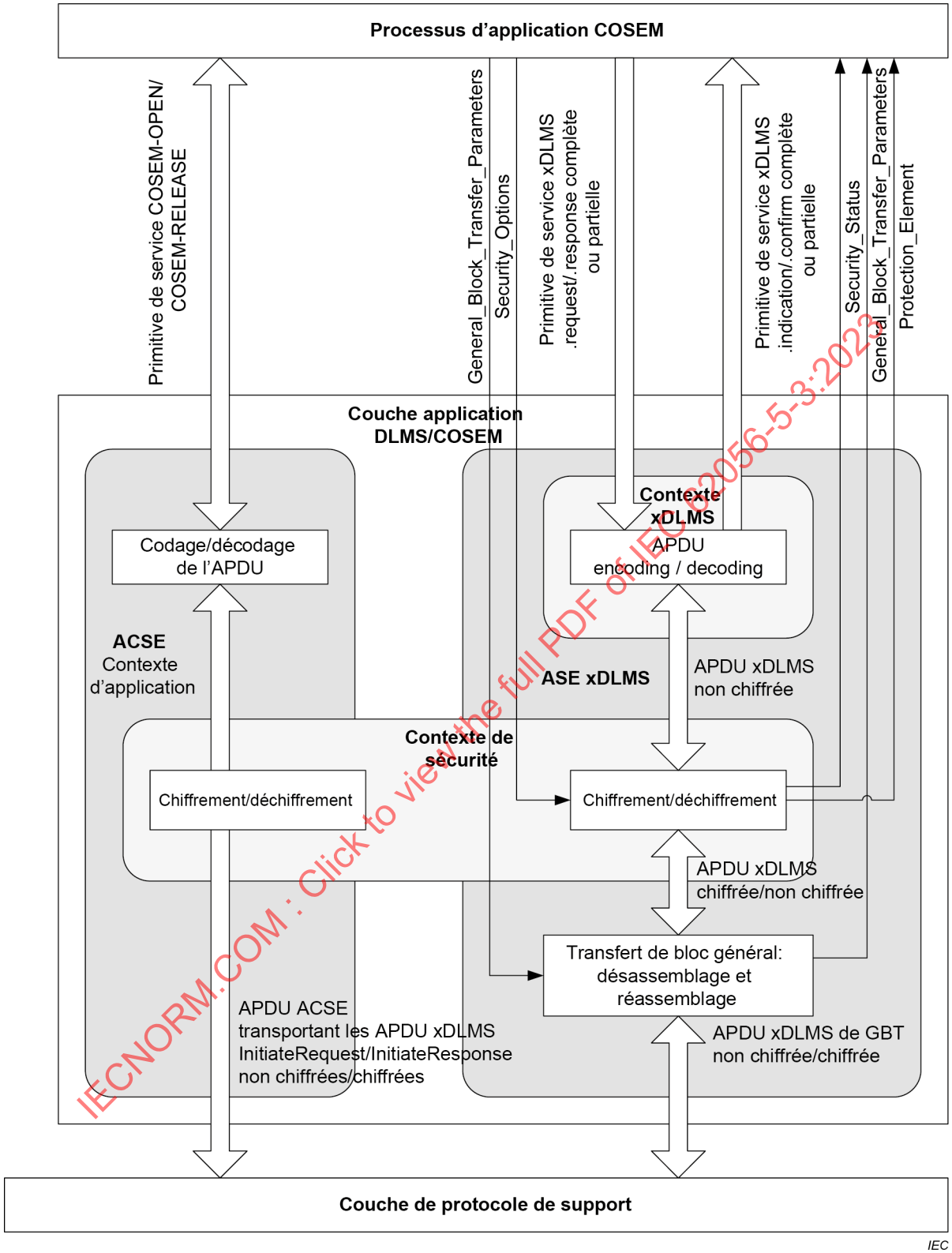
La primitive COSEM-ABORT.indication est générée localement du côté client et du côté serveur pour indiquer à l'AP COSEM qu'une connexion à la couche inférieure a été fermée de manière non sollicitée. L'origine d'un tel événement peut être un événement externe (par exemple, la ligne physique est rompue) ou une action d'un AP gestionnaire des connexions à la couche de support, présent dans certains profils, lorsque la connexion à la couche de support n'est pas gérée par l'AL COSEM. Cela doit entraîner l'annulation par les AP COSEM de toutes les AA existantes, sauf celles préétablies du côté serveur.

Le protocole du service COSEM-ABORT est spécifié en 7.2.5.3.

## 6.5 Paramètres de protection et de transfert de bloc général

Pour contrôler la protection cryptographique des APDU xDLMS et le mécanisme GBT, des paramètres de service supplémentaires sont échangés entre l'AL et l'AP, comme représenté à la Figure 36 et dans le Tableau 39.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

NOTE Pour les services initiés par le client, les primitives de service sont .request, .indication, .response et .confirm. Pour les services non sollicités (initiés par le serveur), les primitives de service sont .request et .indication.

**Figure 36 – Paramètres de service supplémentaires pour contrôler la protection cryptographique et le GBT**

**Tableau 39 – Paramètres de service supplémentaires**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Additional_Service_Parameters | U | – | U (=) | |
| Invocation_Type | O | – | O (=) | – |
| Security_Options | C | – | C (=) | – |
| General_Block_Transfer_Parameters | C | – | C (=) | – |
| Block_Transfer_Streaming | O | – | O (=) | – |
| Block_Transfer_Window | O | – | O (=) | – |
| Service_Parameters | O | – | O (=) | – |
| Additional_Service_Parameters | – | U | – | U (=) |
| Invocation_Type | – | U | – | U (=) |
| Security_Status | – | C | – | C (=) |
| General_Block_Transfer_Parameters | – | C | – | C (=) |
| Block_Transfer_Window | – | O | – | O (=) |
| Service_Parameters | – | O | – | O (=) |
| Protection_Element | – | C | – | C (=) |
| NOTE Les primitives de service disponibles dépendent du type de service. | | | | |

Les paramètres Additional_Service_Parameters ne sont présents que si le chiffrement ou le GBT est utilisé.

Le paramètre Invocation_Type est obligatoire: il indique si l'appel de service est complet ou partiel. Valeurs possibles: COMPLETE, FIRST-PART, ONE-PART et LAST-PART.

NOTE 1 Les appels partiels de service peuvent être utiles lorsque les paramètres de service sont longs. Cependant, il n'y a aucune relation directe entre les appels partiels de service et les APDU de transfert de bloc général (APDU GBT).

Le paramètre Security_Options est conditionnel: il est présent uniquement si le contexte d'application est chiffré, la primitive de service .request/.response est tenue d'être chiffrée et Invocation_Type = COMPLETE ou FIRST-PART. Il détermine la protection à appliquer par l'AL. Voir aussi Tableau 40, 7.3.13 et Figure 62.

Le paramètre General_Block_Transfer_Parameters est conditionnel: il est présent uniquement si le transfert de bloc général (GBT) est utilisé et Invocation_Type = COMPLETE ou FIRST-PART. Il fournit des informations sur les capacités de diffusion en flux du GBT:

- le paramètre Block_Transfer_Streaming n'est présent que dans les primitives de service .request et .response. Il est transmis par l'AP vers l'AL pour indiquer si l'AL est autorisée à envoyer des APDU de transfert de bloc général par flux (TRUE) ou non (FALSE);
- le paramètre Block_Transfer_Window indique la dimension de la fenêtre prise en charge, c'est-à-dire le nombre maximal de blocs qui peuvent être reçus dans une fenêtre.

Le processus de flux lui-même est géré par l'AL. Voir 7.3.13.

Les paramètres Service_Parameters sont obligatoires: ils incluent les paramètres des appels de service xDLMS. Si Invocation_Type != COMPLETE, alors le type d'appel comporte une partie des paramètres de service.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le paramètre Security_Status est conditionnel: il est présent uniquement lorsque la protection cryptographique a été appliquée. Il contient des informations sur la protection qui a été vérifiée/supprimée par l'AL. Il peut être présent dans tous les types d'appels de service. Voir Tableau 40.

Le paramètre Protection_Element est conditionnel: il n'est présent que si l'APDU a été authentifiée ou signée. Voir Tableau 40.

**Tableau 40 – Paramètres de sécurité**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Security_Options | C | – | C (=) | – |
| Security_Options_Element {Security_Options_Element} | O | – | O (=) | |
| Security_Protection_Type | O | – | O (=) | – |
| Glo_Ciphering | S | – | S (=) | – |
| Ded_Ciphering | S | – | S (=) | – |
| General_Glo_Ciphering | S | – | S (=) | – |
| General_Ded_Ciphering | S | – | S (=) | – |
| General_Ciphering | S | – | S (=) | – |
| General_Signing | S | – | S (=) | – |
| Avec General_Glo_Ciphering et General_Ded_Ciphering | | | | |
| System_Title | U | – | U (=) | – |
| Avec General_Ciphering et General_Signing | | | | |
| Transaction_Id | U | – | U (=) | – |
| Originator_System_Title | U | – | U (=) | – |
| Recipient_System_Title | U | – | U (=) | – |
| Date_Time | U | – | U (=) | – |
| Other_Information | U | – | U (=) | – |
| Avec General_Ciphering | | | | |
| Key_Info_Options | C | – | C (=) | – |
| Identified_Key_Options | S | – | S (=) | – |
| Wrapped_Key_Options | S | – | S (=) | – |
| Agreed_Key_Options | S | – | S (=) | – |
| Avec Glo_Ciphering, Ded_Ciphering, General_Glo_Ciphering, General_Ded_Ciphering, General_Ciphering | | | | |
| Security_Control | O | – | O (=) | – |
| Security_Status | – | C | – | C (=) |
| Security_Status_Element {Security_Status_Element} | – | O | – | O (=) |
| Security_Protection_Type | | O | | O (=) |
| Glo_Ciphering | – | S | – | S (=) |
| Ded_Ciphering | – | S | – | S (=) |
| General_Glo_Ciphering | – | S | – | S (=) |
| General_Ded_Ciphering | – | S | – | S (=) |
| General_Ciphering | – | S | – | S (=) |
| General_Signing | – | S | – | S (=) |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Avec General_Glo_Ciphering et General_Ded_Ciphering | | | | |
| System_Title | – | U | – | U (=) |
| Avec General_Ciphering et General_Signing | | | | |
| Transaction_Id | – | U | – | U (=) |
| Originator_System_Title | – | U | – | U (=) |
| Recipient_System_Title | – | U | – | U (=) |
| Date_Time | – | U | – | U (=) |
| Other_Information | – | U | – | U (=) |
| Avec General_Ciphering | | | | |
| Key_Info_Status | – | C | – | C (=) |
| Identified_Key_Status | – | S | – | S (=) |
| Wrapped_Key_Status | – | S | – | S (=) |
| Agreed_Key_Status | – | S | – | S (=) |
| Avec Glo_Ciphering, Ded_Ciphering, General_Ded_Ciphering, General_Glo_Ciphering, General_Ciphering | | | | |
| Security_Control | – | O | – | O (=) |
| L'élément de protection est présent lorsque l'authentification ou la signature numérique est appliquée. | | | | |
| Protection_Element {Protection_Element} | – | C | – | C (=) |
| Invocation_Counter | – | C | – | C (=) |
| Authentication_Tag | – | C | – | C (=) |
| Signature | – | C | – | C (=) |

Le paramètre Security_Options contient un paramètre Security_Options_Element pour chaque type de protection à appliquer. De même, le paramètre Security_Status contient un paramètre Security_Status_Element pour chaque type de protection appliquée. Voir aussi 5.7.3.

Les paramètres Security_Options_Element et Security_Status_Element comprennent les sous-paramètres suivants:

- le sous-paramètre Security_Protection_Type est obligatoire: il identifie l'APDU chiffrée à utiliser (voir Tableau 41);
- le sous-paramètre System_Title est facultatif. Lorsqu'il est présent, il contient le titre système de l'expéditeur. Il peut être présent uniquement avec General_Glo_Ciphering et General_Ded_Ciphering.

NOTE 2 L'inclusion du titre système de l'expéditeur a pour but de permettre à l'autre partie de construire le vecteur d'initialisation lorsque le titre système n'a pas été échangé pendant le processus d'enregistrement spécifique au support ou pendant l'échange AARQ / AARE.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 41 – APDU utilisées avec les types de protections de sécurité (Security_Protection_Type)**

| Security_Protection_Type | APDU |
|---|---|
| Glo_Ciphering | Glo-ciphering spécifique au service |
| Ded_Ciphering | Ded-ciphering spécifique au service |
| General_Glo_Ciphering | general-glo-ciphering |
| General_Ded_Ciphering | general-ded-ciphering |
| General_Ciphering | general-ciphering |
| General_Signing | general-signing |
| Voir aussi Tableau 26. | |

Les cinq paramètres suivants sont présents de manière facultative avec General_Ciphering et General_Signing:

- Transaction_Id: identifie la transaction entre deux parties;
- Originator_System_Title: indique le titre système de l'émetteur de l'APDU protégée;
- Recipient_System_Title: indique le titre système du destinataire, c'est-à-dire l'entité qui vérifie/supprime la protection qui a été appliquée à l'APDU. En cas de diffusion, Recipient_System_Title doit être une chaîne vide;
- le paramètre Date_Time est facultatif. Lorsqu'il est présent, il indique la date et l'heure de l'appel de la primitive de service .request/.response. Sauf indication contraire dans une spécification auxiliaire spécifique à un projet donné, le paramètre Date_Time dans la réponse doit être présent s'il était présent dans la demande et ne doit pas être présent dans la réponse s'il ne l'était pas dans la demande;

  Si l'un quelconque des quatre paramètres susmentionnés n'est pas utilisé, une chaîne d'octets de longueur nulle doit être incluse.
- le paramètre Other_Information est facultatif. Lorsqu'il est présent, il contient des informations supplémentaires concernant la protection. Son contenu peut être spécifié dans les spécifications auxiliaires spécifiques à un projet donné.

Le paramètre Key_Info_Options est conditionnel: lorsque la protection est à appliquer, il contient les informations relatives à la clé symétrique utilisée par l'émetteur/qui est à utiliser par le destinataire. Les informations relatives à la clé sont envoyées/reçues dans le cadre de l'APDU chiffrée:

- Identified_Key_Options (voir 5.5.3): il peut être utilisé lorsque les partenaires partagent la clé; il peut s'agir de la clé globale de cryptage de monodiffusion ou de la clé globale de cryptage de diffusion;
- Wrapped_Key_Options (voir 5.5.4): dans ce cas, une clé enveloppée est envoyée;
- Agreed_Key_Options (voir 5.5.5): dans ce cas, les partenaires utilisent un mécanisme d'agrément de clé Diffie-Hellman pour convenir de la clé;

Security_Control: contient l'octet de contrôle de sécurité (voir Tableau 27).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le paramètre Protection_Element est conditionnel: il doit être présent si l'APDU a été authentifiée ou signée numériquement. Il peut être présent dans tous les types d'appels de service, mais il peut être vide s'il n'est pas encore disponible (ceci peut se produire lorsque le transfert de bloc général est utilisé). Il contient:

- lorsque General_Ciphering est utilisé, le Invocation_Counter, contenant le champ d'appel du vecteur d'initialisation (voir 5.3.3.7.3);
- lorsque l'APDU a été authentifiée, la balise d'authentification;
- lorsque l'APDU a été signée, la signature numérique.

### 6.6 Service GET

*Fonction*

Le service GET est utilisé avec le référencement LN. Il peut être appelé de manière confirmée ou non confirmée. Sa fonction est de lire la valeur d'un ou de plusieurs attributs d'objet d'interface COSEM. Le résultat peut être envoyé dans une seule réponse ou, s'il est trop long, dans plusieurs réponses, avec un transfert de blocs.

*Sémantique*

Les primitives du service GET doivent fournir les paramètres indiqués dans le Tableau 42.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 42 – Paramètres de service du service GET**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Invoke_Id | O | O (=) | O (=) | O (=) |
| Priority | O | O (=) | O (=) | O (=) |
| Service_Class | O | O (=) | O (=) | O (=) |
| Request_Type | O | O (=) | – | – |
| COSEM_Attribute_Descriptor<br>{ COSEM_Attribute_Descriptor } | C | C (=) | – | – |
| COSEM_Class_Id | O | O (=) | | |
| COSEM_Object_Instance_Id | O | O (=) | | |
| COSEM_Object_Attribute_Id | O | O (=) | | |
| Access_Selection_Parameters | U | U (=) | | |
| Access_Selector | O | O (=) | | |
| Access_Parameters | O | O (=) | | |
| Block_Number | C | C (=) | – | – |
| Response_Type | – | – | O | O (=) |
| Result | – | – | O | O (=) |
| Get_Data_Result { Get_Data_Result } | – | – | S | S (=) |
| Data | | | S | S (=) |
| Data_Access_Result | | | S | S (=) |
| DataBlock_G | – | – | S | S (=) |
| Last_Block | | | O | O (=) |
| Block_Number | | | O | O (=) |
| Result | | | O | O (=) |
| Raw_Data | | | S | S (=) |
| Data_Access_Result | | | S | S (=) |
| NOTE Pour les paramètres de sécurité, voir Tableau 40. | | | | |

Le paramètre Invoke_Id identifie l'instance de l'appel de service.

Le paramètre Priority indique le niveau de priorité associé à l'instance de l'appel de service: normale (FALSE) ou élevée (TRUE).

Le paramètre Service_Class indique si le service est confirmé ou non confirmé. La gestion de ce paramètre dépend du profil de communication (voir Annexe A).

L'utilisation des paramètres Request_Type et Response_Type est présentée dans le Tableau 43.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 43 – Types de demandes et de réponses du service GET**

| Type de demande | | Type de réponse | |
|---|---|---|---|
| NORMAL | La valeur d'un attribut unique est demandée. | NORMAL | Le résultat complet est fourni. |
| | | ONE-BLOCK | Un bloc du résultat est fourni. |
| NEXT | Le bloc de données suivant est demandé. | ONE-BLOCK | Voir ci-dessus. |
| | | LAST-BLOCK | Le dernier bloc du résultat est fourni. |
| WITH-LIST | La valeur d'une liste d'attributs est demandée. | WITH-LIST | Le résultat complet est fourni. |
| | | ONE-BLOCK | Voir ci-dessus. |
| NOTE Le même paramètre Response_Type peut être spécifié plusieurs fois, pour indiquer les réponses possibles pour chaque type de demande. | | | |

Le paramètre COSEM_Attribute_Descriptor se rapporte à un attribut d'objet COSEM. Il est présent lorsque Request_Type == NORMAL ou WITH-LIST. Il s'agit d'un paramètre composite:

- le doublet (COSEM_Class_Id, COSEM_Object_Instance_Id) fait référence de façon non ambiguë à une seule instance d'objet COSEM;
- l'élément COSEM_Object_Attribute_Id identifie le ou les attributs de l'instance d'objet. COSEM_Object_Attribute_Id == 0 fait référence à tous les attributs publics de l'objet (fonction Attribute_0; voir 4.2.4.3.7);
- le paramètre Access_Selection_Parameters est uniquement présent lorsque COSEM_Object_Attribute_Id != 0 et si l'accès sélectif à l'attribut donné est disponible (voir 4.2.4.3.5). Les sous-paramètres Access_Selector et Access_Parameters sont définis dans les définitions d'objet d'interface COSEM (voir IEC 62056-6-2:2021).

Une primitive de service GET-REQUEST-NORMAL contient un descripteur d'attribut COSEM unique. Une primitive de service GET-REQUEST-WITH-LIST contient une liste de descripteurs d'attributs COSEM; leur nombre est limité par le paramètre server-max-receive-pdu-size: une primitive de service GET.request doit toujours être contenue dans une seule APDU.

Le paramètre Block_Number est utilisé dans la primitive de service GET-REQUEST-NEXT. Il contient le numéro du dernier bloc de données reçu correctement.

Si la forme codée de la réponse tient dans une seule APDU, le paramètre Result est de type Get_Data_Result:

- le choix Data contient la valeur de l'attribut au moment de l'accès;
- le choix Data_Access_Result indique la raison de l'échec de lecture de cet attribut.

Une primitive de service GET-RESPONSE-NORMAL contient un seul paramètre Get_Data_Result. Une primitive de service GET-RESPONSE-WITH-LIST contient une liste de paramètres Get_Data_Result; leur nombre et leur ordre doivent être identiques à ceux des paramètres COSEM_Attribute_Descriptor dans la demande.

Si COSEM_Object_Attribute_Id == 0 (Attribute_0), le paramètre Data doit être une structure contenant la valeur de tous les attributs publics dans l'ordre de leur apparition dans la spécification d'objet donnée. Pour les attributs auxquels aucun droit d'accès n'est octroyé dans l'AA donnée ou qui ne peuvent pas être accessibles pour une quelconque autre raison, la valeur NULL doit être renvoyée.

Si la forme codée de la réponse ne tient pas dans une seule APDU, elle peut être déplacée en blocs de données à l'aide du mécanisme spécifique au service ou du mécanisme de transfert de bloc général.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Si le mécanisme de transfert de blocs spécifique au service est utilisé, le paramètre Result est de type DataBlock_G. Il contient les informations de contrôle de transfert de blocs et l'élément raw-data:

- l'élément Last_Block indique si le bloc actuel est le dernier (TRUE) ou non (FALSE);
- l'élément Block_Number contient le numéro du bloc réel envoyé;
- l'élément Result (interne) contient le paramètre Raw_Data ou Data_Access_Result. Dans cette formule:
  - si la valeur d'un attribut unique a été demandée, Raw_Data contient une partie de la valeur de l'attribut (Data). Si la valeur Data ne peut pas être fournie, la réponse doit être de type GET-RESPONSE-NORMAL avec Data_Access_Result;
  - si la valeur d'une liste d'attributs a été demandée, Raw_Data contient une partie de la liste de Get_Data_Results (soit Data, soit Data_Access_Result pour chaque attribut);
  - si les données brutes ne peuvent pas être fournies, Data_Access_Result doit apporter la raison. Pour les codes d'erreur, voir 7.3.3.

*Utilisation*

Des exemples de séquences logiques des primitives de service GET sont représentés à la Figure 35:

- pour un GET réussi confirmé, élément a);
- pour un GET non confirmé, élément d);
- pour une tentative échouée en raison d'une erreur locale, élément c).

La primitive GET.request est appelée par l'AP client pour lire la valeur d'un attribut ou de tous les attributs d'un ou de plusieurs objets COSEM du AP serveur. La première demande doit toujours être définie sur le type Request_Type == NORMAL ou WITH-LIST. Une primitive de service GET-REQUEST-NEXT est appelée uniquement lorsque le serveur n'a pas pu livrer l'ensemble des données dans une seule réponse, c'est-à-dire après réception d'une primitive .confirm de type Response_Type == ONE-BLOCK. À réception de la primitive .request, l'AL du client génère l'APDU Get-Request appropriée au type de demande et l'envoie au serveur.

La primitive GET.indication est générée par l'AL du serveur à réception de l'APDU Get-Request.

La primitive GET.response est appelée par l'AP serveur, si la classe Service_Class == Confirmed, afin d'envoyer une réponse à une primitive .indication reçue. Si l'ensemble des données demandées entre dans une seule APDU, la primitive .response est appelée avec le type Response_Type == NORMAL ou WITH-LIST, selon le cas. Sinon, elle est appelée avec Response_Type == ONE-BLOCK, puis avec LAST-BLOCK.

La primitive GET.confirm est générée par l'AL du client pour accuser réception d'une APDU Get-Response.

Le protocole du service GET est spécifié en 7.3.3.

## 6.7 Service SET

*Fonction*

Le service SET est utilisé avec le référencement LN. Il peut être appelé de manière confirmée ou non confirmée. Sa fonction est d'écrire la valeur d'un ou de plusieurs attributs d'objets d'interface COSEM. Les données à écrire peuvent être envoyées dans une seule demande ou, si elles sont trop longues pour être contenues dans une seule requête, dans plusieurs requêtes, grâce au transfert de blocs.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

*Sémantique*

Les primitives de service SET doivent fournir les paramètres indiqués dans le Tableau 44.

**Tableau 44 – Paramètres de service du service SET**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Invoke_Id | O | O (=) | O (=) | O (=) |
| Priority | O | O (=) | O (=) | O (=) |
| Service_Class | O | O (=) | O (=) | O (=) |
| Request_Type | O | O (=) | – | – |
| COSEM_Attribute_Descriptor<br>{ COSEM_Attribute_Descriptor } | C | C (=) | – | – |
| COSEM_Class_Id | O | O (=) | | |
| COSEM_Object_Instance_Id | O | O (=) | | |
| COSEM_Object_Attribute_Id | O | O (=) | | |
| Access_Selection_Parameters | U | U (=) | | |
| Access_Selector | O | O (=) | | |
| Access_Parameters | O | O (=) | | |
| Data {Data } | C | C (=) | – | – |
| DataBlock_SA | C | C (=) | – | – |
| Last_Block | O | O (=) | | |
| Block_Number | O | O (=) | | |
| Raw_Data | O | O (=) | | |
| Response_Type | – | – | O | O (=) |
| Result { Result } | – | – | C | C (=) |
| Block_Number | – | – | C | C (=) |
| NOTE Pour les paramètres de sécurité, voir Tableau 40. | | | | |

Le paramètre Invoke_Id identifie l'instance de l'appel de service.

Le paramètre Priority indique le niveau de priorité associé à l'instance de l'appel de service: normale (FALSE) ou élevée (TRUE).

Le paramètre Service_Class indique si le service est confirmé ou non confirmé. La gestion de ce paramètre dépend du profil de communication (voir Annexe A).

L'utilisation des paramètres Request_Type et Response_Type est présentée dans le Tableau 45.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 45 – Types de demandes et de réponses du service SET**

| Type de demande | | Type de réponse | |
|---|---|---|---|
| NORMAL | La référence d'un attribut unique et l'ensemble des données à écrire sont envoyés. | NORMAL | Le résultat est fourni. |
| FIRST-BLOCK | La référence d'un attribut unique et le premier bloc de données à écrire sont envoyés. | ACK-BLOCK | La bonne réception du bloc est confirmée. |
| ONE-BLOCK | L'un des blocs de données à écrire est envoyé. | | |
| LAST-BLOCK | Le dernier bloc de données à écrire est envoyé. | LAST-BLOCK | La bonne réception du dernier bloc est confirmée et le résultat est fourni. |
| | | LAST-BLOCK-WITH-LIST | La bonne réception du dernier bloc est confirmée et la liste des résultats est fournie. |
| WITH-LIST | La référence d'une liste d'attributs et l'ensemble des données à écrire sont envoyés. | WITH-LIST | La liste des résultats est fournie. |
| FIRST-BLOCK-WITH-LIST | La référence d'une liste d'attributs et le premier bloc des données à écrire sont envoyés. | ACK-BLOCK | La bonne réception du bloc est confirmée. |
| NOTE Le même type de réponse (Response_Type) peut figurer plusieurs fois, pour indiquer les réponses possibles pour chaque demande. | | | |

Le paramètre COSEM_Attribute_Descriptor se rapporte à un attribut d'objet COSEM. Il est présent lorsque Request_Type == NORMAL, FIRST-BLOCK, WITH-LIST et FIRST-BLOCK-WITH-LIST. Il s'agit d'un paramètre composite:

- le doublet (COSEM_Class_Id, COSEM_Object_Instance_Id) fait référence de façon non ambiguë à une seule instance d'objet COSEM;
- COSEM_Object_Attribute_Id identifie le ou les attributs de l'instance d'objet. COSEM_Object_Attribute_Id == 0 fait référence à tous les attributs publics de l'objet (fonction Attribute_0; voir 4.2.4.3.7);
- le paramètre Access_Selection_Parameters est uniquement présent lorsque COSEM_Object_Attribute_Id != 0 et si l'accès sélectif à l'attribut donné est disponible (voir 4.2.4.3.5). Les sous-paramètres Access_Selector et Access_Parameters sont définis dans les définitions d'objet d'interface COSEM de l'IEC 62056-6-2:2021.

Une primitive de service SET-REQUEST-NORMAL ou SET-REQUEST-WITH-FIRST-BLOCK contient un seul descripteur d'attribut COSEM. Une primitive de service SET-REQUEST-WITH-LIST ou SET-REQUEST-FIRST-BLOCK-WITH-LIST contient une liste de descripteurs d'attributs COSEM; leur nombre est limité par le paramètre server-max-receive-pdu-size: tous les descripteurs d'attributs COSEM ainsi qu'une partie des données à écrire doivent être contenus dans une seule APDU.

Le paramètre Data contient les données nécessaires pour écrire la valeur des attributs référencés. Le nombre et l'ordre des paramètres Data doivent être identiques à ceux des paramètres COSEM_Attribute_Descriptor.

Si COSEM_Object_Attribute_Id == 0 (Attribute_0), le paramètre Data envoyé doit être une structure des attributs publics respectant leur ordre d'apparition dans la spécification d'objet donnée et contenant pour chacun des attributs une valeur ou la valeur NULL, indiquant que l'attribut donné n'a pas besoin d'être défini.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Si la forme codée du paramètre Data ne tient pas dans une seule APDU, elle peut être transportée en blocs à l'aide du mécanisme spécifique au service ou du mécanisme de transfert de bloc général.

Si le mécanisme de transfert de blocs spécifique au service est utilisé, le paramètre DataBlock_SA contient les informations de contrôle de transfert de blocs et l'élément raw-data:

- l'élément Last_Block indique si le bloc actuel est le dernier (TRUE) ou non (FALSE);
- l'élément Block_Number contient le numéro du bloc réel envoyé;
- l'élément Raw_Data contient une partie des données à écrire.

Les paramètres Result figurent dans la primitive .response lorsque Response_Type != ACK-BLOCK. Leur nombre et leur ordre doivent être identiques à ceux des paramètres COSEM_Attribute_Descriptor de la demande. Chaque paramètre Result doit contenir l'information "réussite" (success) ou la raison de l'échec d'écriture de l'attribut référencé (Data_Access_Result). Dans la primitive .request, lorsque COSEM_Object_Attribute_Id == 0 (Attribute_0), le paramètre Result doit contenir une liste de résultats, soit l'information "réussite" (success), soit la raison de l'échec d'écriture de l'attribut (Data_Access_Result), pour chaque attribut public, dans l'ordre où ils apparaissent dans la spécification d'objet correspondante.

Le paramètre Block_Number doit être présent lorsque le paramètre Response_Type == ACK-BLOCK, LAST-BLOCK ou LAST-BLOCK-WITH-LIST. Il contient le numéro du dernier bloc de données reçu correctement.

*Utilisation*

Un exemple de séquence logique des primitives de service SET est représenté à la Figure 35:

- pour un SET réussi confirmé, élément a);
- pour un SET non confirmé, élément d);
- pour une tentative échouée en raison d'une erreur locale, élément c).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La primitive SET.request est appelée par l'AP client pour écrire la valeur d'un ou de tous les attributs d'un ou de plusieurs objets COSEM du AP serveur. Si l'ensemble des données à envoyer tient dans une seule APDU, la primitive .request doit être appelée avec Request_Type == NORMAL ou WITH-LIST, selon le cas. Sinon, elle doit être appelée avec Request_Type == FIRST-BLOCK ou FIRST-BLOCK-WITH-LIST, puis avec Request_Type == ONE-BLOCK et enfin avec LAST-BLOCK, selon le cas. À réception de la primitive .request, l'AL du client génère l'APDU Set-Request correspondant au Request_Type et l'envoie au serveur.

La primitive SET.indication est générée par l'AL du serveur à réception de l'APDU Set-Request.

La primitive SET.response est appelée par l'AP serveur, si Service_Class == Confirmed, afin d'envoyer une réponse à une primitive .indication reçue. Si les données ont été envoyées dans une seule APDU, la primitive .response est appelée avec Response_Type == NORMAL ou WITH-LIST, selon le cas. Sinon, elle est appelée avec Response_Type == ACK-BLOCK, puis avec LAST-BLOCK ou LAST-BLOCK-WITH-LIST, selon le cas.

La primitive SET.confirm est générée par l'AL du client pour accuser réception d'une APDU Set-Response.

Le protocole du service SET est spécifié en 7.3.4.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 6.8 Service ACTION

*Fonction*

Le service ACTION est utilisé avec le référencement LN. Il peut être appelé de manière confirmée ou non confirmée. Sa fonction est d'appeler une ou plusieurs méthodes d'objets d'interface COSEM. Il se décompose en deux phases:

- lors de la première phase, le client envoie la ou les références de la ou des méthodes à appeler, avec les paramètres d'appel de méthode nécessaires;
- lors de la deuxième phase, après l'appel des méthodes, le serveur renvoie le résultat et les paramètres de retour générés par l'appel de la ou des méthodes, le cas échéant.

Si les paramètres d'appel de méthode sont trop longs pour être contenus dans une seule demande, ils sont envoyés en plusieurs demandes (transfert de blocs du client vers le serveur). Si les paramètres de résultat et de retour sont trop longs pour être contenus dans une seule réponse, ils sont renvoyés en plusieurs réponses (transfert de blocs du serveur vers le client).

*Sémantique*

Les primitives du service ACTION doivent fournir les paramètres indiqués dans le Tableau 46.

**Tableau 46 – Paramètres de service du service ACTION**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Invoke_Id | O | O (=) | O (=) | O (=) |
| Priority | O | O (=) | O (=) | O (=) |
| Service_Class | O | O (=) | O (=) | O (=) |
| Request_Type | O | O (=) | – | – |
| COSEM_Method_Descriptor<br>{ COSEM_Method_Descriptor } | C | C (=) | – | – |
| COSEM_Class_Id | O | O (=) | | |
| COSEM_Object_Instance_Id | O | O (=) | | |
| COSEM_Object_Method_Id | O | O (=) | | |
| Method_Invocation_Parameters<br>{ Method_Invocation_Parameters } | U | U (=) | – | – |
| Response_Type | - | - | O | O (=) |
| Action_Response { Action_Response } | – | – | O | O (=) |
| Result | | | O | O (=) |
| Response_Parameters | | | U | U (=) |
| Data | | | S | S (=) |
| Data_Access_Result | | | S | S (=) |
| DataBlock_SA | C | C (=) | C | C (=) |
| Last_Block | O | O (=) | O | O (=) |
| Block_Number | O | O (=) | O | O (=) |
| Raw_Data | O | O (=) | O | O (=) |
| Block_Number | C | C (=) | C | C (=) |
| NOTE Pour les paramètres de sécurité, voir Tableau 40. | | | | |

Le paramètre Invoke_Id identifie l'instance de l'appel de service.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le paramètre Priority indique le niveau de priorité associé à l'instance de l'appel de service: normale (FALSE) ou élevée (TRUE).

Le paramètre Service_Class indique si le service est confirmé ou non confirmé. La gestion de ce paramètre dépend du profil de communication (voir Annexe A).

L'utilisation des paramètres Request_Type et Response_Type est présentée dans le Tableau 47.

**Tableau 47 – Types de demandes et de réponses du service ACTION**

| Type de demande | | Type de réponse | |
|---|---|---|---|
| NORMAL | La référence de méthode unique et le paramètre d'appel de méthode complet sont envoyés. | NORMAL | Le résultat et le paramètre de retour complet sont envoyés. |
| | | ONE-BLOCK | Un bloc du résultat et du paramètre de retour est envoyé. |
| NEXT | Le bloc de données suivant est demandé. | ONE-BLOCK | Voir ci-dessus. |
| | | LAST-BLOCK | Le dernier bloc du ou des résultats et du ou des paramètres de retour est envoyé. |
| FIRST-BLOCK | La référence d'une méthode unique et le premier bloc des paramètres d'appel de méthode sont envoyés. | NEXT | La bonne réception du bloc est confirmée. |
| ONE-BLOCK | Un bloc des paramètres d'appel de méthode est envoyé. | | |
| LAST-BLOCK | Le dernier bloc des paramètres d'appel de méthode est envoyé. | NORMAL | Voir ci-dessus. |
| | | ONE-BLOCK | Voir ci-dessus. |
| WITH-LIST | La référence d'une liste des méthodes et la liste complète des paramètres d'appel de méthode sont envoyées. | WITH-LIST | La liste complète des résultats et les paramètres de retour sont envoyés. |
| | | ONE-BLOCK | Voir ci-dessus. |
| WITH-LIST-AND-FIRST-BLOCK | La référence d'une liste de méthodes et du premier bloc des paramètres d'appel de méthode est envoyée. | NEXT | La bonne réception du bloc est confirmée. |
| NOTE Le même type de réponse (Response_Type) peut figurer plusieurs fois, pour indiquer les réponses possibles pour chaque demande. | | | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le paramètre COSEM_Method_Descriptor fait référence à une méthode d'objet COSEM. Il est présent lorsque Request_Type == NORMAL, FIRST-BLOCK, WITH-LIST et WITH-LIST-AND-FIRST-BLOCK. Il s'agit d'un paramètre composite:

- le doublet (COSEM_Class_Id, COSEM_Object_Instance_Id) fait référence de façon non ambiguë à une seule instance d'objet COSEM;
- COSEM_Method_Id identifie une méthode de l'objet COSEM référencé.

Une primitive de service ACTION-REQUEST-NORMAL ou ACTION-REQUEST-FIRST-BLOCK doit contenir un seul descripteur de méthode COSEM. Une primitive de service ACTION-REQUEST-WITH-LIST ou ACTION-REQUEST-WITH-LIST-AND-FIRST-BLOCK doit contenir une liste des descripteurs de méthode COSEM; leur nombre est limité par le paramètre server-max-receive-pdu-size: toutes les références de méthode COSEM (ainsi qu'une partie des paramètres d'appel de méthode) doivent être contenues dans une seule APDU.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le paramètre Method_Invocation_Parameter contient le ou les paramètres nécessaires pour appeler la ou les méthodes référencées.

- si Request_Type == NORMAL, le paramètre Method_Invocation_Parameter est facultatif;
- si Request_Type == WITH-LIST, la primitive de service doit contenir une liste des paramètres Method_Invocation_Parameters. Le nombre et l'ordre des paramètres d'appel de méthode doivent être identiques à ceux des paramètres COSEM_Method_Descriptor. Si l'appel d'une méthode n'exige pas de paramètres supplémentaires, ceux-ci doivent cependant être présents et doivent contenir des données NULL.

Si la forme codée du ou des descripteurs de méthode COSEM et du ou des paramètres d'appel de méthode ne tient pas dans une APDU unique, elle peut être transportée en blocs à l'aide du mécanisme spécifique au service ou du mécanisme de transfert de bloc général.

Si le mécanisme de transfert de blocs spécifique au service est utilisé, le paramètre DataBlock_SA contient les informations de contrôle de transfert de blocs et l'élément raw-data:

- l'élément Last_Block indique si le bloc actuel est le dernier (TRUE) ou non (FALSE);
- l'élément Block_Number contient le numéro du bloc réel envoyé;
- l'élément Raw_Data contient une partie des paramètres d'appel de méthode.

Les paramètres Action_Response sont présents dans la primitive .response lorsque Response_Type == NORMAL ou WITH-LIST. Leur nombre et leur ordre doivent être identiques à ceux des descripteurs de méthode COSEM. Ils contiennent deux éléments:

- le paramètre Result. Il contient l'information "réussite" (success) ou la raison de l'échec de l'appel de la méthode référencée (Action-Result);
- le ou les paramètres Response_Parameters. Chaque paramètre de réponse doit contenir les données retournées par suite de l'appel de la méthode, ou une raison de l'échec du retour des paramètres (Data-Access-Result). Si l'appel de l'une des méthodes ne renvoie aucun paramètre, des données NULL doivent être retournées.

Si la réponse ne tient pas dans une seule APDU, elle peut être déplacée en blocs à l'aide du mécanisme spécifique au service ou du mécanisme de transfert de bloc général.

Si le mécanisme de transfert de blocs spécifique au service est utilisé, le paramètre DataBlock_SA contient les informations de contrôle de transfert de blocs et l'élément raw-data:

- l'élément Last_Block indique si le bloc actuel est le dernier (TRUE) ou non (FALSE);
- l'élément Block_Number contient le numéro du bloc réel envoyé;
- l'élément Raw_Data contient une partie de la réponse:
  - si une seule méthode est appelée, Raw_Data contient le résultat et les paramètres Response_Parameters, le cas échéant. Si aucun paramètre Response_Parameters n'est retourné, la réponse doit être de type ACTION-RESPONSE-NORMAL;
  - si une liste de méthodes a été appelée, Raw_Data contient une partie de la liste des paramètres Action_Response et des données facultatives.

Le paramètre Block_Number d'une primitive de service ACTION-REQUEST-NEXT doit contenir le numéro du dernier bloc de données reçu correctement en provenance du serveur.

Le paramètre Block_Number inclus dans une primitive de service ACTION-RESPONSE-NEXT doit contenir le numéro du dernier bloc de données reçu correctement en provenance du client.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

*Utilisation*

Des exemples de séquences logiques de primitives de service ACTION sont représentés à la Figure 35:

- pour une ACTION réussie confirmée, élément a);
- pour une ACTION non confirmée, élément d);
- pour une tentative échouée en raison d'une erreur locale, élément c).

Lors de la première phase, la primitive ACTION.request est appelée par l'AP client pour appeler une ou plusieurs méthodes d'un ou de plusieurs objets d'interface COSEM du AP serveur. Si la liste complète des descripteurs de méthode et des paramètres d'appel de méthode COSEM est contenue dans une seule APDU, la primitive .request est appelée avec Request_Type == NORMAL ou WITH-LIST, selon le cas. Sinon, elle est appelée avec Request_Type == FIRST-BLOCK ou WITH-LIST-AND-FIRST-BLOCK, selon le cas, puis avec Request_Type == ONE-BLOCK, et enfin avec LAST-BLOCK. À réception de la primitive .request, l'AL du client génère l'APDU Action-Request correspondant au type de demande (Request_Type) et l'envoie au serveur.

La primitive ACTION.indication est générée par l'AL du serveur à réception de l'APDU Action-Request.

Lors du transfert de blocs des paramètres d'appel de la méthode, l'AP serveur appelle la primitive ACTION.response avec Request_Type == NEXT jusqu'à réception du dernier bloc.

La primitive ACTION.confirm est générée par l'AL du client à réception de l'APDU Action-Response.

Une fois tous les paramètres d'appel de méthode transférés, le serveur appelle les méthodes des objets d'interface COSEM référencés, puis la deuxième phase commence.

Si la réponse complète est contenue dans une seule APDU, la primitive ACTION.response est appelée par l'AP serveur avec Response_Type == NORMAL ou WITH-LIST, selon le cas. Sinon, elle est appelée avec Response_Type == ONE-BLOCK, puis avec LAST-BLOCK. À réception de la primitive .response, l'AL du serveur génère l'APDU Action-Response correspondant au type de réponse (Response_Type) et l'envoie au client.

La primitive ACTION.confirm est générée par l'AL du client pour confirmer la réception de l'APDU Action-Response.

Lors du transfert de blocs des paramètres de retour, l'AP client appelle la primitive .request avec Request_Type == NEXT jusqu'à réception du dernier bloc.

Le protocole du service ACTION est spécifié en 7.3.5.

## 6.9 Service ACCESS

### 6.9.1 Vue d'ensemble — Principales fonctionnalités

#### 6.9.1.1 Généralités

Le service ACCESS est un service unifié qui peut être utilisé pour accéder à plusieurs attributs et/ou méthodes d'objet COSEM avec une seule primitive .request/.response. Le but de sa mise en œuvre est d'améliorer la messagerie xDLMS tout en maintenant la coexistence avec les services xDLMS existants.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 6.9.1.2 Service WITH-LIST unifié pour améliorer l'efficacité

Le service ACCESS est un service unifié qui utilise le référencement LN et qui peut être utilisé pour lire ou écrire plusieurs attributs d'objet COSEM et/ou appeler plusieurs méthodes avec une seule primitive .request/.response. Chaque demande contient une liste des demandes et des données liées. Chaque réponse contient une liste des données de retour et le résultat de la demande.

NOTE Le référencement SN n'est pas pris en charge actuellement. Il peut être ajouté en introduisant de nouvelles variantes du service.

Alors que les demandes de service GET- / SET- / ACTION-WITH-LIST peuvent uniquement comprendre un type de demande (GET, SET et/ou ACTION) sur la liste, les demandes de service ACCESS peuvent comprendre différents types de demandes. Ceci permet de réduire le nombre d'échanges et améliore ainsi l'efficacité.

Le traitement de la liste de demandes commence avec la première demande de la liste et continue avec le traitement des demandes suivantes jusqu'à la fin de la liste.

#### 6.9.1.3 Variantes spécifiques pour l'accès sélectif

Les primitives de service GET/SET .request doivent toujours contenir les paramètres Access_Selection_Parameters, même lorsque l'accès sélectif n'est pas disponible ou nécessaire. En revanche, le service ACCESS fournit des variantes spécifiques permettant d'accéder aux attributs avec ou sans accès sélectif. Ceci pare au besoin d'inclure les paramètres Access_Selection_Parameters lorsque l'accès sélectif n'est pas disponible ou nécessaire, ce qui réduit la charge utile et améliore l'efficacité.

#### 6.9.1.4 Paramètre Long_Invoke_Id

Le service ACCESS utilise le paramètre long_invoke_id.

NOTE La description des circonstances dans lesquelles les paramètres Long_Invoke_id sont utilisés ne relève pas du domaine d'application du présent document.

#### 6.9.1.5 Réponses autodescriptives

Lorsque cela est demandé par le client, la primitive de service ACCESS.response contient non seulement la réponse à chaque demande, c'est-à-dire le résultat de l'accès à chaque attribut/méthode et les données de retour, mais aussi le paramètre de service Access_Request_Specification (contenant les références aux attributs/méthodes et, le cas échéant, les paramètres Access_Selection), rendant la primitive de service .response autodescriptive. Ces réponses autodescriptives peuvent être stockées et traitées par elles-mêmes, sans nécessiter d'appairer des réponses et des demandes.

#### 6.9.1.6 Gestion des défaillances

En cas de services GET- / SET- / ACTION-WITH-LIST, le client ne peut pas contrôler ce qu'il convient qu'il se produise si l'une des demandes échoue. En revanche, le service ACCESS permet au client de contrôler s'il convient de traiter ou non les demandes qui suivent la demande qui a échoué sur la liste.

#### 6.9.1.7 Horodatage comme paramètre de service

Les services xDLMS spécifiés précédemment ne fournissent pas de paramètre de service dans la primitive de service .request ou .response permettant de contenir un horodatage.

En revanche, les primitives de service ACCESS fournissent un paramètre de service pour contenir l'horodatage comprenant la date et l'heure d'appel de la primitive de service. Ceci réduit davantage la charge utile.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 6.9.1.8 Présence de données dans les primitives de service

Il existe des différences importantes entre les services GET / SET / ACTION et le service ACCESS concernant les données dans les primitives de service:

- service GET: les données ne sont pas présentes dans la demande. Dans la réponse, les données ou le résultat (Data-Access-Result) sont retournés;
- service SET: les données sont présentes dans la demande. Dans la réponse, seul le résultat (Data-Access-Result) est retourné;
- service ACTION: les paramètres d'appel de méthode sont facultatifs dans la demande. Dans la réponse, le résultat de l'appel de la méthode (Action-Result) et, facultativement, le résultat du renvoi des paramètres de retour (Data ou Data-Access-Result) sont retournés;
- service ACCESS: les données sont associées à chaque référence aux attributs/méthodes dans la demande. Si les données ne sont pas nécessaires pour une demande particulière, des données NULL sont incluses. Dans la réponse, les données et le résultat sont retournés. S'il n'existe aucune donnée à retourner pour une réponse particulière, des données NULL sont incluses. En cas d'accès à une méthode, Access-Response-Action (Action-Result) transmet le résultat de l'appel de la méthode et le résultat du renvoi des paramètres de retour.

### 6.9.2 Spécification de service

*Fonction*

Le service ACCESS est un service unifié qui utilise le référencement LN et qui peut être utilisé pour lire ou écrire plusieurs attributs d'objet COSEM et/ou appeler plusieurs méthodes avec une seule primitive .request/.response. Chaque demande contient une liste des demandes et des données liées. Chaque réponse contient une liste des données de retour et le résultat de la demande. Il peut être appelé de manière confirmée ou non confirmée. Il peut être utilisé avec les mécanismes de transfert de bloc général et de chiffrement général.

L'utilisation du bloc de conformité est comme suit:

- le bit 17, *access*, indique la prise en charge du service ACCESS;
- le bit 1, *general-protection*, indique la disponibilité des APDU de protection générale;
- le bit 2, *general-block-transfer*, indique la disponibilité du mécanisme GBT;
- le bit 8, *attribute0-supported-with-set*, et le bit 10, *attribute0-supported-with-get* (10), ne sont pas pertinents: attribute0 est toujours pris en charge;
- le bit 9, *priority-mgmt-supported*, est pertinent;
- le bit 14, *multiple-references*, n'est pas pertinent: le service ACCESS prend toujours en charge plusieurs références;
- le bit 21, *selective-access*, est pertinent. Les paramètres de choix d'accès peuvent être utilisés uniquement si l'utilisation de l'accès sélectif a été négociée avec succès.

*Sémantique des primitives de service*

Les primitives de service ACCESS doivent fournir les paramètres indiqués dans le Tableau 48.

Les paramètres Long_Invoke_Id, Self_Descriptive, Processing_Option, Service_Class et Priority sont obligatoires. Leur valeur dans les primitives de service .indication, .response et .confirm doit être la même que celle dans la primitive de service .request. Elle est contenue par les bits du champ long-invoke-id-and-priority de l'APDU access-request/access-response:

- long-invoke-id (bits 0-23) identifie l'instance de l'appel de service;

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- self-descriptive (bit 28) indique si la réponse de service doit être non autodescriptive (FALSE) ou autodescriptive (TRUE). Lorsqu'il est défini sur TRUE, le paramètre Access_Response_Body doit contenir le paramètre Access_Request_Specification;

NOTE 1 Le paramètre Access_Request_List_Of_Data n'est pas compris dans la primitive de service .response.

- processing-option (bit 29) spécifie que faire lorsque le traitement d'une demande sur la liste échoue. Lorsqu'il est défini sur FALSE, le traitement continue. Lorsqu'il est défini sur TRUE, le traitement est interrompu, c'est-à-dire que les demandes sur la liste qui suivent la demande qui a échoué sur la liste ne doivent pas être traitées. Comme décrit en 6.9.1.2, le traitement de la liste de demandes doit commencer avec la première demande de la liste et doit continuer avec le traitement des demandes suivantes jusqu'à la fin de la liste;
- service-class (bit 30) indique si l'appel de service est confirmé (TRUE) ou non confirmé (FALSE);

NOTE 2 Le paramètre Service_Class s'applique à l'appel de service, pas aux demandes individuelles figurant sur la liste.

NOTE 3 En fonction du profil de communication, le paramètre Service_Class peut également déterminer le type de cadre à utiliser pour contenir l'APDU.

- priority (bit 31) indique le niveau de priorité associé à l'instance de l'appel de service. Il peut s'agir d'une priorité normale (FALSE) ou élevée (TRUE).

NOTE 4 Le paramètre Priority s'applique à l'appel de service, pas aux demandes individuelles figurant sur la liste.

Le paramètre de service Date_Time est facultatif. Lorsqu'il est présent, il doit contenir la date et l'heure de l'appel du service .request/.response. Il est contenu par le champ date-time (de type OCTET STRING) de l'APDU access-request/access-response. Lorsqu'il n'est pas présent, la chaîne d'octets (OCTET STRING) doit avoir une longueur de 0. Sauf indication contraire dans une spécification auxiliaire spécifique à un projet donné, le paramètre Date_Time doit être présent dans la réponse s'il était présent dans la demande et ne doit pas être présent dans la réponse s'il n'était pas présent dans la demande.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 48 – Paramètres de service du service ACCESS**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Long_Invoke_Id | O | O (=) | O (=) | O (=) |
| Self_Descriptive | O | O (=) | O (=) | O (=) |
| Processing_Option | O | O (=) | O (=) | O (=) |
| Service_Class | O | O (=) | O (=) | O (=) |
| Priority | O | O (=) | O (=) | O (=) |
| Date_Time | U | U (=) | U | U (=) |
| Access_Request_Body | O | O (=) | – | – |
| Access_Request_Specification | O | O (=) | – | – |
| { Access_Request_Specification } | | | | |
| Access_Request_Get | U | U (=) | – | – |
| COSEM_Attribute_Descriptor | O | O (=) | – | – |
| Access_Request_Set | U | U (=) | – | – |
| COSEM_Attribute_Descriptor | O | O (=) | – | – |
| Access_Request_Action | U | U (=) | – | – |
| COSEM_Method_Descriptor | O | O (=) | – | – |
| Access_Request_Get_With_Selection | U | U (=) | – | – |
| COSEM_Attribute_Descriptor | O | O (=) | – | – |
| Access_Selection | O | O (=) | – | – |
| Access_Selector | O | O (=) | – | – |
| Access_Parameters | O | O (=) | – | – |
| Access_Request_Set_With_Selection | U | U (=) | – | – |
| COSEM_Attribute_Descriptor | O | O (=) | – | – |
| Access_Selection | O | O (=) | – | – |
| Access_Selector | O | O (=) | – | – |
| Access_Parameters | O | O (=) | – | – |
| Access_Request_List_Of_Data | O | O (=) | – | – |
| Data { Data } | | | – | – |
| Access_Response_Body | – | – | O | O (=) |
| Access_Request_Specification | – | – | C (=) [1] | C (=) |
| { Access_Request_Specification } | | | | |
| Access_Response_List_Of_Data | – | – | O | O (=) |
| Data { Data } | | | | |
| Access_Response_Specification | – | – | O | O (=) |
| { Access_Response_Specification} | | | | |
| Access_Response_Get | – | – | C | C (=) |
| Result | – | – | O | O (=) |
| Access_Response_Set | – | – | C | C (=) |
| Result | – | – | O | O (=) |
| Access_Response_Action | – | – | C | C (=) |
| Result | – | – | O | O (=) |

[1] Lorsque le paramètre de service Access_Request_Specification est présent dans Access_Response_Body, sa valeur doit être la même que celle de la primitive .request/.indication.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le paramètre Access_Request_Body contient les sous-paramètres Access_Request_Specification et Access_Request_List_Of_Data.

Le paramètre Access_Request_Specification contient une liste de spécifications de demande. La liste peut avoir 0 ou plusieurs éléments. Chaque demande peut être l'une des suivantes:

Sans accès sélectif:

- Access_Request_Get contient le COSEM_Attribute_Descriptor d'un attribut à lire;
- Access_Request_Set contient le COSEM_Attribute_Descriptor d'un attribut à écrire;
- Access_Request_Action contient le COSEM_Method_Descriptor d'une méthode à appeler.

Avec accès sélectif:

- Access_Request_Get_With_Selection contient le COSEM_Attribute_Descriptor d'un attribut à lire avec l'accès sélectif, et le paramètre Access_Selection qui contient Access_Selector et Access_Parameters;
- Access_Request_Set_With_Selection contient le COSEM_Attribute_Descriptor d'un attribut à écrire avec l'accès sélectif, et le paramètre Access_Selection qui contient Access_Selector et Access_Parameters.

Il convient de ne pas utiliser Access_Request_Get_/Set_ With_Selection si l'accès sélectif à l'attribut n'est pas disponible ou si COSEM_Attribute_Descriptor identifie tous les attributs (Attribute_0).

Le paramètre COSEM_Attribute_Descriptor est un paramètre composite:

- le doublet (COSEM_Class_Id, COSEM_Object_Instance_Id) fait référence de façon non ambiguë à une seule instance d'objet COSEM;
- l'élément COSEM_Object_Attribute_Id identifie l'attribut de l'instance d'objet. COSEM_Object_Attribute_Id == 0 fait référence à tous les attributs publics de l'objet.

Le paramètre Access_Selection contient les sous-paramètres Access_Selector et Access_Parameters. Les valeurs possibles sont définies dans les définitions pertinentes de la classe d'interface COSEM.

Le paramètre Access_Request_List_Of_Data contient la liste des données liées à la liste des paramètres Access_Request_Specification. Les données dépendent du type de demande d'accès:

- Access_Request_Get faisant référence à un ou à tous les attributs (Attribute_0): null-data (données NULL);
- Access_Request_Set: les données correspondantes contiennent la valeur à écrire. En cas de référence à tous les attributs (Attribute_0), les données doivent être une structure. Le nombre d'éléments dans la structure doit être identique au nombre d'attributs stipulés dans la spécification IC COSEM pertinente. Chaque élément dans la structure contient la valeur à écrire ou les données NULL indiquant que l'attribut donné ne nécessite pas d'être écrit;
- Access_Request_Action: les données correspondantes contiennent les paramètres d'appel de méthode ou, si cela n'est pas exigé, les données NULL.

Le nombre et l'ordre des éléments sur les deux listes doivent être les mêmes.

Le paramètre Access_Response_Body contient les sous-paramètres suivants:

- Access_Request_Specification (facultatif, uniquement lorsque Self_Descriptive == TRUE);
- Access_Response_List_Of_Data;
- Access_Response_Specification.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

À noter que dans la réponse, Access_Request_List_Of_Data vient en premier lieu, suivi de Access_Response_Specification. Si les données sont fournies, mais que le résultat indique un échec, il convient que le client supprime les données.

Le paramètre Access_Request_Specification, lorsqu'il est présent, doit être le même que dans les primitives .request/.indication.

Le paramètre Access_Response_List_Of_Data contient les données résultant du traitement des demandes. Le nombre d'éléments sur la liste doit être le même que sur la liste Access_Request_Specification. Les données dépendent du type de demande d'accès:

- Access_Request_Get faisant référence à un seul attribut: la valeur de l'attribut demandé, ou des données NULL lorsque la valeur de l'attribut ne peut pas être retournée;
- Access_Request_Get faisant référence à tous les attributs (Attribute_0): les données doivent être une structure, contenant la valeur de chaque attribut. Le nombre d'éléments dans la structure doit être identique au nombre d'attributs stipulés dans la spécification IC COSEM pertinente. Si la valeur d'un attribut ne peut pas être retournée, des données NULL doivent être incluses pour cet attribut. Si aucune valeur d'attribut ne peut être retournée, une donnée NULL unique doit être retournée;
- Access_Request_Set faisant référence à un ou à tous les attributs (Attribute_0): données NULL;
- Access_Request_Action: paramètres de retour ou, lorsqu'ils ne sont pas fournis, données NULL.

Le paramètre Access_Response_Specification contient le résultat de chaque demande. Le nombre d'éléments sur cette liste doit être identique à celui sur la liste Access_Request_Specification:

- Access_Request_Get faisant référence à un seul attribut: résultat de la lecture de l'attribut: *success* ou raison de l'échec;
- Access_Request_Get faisant référence à tous les attributs (Attribute_0): le résultat doit être *success* si la valeur de tous les attributs auxquels le droit d'accès est octroyé peut être retournée. Sinon, il doit être *Data-Access-Result* donnant une raison pour l'échec;
- Access_Request_Set faisant référence à un seul attribut: le résultat de l'écriture de l'attribut: *success* ou raison de l'échec;
- Access_Request_Set faisant référence à tous les attributs (Attribute_0): le résultat doit être *success* si la valeur de tous les attributs auxquels le droit d'accès est octroyé peut être écrite. Sinon, il doit être *Data-Access-Result* donnant une raison pour l'échec;
- Access_Response_Action contient le résultat de l'appel d'une méthode: *success* ou raison de l'échec. Le résultat doit être *success* si la méthode peut être appelée avec succès et, lorsque la spécification IC stipule les paramètres de retour, ceux-ci peuvent être retournés avec succès. Sinon, il doit être *Action-Result* donnant une raison pour l'échec.

Si le paramètre Processing_Option est défini sur TRUE et le traitement d'une demande sur la liste échoue, alors pour toutes les demandes qui suivent celle qui a échoué, Access_Response_List_Of_Data doit contenir des données NULL et Access_Response_Specification doit contenir la raison de l'échec.

*Utilisation*

Des exemples de séquences logiques des primitives de service ACCESS sont représentés à la Figure 35:

- pour un ACCESS réussi confirmé, élément a);
- pour un ACCESS non confirmé, élément d);
- pour une tentative échouée en raison d'une erreur locale, élément c).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La primitive ACCESS.request est appelée par l'AP client pour lire ou écrire la valeur d'une liste d'attributs d'objet COSEM et/ou pour appeler une liste de méthodes.

La primitive ACCESS.indication est générée par l'AL du serveur à réception d'une APDU Access-Request.

La primitive ACCESS.response est appelée par l'AP serveur (si Service_Class == Confirmed) pour envoyer une réponse à une primitive .indication reçue.

La primitive ACCESS.confirm est générée par l'AL du client pour accuser réception d'une APDU access-response.

Lorsque la demande ou la réponse ne tient pas dans une seule APDU, le mécanisme de transfert de bloc général peut être utilisé. Voir 4.2.4.4.9.

Si la réponse est trop longue pour tenir dans une seule APDU, mais le GBT n'est pas pris en charge, la réponse peut être une liste de données NULL et une liste de résultats indiquant la raison de l'échec.

Lorsque la protection cryptographique est exigée, les APDU access-request/access-response peuvent être transportées dans des APDU general-ded-ciphering, general-glo-ciphering, general-ciphering ou general-signing selon le type protection à appliquer. Voir 6.5.

Le protocole du service ACCESS est spécifié en 7.3.6.

### 6.10 Service DataNotification

*Fonction*

Le service DataNotification est un service non sollicité, confirmé ou non confirmé. Il est utilisé par le serveur pour transmettre par opération Push des données au client. Il s'agit d'un service non confirmé. Le processus Push est configuré par les objets "Push setup" (voir 4.4.8 de l'IEC 62056-6-2:2021).

*Sémantique des primitives de service*

Les primitives de service DataNotification doivent fournir les paramètres indiqués dans le Tableau 49.

**Tableau 49 – Paramètres de service des primitives de service DataNotification**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Long_Invoke_Id | O | O (=) | O (=) | O (=) |
| Self_Descriptive | – | – | – | – |
| Processing_Option | – | – | – | – |
| Service_Class | O | O (=) | O (=) | O (=) |
| Priority | O | O (=) | – | – |
| Date_Time | U | U (=) | U | U (=) |
| Notification_Body | O | O (=) | – | – |
| Result | - | - | - | O |

Le paramètre Long_Invoke_Id identifie l'instance de l'appel de service.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Les paramètres Self_Descriptive, Processing_Option et Service_Class ne sont pas utilisés dans le cas de ce service.

Le paramètre Service_Class indique si la primitive de service DataNotification.request est appelée de manière confirmée ou non confirmée.

Le paramètre Priority indique le niveau de priorité associé à l'instance de l'appel de service: normale (FALSE) ou élevée (TRUE).

Le paramètre Date_Time indique le moment auquel la primitive de service DataNotification.request/.response est appelée. Lorsque la primitive DataNotification.confirm est appelée à la suite d'une confirmation de couche inférieure, aucun paramètre Date_Time n'est fourni.

Le paramètre Date_Time indique le moment auquel la primitive de service DataNotification.request est appelée. Il s'agit d'une chaîne d'octets, qui peut être de longueur nulle lorsque la transmission de Date_Time n'est pas exigée.

Le paramètre Notification_Body contient les données Push.

Le paramètre Result indique le résultat de confirmation; il est CONFIRMED si Service_Class == Confirmed et à réception de l'APDU Data-Notification-Confirm; ou si Service_Class == Unconfirmed et à réception de la confirmation de la couche de protocole de support. Le paramètre Result est SUPPORTING_LAYER_FAILED en cas de défaillance de la couche de protocole de support.

Le protocole du service DataNotification est spécifié en 7.3.7.

*Utilisation*

Des exemples de séquence logique de primitives de service DataNotification sont représentés à la Figure 35, éléments a), b) et d).

La primitive .request est appelée par l'AP serveur pour transmettre par opération Push des données à l'AP client distant. À réception de la primitive .request, l'AL du serveur construit l'APDU DataNotification.

La primitive .indication est générée par l'AL du client à réception d'une APDU DataNotification.

La primitive DataNotification.response est appelée par l'AP client pour confirmer la réception d'une DataNotification pouvant être confirmée. À la réception de la primitive .response, l'AL du client construit l'APDU Data-Notification-Confirm et l'envoie au serveur.

La primitive DataNotification.confirm est appelée par l'AL du serveur pour transporter la réception d'une APDU Data-Notification-Confirm ou une confirmation de couche inférieure vers l'AP serveur.

Le protocole du service DataNotification est spécifié en 7.3.6.

### 6.11 Service EventNotification

*Fonction*

Le service EventNotification est un service de type non-client/serveur, non sollicité. Il est demandé par le serveur lorsqu'un événement se produit, afin d'informer le client de la valeur d'un attribut, comme s'il était demandé par le COSEM. Il s'agit d'un service non confirmé.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

*Sémantique des primitives de service*

Les primitives du service EventNotification doivent fournir les paramètres indiqués dans le Tableau 50.

**Tableau 50 – Paramètres de service des primitives de service EventNotification**

| | .request | .indication |
|---|---|---|
| Time | U | U (=) |
| Application_Addresses | U | U (=) |
| COSEM_Attribute_Descriptor | O | O (=) |
| COSEM_Class_Id | O | O (=) |
| COSEM_Object_Instance_Id | O | O (=) |
| COSEM_Object_Attribute_Id | O | O (=) |
| Attribute_Value | O | O (=) |

Le paramètre facultatif Time indique l'heure à laquelle la primitive de service EventNotification.request a été émise.

Le paramètre Application_Addresses est facultatif. Il est présent uniquement lorsque le service EventNotification est appelé en dehors d'une AA établie. Dans ce cas, il contient tous les paramètres spécifiques au protocole exigés pour identifier les AP de l'expéditeur et du destinataire.

Lorsque la primitive .request ne contient pas le paramètre facultatif Application_Addresses, des adresses par défaut doivent être utilisées. Il s'agit des adresses par défaut du dispositif logique de gestion de serveur et du AP de gestion du client. Les deux AP sont toujours présents et dans tout profil de protocole. Ils sont associés à des adresses connues prédéfinies.

Le triplet (COSEM_Class_Id, COSEM_Object_Instance_Id, COSEM_Object_Attribute_Id) fait référence de façon non ambiguë à un seul attribut d'une instance d'objet d'interface COSEM.

Le paramètre Attribute_Value définit la valeur de cet attribut. Des informations supplémentaires sur l'événement notifié peuvent être obtenues en interrogeant cet objet d'interface COSEM.

Si la forme codée de la demande ne tient pas dans une seule APDU, elle peut être déplacée en blocs de données à l'aide du mécanisme de transfert de bloc général.

*Utilisation*

Un exemple de séquence logique de primitives de service EventNotification est représenté à la Figure 35, éléments f) et g).

La primitive .request est appelée par l'AP serveur pour envoyer la valeur d'un attribut d'objet d'interface COSEM à l'AP client distant. À réception de la primitive .request, l'AL du serveur construit l'APDU EventNotificationRequest.

Dans certains cas, le ou les protocoles de couche inférieure de support ne permettent pas l'envoi réel et non sollicité d'une unité de données de protocole. Dans ces cas, le client est tenu de solliciter explicitement l'envoi d'une trame EventNotification, en appelant la primitive de service Trigger_EventNotification_Sending.

La primitive EventNotification.indication est générée par l'AL du client à réception d'une APDU EventNotificationRequest.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le protocole du service EventNotification est spécifié en 7.3.8.

### 6.12 Service TriggerEventNotificationSending

*Fonction*

La fonction du service TriggerEventNotificationSending est de déclencher l'envoi de la trame contenant l'APDU EventNotification.request sur le serveur, par le biais du client.

Ce service est nécessaire dans le cadre de protocoles, lorsque le serveur n'est pas en mesure d'envoyer une APDU EventNotification.request réelle non sollicitée.

*Sémantique des primitives de service*

La primitive de service TriggerEventNotificationSending.request doit spécifier les paramètres indiqués dans le Tableau 51.

**Tableau 51 – Paramètres de service de la primitive de service TriggerEventNotificationSending.request**

| | .request |
|---|---|
| Protocol_Parameters | O |

Le paramètre Protocol_Parameters contient toutes les informations relatives à la couche de protocole inférieure, exigées pour déclencher l'envoi par le serveur d'une trame en attente contenant une APDU EventNotification.request. Ces informations incluent l'identifiant de protocole et tous les paramètres de couches inférieures exigés.

*Utilisation*

À réception d'un appel de service TriggerEventNotificationSending.request envoyé par l'AP client, l'AL du client doit appeler le service de couche de support correspondant pour qu'il envoie un message déclencheur au serveur.

### 6.13 Spécification d'accès variable

Variable_Access_Specification est un paramètre des primitives de service .request et .indication suivantes: xDLMS Read / Write / UnconfirmedWrite / InformationReport. Ses variantes sont présentées dans le Tableau 52:

- Variable_Name identifie une variable nommée selon la DLMS®;
- Parameterized_Access permet d'intégrer des paramètres supplémentaires;
- Block_Number_Access contient un numéro de bloc;
- Read_Data_Block_Access contient les informations de contrôle de transfert de blocs et des données brutes;
- Write_Data_Block_Access contient les informations de contrôle de transfert de blocs.

L'utilisation des différentes variantes dépend du service utilisé. Elle est détaillée dans les spécifications de service SN correspondantes.

**Tableau 52 – Spécification d'accès variable**

| Variable_Access_Specification | Read .request | Write .request | Unconfirmed Write.request | Information Report |
|---|---|---|---|---|
| Kind_Of_Access | O | O | O | O |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| | | | | |
|---|---|---|---|---|
| Variable_Name | S | S | S | O |
| Detailed_Access | Non utilisé dans DLMS®/COSEM | | | |
| Parameterized_Access | S | S | S | – |
| Variable_Name | O | O | O | |
| Selector | U | U | U | |
| Parameter | U | U | U | |
| Block_Number_Access | S | – | – | – |
| Block_Number | O | | | |
| Read_Data_Block_Access | S | – | – | – |
| Last_Block | O | | | |
| Block_Number | O | | | |
| Raw_Data | O | | | |
| Write_Data_Block_Access | – | S | – | – |
| Last_Block | | O | | |
| Block_Number | | O | | |

## 6.14 Service Read

*Fonction*

Le service Read est utilisé avec le référencement SN. Il s'agit d'un service confirmé. Ses fonctions sont les suivantes:

- lire la valeur d'un ou de plusieurs attributs d'objets d'interface COSEM. Dans ce cas, la forme codée de la primitive de service .request doit être contenue dans une APDU unique. Le résultat peut être envoyé dans une seule réponse ou, s'il est trop long, dans plusieurs réponses, avec un transfert de blocs;
- appeler une ou plusieurs méthodes d'objet d'interface COSEM, lorsque des paramètres de retour sont attendus. Dans ce cas, si la primitive de service .request (y compris les références de méthode et les paramètres d'appel de méthode) ou .response (y compris les résultats et les paramètres de retour) est trop longue pour être contenue dans une seule APDU, alors le transfert de blocs sur plusieurs demandes et/ou réponses peut être utilisé.

Le service Read est spécifié dans l'IEC 61334-4-41:1996, 10.4 et Annexe A. Afin de fournir des informations complètes et cohérentes avec la spécification de services à l'aide du référencement LN, la spécification est reproduite ici, avec les extensions adaptées à DLMS®/COSEM.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

*Sémantique des primitives de service*

Les primitives du service Read doivent fournir les paramètres de service indiqués dans le Tableau 53.

**Tableau 53 – Paramètres de service du service Read**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Variable_Access_Specification<br>{ Variable_Access_Specification } | O | O (=) | – | – |
| Variable_Name | S | S (=) | | |
| Parameterized_Access | S | S (=) | | |
| Variable_Name | O | O (=) | | |

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Selector | U | U (=) | | |
| Parameter | U | U (=) | | |
| **Read_Data_Block_Access** | S | S (=) | | |
| Last_Block | O | O (=) | | |
| Block_Number | O | O (=) | | |
| Raw_Data | O | O (=) | | |
| **Block_Number_Access** | S | S (=) | | |
| Block_Number | O | O (=) | | |
| Result (+) | | | S | S (=) |
| Read_Result { Read_Result } | – | – | O | O (=) |
| Data | | | S | S (=) |
| Data_Access_Error | | | S | S (=) |
| **Data_Block_Result** | | | S | S (=) |
| Last_Block | | | O | O (=) |
| Block_Number | | | O | O (=) |
| Raw_Data | | | O | O (=) |
| **Block_Number** | | | S | S (=) |
| Result (–) | – | – | S | S (=) |
| Error_Type | | | O | O (=) |
| NOTE Pour les paramètres de sécurité, voir Tableau 40. | | | | |

L'utilisation des diverses variantes du paramètre de service Variable-Access-Specification de la primitive de service Read.request et les différents choix de la primitive Read.response sont détaillés dans le Tableau 54.

Si la forme codée de la réponse ne tient pas dans une seule APDU, elle peut être transportée en blocs de données à l'aide du mécanisme de transfert de bloc général.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 54 – Utilisation des variantes du paramètre Variable_Access_Specification et des choix pour Read.response**

| Read.request Variable_Access_Specification | | CHOIX Read.response | |
|---|---|---|---|
| Variable_Name (Variable_Name} | Référence une liste[1] d'attributs d'objets COSEM. | Data {Data} | Indique la valeur du ou des attributs référencés. |
| | | Data_Access_Error {Data_Access_Error} | Indique la raison de l'échec de lecture. |
| | | Data_Block_Result | Fournit les informations de contrôle de transfert de blocs et un bloc de données brutes. |
| Parameterized_Access {Parameterized_Access} | Référence une liste[1] d'attributs d'objets COSEM à lire de façon sélective. | Data {Data} | Voir ci-dessus. |
| | | Data_Access_Error {Data_Access_Error} | |
| | | Data_Block_Result | |
| | Référence une liste[1] de méthodes d'objet COSEM, avec les paramètres d'appel de méthode. | Data {Data} | Présente les paramètres de retour d'appel de méthode.<br>NOTE 1 Si des paramètres sont retournés, cela implique que l'appel de méthode a réussi. |
| | | Data_Access_Error {Data_Access_Error} | Indique la raison de l'échec de l'appel de méthode. |
| | | Data_Block_Result | Voir ci-dessus. |
| Read_Data_Block_ Access | Contient les informations de contrôle de transfert de blocs et une partie de la forme codée des références de méthode COSEM et des paramètres d'appel de méthode. | Block_Number | Contient le numéro du dernier bloc de données reçu. |
| Block_Number_Access | Contient le numéro du dernier bloc de données reçu. | Data_Block_Result | Voir ci-dessus. |

NOTE 2 Un même choix Read.response peut figurer plusieurs fois, afin de représenter les réponses possibles à chaque demande.

[1] Une liste peut contenir un ou plusieurs éléments.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La primitive de service Read .request peut contenir un ou plusieurs paramètres Variable_Access_Specification.

- La variante Variable_Name permet de référencer un attribut d'objet COSEM complet à lire. La demande peut inclure un ou plusieurs noms de variables;
- La variante Parameterized_Access est utilisée soit:
  - pour référencer un attribut d'objet COSEM à lire de façon sélective. Dans ce cas, l'élément Variable_Name référence l'attribut d'objet COSEM, les éléments Selector et Parameter contiennent respectivement le sélecteur d'accès et les paramètres d'accès comme indiqué dans la spécification de l'attribut; soit
  - pour référencer une méthode d'objet COSEM à appeler. Dans ce cas, l'élément Variable_Name référence la méthode, l'élément Selector est égal à zéro et l'élément Parameter contient les paramètres d'appel de méthode (le cas échéant) ou des données NULL;
  - la demande peut inclure un ou plusieurs paramètres d'accès paramétré.

NOTE 1 Avec cette spécification, le service Read peut transporter des informations dans les deux directions, tout comme le service ACTION utilisé dans le référencement LN: les paramètres d'appel de méthode depuis le client vers le serveur et les paramètres de retour depuis le serveur vers le client.

- La variante Read_Data_Block_Access est utilisée lorsqu'une ou plusieurs méthodes d'objet COSEM sont appelées et lorsque la forme codée de la demande ne tient pas dans une seule APDU. La demande peut inclure un seul paramètre Read_Data_Block_Access. Il contient les informations de contrôle de transfert de blocs et l'élément raw-data:
  - l'élément Last_Block indique si le bloc donné est le dernier (TRUE) ou non (FALSE);
  - l'élément Block_Number contient le numéro du bloc réel envoyé;
  - l'élément Raw_Data contient une partie de la forme codée de la liste des paramètres Variable_Access_Specification (comme en l'absence de transfert de bloc), y compris les références de méthode et les paramètres d'appel de méthode. Ici, seules les variantes Variable_Name et Parameterized_Access sont autorisées.
- La variante Block_Number_Access est appliquée lorsque le serveur utilise le transfert de blocs pour envoyer une réponse longue, pour confirmer la réception d'un bloc de données et pour demander le bloc de données suivant. La demande peut inclure un paramètre Block_Number_Access unique. Il contient le numéro du dernier bloc de données reçu correctement.

Le paramètre Result (+) indique que le service demandé a réussi.

Sans transfert de bloc, les primitives de service .response/.confirm contiennent un ou plusieurs paramètres Read_Result. Leur nombre et leur ordre doivent être identiques à ceux des paramètres Variable_Name/Parameterized_Access des primitives .request/.indication.

Si le service Read est utilisé pour lire un ou des attributs, alors:

- le choix Data permet d'insérer la valeur de l'attribut au moment de l'accès;
- l'élément Data_Access_Error indique la raison de l'échec de lecture de cet attribut.

Si le service Read est utilisé pour appeler une ou des méthodes, alors:

- le choix Data intègre les paramètres de retour (si des données sont renvoyées, cela implique que l'appel de méthode a réussi). Si aucun paramètre de retour n'est défini, Data doit contenir des données NULL;

  NOTE 2 Cependant, si aucune donnée de retour n'est attendue, le service Write est utilisé pour appeler des méthodes.

- le choix Data_Access_Error indique la raison de l'échec de l'appel de cette méthode.

Dans le cas d'un transfert de blocs, la primitive .response/.confirm contient un paramètre Read_Result unique. Le choix Data_Block_Result contient un bloc de la réponse:

- l'élément Last_Block indique si le bloc donné est le dernier (TRUE) ou non (FALSE);
- l'élément Block_Number doit contenir le numéro du bloc envoyé;
- l'élément Raw_Data contient une partie de la forme codée de la liste Read_Results.

Si le bloc de données ne peut pas être fourni, la primitive .response doit contenir un seul paramètre Result à l'aide du choix Data_Access_Error, contenant un message d'erreur approprié, par exemple (14) data-block-unavailable.

Si le numéro de bloc de la demande n'est pas celui attendu, ou si le bloc suivant ne peut pas être fourni, la primitive de service Read.response doit être retournée avec un seul paramètre Read_Result, à l'aide du choix Data_Access_Error, contenant un code approprié, par exemple (19) data-block-number-invalid.

Le choix Block_Number est sélectionné lorsque le service Read est utilisé pour appeler une ou plusieurs méthodes; la demande est envoyée en plusieurs blocs pour confirmer la bonne

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

réception du bloc de données et demander le bloc suivant. Il contient le numéro du dernier bloc reçu.

Le paramètre Result (–) indique que le service demandé précédemment a échoué. Le paramètre Error_Type indique la raison de l'échec. Dans ce cas, le serveur doit renvoyer une APDU ConfirmedServiceError au lieu d'une APDU ReadResponse.

*Utilisation*

Un exemple de séquence logique des primitives de service Read est représenté à la Figure 35, élément a).

La primitive Read.request est appelée par suite de l'appel d'une primitive GET.request ou ACTION.request par l'AP client; elle la mappe sur une primitive Read.request à l'aide de l'ASE SN_MAPPER. L'AL du client génère ensuite l'APDU ReadRequest et l'envoie au serveur. Pour la mise en correspondance de services LN / SN, voir 6.19.

La primitive Read.indication est générée par l'AL du serveur à réception d'une APDU ReadRequest.

La primitive Read.response est appelée par l'AP serveur afin d'envoyer une réponse à une primitive Read.indication précédemment reçue. L'AL du serveur génère alors l'APDU ReadResponse et l'envoie au client.

La primitive Read.confirm est générée par l'AL du client après réception d'une APDU ReadResponse. Elle est ensuite mise en correspondance avec une primitive GET.confirm ou ACTION.confirm par l'ASE SN_MAPPER. La primitive GET.confirm ou ACTION.confirm est ensuite générée.

Le protocole du service Read est spécifié en 7.3.9.

### 6.15 Service Write

*Fonction*

Le service Write est utilisé avec le référencement SN. Il s'agit d'un service confirmé. Ses fonctions sont les suivantes:

- écrire la valeur d'un ou de plusieurs attributs d'objets d'interface COSEM;
- appeler une ou plusieurs méthodes d'objet d'interface COSEM, lorsqu'aucun paramètre de retour n'est attendu.

Dans les deux cas, si la forme codée de la primitive de service .request ne tient pas dans une seule APDU, elle peut alors être envoyée en plusieurs demandes avec le transfert de blocs. La primitive de service .response doit toujours tenir dans une seule APDU.

Le service Write est spécifié dans l'IEC 61334-4-41:1996, 10.5 et Annexe A. Afin de fournir des informations complètes et cohérentes avec la spécification de services à l'aide du référencement LN, la spécification est reproduite ici, avec les extensions adaptées à DLMS®/COSEM.

*Sémantique des primitives de service*

Les primitives du service Write doivent fournir les paramètres de service indiqués dans le Tableau 55.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 55 – Paramètres de service du service Write**

| | .request | .indication | .response | .confirm |
|---|---|---|---|---|
| Variable_Access_Specification<br>{ Variable_Access_Specification } | O | O (=) | – | – |
| Variable_Name | S | S (=) | | |
| Parameterized_Access | S | S (=) | | |
| Variable_Name | O | O (=) | | |
| Selector | O | O (=) | | |
| Parameter | O | O (=) | | |
| Write_Data_Block_Access | S | S (=) | – | – |
| Last_Block | O | O (=) | | |
| Block_Number | O | O (=) | | |
| Data { Data } | O | O (=) | – | – |
| Result (+) | – | – | S | S (=) |
| Write_Result { Write_Result } | - | - | S | S (=) |
| Success | | | S | S (=) |
| Data_Access_Error | | | S | S (=) |
| **Block_Number** | | | S | S (=) |
| Result (–) | | | S | S (=) |
| Error_Type | | | O | O (=) |
| NOTE Pour les paramètres de sécurité, voir Tableau 40. | | | | |

L'utilisation des diverses variantes du paramètre de service Variable-Access-Specification de la primitive de service Write.request et les différents choix de primitive Write.response sont détaillés dans le Tableau 56. L'utilisation du paramètre de service Data est également expliquée.

Si la forme codée de la demande ne tient pas dans une seule APDU, elle peut être déplacée en blocs de données à l'aide du mécanisme spécifique au service ou du mécanisme de transfert de bloc général.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 56 – Utilisation des variantes de Variable_Access_Specification et des choix pour Write.response**

| Write.request Variable_Access_Specification | | CHOIX Write.response | |
|---|---|---|---|
| Variable_Name {Variable_Name} | Référence une liste[1] d'attributs d'objets COSEM.<br>Le paramètre de service Data contient les données à écrire ou le ou les paramètres d'appel de méthode. | Success {Success} | Indique que l'attribut référencé a pu être écrit avec succès. |
| | | Data_Access_Error {Data_Access_Error} | Indique la raison de l'échec d'écriture. |
| Parameterized_Access {Parameterized_Access} | Référence une liste[1] d'attributs d'objet COSEM à écrire de façon sélective.<br>Le paramètre de service Data contient les données à écrire. | Success {Success} | Voir ci-dessus. |
| | | Data_Access_Error {Data_Access_Error} | |
| Write_Data_Block_Access | Contient les informations de contrôle de transfert de blocs.<br>Le paramètre de service Data contient les données brutes, comprenant la forme codée de la liste[1] d'attributs d'objet COSEM ou de références de méthode, et la liste des données à écrire ou la liste des paramètres d'appel de méthode. | Block_Number | Contient le numéro du dernier bloc de données reçu. |

NOTE Un même choix Write.response peut figurer plusieurs fois, afin de représenter les réponses possibles à chaque demande.

[1] Une liste peut contenir un ou plusieurs éléments.

La primitive de service Write.request peut contenir un ou plusieurs paramètres Variable_Access_Specification:

- la variante Variable_Name permet de référencer un attribut d'objet COSEM complet à écrire ou une méthode d'objet COSEM à appeler. La demande peut inclure un ou plusieurs noms de variables;
- la variante Parameterized_Access est utilisée pour référencer un attribut d'objet COSEM à écrire de façon sélective. Dans ce cas, l'élément Variable_Name référence l'attribut d'objet COSEM, les éléments Selector et Parameter contiennent respectivement le sélecteur d'accès et les paramètres d'accès comme indiqué dans la spécification de l'attribut. La demande peut inclure un ou plusieurs paramètres Parameterized_Access.

Le paramètre de service Data inclut la ou les valeurs à écrire sur le ou les attributs, ou le ou les paramètres d'appel de la ou des méthodes à appeler. Le nombre et l'ordre des paramètres Data doivent être identiques à ceux des paramètres Variable_Access_Specification.

Si la primitive de service Write.request ne tient pas dans une seule APDU, le transfert de blocs peut être utilisé. Dans ce cas:

- la variante Write_Data_Block_Access du paramètre Variable_Access_Specification contient les informations de contrôle de transfert de blocs:
  - l'élément Last_Block indique si le bloc donné est le dernier (TRUE) ou non (FALSE);
  - l'élément Block_Number contient le numéro du bloc réel envoyé;
- le paramètre Data contient une partie de la liste des références d'attribut et la liste des données à écrire, ou une partie de la liste de références de méthode et la liste des paramètres d'appel de méthode;
- la demande inclut un paramètre Write_Data_Block_Access unique et un paramètre Data unique.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le paramètre Result (+) indique que le service demandé a réussi.

Les primitives de service .response/.confirm contiennent une liste des paramètres Write_Result. Leur nombre et leur ordre doivent être identiques à ceux des paramètres Variable_Name/Parameterized_Access des primitives de service .request/.indication.

Sans transfert de blocs, et en cas de transfert de blocs, après réception du dernier bloc:

- lorsque le service Write est utilisé pour écrire un ou plusieurs attributs, chaque élément indique soit que l'accès en écriture a réussi (Success), soit la raison de l'échec d'écriture de cette variable (Data_Access_Error);
- lorsque le service Write est utilisé pour appeler une ou plusieurs méthodes, chaque élément indique soit que l'accès d'appel a réussi (Success), soit la raison de l'échec de l'appel de cette méthode (Data_Access_Error).

Le choix Block_Number est utilisé lors du transfert de blocs pour confirmer la réception d'un bloc de données et pour demander le bloc suivant. Il contient le numéro du dernier bloc reçu.

Si le numéro de bloc de la demande n'est pas celui attendu, ou si le bloc n'a pas pu être bien reçu, la primitive de service Write.response doit être retournée avec un seul paramètre Write_Result, à l'aide du choix Data_Access_Error, contenant un code approprié, par exemple (19) data-block-number-invalid.

Le paramètre Result (–) indique que le service demandé a échoué. Le paramètre Error_Type indique la raison de l'échec. Dans ce cas, le serveur doit retourner une APDU ConfirmedServiceError au lieu d'une APDU WriteResponse.

*Utilisation*

Un exemple de séquence logique des primitives de service Write est représenté à la Figure 35, élément a).

La primitive Write.request est appelée par suite de l'appel d'une primitive SET.request ou ACTION.request par l'AP client; elle la mappe sur une primitive Write.request à l'aide de l'ASE SN_MAPPER. L'AL du client génère ensuite l'APDU WriteRequest et l'envoie au serveur. Pour la mise en correspondance de services LN / SN, voir 6.19.

La primitive Write.indication est générée par l'AL du serveur à réception d'une APDU WriteRequest.

La primitive Write.response est appelée par l'AP serveur afin d'envoyer une réponse à une primitive Write.indication reçue précédemment. L'AL du serveur génère alors l'APDU WriteResponse et l'envoie au client.

La primitive Write.confirm est générée par l'AL du client après réception d'une APDU WriteResponse. Elle est ensuite mise en correspondance avec une primitive SET.confirm ou ACTION.confirm par l'ASE SN_MAPPER. La primitive SET.confirm ou ACTION.confirm est ensuite générée.

Le protocole du service Write est spécifié en 7.3.10.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 6.16 Service UnconfirmedWrite

*Fonction*

Le service UnconfirmedWrite est utilisé avec le référencement SN. Il s'agit d'un service non confirmé. Ses fonctions sont les suivantes:

- écrire la valeur d'un ou plusieurs attributs d'objets COSEM;
- appeler une ou plusieurs méthodes d'objet d'interface COSEM, lorsqu'aucun paramètre de retour n'est attendu.

La primitive de service UnconfirmedWrite.request doit toujours tenir dans une seule APDU.

Le service UnconfirmedWrite est spécifié dans l'IEC 61334-4-41:1996, 10.6 et Annexe A. Afin de fournir des informations complètes et cohérentes avec la spécification de services à l'aide du référencement LN, la spécification est reproduite ici, avec les extensions adaptées à DLMS®/COSEM.

*Sémantique des primitives de service*

Les primitives du service UnconfirmedWrite doivent fournir les paramètres de service indiqués dans le Tableau 57.

**Tableau 57 – Paramètres de service du service UnconfirmedWrite**

| | .request | .indication |
|---|---|---|
| Variable_Access_Specification<br>{ Variable_Access_Specification } | O | O (=) |
| Variable_Name | S | S (=) |
| Parameterized_Access | S | S (=) |
| Variable_Name | O | O (=) |
| Selector | O | O (=) |
| Parameter | O | O (=) |
| Data { Data } | O | O (=) |
| NOTE Pour les paramètres de sécurité, voir Tableau 40. | | |

L'utilisation des diverses variantes du paramètre de service Variable-Access-Specification de la primitive de service UnconfirmedWrite.request est présentée dans le Tableau 58. L'utilisation du paramètre de service Data est également expliquée.

Si la forme codée de la demande ne tient pas dans une seule APDU, elle peut être déplacée en blocs de données à l'aide du mécanisme de transfert de bloc général.

**Tableau 58 – Utilisation des variantes de Variable_Access_Specification**

| **UnconfirmedWrite.request Variable_Access_Specification** | |
|---|---|
| Variable_Name<br>{Variable_Name} | Référence un attribut d'objet COSEM.<br>Le paramètre de service Data contient les données à écrire ou le ou les paramètres d'appel de méthode. |
| Parameterized_Access<br>{Parameterized_Access} | Référence un attribut d'objet COSEM avec accès sélectif.<br>Le paramètre de service Data contient les données à écrire. |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La primitive de service UnconfirmedWrite.request peut contenir un ou plusieurs paramètres Variable_Access_Specification.

- La variante Variable_Name permet de référencer un attribut d'objet COSEM complet à écrire ou une méthode d'objet COSEM à appeler.
- La variante Parameterized_Access est utilisée pour référencer un attribut d'objet COSEM à écrire de façon sélective. Dans ce cas, l'élément Variable_Name référence l'attribut d'objet COSEM, les éléments Selector et Parameter contiennent respectivement le sélecteur d'accès et les paramètres d'accès comme indiqué dans la spécification de l'attribut.

Le paramètre de service Data inclut la ou les valeurs à écrire sur le ou les attributs, ou le ou les paramètres d'appel de la ou des méthodes à appeler. Le nombre et l'ordre des paramètres Data doivent être identiques à ceux des paramètres Variable_Access_Specification.

*Utilisation*

Un exemple de séquence logique des primitives de service Write est représenté à la Figure 35, élément d).

La primitive UnconfirmedWrite.request est appelée par suite de l'appel d'une primitive SET.request ou ACTION.request avec le paramètre Service_Class == Unconfirmed par l'AP client; elle la mappe vers une primitive UnconfirmedWrite.request à l'aide de l'ASE SN_MAPPER. L'AL du client génère ensuite l'APDU UnconfirmedWriteRequest et l'envoie au serveur.

La primitive UnconfirmedWrite.indication est générée par l'AL du serveur à réception d'une APDU WriteRequest.

Le protocole du service UnconfirmedWrite est spécifié en 7.3.11.

### 6.17 Service InformationReport

*Fonction*

Le service InformationReport est un service de type non-client/serveur, non sollicité. Il est demandé par un serveur utilisant le référencement SN lorsqu'un événement se produit, afin d'informer le client de la valeur d'une ou plusieurs variables nommées selon la DLMS®, mises en correspondance avec des attributs d'objet d'interface COSEM, comme si les valeurs avaient été demandées par le client. Il s'agit d'un service non confirmé.

Le service InformationReport est détaillé dans l'IEC 61334-4-41:1996, 10.7 et Annexe A. Afin de fournir des informations complètes et cohérentes sur la spécification de services à l'aide du référencement LN, la spécification du service InformationReport est reproduite ici, avec les extensions adaptées aux caractéristiques DLMS®/COSEM.

*Sémantique des primitives de service*

Les primitives du service InformationReport doivent fournir les paramètres indiqués dans le Tableau 59.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 59 – Paramètres de service du service InformationReport**

| | .request | .indication |
|---|---|---|
| Current_Time | O | O (=) |
| Variable_Access_Specification<br>{ Variable_Access_Specification }<br><br>Variable_Name | O<br><br>O | O (=)<br><br>O (=) |
| Data { Data } | O | O (=) |

Le paramètre Current_Time indique l'heure à laquelle la primitive de service InformationReport.request a été émise.

Le paramètre Variable_Access_Specification du choix Variable_Name indique une ou plusieurs variables nommées selon la DLMS® (mises en correspondance sur des attributs d'objet d'interface COSEM) dont les valeurs sont envoyées par le serveur.

Le paramètre Data contient la valeur de la ou des variables nommées selon la DLMS®, qui respectent l'ordre du ou des paramètres Variable_Access_Specification.

Le protocole du service InformationReport est spécifié en 7.3.12.

### 6.18 Services de gestion de couches côté client: Service SetMapperTable.request

*Fonction*

La fonction du service SetMapperTable est de gérer l'ASE SN_MAPPER du client.

*Sémantique des primitives de service*

Il n'existe qu'une seule primitive: .request. Elle doit spécifier les paramètres suivants (voir Tableau 60):

**Tableau 60 – Paramètres de service des primitives de service SetMapperTable.request**

| | .request |
|---|---|
| Mapping_Table | O |

Le paramètre Mapping_table est obligatoire. Il contient les éléments de l'attribut *object_list* du serveur et de l'AA demandés. La structure du contenu est définie dans l'IEC 62056-6-2:2021.

*Utilisation*

Le service SetMapperTable.request est appelé par l'AP client afin de fournir des informations de mise en correspondance avec l'ASE SN_MAPPER du client. Ce service ne provoque aucune transmission de données entre le client et le serveur. L'AP client utilise cette primitive de service afin d'améliorer l'efficacité du processus de mise en correspondance si le référencement SN est utilisé.

### 6.19 Récapitulatif des services et de la mise en correspondance de services de transfert de données LN/SN

Le Tableau 61 et le Tableau 62 récapitulent les services de couche application DLMS®/COSEM.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 61 – Récapitulatif des services ACSE**

| Côté client | Côté serveur |
|---|---|
| COSEM-OPEN.request | COSEM-OPEN.indication |
| COSEM-OPEN.confirm | COSEM-OPEN.response |
| COSEM-RELEASE.request | COSEM-RELEASE.indication |
| COSEM-RELEASE.confirm | COSEM-RELEASE.response |
| COSEM-ABORT.indication | COSEM-ABORT.indication |

**Tableau 62 – Récapitulatif des services xDLMS**

| Côté client | Côté serveur |
|---|---|
| **Référencement LN** | |
| GET.request | GET.indication |
| GET.confirm | GET.response |
| SET.request | SET.indication |
| SET.confirm | SET.response |
| ACTION.request | ACTION.indication |
| ACTION.confirm | ACTION.response |
| ACCESS.request | ACCESS.indication |
| ACCESS.confirm | ACCESS.response |
| EventNotification.indication | EventNotification.request |
| TriggerEventNotificationSending.request | – |
| DataNotification.indication | DataNotification.request |
| DataNotification.response | DataNotification.confirm |
| **Référencement SN** | |
| Read.request | Read.indication |
| Read.confirm | Read.response |
| Write.request | Write.indication |
| Write.confirm | Write.response |
| UnconfirmedWrite.request | UnconfirmedWrite.indication |
| InformationReport.indication | InformationReport.request |
| DataNotification.indication | DataNotification.request |
| DataNotification.response | DataNotification.confirm |
| NOTE Le service DataNotification peut être utilisé dans les contextes d'application utilisant le référencement SN ou ceux utilisant le référencement LN. | |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

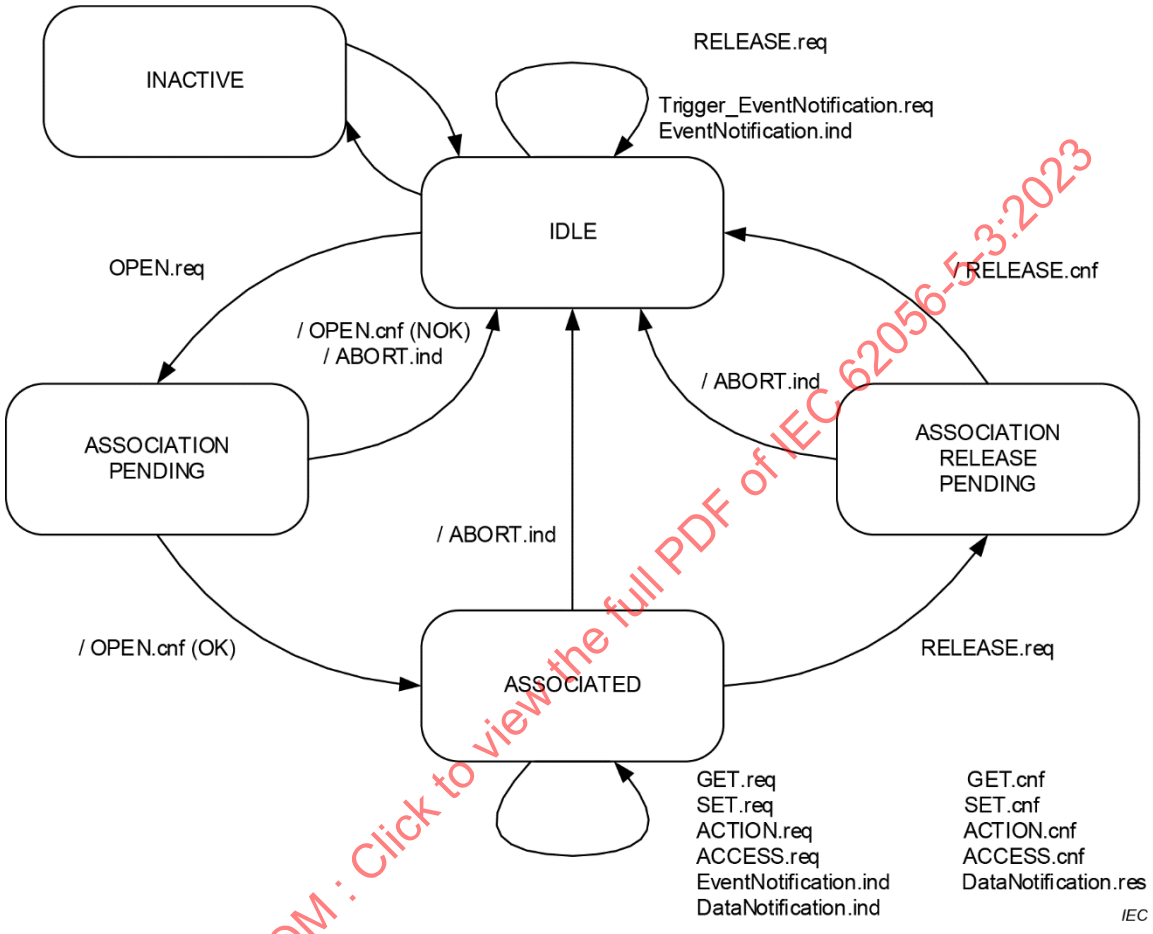
Lorsque le serveur utilise le référencement SN, une mise en correspondance entre les primitives de service utilisant le référencement LN et le référencement SN s'effectue côté client. Cette mise en correspondance est spécifiée en 7.3.9, 7.3.10, 7.3.11 et 7.3.12.

# 7 Spécification du protocole de couche application DLMS®/COSEM

## 7.1 Fonction de commande

### 7.1.1 Définitions des états de la fonction de commande côté client

La Figure 37 représente le diagramme d'états de la CF côté client (voir également Figure 9).




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

NOTE Sur les diagrammes d'états des CF côté client et côté serveur, les conventions suivantes sont utilisées:

– les primitives de service sans caractère initial "/" sont des "stimulants": l'appel de ces primitives constitue l'origine du changement d'état;

– les primitives de service avec un caractère initial "/" sont des "valeurs de sortie": la génération de ces primitives s'effectue au moment du changement d'état.

**Figure 37 – Diagramme d'états partiel pour la fonction de commande côté client**

Les définitions d'état de la CF côté client (et de l'AL où figure la CF) sont les suivantes:

INACTIVE Dans cet état, la CF ne présente aucune activité: elle ne fournit aucun service à l'AP et elle n'utilise aucun service de la couche de protocole de support.

| | |
|---|---|
| IDLE | Dans cet état, la CF ne présente aucune AA existante, aucune AA en cours de libération, ni aucune AA en cours d'établissement [5]. Néanmoins, certains échanges de données entre le client et le serveur (si le canal physique a été établi précédemment) sont possibles. Dans cet état, la CF peut gérer le service EventNotification.<br><br>NOTE Les transitions entre les états INACTIVE et IDLE sont contrôlées en dehors du protocole. Par exemple, la CF peut être considérée comme effectuant la transition de l'état INACTIVE à l'état IDLE en étant instanciée et placée en haut de la couche de protocole de support. La transition inverse s'effectue en supprimant l'instance donnée de la CF. |
| ASSOCIATION PENDING | La CF quitte l'état IDLE et passe à cet état lorsque l'AP demande l'établissement d'une AA en appelant la primitive COSEM-OPEN.request (OPEN.req). La CF peut quitter cet état et soit passer à l'état ASSOCIATED, soit revenir à l'état IDLE. Elle génère la primitive COSEM-OPEN.confirm, (/OPEN.cnf(OK)) ou (/OPEN.cnf(NOK)), en fonction du résultat de la demande d'association. La CF quitte également cet état et revient à l'état IDLE lorsqu'elle génère la primitive COSEM-ABORT.indication (/ABORT.ind). |
| ASSOCIATED | La CF passe à cet état lorsque l'AA a été établie avec succès. Tous les services et APDU xDLMS sont disponibles dans cet état. La CF reste à cet état jusqu'à ce que l'AP demande la libération de l'AA en appelant la primitive COSEM-RELEASE.request (RELEASE.req). La CF quitte également cet état et revient à l'état IDLE lorsqu'elle génère la primitive COSEM-ABORT.indication (/ABORT.ind). |
| ASSOCIATION RELEASE PENDING | La CF quitte l'état ASSOCIATED et passe à cet état lorsque l'AP demande la libération de l'AA en appelant la primitive COSEM-RELEASE.request (RELEASE.req). La CF reste dans cet état et attend que le serveur réponde à cette demande. Le serveur n'est pas autorisé à refuser une demande de libération. Après avoir quitté cet état, la CF passe donc toujours à l'état IDLE. La CF peut quitter cet état en générant la primitive COSEM-RELEASE.confirm après réception de la réponse envoyée par le serveur ou en la générant localement (/RELEASE.cnf). La CF quitte également cet état et revient à l'état IDLE lorsqu'elle génère la primitive COSEM-ABORT.indication (/ABORT.ind). |

### 7.1.2 Définitions des états de la fonction de commande côté serveur

La Figure 38 représente le diagramme d'états de la CF côté serveur (voir Figure 9).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

[5] À noter qu'il s'agit du diagramme d'états pour l'AL: les connexions de couche inférieure, y compris la connexion physique, ne sont pas prises en compte. D'un autre côté, l'établissement de la connexion physique s'effectue en dehors du protocole.

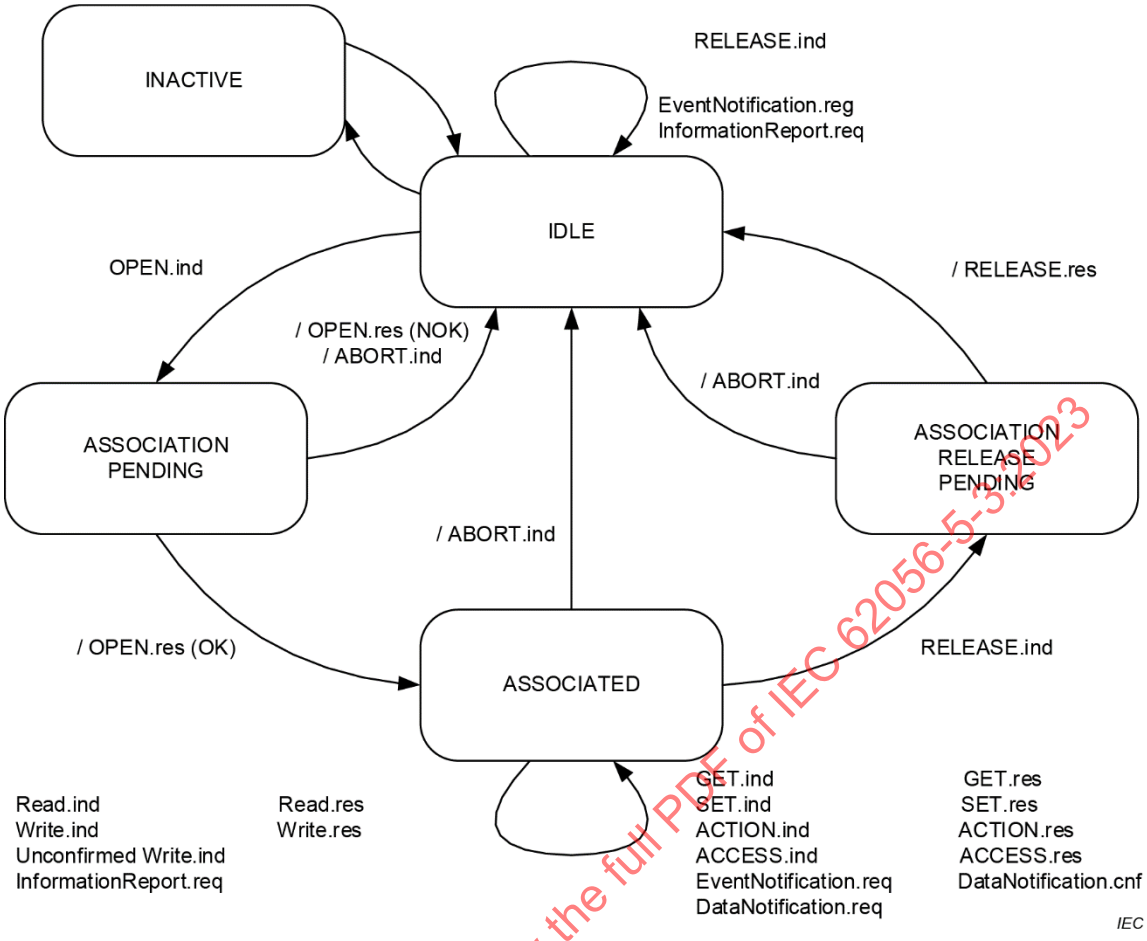



**Figure 38 – Diagramme d'états partiel pour la fonction de commande côté serveur**

INACTIVE — Dans cet état, la CF ne présente aucune activité: elle ne fournit aucun service à l'AP et elle n'utilise aucun service de la couche de protocole de support.

IDLE — Dans cet état, la CF ne présente aucune AA existante, aucune AA en cours de libération ni aucune AA en cours d'établissement [6]. Néanmoins, certains échanges de données entre le client et le serveur (si le canal physique a été établi précédemment) sont possibles. La CF peut gérer les services EventNotification/InformationReport.

ASSOCIATION PENDING — La CF quitte l'état IDLE et passe à cet état lorsque le client demande l'établissement d'une AA et lorsque l'AL du serveur génère la primitive COSEM-OPEN.indication (/OPEN.ind). La CF peut quitter cet état et passer soit à l'état ASSOCIATED, soit à l'état IDLE, en fonction du résultat de la demande d'association, et appeler la primitive COSEM-OPEN.response, (/OPEN.res(OK)) ou (/OPEN.res(NOK)). La CF quitte également cet état et revient à l'état IDLE lorsqu'elle génère la primitive COSEM-ABORT.indication (/ABORT.ind).

ASSOCIATED — La CF passe à cet état lorsque l'AA a été établie avec succès. Tous les services et APDU xDLMS sont disponibles dans cet état. La CF reste dans cet état jusqu'à ce que le client demande la libération de l'AA et que l'AL du serveur génère la primitive COSEM-RELEASE.ind (/RELEASE.ind). La CF quitte également cet état et revient à l'état IDLE lorsqu'elle génère la primitive COSEM-ABORT.indication (/ABORT.ind).

6 À noter qu'il s'agit du diagramme d'états pour l'AL: les connexions de couche inférieure, y compris la connexion physique, ne sont pas prises en compte. D'un autre côté, l'établissement de la connexion physique s'effectue en dehors du protocole.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| | |
|---|---|
| ASSOCIATION RELEASE PENDING | La CF quitte l'état ASSOCIATED et passe dans cet état lorsque le client demande la libération de l'AA et l'AP serveur reçoit la primitive COSEM-RELEASE.indication (/RELEASE.ind). La CF reste dans cet état, en attendant que l'AP accepte la demande de libération. Le serveur n'est pas autorisé à refuser une demande de libération. Après avoir quitté cet état, la CF passe donc toujours à l'état IDLE. La CF peut quitter cet état lorsque l'AP accepte la libération de l'AA, puis appelle la primitive COSEM-RELEASE.response (RELEASE.res). La CF quitte également cet état et revient à l'état IDLE lorsqu'elle génère la primitive COSEM-ABORT.indication (/ABORT.ind). |

## 7.2 Services ACSE et APDU

### 7.2.1 Unités fonctionnelles ACSE, services et paramètres de service

L'ACSE AL DLMS®/COSEM est basé sur l'ACSE orienté connexion, comme spécifié dans l'ISO/IEC 15953 et l'ISO/IEC 15954.

Les unités fonctionnelles permettent de négocier les exigences d'utilisateur ACSE lors de l'établissement de l'association. Cinq unités fonctionnelles sont définies:

- unité fonctionnelle Kernel;
- unité fonctionnelle d'authentification;
- unité fonctionnelle de négociation du contexte d'objet ASO;

  NOTE 1 L'ISO/IEC 15953:1999 et l'ISO/IEC 15954:1999 utilisent les termes "contexte d'objet ASO". Dans DLMS®/COSEM, le terme "contexte d'application" est utilisé conformément à l'ISO/IEC 8649/l'ISO/IEC 8650-1.

- unité fonctionnelle d'association de niveau supérieur;
- unité fonctionnelle d'association imbriquée.

L'AL DLMS®/COSEM utilise uniquement les unités fonctionnelles Kernel et d'authentification.

Les paramètres d'exigences ACSE des APDU AARQ et AARE permettent de sélectionner les unités fonctionnelles de l'association.

L'unité fonctionnelle Kernel est toujours disponible et comprend les services fondamentaux A-ASSOCIATE, A-RELEASE.

L'unité fonctionnelle d'authentification prend en charge l'authentification pendant l'établissement de l'association. La disponibilité de cette unité fonctionnelle est négociée pendant l'établissement de l'association. Cette unité fonctionnelle ne comprend pas de services supplémentaires. Elle ajoute des paramètres au service A-ASSOCIATE.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 63 – APDU d'unité fonctionnelle et leurs champs**

| Unité fonctionnelle | Service | APDU | Nom du champ | Présence |
|---|---|---|---|---|
| Kernel | A-ASSOCIATE | AARQ | protocol-version | F |
| | | | application-context-name | O |
| | | | called-AP-title | U |
| | | | called-AE-qualifier | U |
| | | | called-AP-invocation-identifier | U |
| | | | called-AE-invocation-identifier | U |
| | | | calling-AP-title | U |
| | | | calling-AE-qualifier | U |
| | | | calling-AP-invocation-identifier | U |
| | | | calling-AE-invocation-identifier | U |
| | | | implementation-information | F |
| | | | user-information 2) | O |
| | | | (contenant une APDU xDLMS Initiate.request) | |
| | | | dedicated-key | U |
| | | | response-allowed | U |
| | | | proposed-quality-of-service | U |
| | | | proposed-dlms-version-number | O |
| | | | proposed-conformance | O |
| | | | client-max-receive-pdu-size | O |
| | | AARE | protocol-version | F |
| | | | application-context-name | O |
| | | | result | O |
| | | | result-source-diagnostic | O |
| | | | responding-AP-title | U |
| | | | responding-AE-qualifier | U |
| | | | responding-AP-invocation-identifier | U |
| | | | responding-AE-invocation-identifier | U |
| | | | implementation-information | F |
| | | | user-information 3) | O |
| | | | (contenant une APDU xDLMS initiateResponse) | S |
| | | | negotiated-quality-of-service | U |
| | | | negotiated-dlms-version-number | O |
| | | | negotiated-conformance | O |
| | | | server-max-receive-pdu-size | O |
| | | | vaa-name | O |
| | | | (ou contenant une APDU confirmedServiceError) | S |
| | A-RELEASE | RLRQ | reason | U |
| | | | user-information | U |
| | | RLRE | reason | U |
| | | | user-information | U |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| Unité fonctionnelle | Service | APDU | Nom du champ | Présence |
|---|---|---|---|---|
| Authentification | A-ASSOCIATE | AARQ | sender-acse-requirements | U |
| | | | mechanism-name | U |
| | | | calling-authentication-value | U |
| | | AARE | responder-acse-requirements | U |
| | | | mechanism-name | U |
| | | | responding-authentication-value | U |
| NOTE 1 Ce tableau se base sur le Tableau 2 et le Tableau 3 de l'ISO/IEC 15954:1999. Les champs sont répertoriés dans l'ordre dans lequel ils apparaissent dans les APDU ACSE. | | | | |
| O La présence est obligatoire.<br>F La présence est une option ACPM.<br>U La présence est une option dépendant de l'utilisateur des services ACSE ("ACSE service-user").<br>S Le paramètre est choisi parmi les autres paramètres S comme réponse interne de l'environnement ASE du serveur. | | | | |
| NOTE 2 Conformément à l'ISO/IEC 15953:1999, le paramètre d'informations sur l'utilisateur (user-information) est facultatif. Cependant, dans l'environnement DLMS®/COSEM, il est obligatoire dans les APDU AARQ/AARE. | | | | |
| Il y a quelques modifications dans l'ISO/IEC 15953:1999 et l'ISO/IEC 15954:1999 par rapport à l'ISO/IEC 8649/l'ISO/IEC 8650-1:<br>– dans l'ISO/IEC 15954, la version de protocole est obligatoire dans l'AARQ et facultative dans l'AARE. Dans DLMS®/COSEM, elle reste obligatoire pour la rétrocompatibilité;<br>– en lieu et place de "application-context-name" (nom de contexte d'application), "ASO-context-name" (nom de contexte d'objet ASO) est utilisé. Dans DLMS®/COSEM, "nom de contexte d'application" est conservé. En 7.1.5.2 de l'ISO/IEC 15954:1999, il est spécifié que le nom du contexte d'objet ASO est facultatif. Si la rétrocompatibilité avec d'anciennes réalisations de l'élément ACSE est souhaitée, ils doivent être présents. Par conséquent, dans DLMS®/COSEM, il est obligatoire;<br>– dans l'ISO/IEC 15954, les paramètres result et result-source-diagnostic sont facultatifs. En 7.1.5.8 et 7.1.5.9 de l'ISO/IEC 15954:1999, il est spécifié ceci: Résultat (Result) et Source du résultat – Diagnostic (Result-source-diagnostic) sont facultatifs. Si la rétrocompatibilité avec d'anciennes réalisations de l'élément ACSE est souhaitée, ils doivent être présents. Par conséquent, dans DLMS®/COSEM, ces paramètres sont obligatoires. | | | | |

Le Tableau 63 présente les services, les APDU et les champs d'APDU associés aux unités fonctionnelles ACSE, utilisés par l'AL DLMS®/COSEM. La syntaxe abstraite des APDU ACSE est spécifiée à l'Article 8.

En général, la valeur de chaque champ de l'APDU AARQ est déterminée par les paramètres de la primitive de service COSEM-OPEN.request. De la même façon, la valeur de chaque champ de l'AARE est déterminée par la primitive COSEM-OPEN.response. Le service COSEM-OPEN est spécifié en 6.2.

Les champs des APDU AARQ et AARE sont spécifiés ci-dessous. La gestion de ces champs est spécifiée en 7.2.4.1.

- protocol-version: l'AL DLMS®/COSEM utilise la version 1 de la valeur par défaut. Pour plus de détails, voir ISO/IEC 15954:1999.
- application-context-name**:** les noms de contexte d'application COSEM sont indiqués en 7.2.2.2.

NOTE 2 L'ISO/IEC 15953:1999 et l'ISO/IEC 15954:1999 utilisent "nom de contexte d'objet ASO".

- titres called, calling et responding, qualificateurs et identificateurs d'invocation: ces champs facultatifs contiennent la valeur des paramètres correspondants du service COSEM-OPEN. Pour plus de détails, voir ISO/IEC 15954:1999.
- implementation-information: ce champ n'est pas utilisé par l'AL DLMS®/COSEM. Pour plus de détails, voir ISO/IEC 15954:1999.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

- user-information**:** l'APDU AARQ contient une APDU xDLMS InitiateRequest où sont définis les éléments du paramètre Proposed_xDLMS_Context de la primitive de service COSEM-OPEN.request. L'APDU AARE contient une APDU xDLMS InitiateResponse où figurent les éléments du paramètre Negotiated_xDLMS_Context ou une APDU xDLMS confirmedServiceError où figurent les éléments du paramètre xDLMS_Initiate_Error de la primitive de service COSEM-OPEN.response.
- sender-acse-requirements et responder-acse-requirements**:** ce champ permet de sélectionner les unités fonctionnelles facultatives des APDU AARQ/AARE. Dans COSEM, seule l'unité fonctionnelle d'authentification est utilisée. Lorsqu'elle est présente, elle contient la valeur BIT STRING { authentication (0) }. Bit positionné: l'unité fonctionnelle d'authentification est sélectionnée.
- mechanism-name**:** les noms des mécanismes d'authentification COSEM sont spécifiés en 7.2.2.3.
- calling-authentication-value et responding-authentication-value**:** voir 5.2.2.2.
- result: la valeur de ce champ est déterminée par l'AP COSEM (accepteur) ou l'AL DLMS®/COSEM (ACPM) comme indiqué ci-dessous. Elle est utilisée pour déterminer la valeur du paramètre Result de la primitive COSEM-OPEN.confirm:
  - si l'APDU AARQ est rejetée par l'ACPM (c'est-à-dire que la primitive COSEM-OPEN.indication n'est pas émise par l'AL DLMS®/COSEM), la valeur "rejected (permanent)" ou "rejected (transient)" est attribuée par l'ACPM;
  - sinon, la valeur est déterminée par le paramètre Result de l'APDU COSEM-OPEN.response.
- result-source-diagnostic: ce champ contient la valeur Result source et la valeur Diagnostic. Il est utilisé pour déterminer la valeur du paramètre Failure_Type de la primitive COSEM-OPEN.confirm:
  - valeur correspondant à la source du résultat (Result-source): si l'AARQ est rejetée par l'ACPM, (c'est-à-dire que la primitive COSEM-OPEN.indication n'est pas émise par l'AL DLMS®/COSEM), l'ACPM attribue la valeur "ACSE service-provider". Sinon, l'ACPM attribue la valeur "ACSE service-user";
  - valeur Diagnostic: si l'AARQ est rejetée par l'ACPM, la valeur appropriée est attribuée par l'ACPM. Sinon, la valeur est déterminée par le paramètre Failure_Type de la primitive COSEM-OPEN.response. Si le paramètre Diagnostic ne figure pas dans la primitive .response, l'ACPM attribue la valeur "null".

Les paramètres des APDU RLRQ/RLRE (utilisés lorsque le service COSEM-RELEASE [voir 6.3] est appelé avec le paramètre Use_RLRQ_RLRE == TRUE) sont spécifiés ci-dessous.

- reason**:** indique la valeur adéquate, comme indiqué en 6.2.
- user-information**:** si ces informations sont présentes, elles contiennent une APDU xDLMS InitiateRequest / InitiateResponse, où figurent les éléments du paramètre Proposed_xDLMS_Context / Negotiated_XDLMS_Context, respectivement des primitives de service COSEM-RELEASE.request/.response. Voir 6.2.

### 7.2.2 Noms COSEM enregistrés

#### 7.2.2.1 Généralités

Dans un environnement OSI, de nombreux types d'objets réseau doivent être identifiés par des noms non ambigus au niveau mondial. Ces objets réseau sont des syntaxes abstraites, des syntaxes de transfert, des contextes d'application, des noms de mécanismes d'authentification, etc. Généralement, les noms de ces objets sont attribués par le comité qui développe la norme de base ISO particulière, ou durant des ateliers pour implémenteurs. Il convient de déposer ces noms d'objets. Pour DLMS®/COSEM, ces noms d'objets sont attribués par l'UA DLMS®. Ils sont spécifiés ci-dessous.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La décision nº 1999.01846 de l'OFCOM, Suisse, attribue le préfixe suivant aux identificateurs d'objet définis par l'association DLMS® User Association.

| { joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS®-UA(8) } |
|---|
| NOTE Comme spécifié dans l'UIT-T X.660, A.2.4, pour des raisons historiques, les identificateurs secondaires ccitt et joint-iso-ccitt sont respectivement synonymes des identificateurs itu-t et joint-iso-itu-t, et peuvent donc apparaître dans des valeurs d'identificateurs d'objet ASN.1 et permettent en outre d'identifier la valeur entière primaire correspondante. |

Pour DLMS®/COSEM, les identificateurs d'objet sont spécifiés afin de désigner les éléments suivants:

- noms de contexte d'application COSEM;
- noms de mécanismes d'authentification COSEM;
- ID d'algorithmes cryptographiques.

#### 7.2.2.2 Contexte d'application COSEM

Pour établir un échange d'informations efficace à l'intérieur de l'AA, la paire d'appels AE doit avoir mutuellement connaissance de l'échange, et respecter un ensemble de règles relatives à cet échange. Cet ensemble de règles communes est appelé le contexte d'application de l'AA. Le contexte d'application relatif à une AA est déterminé lors de son établissement.

Un seul contexte d'application est défini par AA. Cependant, l'ensemble de règles constitutives du contexte d'application d'une AA peut contenir des règles modifiant cet ensemble pour la durée de vie de l'AA.

Les méthodes suivantes peuvent être appliquées:

- définition d'un contexte d'application préexistant;
- transfert d'une description réelle du contexte d'application.

Dans un environnement COSEM, il est prévu qu'un contexte d'application préexiste. Il est référencé sous son nom lors de l'établissement d'une AA. Le nom de contexte d'application est défini comme suit: IDENTIFICATEUR D'OBJET ASN.1. COSEM identifie le nom de contexte d'application en fonction de la valeur suivante d'identificateur d'objet:

| COSEM_Application_Context_Name::= |
|---|
| {joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS®-UA(8) application-context(1) context_id(x)} |

La signification de ce contexte général d'application COSEM est la suivante:

- deux ASE sont présents dans l'appel AE, l'ACSE et l'ASE xDLMS;
- l'ASE xDLMS est telle que spécifiée dans l'IEC 61334-4-41:1996;

NOTE Avec les extensions COSEM sur DLMS®, voir 4.2.4.

- la syntaxe de transfert est de type A-XDR.

Les ID de contexte (context_id) spécifiques et l'utilisation des APDU chiffrées et non chiffrées sont présentés dans le Tableau 64 ci-dessous:

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 64 – Noms de contexte d'application COSEM**

| Nom de contexte d'application | context_id | APDU non chiffrées | APDU chiffrées |
|---|---|---|---|
| Logical_Name_Referencing_No_Ciphering::= | context_id(1) | Oui | Non |
| Short_Name_Referencing_No_Ciphering::= | context_id(2) | Oui | Non |
| Logical_Name_Referencing_With_Ciphering::= | context_id(3) | Oui | Oui |
| Short_Name_Referencing_With_Ciphering::= | context_id(4) | Oui | Oui |

Pour établir avec succès une AA, il convient que le paramètre application-context-name des APDU AARQ et AARE contienne l'un des noms "valides". Le client propose un nom de contexte d'application en utilisant le paramètre Application_Context_Name de la primitive COSEM-OPEN.request. Le serveur peut retourner toute valeur, soit la valeur proposée, soit la valeur qu'il prend en charge.

#### 7.2.2.3 Nom de mécanisme d'authentification COSEM

L'authentification du client, du serveur ou les deux est l'un des aspects de sécurité traités par la spécification DLMS®/COSEM. Trois niveaux de sécurité d'authentification sont spécifiés:

- authentification sans sécurité (niveau de sécurité le plus faible) (voir 5.2.2.2.2);
- authentification de niveau de sécurité faible (LLS) (voir 5.2.2.2.3);
- authentification de niveau de sécurité élevé (HLS) (voir 5.2.2.2.4).

DLMS®/COSEM identifie les mécanismes d'authentification en fonction de la valeur générale suivante d'identificateur d'objet:

```
COSEM_Authentication_Mechanism_Name::=

{joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS®-UA(8) authentication_mechanism_name(2) mechanism_id(x)}
```

La valeur de l'élément mechanism_id sélectionne l'un des mécanismes de sécurité spécifiés. Voir Tableau 65.

**Tableau 65 – Noms de mécanismes d'authentification COSEM**

| | |
|---|---|
| COSEM_lowest_level_security_mechanism_name::= | mechanism_id(0) |
| COSEM_low_level_security_mechanism_name::= | mechanism_id(1) |
| COSEM_high_level_security_mechanism_name::= | mechanism_id(2) |
| COSEM_high_level_security_mechanism_name_using_MD5::= | mechanism_id(3) |
| COSEM_high_level_security_mechanism_name_using_SHA-1::= | mechanism_id(4) |
| COSEM_high_level_security_mechanism_name_using_GMAC::= | mechanism_id(5) |
| COSEM_high_level_security_mechanism_name_using_SHA-256::= | mechanism_id(6) |
| COSEM_high_level_security_mechanism_name_using_ECDSA::= | mechanism_id(7) |
| NOTE 1 Avec l'élément mechanism_id(2), la méthode de traitement de l'épreuve est secrète.<br>NOTE 2 L'utilisation des mécanismes d'authentification 3 et 4 n'est pas recommandée pour de nouvelles implémentations. | |

Lorsque l'élément Authentication_Mechanism_Name est présent dans le service COSEM-OPEN, l'unité fonctionnelle d'authentification du service A-ASSOCIATE doit être sélectionnée. Le processus d'authentification LLS et HLS est décrit en 5.2.2.2 et 5.7.3.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 7.2.2.4 ID d'algorithmes cryptographiques

Les ID d'algorithmes cryptographiques identifient l'algorithme pour lequel une clé symétrique secrète dérivée est utilisée. Voir 5.3.4.6.5.

Les algorithmes cryptographiques sont identifiés par la valeur générale suivante d'identificateur d'objet:

| |
|---|
| COSEM_Cryptographic_Algorithm_Id::= |
| {joint-iso-ccitt(2) country(16) country-name(756) identified-organization(5) DLMS®-UA(8) cryptographic-algorithms (3) algorithm_id(x)} |

Les valeurs des algorithm_id sont présentées dans le Tableau 66. Voir aussi Tableau 8.

**Tableau 66 – ID d'algorithmes cryptographiques**

| | |
|---|---|
| COSEM_cryptographic_algorithm_name_aes-gcm-128::= | algorithm_id(0) |
| COSEM_cryptographic_algorithm_name_aes-gcm-256::= | algorithm_id(1) |
| COSEM_cryptographic_algorithm_name_aes-wrap-128::= | algorithm_id(2) |
| COSEM_cryptographic_algorithm_name_aes-wrap-256::= | algorithm_id(3) |

### 7.2.3 Règles de codage d'APDU

#### 7.2.3.1 Codage des APDU ACSE

Les APDU ACSE doivent être codées en BER (ISO/IEC 8825-1). Le paramètre user-information de ces APDU doit contenir l'APDU xDLMS InitiateRequest / InitiateResponse / confirmedServiceError, selon le cas, codée en A-XDR, puis coder la CHAÎNE D'OCTETS résultante en BER.

Des exemples de codages d'APDU AARQ/AARE sont fournis à l'Annexe D.

#### 7.2.3.2 Codage des APDU xDLMS

Les APDU xDLMS doivent être codées en A-XDR, comme indiqué dans l'IEC 61334-6.

#### 7.2.3.3 XML

Selon le paramétrage de l'objet "Push setup", l'APDU DataNotification peut être codée comme document XML à l'aide du schéma XML spécifié à l'Article 9.

NOTE L'utilisation de XML pour coder les autres APDU ne relève pas du domaine d'application du présent document.

### 7.2.4 Protocole d'établissement d'association d'applications

#### 7.2.4.1 Protocole d'établissement d'associations d'applications confirmées

L'établissement d'AA à l'aide du service A-Associate de l'ACSE est l'élément clé de l'interopérabilité COSEM. Les éléments d'une AA sont:

- un AP client, qui propose les AA;
- un AP serveur, qui les accepte ou non.

NOTE 1 Pour prendre en charge des services de diffusion ou multidiffusion, une AA peut également être établie entre un AP client et un groupe d'AP serveur.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La Figure 39 représente le MSC lorsque:

- la primitive COSEM-OPEN.request demande une AA confirmée;
- la connexion de couches inférieures de support est exigée pour l'établissement de cette AA.

Un AP client souhaitant établir une AA confirmée appelle la primitive COSEM-OPEN.request de l'ASO avec le paramètre Service_Class == Confirmed. À noter que la PhL est tenue d'être connectée avant l'appel du service COSEM-OPEN. Le paramètre response-allowed de l'APDU xDLMS InitiateRequest est défini sur TRUE. L'AL du client attend une APDU AARE avant de générer la primitive .confirm, avec un résultat positif ou négatif.

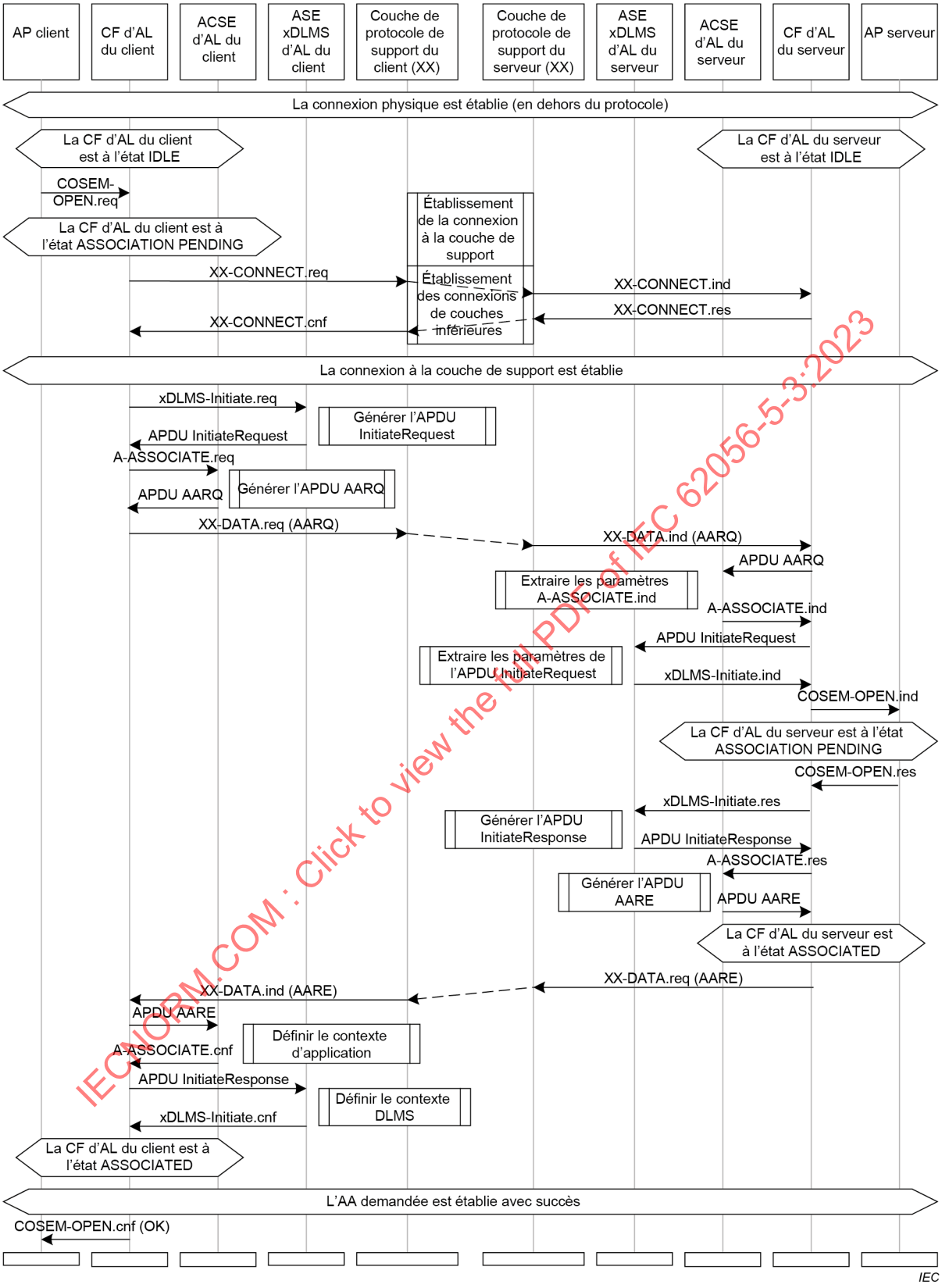
La CF du client passe à l'état ASSOCIATION PENDING. Elle examine ensuite le paramètre Protocol_Connection_Parameters. Si celui-ci indique que l'établissement de la connexion à la couche de support est exigé, la CF établit la connexion. La CF assemble alors (à l'aide de l'ACSE et de l'ASE xDLMS) l'APDU AARQ contenant les paramètres de la primitive COSEM-OPEN.request envoyés par l'AP et l'envoie ensuite au serveur.

La CF de l'AL du serveur indique l'APDU AARQ reçue à l'ACSE. Elle extrait les paramètres associés à l'ACSE, puis redonne le contrôle à la CF. La CF transfère ensuite le contenu du paramètre user-information de l'APDU AARQ (contenant une APDU xDLMS InitiateRequest) vers xDLMS_ASE. Elle extrait les paramètres de cette APDU, puis redonne le contrôle à la CF. La CF génère la primitive COSEM-OPEN.indication sur l'AP serveur avec les paramètres de l'APDU reçue et passe à l'état ASSOCIATION PENDING.

NOTE 2 Certains paramètres de service de la primitive COSEM-OPEN.indication (informations d'adresse, User_Information) ne proviennent pas de l'APDU AARQ, mais de la trame de couche de support contenant l'APDU AARQ. Sous certains profils de communication, le paramètre Service_Class du service COSEM-OPEN est associé au type de trame de la couche de support. Dans d'autres profils de communication, il est associé au champ response-allowed de l'APDU xDLMS-Initiate.request. Voir aussi Annexe A.

NOTE 3 Les ASE extraient uniquement les paramètres, leur interprétation et la décision d'accepter ou non l'AA proposée incombent à l'AP serveur.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure 39 – MSC pour l'établissement réussi d'une AA précédé de l'établissement réussi d'une connexion de couches inférieures**

L'AP serveur analyse les champs de l'APDU AARQ comme décrit ci-dessous.

Champs de l'unité fonctionnelle Kernel:

- application-context-name: contient le nom COSEM_Application_Context_Name proposé par le client pour l'association;
- calling-AP-title: lorsque le contexte d'application proposé utilise le chiffrement, il doit contenir le titre système client. Si un titre système client a déjà été envoyé lors du processus d'enregistrement, par exemple dans le profil PLC S-FSK, le champ calling-AP-title doit contenir le même titre système. Sinon, l'AA doit être rejetée et des informations de diagnostic appropriées doivent être envoyées;
- calling-AE-invocation-identifier: ce champ prend en charge le processus d'identification des utilisateurs du client (voir 4.4.2 de l'IEC 62056-6-2:2021);
- calling-AE-qualifier: ce champ peut être utilisé pour transporter le certificat de clé publique de la clé de signature numérique du client;

Champs de l'unité fonctionnelle d'authentification (le cas échéant):

- sender-acse-requirements:
  a) si ce champ n'est pas présent, ou s'il est présent, mais que bit 0 = 0, l'unité fonctionnelle d'authentification n'est pas sélectionnée. Tous les champs suivants de l'unité fonctionnelle d'authentification peuvent être ignorés;
  b) s'il est présent, mais que bit 0 = 1, l'unité fonctionnelle d'authentification est sélectionnée;
- mechanism-name: contient le nom COSEM_Authentication_Mechanism_Name proposé par le client pour l'association;
- calling-authentication-value: contient la valeur d'authentification générée par le client.

Si la valeur des champs mechanism-name ou calling-authentication-value ne sont pas acceptables, l'AA proposée doit être refusée.

Lorsque l'analyse syntaxique des champs des unités fonctionnelles Kernel et d'authentification est terminée, le serveur passe à l'analyse syntaxique des paramètres de l'APDU xDLMS InitiateRequest, contenus dans le champ user-information de l'AARQ:

- dedicated-key: contient la clé dédiée à utiliser dans l'AA en cours d'établissement;
- response-allowed: Si l'AA proposée est confirmée, la valeur de ce paramètre est TRUE (par défaut) et le serveur doit renvoyer une APDU AARE. Sinon, le serveur ne doit pas répondre. Voir aussi Annexe A;
- proposed-dlms-version-number (voir 4.2.4);
- proposed-conformance (voir 7.3.1);
- client-max-receive-pdu-size (voir 4.2.4).

Si tous les éléments de l'AA proposée sont acceptables, l'AP serveur appelle la primitive de service COSEM-OPEN.response avec les paramètres suivants:

- Application_Context_Name: le même nom que celui proposé;
- Result: accepted;
- Failure_Type: Result-source: acse-service-user; Diagnostic: null;
- Responding_AP_title: si le contexte d'application négocié utilise le chiffrement, il doit contenir le titre système serveur. Si un titre système serveur a déjà été envoyé lors d'un processus d'enregistrement, comme dans le cas d'un profil PLC S-FSK, le paramètre Responding_AP_Title doit contenir le même titre système. Sinon, le client doit annuler l'AA;
- Responding_AE_Qualifier: ce champ peut être utilisé pour transporter le certificat de clé publique de la clé de signature numérique du serveur;
- champs de l'unité fonctionnelle d'authentification de l'AARE:

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

  - (Responder_)ACSE_Requirements:
    i) lorsque l'authentification sans sécurité (niveau de sécurité le plus faible) ou l'authentification de niveau de sécurité faible (LLS) est utilisée, ce champ ne doit pas être présent ou, s'il est présent, le bit 0 (authentification) doit être défini sur 0. Tous les champs suivants de l'unité fonctionnelle d'authentification peuvent être ignorés;
    ii) lorsque l'authentification de niveau de sécurité élevé (HLS) est utilisée, ce champ doit être présent et le bit 0 (authentification) doit être défini sur 1;
  - Security_Mechanism_Name: doit contenir le nom COSEM_Authentication_Mechanism_Name négocié;
  - Responding_Authentication_Value: contient la valeur d'authentification générée par le serveur (StoC);
- Negotiated_xDLMS_context.

La CF assemble l'APDU AARE (à l'aide de l'ASE xDLMS et de l'ACSE) et l'envoie à l'AL du client par l'intermédiaire des protocoles de couche inférieure de support, puis passe à l'état ASSOCIATED. L'AA proposée est alors établie, le serveur est en mesure de recevoir une ou des demandes de service de transfert de données xDLMS (confirmées et non confirmées) et d'envoyer des réponses à ces demandes de service confirmées dans cette AA.

Côté client, les paramètres de l'APDU AARE reçus sont extraits à l'aide de l'ACSE et de l'ASE xDLMS puis sont transmis à l'AP client par l'intermédiaire de la primitive de service COSEM-OPEN.confirm. Au même moment, l'AL du client passe à l'état ASSOCIATED. L'AA est alors établie avec le contexte d'application et le contexte xDLMS négociés.

Si le contexte d'application proposé par le client n'est pas acceptable ou si l'authentification du client échoue, la primitive COSEM-OPEN.response est appelée avec les paramètres suivants:

- Application_Context_Name: le même nom que celui proposé ou celui pris en charge par le serveur;
- Result: rejected-permanent ou rejected-transient;
- Failure_Type: Result-source: acse-service-user; Diagnostic: une valeur appropriée;
- User_Information: une APDU xDLMS InitiateResponse avec les paramètres du contexte xDLMS pris en charge par le serveur.

Si le contexte d'application proposé par le client est acceptable et si l'authentification du client réussit, mais le contexte xDLMS ne peut pas être accepté, la primitive COSEM-OPEN.response doit être appelée avec les paramètres suivants:

- Application_Context_Name: le même nom que celui proposé;
- Result: rejected-permanent ou rejected-transient;
- Failure_Type: Result-source: acse-service-user; Diagnostic: no-reason-given;
- xDLMS_Initiate_Error, indiquant la raison du refus du contexte xDLMS proposé.

Dans ces deux cas, lors de l'appel d'une primitive .response, la CF assemble et envoie l'APDU AARE au client par l'intermédiaire des protocoles de couche inférieure de support. L'AA proposée n'est pas établie et la CF du serveur revient à l'état IDLE.

Côté client, les paramètres de l'APDU AARE reçus sont extraits à l'aide de l'ACSE et de l'ASE xDLMS puis sont transmis à l'AP client par l'intermédiaire de la primitive COSEM-OPEN.confirm. L'AA proposée n'est pas établie et la CF du client revient à l'état IDLE.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

L'ACSE du serveur peut ne pas être en mesure de prendre en charge l'association demandée, par exemple si la syntaxe AARQ ou la version de protocole ACSE ne sont pas acceptables. Dans cette situation, il renvoie une primitive COSEM-OPEN.response au client avec le paramètre Result approprié. La valeur symbolique "ACSE service-provider" est attribuée au paramètre Result Source de façon appropriée. La primitive COSEM-OPEN.indication n'est pas émise. L'association n'est pas établie.

#### 7.2.4.2 Appels de service COSEM-OPEN répétés

Si une primitive COSEM-OPEN.request est appelée par l'AP client référençant une AA déjà établie, l'AL confirme localement et négativement cette demande au motif que l'AA demandée existe déjà. À noter que c'est toujours le cas pour des AA préétablies (voir 7.2.4.4).

Si néanmoins une AL de serveur reçoit une APDU AARQ référençant une AA déjà existante, elle supprime simplement cette AARQ, ou peut répondre avec l'APDU facultative ExceptionResponse, si elle est implémentée.

#### 7.2.4.3 Établissement d'associations d'applications non confirmées

Un AP client souhaitant établir une AA non confirmée appelle la primitive COSEM-OPEN.request de l'ASO avec le paramètre Service_Class == Unconfirmed. Le paramètre response-allowed de l'APDU xDLMS InitiateRequest, contenu dans le paramètre user-information de l'AARQ, est défini sur FALSE. L'AL du client n'attend aucune réponse du serveur: la primitive .confirm est générée localement. Sinon, la procédure est identique à celle de l'établissement d'AA confirmées.

L'établissement d'AA non confirmées n'exige pas la réponse du AP serveur à la demande d'association venant du client; ainsi, dans certains cas, par exemple dans le cas de communications unidirectionnelles ou de diffusion, l'établissement d'AA non confirmées est la seule possibilité.

Lorsqu'une AA non confirmée est établie, les services de transfert de données xDLMS utilisant le référencement LN peuvent être appelés uniquement de façon non confirmée, jusqu'à la libération de l'association. Avec le référencement SN, seul le service UnconfirmedWrite peut être utilisé.

#### 7.2.4.4 Associations d'applications préétablies

Le but des AA préétablies est de simplifier l'échange de données. Les phases d'établissement et de libération d'AA (phases 1 et 3 de la Figure 4) utilisant les services COSEM-OPEN et COSEM-RELEASE sont éliminées, et seuls les services de transfert de données sont utilisés.

La présente Norme internationale n'indique pas comment établir ce type d'AA: cela peut être considéré comme ayant déjà été fait. Les AA préétablies peuvent être considérées comme existantes dès que les couches inférieures sont en mesure de transmettre des APDU entre le client et le serveur.

Comme pour toutes les AA, les dispositifs logiques contiennent un objet d'interface Association LN / SN pour les associations préétablies.

Les AA préétablies peuvent être confirmées ou non confirmées (en fonction de la façon dont elles ont été préétablies).

Les AA préétablies ne peuvent pas être libérées.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 7.2.5 Protocole de libération d'association d'applications

#### 7.2.5.1 Vue d'ensemble

Une AA existante peut être libérée avec ou sans perte de données. La libération sans perte de données est déclenchée par l'AP client. La libération avec perte de données se produit lorsqu'un événement inattendu survient, par exemple lors de la détection d'une déconnexion physique.

#### 7.2.5.2 Libération sans perte de données d'une association d'applications

DLMS®/COSEM propose deux mécanismes de libération d'AA:

- déconnexion de la couche de support de l'AL;
- utilisation du service A-Release de l'ACSE.

Le premier mécanisme doit être pris en charge sous tous les profils où la couche de support de l'AL est orientée vers la connexion.

EXEMPLE Profil à 3 couches orienté connexion basé sur HDLC.

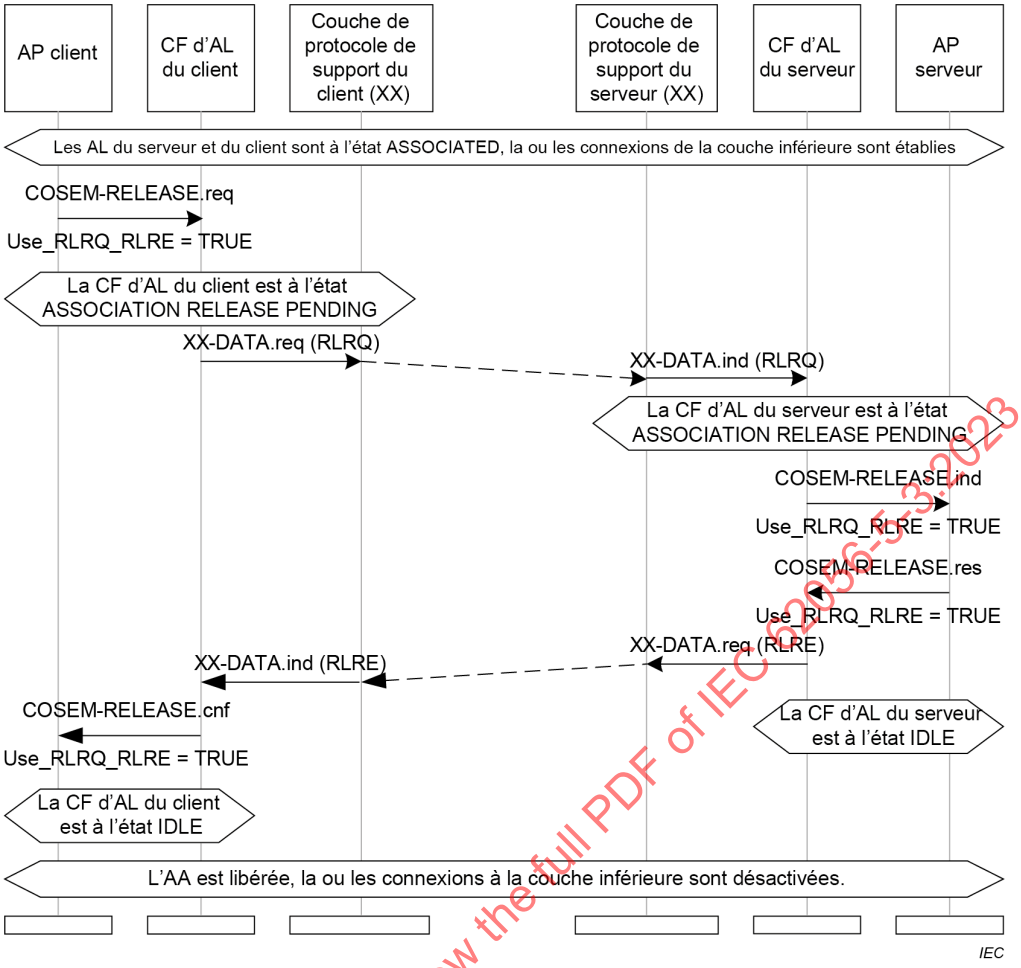
Pour libérer une AA de cette façon, le service COSEM-RELEASE doit être appelé avec le paramètre Use_RLRQ_RLRE non présent ou défini sur FALSE. Lorsque la couche de support se déconnecte, toutes les AA générées sur cette connexion de couche de support doivent être libérées.

Le deuxième mécanisme peut être utilisé pour libérer une AA sans déconnecter la couche de support. Il doit être pris en charge sous tous les profils lorsque la couche de support n'est pas connectée. Il peut également être utilisé lorsque la couche de support est orientée vers la connexion, mais la connexion n'est pas gérée par l'AL, ou lorsque la déconnexion de la couche de support n'est pas effective, car d'autres applications peuvent l'utiliser, ou lorsqu'il est nécessaire de sécuriser le service COSEM-RELEASE. C'est la seule façon de libérer des AA non confirmées.

Pour libérer une AA de cette façon, le service COSEM-RELEASE doit être appelé avec le paramètre Use_RLRQ_RLRE défini sur TRUE. Comme indiqué en 6.3, le service COSEM-RELEASE peut être sécurisé en insérant une APDU xDLMS InitiateRequest/InitiateResponse chiffrée dans le champ user-information des APDU RLRQ/RLRE, respectivement, empêchant ainsi une attaque potentielle par déni de service.

Un exemple de libération d'une AA à l'aide du service A-RELEASE d'ACSE est représenté à la Figure 40.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




NOTE La libération d'une AA peut exiger la communication interne entre les ASE de la CF. Ceci n'est pas représenté aux figures de la Figure 40 à la Figure 42.

**Figure 40 – Libération d'AA sans perte de données à l'aide du service A-RELEASE**

Un AP client souhaitant libérer une AA à l'aide du service A-RELEASE doit appeler la primitive de service COSEM-RELEASE.request avec le paramètre Use_RLRQ_RLRE == TRUE. La CF du client passe à l'état ASSOCIATION RELEASE PENDING. Elle génère ensuite une APDU RLRQ et l'envoie au serveur. Si l'AA à libérer a été établie dans un contexte chiffré, le champ user-information de l'APDU RLRQ peut contenir une APDU xDLMS InitiateRequest chiffrée (voir 6.3).

Lorsque la CF d'AL du serveur reçoit une APDU RLRQ, elle contrôle d'abord si le champ user-information contient une APDU xDLMS InitiateRequest chiffrée. Si oui, elle tente de la déchiffrer. Si elle réussit, elle passe à l'état ASSOCIATION RELEASE PENDING et génère une primitive COSEM-RELEASE.indication avec le paramètre Use_RLRQ_RLRE == TRUE. Sinon, elle supprime en mode silencieux l'APDU RLRQ et reste à l'état ASSOCIATED.

La primitive .response est appelée par l'AP serveur pour indiquer si la libération de l'AA est acceptée ou non, mais uniquement si l'AA à libérer est confirmée. À noter que l'AP serveur ne peut pas refuser une demande de libération. À réception de la primitive .response, la CF de l'AL du serveur génère une APDU RLRE et l'envoie au client. Si l'APDU RLRQ contient une APDU xDLMS InitiateRequest chiffrée, l'APDU RLRE doit contenir une APDU xDLMS InitiateResponse chiffrée. La CF de l'AL du serveur revient à l'état IDLE.

La primitive .confirm est générée par la CF de l'AL du client à réception de l'APDU RLRE. La couche de support n'est pas déconnectée. La CF de l'AL du client revient à l'état IDLE.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Si l'APDU RLRE reçue contient une APDU xDLMS InitiateResponse chiffrée, mais ne peut pas être déchiffrée, l'APDU RLRE doit être supprimée. C'est au client de gérer la situation.

La Figure 41 fournit un exemple de libération sans perte de données d'une AA confirmée, en déconnectant la couche inférieure correspondante.

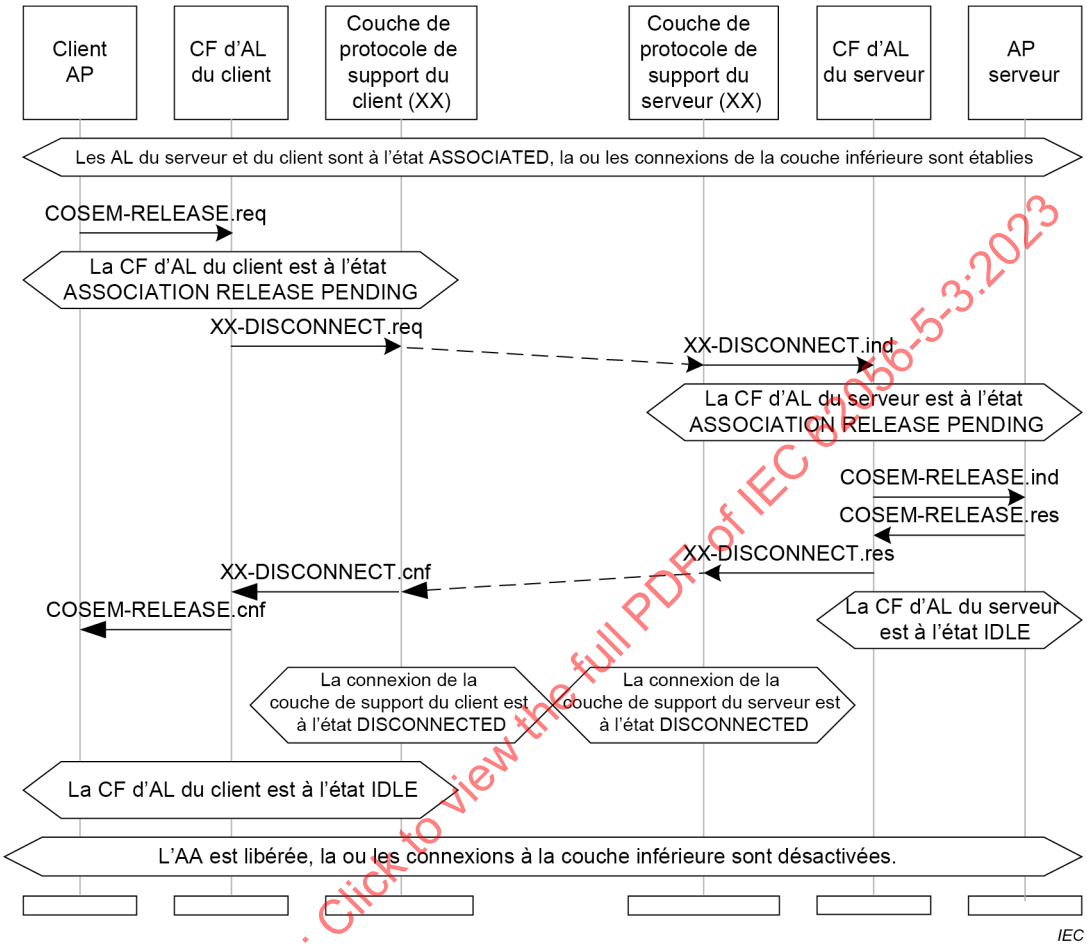



**Figure 41 – Libération d'AA sans perte de données par déconnexion de la couche de support**

Un AP client souhaitant libérer une AA sans utiliser le service A-RELEASE appelle la primitive COSEM-RELEASE.request avec le paramètre Use_RLRQ_RLRE == FALSE ou non présent. La CF de l'AL du client passe à l'état ASSOCIATION RELEASE PENDING.

Sous les profils de communication où le service RLRQ est obligatoire, le fait d'appeler la primitive .request avec le paramètre Use_RLRQ_RLRE absent ou avec le paramètre Use_RLRQ_RLRE == FALSE peut générer une erreur: la primitive .request doit être confirmée localement et négativement. La CF de l'AL du client revient à l'état IDLE.

Lorsque la CF de l'AL du client reçoit la primitive .request, elle envoie une primitive XX-DISCONNECT.request au serveur.

Lorsque la CF d'AL du serveur reçoit la primitive XX-DISCONNECT.request, la CF passe à l'état ASSOCIATION RELEASE PENDING. La primitive COSEM-RELEASE.indication est générée par la CF d'AL du serveur avec le paramètre Use_RLRQ_RLRE absent ou avec le paramètre Use_RLRQ_RLRE == FALSE.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La primitive COSEM-RELEASE.response est appelée par l'AP serveur pour indiquer si la libération de l'AA est acceptée ou non. À noter que l'AP serveur ne peut pas refuser une demande de libération. À réception de cette primitive, la CF d'AL du serveur envoie la primitive XX-DISCONNECT.response au client et revient à l'état IDLE.

La primitive COSEM-RELEASE.confirm est générée par l'AL du client à réception de la primitive XX-DISCONNECT.confirm. La couche de support est déconnectée. La CF de l'AL du client revient à l'état IDLE.

#### 7.2.5.3 Libération avec perte de données d'une association d'applications

Plusieurs événements peuvent entraîner une libération d'AA avec perte de données: la détection d'une déconnexion d'une couche inférieure (y compris la connexion physique), la détection d'une erreur locale, etc.

La libération avec perte de données (l'abandon) d'une AA est signalée à l'AP COSEM par l'intermédiaire du service COSEM-ABORT. Le paramètre Diagnostics de ce service indique la raison de la libération d'AA avec perte de données. La libération avec perte de données d'une AA n'est pas sélective: si elle se produit, toutes les associations existantes (à l'exception de celles préétablies) doivent être abandonnées.

La Figure 42 représente le tableau de séquences de messages générées par l'abandon d'une AA en raison de la détection d'une déconnexion physique.

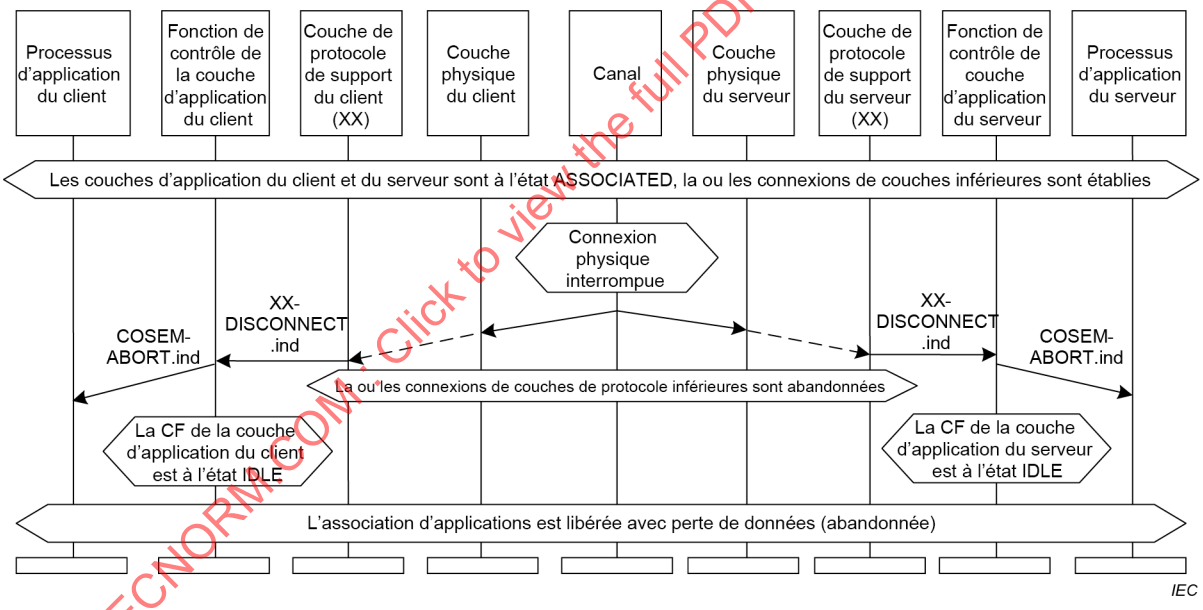



**Figure 42 – Abandon d'une AA après la primitive PH-ABORT.indication**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

## 7.3 Protocole des services de transfert de données

### 7.3.1 Négociation de services et d'options — Bloc de conformité

Le bloc de conformité permet aux clients et aux serveurs utilisant le même protocole DLMS®/COSEM, mais prenant en charge des capacités différentes, de négocier un ensemble de capacités de sorte qu'ils puissent communiquer. Il est contenu dans le paramètre DLMS_Conformance du service COSEM-OPEN.

Dans DLMS®/COSEM, aucun service ni aucune option n'est obligatoire: ceux à utiliser sont négociés par l'intermédiaire du service COSEM-OPEN (le paramètre proposed-conformance de l'APDU xDLMS InitiateRequest et le paramètre negotiated-conformance de l'APDU xDLMS InitiateResponse). Un service implémenté doit se conformer totalement à sa spécification. Si

un service ou une option n'est pas présent dans le bloc de conformité négocié, il/elle peut ne pas être demandé par le client.

Le bloc de conformité xDLMS peut être différencié du bloc de conformité DLMS® indiqué dans l'IEC 61334-4-41:1996 au moyen de cette balise: APPLICATION 31. Voir Tableau 67.

**Tableau 67 – Bloc de conformité xDLMS**

| Bit de bloc de conformité | Réservé | Référencement LN | Référencement SN |
|---|---|---|---|
| 0 | x | | |
| 1 | | general-protection [1] | general-protection [1] |
| 2 | | general-block-transfer | general-block-transfer |
| 3 | | | read |
| 4 | | | write |
| 5 | | | unconfirmed-write |
| 6 | | delta-value-encoding | delta-value-encoding |
| 7 | x | | |
| 8 | | attribute0-supported-with-set | |
| 9 | | priority-mgmt-supported | |
| 10 | | attribute0-supported-with-get | |
| 11 | | block-transfer-with-get-or-read | block-transfer-with-get-or-read |
| 12 | | block-transfer-with-set-or-write | block-transfer-with-set-or-write |
| 13 | | block-transfer-with-action | |
| 14 | | multiple-references | multiple-references |
| 15 | | | information-report |
| 16 | | data-notification | data-notification |
| 17 | | access | |
| 18 | | | parameterized-access |
| 19 | | get | |
| 20 | | set | |
| 21 | | selective-access | |
| 22 | | event-notification | |
| 23 | | action | |

[1] Le bit general-protection inclut general-glo-ciphering, general-ded-ciphering, general-ciphering et general-signing.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 7.3.2 Appels de service confirmés et non confirmés

#### 7.3.2.1 Appels de service par le client

Généralement, les services de transfert de données peuvent être appelés de façon confirmée ou non confirmée. La séquence temporelle des primitives de service correspond aux représentations suivantes:

- Figure 35, élément a) pour les appels de service confirmés;
- Figure 35, élément d) pour les appels de service non confirmés.

Un AP client souhaitant accéder à un attribut ou à une méthode d'interface d'objet COSEM appelle la primitive de service .request appropriée. L'AL du client génère l'APDU correspondant à la primitive .request et l'envoie au serveur.

À réception de la primitive .indication, l'AP serveur vérifie si le service peut être fourni ou non (validité, droits d'accès du client, disponibilité, etc.). S'il le peut, il applique localement le service exigé sur l'objet "réel" correspondant. Dans le cas de services confirmés, l'AP serveur appelle la primitive .response appropriée. L'AL du serveur génère l'APDU correspondant à la primitive .response et l'envoie au client. L'AL du client génère la primitive .confirm.

Si une demande de service confirmée ne peut pas être traitée par l'AL du serveur (par exemple si la demande a été reçue sans qu'une AA ait été préalablement établie ou si la demande présente une erreur), soit elle est ignorée, soit l'AL du serveur y répond, si possible, avec une APDU ConfirmedServiceError ou, si elle est implémentée, une APDU ExceptionResponse. Ces APDU peuvent contenir des informations de diagnostic sur la raison de l'échec du traitement de la demande. Elles sont définies à l'Article 8.

Dans les AA confirmées, les services de transfert de données de type client/serveur peuvent être appelés de manière confirmée ou non confirmée.

Dans les AA non confirmées, les services de transfert de données de type client/serveur peuvent être appelés de manière non confirmée uniquement. Les collisions dues à des réponses multiples possibles dans le cas d'une multidiffusion et/ou d'une diffusion peuvent ainsi être évitées.

Dans le cas de services non confirmés, trois types différents d'adresses de destination sont possibles: individuel, groupe ou diffusion. En fonction du type de l'adresse de destination, la station destinataire doit gérer les APDU entrantes différemment, comme suit:

- XX-APDU avec l'adresse individuelle d'un dispositif logique COSEM. Si elles sont reçues dans une AA établie, elles doivent être envoyées au dispositif logique COSEM destinataire, sinon, elles doivent être ignorées;
- XX-APDU avec une adresse de groupe de dispositifs logiques COSEM. Elles doivent être envoyées au groupe de dispositifs logiques COSEM destinataire. Cependant, le message reçu doit être ignoré s'il n'existe aucune AA établie entre un client et le groupe de dispositifs logiques COSEM destinataire;
- les XX- APDU avec adresse de diffusion doivent être envoyées vers tous les dispositifs logiques COSEM destinataires. Cependant, le message reçu doit être ignoré si aucune AA n'est établie entre le client et l'adresse correspondant à toutes les stations.

NOTE Des AA non confirmées établies entre un client et un groupe de dispositifs logiques sont établies avec un service COSEM-OPEN avec Service_Class == Unconfirmed et un groupe d'adresses de dispositifs logiques (par exemple, adresse de diffusion).

#### 7.3.2.2 Appels de service par le serveur (services non sollicités)

Les services non sollicités qui peuvent être appelés par le serveur sont les suivants:

- InformationReport;
- EventNotification;
- DataNotification.

Les services InformationReport et EventNotification peuvent être appelés uniquement de manière non confirmée. Le diagramme de séquence temporelle correspondant est représenté à la Figure 35, élément d).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le service DataNotification peut être appelé de trois manières différentes:

a) non confirmé, réessayer en cas de défaillance de la couche de protocole de support;

b) non confirmé, réessayer en cas d'absence de confirmation de la couche de protocole de support;

c) confirmé, réessayer en cas d'absence de confirmation.

La séquence temporelle des primitives de service correspond aux représentations suivantes:

– dans le cas a), Figure 35, élément d);
– dans le cas b), Figure 35, élément b);
– dans le cas c), Figure 35, élément a).

Le protocole du service DataNotification est spécifié en 6.10.

### 7.3.3 Protocole du service GET

Lorsque l'AP client souhaite lire la valeur d'un ou plusieurs attributs d'objet d'interface COSEM, il utilise le service GET.

Comme expliqué en 6.6, la forme codée de la demande doit toujours être contenue dans une seule APDU.

Cependant, le résultat peut être trop long pour tenir sur une seule APDU. Dans ce cas, le mécanisme spécifique au service ou le mécanisme de transfert de bloc général peut être utilisé. Il est négocié par l'intermédiaire du bit 2 ou du bit 11 du bloc de conformité (voir 7.3.1).

NOTE Dans certains profils de communication DLMS®/COSEM, une segmentation est disponible pour le transfert de longues APDU.

Les types de primitives de service GET et les APDU correspondantes sont présentés dans le Tableau 68.

**Tableau 68 – Types de service GET et APDU**

| GET .req/.ind | APDU Request | APDU Response | GET .res/.cnf |
|---|---|---|---|
| NORMAL | Get-Request-Normal | Get-Response-Normal | NORMAL |
| | | Get-Response-With-Datablock avec Last-Block = FALSE | ONE-BLOCK |
| NEXT | Get-Request-Next | Get-Response-With-Datablock avec Last-Block = FALSE | ONE-BLOCK |
| | | Get-Response-With-Datablock avec Last-Block = TRUE | LAST-BLOCK |
| WITH-LIST | Get-Request-With-List | Get-Response-With-List | WITH-LIST |
| | | Get-Response-With-Datablock avec Last-Block = FALSE | ONE-BLOCK |

La Figure 43 représente le MSC pour un service GET confirmé en cas de succès, sans transfert de blocs.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

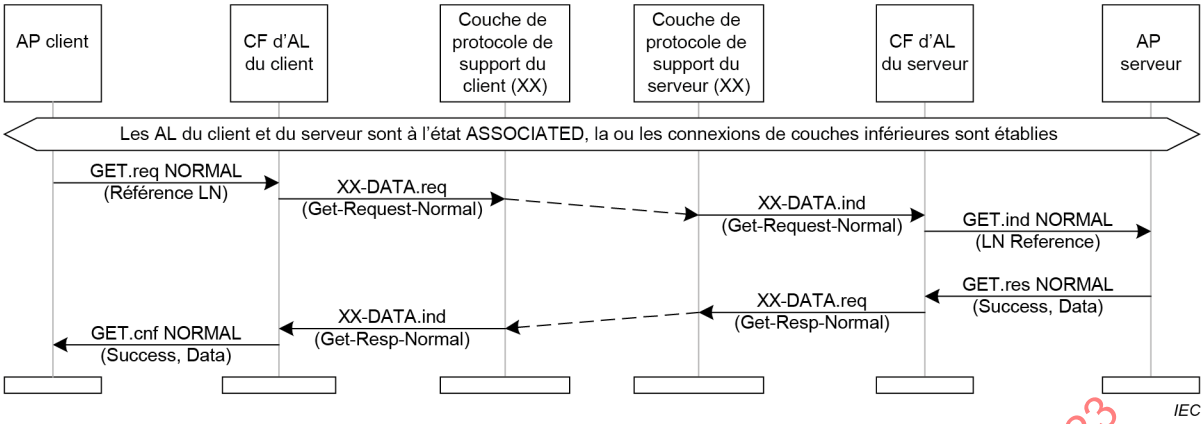
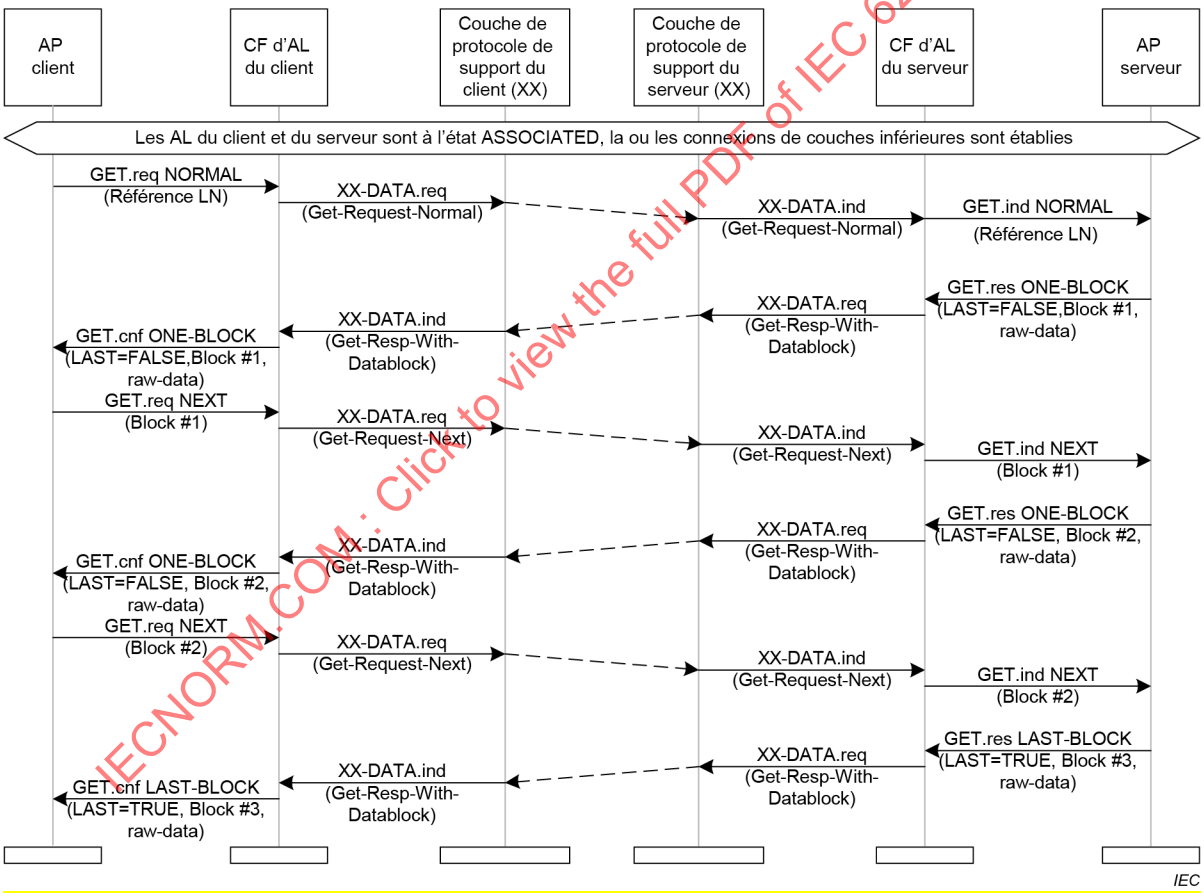



**Figure 43 – MSC du service GET**

La Figure 44 représente le MSC d'un service GET confirmé en cas de succès; le résultat est retourné en trois blocs.




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure 44 – MSC du service GET avec transfert de blocs**

La primitive de service GET.request est appelée avec le type Request_Type == NORMAL ou WITH-LIST, selon le cas. Dans ce cas, les données à retourner sont trop longues pour être contenues dans une seule APDU. L'AP serveur les envoie donc en plusieurs blocs. Les données sont d'abord codées, comme si elles étaient adaptées pour être contenues dans une seule APDU:

- si la valeur d'un attribut unique a été demandée, seuls le type et la valeur doivent être codés (Data). Si l'attribut Data ne peut pas être fourni, la réponse doit être de type GET-NORMAL avec l'attribut Data_Access_Result;

- si la valeur d'une liste d'attributs a été demandée, la liste de résultats doit être codée pour chaque attribut (Data ou Data_Access_Result).

Le résultat est une série d'octets, $B_1$, $B_2$, $B_3$…., $B_N$. Le serveur peut générer la réponse complète à réception de la première primitive GET.indication ou de façon dynamique (à la volée).

L'AP serveur assemble ensuite la structure DataBlock_G:

- Last_Block == FALSE;
- Block_Number == 1;
- Result (Raw_Data) == les K premiers octets des données codées: $B_1$, $B_2$, $B_3$…., $B_K$.

Il est recommandé de commencer la numérotation des blocs à partir de 1.

L'AP serveur appelle alors la primitive de service GET-RESPONSE-ONE-BLOCK avec Response_Type == ONE-BLOCK contenant cette structure DataBlock-G.

À réception de la primitive .response, l'AL du serveur génère une APDU Get-Response-With-Datablock contenant les paramètres de la primitive .response et l'envoie au client.

À réception de cette APDU, l'AL du client génère une primitive .confirm avec Response_Type == ONE-BLOCK. L'AP client est alors informé que la réponse sera fournie en plusieurs blocs. Elle stocke le bloc de données reçu ($B_1$, $B_2$, $B_3$…., $B_K$), puis accuse réception des données et demande le bloc suivant en appelant une primitive de service GET-REQUEST-NEXT. Le numéro de bloc doit être identique au numéro du bloc de données reçu. L'AL du client génère ensuite une APDU Get-Request-Next et l'envoie au serveur.

Lorsque l'AL du serveur appelle la primitive GET.indication avec Request_Type == NEXT, l'AP serveur prépare et envoie le bloc de données suivant, y compris $B_{K+1}$, $B_{K+2}$, $B_{K+3}$…., $B_L$, avec le paramètre block-number == 2. Cet échange bloc de données-accusés de réception se poursuit jusqu'à l'envoi du dernier bloc, contenant $B_M$, $B_{M+1}$, $B_{M+2}$…., $B_N$. La dernière primitive GET.response est appelée avec le paramètre Response_Type == LAST-BLOCK: dans la structure DataBlock-G, Last_Block == TRUE. Le client n'accuse pas réception de ce dernier bloc de données.

Tout au long de la procédure, les paramètres Invoke_Id et Priority doivent être identiques dans chaque primitive.

Si lors d'un transfert de données long, le serveur reçoit une autre demande de service, celle-ci est traitée en fonction des règles de priorité et des paramètres de gestion des priorités (bit 9 du bloc de conformité).

Si une erreur se produit lors d'un transfert de données long, le transfert est annulé. Les situations d'erreur sont les suivantes.

a) Le serveur n'est pas en mesure de fournir le bloc de données suivant pour une raison quelconque. Dans ce cas, l'AP serveur doit appeler une primitive de service GET-RESPONSE-LAST-BLOCK. Le paramètre Result doit contenir une structure DataBlock_G avec:
   - Last_Block == TRUE;
   - Block_Number == numéro de bloc confirmé par le client +1;
   - Result == Data_Access_Result, indiquant la raison de l'échec.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

b) Le paramètre Block_Number d'une primitive de service GET-REQUEST-NEXT n'est pas égal au numéro du bloc précédent envoyé par le serveur. Le serveur interprète cette erreur comme si le client souhaitait annuler le transfert en cours. Dans ce cas, l'AP serveur doit appeler une primitive de service GET-RESPONSE-LAST-BLOCK. Le paramètre Result doit contenir une structure DataBlock_G avec:

- Last_Block == TRUE;
- Block_Number == le numéro du bloc reçu du client;
- Result == Data-Access-Result, long-get-aborted.

c) Le serveur peut recevoir une APDU Get-Request-Next lorsqu'aucun transfert de données long n'est en cours. Dans ce cas, l'AP serveur doit appeler une primitive de service GET-RESPONSE-LAST-BLOCK. Le paramètre Result doit contenir une structure DataBlock_G avec:

- Last-block == TRUE;
- Block-Number == le numéro du bloc reçu du client;
- Result == Data-Access-Result, no-long-get-in-progress.

d) Le numéro de bloc envoyé par le serveur n'est pas égal au numéro suivant de la séquence. Dans ce cas, le client doit abandonner le transfert de blocs [voir situation d'erreur b)].

Si, lors des situations d'erreur décrites ci-dessus, le serveur n'est pas en mesure d'appeler une primitive de service GET-RESPONSE-LAST-BLOCK pour quelque raison que ce soit, il doit appeler une primitive de service GET-RESPONSE-NORMAL, avec le paramètre Data-Access-Result indiquant la raison de l'échec. Le serveur doit envoyer une APDU Get-Response-Normal.

Le MSC de la situation d'erreur b), transfert long abandonné, est représenté à la Figure 45:

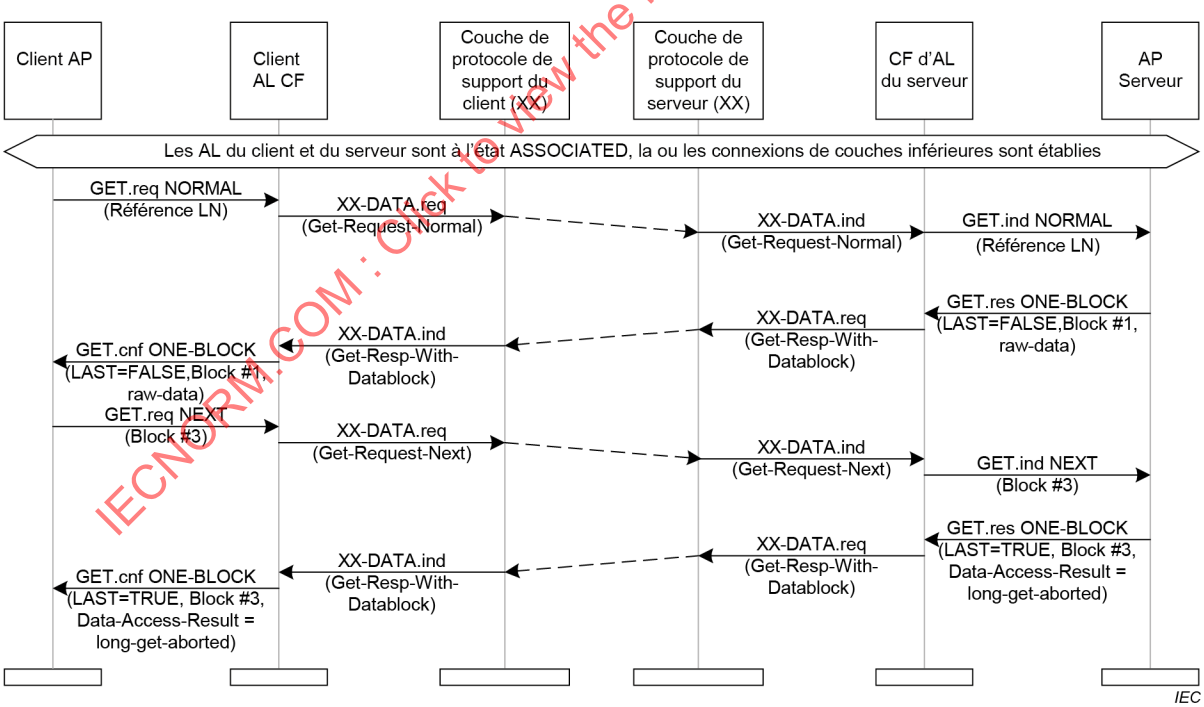



**Figure 45 – MSC du service GET avec transfert de blocs, transfert long abandonné**

### 7.3.4 Protocole du service SET

Lorsque l'AP client souhaite écrire la valeur d'un ou plusieurs attributs d'objet d'interface COSEM, il utilise le service SET.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Comme expliqué en 6.7, la forme codée de la demande peut être contenue dans une seule demande, ou non. Dans ce cas, le mécanisme spécifique au service ou le mécanisme de transfert de bloc général peut être utilisé. Il est négocié par l'intermédiaire du bit 2 ou du bit 12 du bloc de conformité (voir 7.3.1).

NOTE Dans certains profils de communication DLMS®/COSEM, une segmentation est disponible pour le transfert de longues APDU.

Les types de primitives de service SET et les APDU correspondantes sont présentés dans le Tableau 69.

**Tableau 69 – Types de service SET et APDU**

| SET .req/.ind | APDU Request | APDU Response | SET .res/.cnf |
|---|---|---|---|
| NORMAL | Set-Request-Normal | Set-Response-Normal | NORMAL |
| FIRST-BLOCK | Set-Request-With-First-Datablock<br>Last-Block = FALSE | Set-Response-Datablock | ACK-BLOCK |
| ONE-BLOCK | Set-Request-With-Datablock<br>Last-Block = FALSE | | |
| LAST-BLOCK | Set-Request-With-Datablock<br>Last-Block = TRUE | Set-Response-Last-Datablock | LAST-BLOCK |
| | | | LAST-BLOCK-WITH-LIST |
| WITH-LIST | Set-Request-With-List | Set-Response-With-List | WITH-LIST |
| FIRST-BLOCK-WITH-LIST | Set-Request-With-List-And-With-First-Datablock | Set-Response-Datablock | ACK-BLOCK |

La Figure 46 inclut le MSC du service SET confirmé, en cas de succès et sans transfert de bloc.

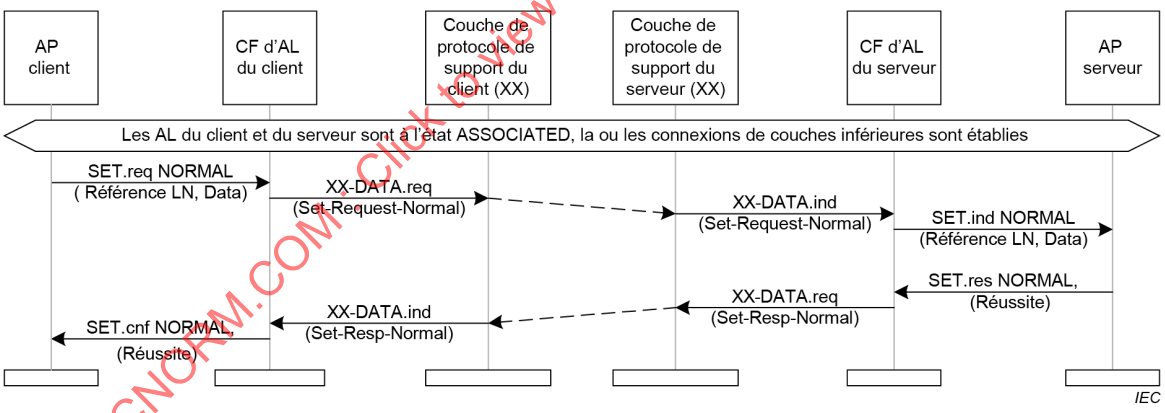



**Figure 46 – MSC du service SET**

La Figure 47 représente le MSC d'un service SET confirmé en cas de succès, avec envoi de la demande en trois blocs.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

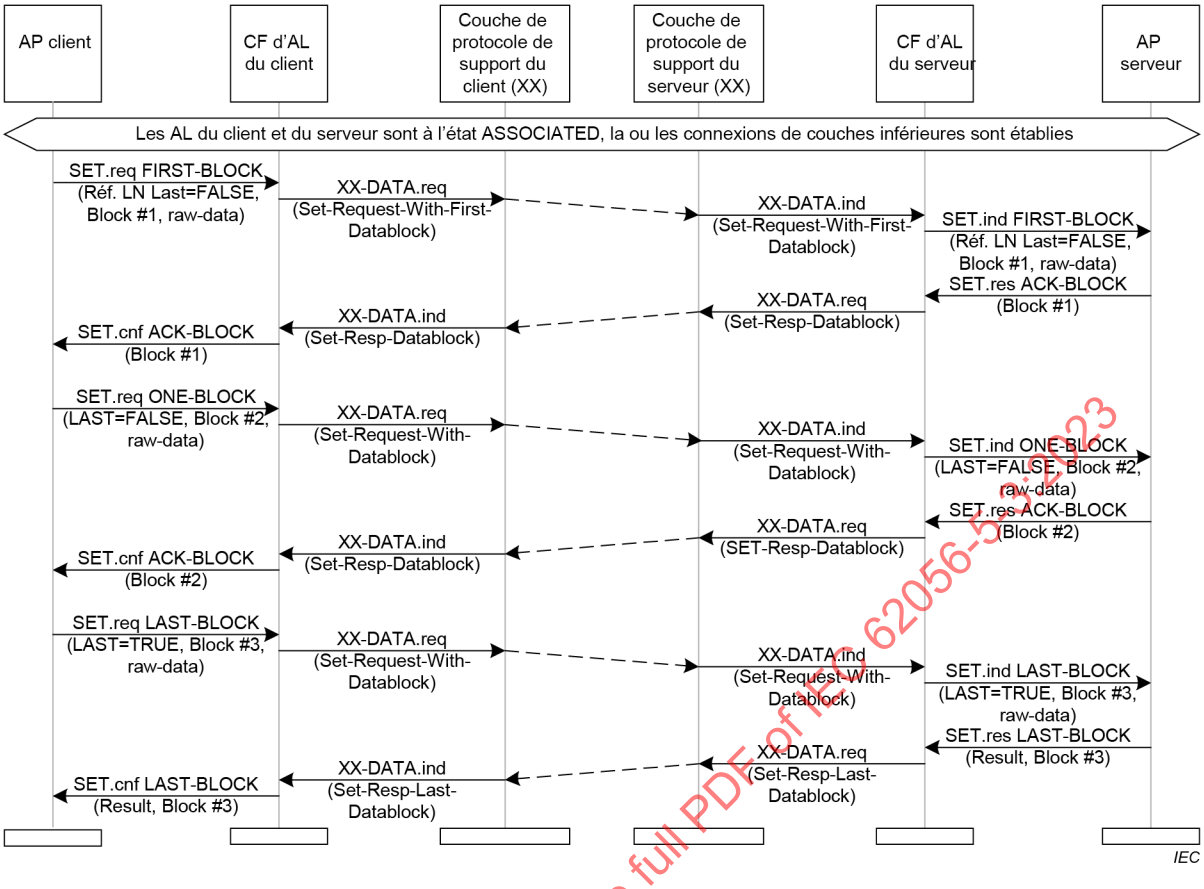



**Figure 47 – MSC du service SET avec transfert de blocs**

Dans ce cas, les données à envoyer sont trop longues pour être contenues dans une seule APDU. L'AP client les envoie donc en plusieurs blocs. Les données sont d'abord codées comme si elles étaient adaptées pour être contenues dans une seule APDU. Le résultat est une série d'octets, $B_1$, $B_2$, $B_3$...., $B_N$. Le client peut générer la demande complète ($B_1$, $B_2$, $B_3$....$B_N$) en une étape, ou de façon dynamique (à la volée).

L'AP client assemble ensuite la structure DataBlock_SA:

- Last_Block == FALSE;
- Block_Number == 1;
- Raw_Data == les K premiers octets de données codées: $B_1$, $B_2$, $B_3$...., $B_K$.

L'AP client appelle alors, suivant le cas, la primitive de service SET-REQUEST-FIRST-BLOCK ou SET-REQUEST-FIRST-BLOCK-WITH-LIST, qui contient la ou les références d'attribut et la structure DataBlock-SA.

À réception de la primitive .request, l'AL du client génère l'APDU Set-Request appropriée contenant les paramètres de la primitive .request et l'envoie au serveur.

Le serveur stocke le bloc de données reçu, puis en accuse réception et demande le bloc suivant en appelant une primitive SET.response avec le paramètre Response_Type == ACK-BLOCK et avec le même numéro de bloc que celui du bloc reçu.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Pour envoyer le bloc de données suivant contenant $B_{K+1}$, $B_{K+2}$, $B_{K+3}$…., $B_{L,}$, l'AP client appelle une primitive de service SET-REQUEST-ONE-BLOCK. Cet échange de blocs de données-accusés de réception se poursuit jusqu'à l'envoi du dernier bloc de données contenant $B_M$, $B_{M+1}$, $B_{M+2}$…., $B_N$ par l'appel de la primitive de service SET-REQUEST-LAST-BLOCK avec le paramètre Last_Block == TRUE.

Lorsque ces primitives sont appelées, l'AL du client génère une APDU Set-Request-With-Datablock contenant une structure DataBlock_SA et envoie ces APDU au serveur.

Lorsque l'AP serveur reçoit le dernier bloc de données, il appelle une primitive de service SET-RESPONSE-LAST-BLOCK ou SET-RESPONSE-LAST-BLOCK-WITH-LIST, suivant le cas. Le paramètre Result contient le résultat de l'appel de service SET complet. Le paramètre Block_Number confirme la réception du dernier bloc.

Tout au long de la procédure, les paramètres Invoke_Id et Priority doivent être identiques dans chaque primitive.

Si lors d'un transfert de données long, le serveur reçoit une autre demande de service, celle-ci est traitée en fonction des règles de priorité et des paramètres de gestion des priorités (bit 9 du bloc de conformité).

Si une erreur se produit lors d'un transfert de données long, le transfert est annulé. Les situations d'erreur sont les suivantes:

a) le serveur n'est pas en mesure de gérer le bloc de données reçu, pour une raison quelconque. Dans ce cas, l'AP serveur doit appeler une primitive de service SET-RESPONSE-LAST-BLOCK ou SET-RESPONSE-LAST-BLOCK-WITH-LIST, suivant le cas. Le paramètre Result indique la raison de l'abandon du transfert;

b) le paramètre Block_Number d'une primitive de service SET-REQUEST-ONE-BLOCK n'est pas égal au numéro de bloc attendu par le serveur (dernier reçu + 1). Le serveur interprète cette erreur comme si le client souhaitait annuler le transfert en cours. Dans ce cas, l'AP serveur doit appeler une primitive de service SET-RESPONSE-LAST-BLOCK ou SET-RESPONSE-LAST-BLOCK-WITH-LIST, suivant le cas, avec le paramètre Result défini comme suit: Data_Access_Result == long-set-aborted;

c) le serveur peut recevoir une APDU Set-Request-With-Datablock lorsqu'aucun transfert de données long n'est en cours. Dans ce cas, l'AP serveur doit appeler une primitive de service SET-RESPONSE-LAST-BLOCK avec le paramètre Result défini comme suit: Data_Access_Result == no-long-set-in-progress.

Si, dans les situations d'erreur décrites ci-dessus, le serveur n'est pas en mesure d'appeler une primitive de service SET-RESPONSE-LAST-BLOCK pour une raison quelconque, il appelle une primitive de service SET-RESPONSE-NORMAL avec le paramètre Data-Access-Result indiquant la raison de l'échec.

### 7.3.5 Protocole du service ACTION

Lorsque l'AP client souhaite appeler une ou plusieurs méthodes d'objet d'interface COSEM, il utilise le service ACTION. Comme expliqué en 6.8, le service ACTION se divise en deux phases.

Si les références de méthode et les paramètres d'appel de méthode ou paramètres de retour ne tiennent pas dans une seule APDU, le mécanisme spécifique au service ou le mécanisme de transfert de bloc général peut être utilisé. Il est négocié via le bit 2 ou le bit 13 du bloc de conformité (voir 7.3.1).

NOTE Dans certains profils de communication DLMS®/COSEM, une segmentation est disponible pour le transfert de longues APDU.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Les types de primitives de service ACTION et les APDU correspondantes sont présentés dans le Tableau 70.

**Tableau 70 – Types de service ACTION et APDU**

| ACTION .req/.ind | APDU Request | APDU Response | SET .res/.cnf |
|---|---|---|---|
| NORMAL | Action-Request-Normal | Action-Response-Normal | NORMAL |
| | | Action-Response-With-Pblock | ONE-BLOCK |
| NEXT | Action-Request-Next-Pblock | Action-Response-With-Pblock | ONE-BLOCK |
| | | Action-Response-With-Pblock | LAST-BLOCK |
| FIRST-BLOCK | Action-Request-With-First-Pblock | Action-Response-Next-Pblock | NEXT |
| ONE-BLOCK | Action-Request-With-Pblock | | |
| LAST-BLOCK | Action-Request-With-Pblock | Action-Response-Normal | NORMAL |
| | | Action-Response-With-Pblock | ONE-BLOCK |
| WITH-LIST | Action-Request-With-List | Action-Response-With-List | WITH-LIST |
| | | Action-Response-With-Pblock | ONE-BLOCK |
| WITH-LIST-AND-FIRST-BLOCK | Action-Request-With-List-And-With-First-Pblock | Action-Response-Next-Pblock | NEXT |

La Figure 48 représente le MSC du service ACTION confirmé, en cas de succès et sans transfert de blocs.

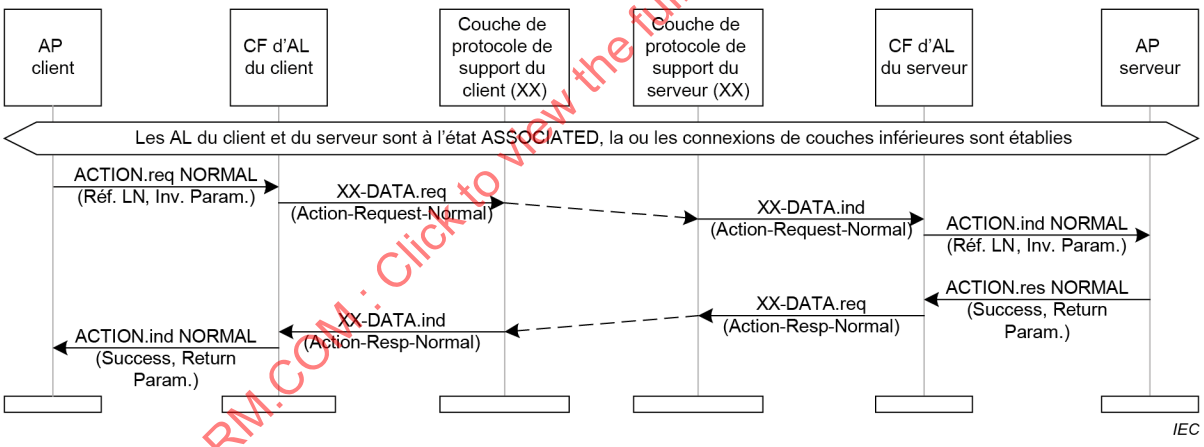



**Figure 48 – MSC du service ACTION**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le service ACTION peut transporter des données dans les deux directions:

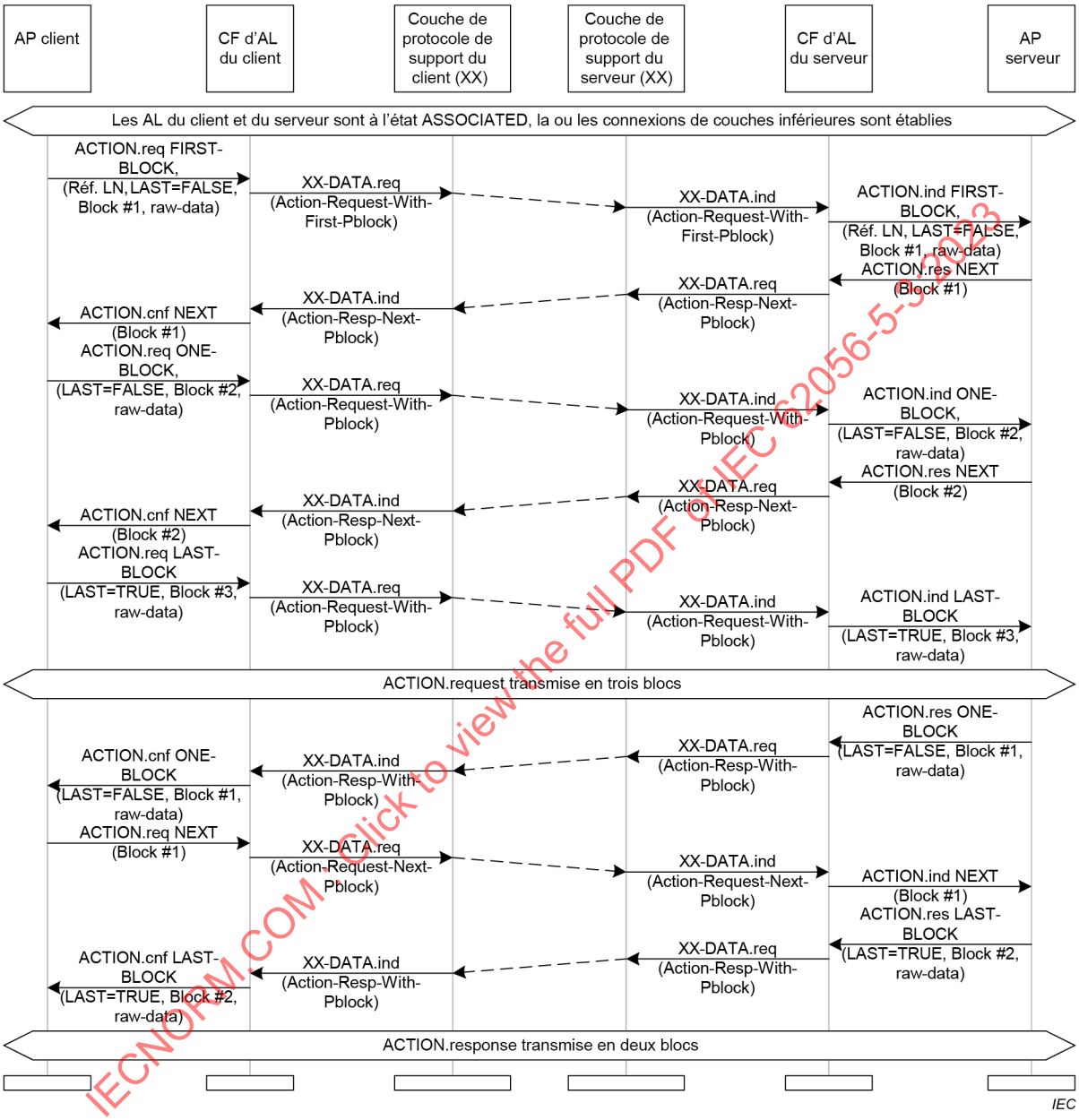
- lors de la première phase, le client envoie la primitive ACTION.request avec les paramètres d'appel de méthode de la ou des méthodes référencées, et le serveur en accuse réception. Le processus est essentiellement le même que dans le cas du service SET;
- lors de la deuxième phase, le serveur envoie la primitive ACTION.response avec le résultat de l'appel de méthode et les paramètres de retour. Le processus est essentiellement le même que dans le cas du service GET.

Tout au long de la procédure, les paramètres Invoke_Id et Priority doivent être identiques dans chaque primitive.

Si lors d'un transfert de données long, le serveur reçoit une autre demande de service, celle-ci est traitée en fonction des règles de priorité et des paramètres de gestion des priorités (bit 9 du bloc de conformité).

La Figure 49 représente le MSC correspondant au transfert de blocs dans les deux directions.

Si une erreur se produit lors d'un transfert de données long, le transfert doit être annulé. Les situations d'erreur sont identiques à celles des services GET et SET.




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure 49 – MSC du service ACTION avec transfert de bloc**

### 7.3.6 Protocole du service ACCESS

Le client peut utiliser le service ACCESS pour lire ou écrire la valeur d'un ou plusieurs attributs d'objet COSEM ou pour appeler une ou plusieurs méthodes.

Le protocole du service ACCESS est spécifié par des tableaux de séquences de messages, y compris les cas dans lesquels il est utilisé avec le transfert de bloc général et la protection générale de messages.

NOTE Voir aussi 4.2.4.4.7, 6.5 et 0.

La Figure 50 représente le MSC d'un service ACCESS utilisé pour obtenir les valeurs d'attributs d'un objet COSEM. La demande est adaptée pour être contenue dans une seule APDU. La réponse est trop longue. Le serveur la renvoie donc à l'aide du mécanisme GBT: des parties de l'APDU access-response sont contenues par le champ block-data des APDU de transfert de bloc général.

Lorsque le client reçoit la première APDU de transfert de bloc général, il passe au GBT et annonce que la diffusion en flux avec fenêtre de taille 3 est prise en charge.

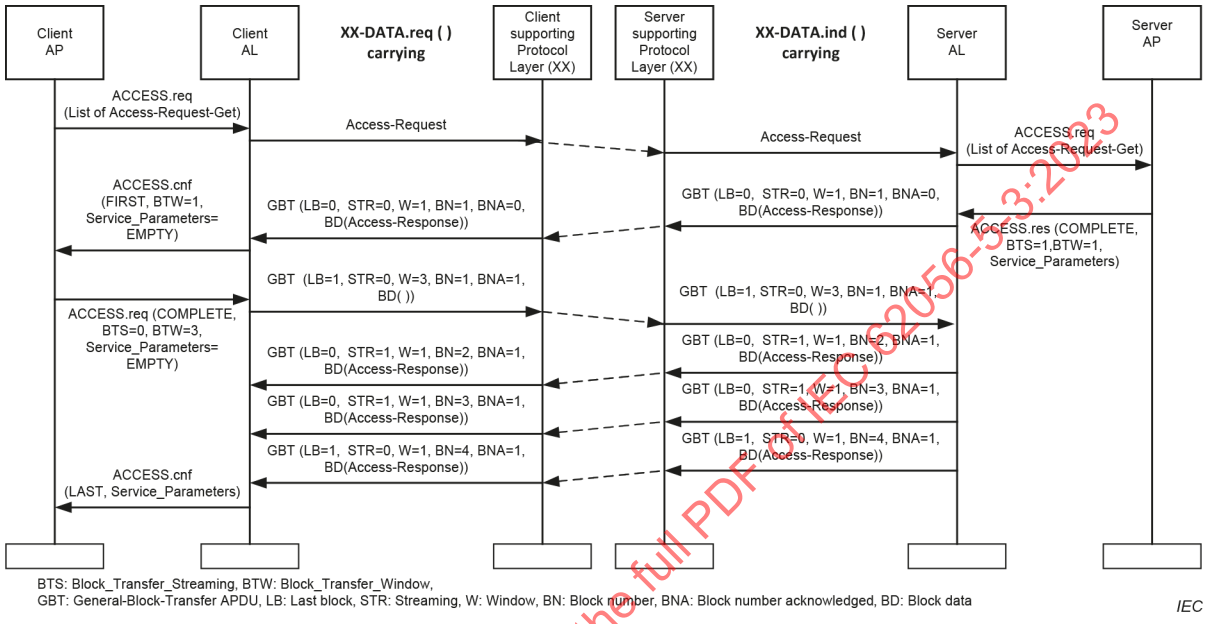



**Figure 50 – Service ACCESS avec réponse longue**

La Figure 51 représente le MSC d'un service ACCESS utilisé pour contenir une liste de demandes, qui ne tient pas dans une seule APDU. Par conséquent, le GBT est utilisé. Comme la réponse est longue, le serveur utilise également le GBT. Les deux parties savent a priori que leur homologue prend en charge le flux avec une taille de fenêtre = 3.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

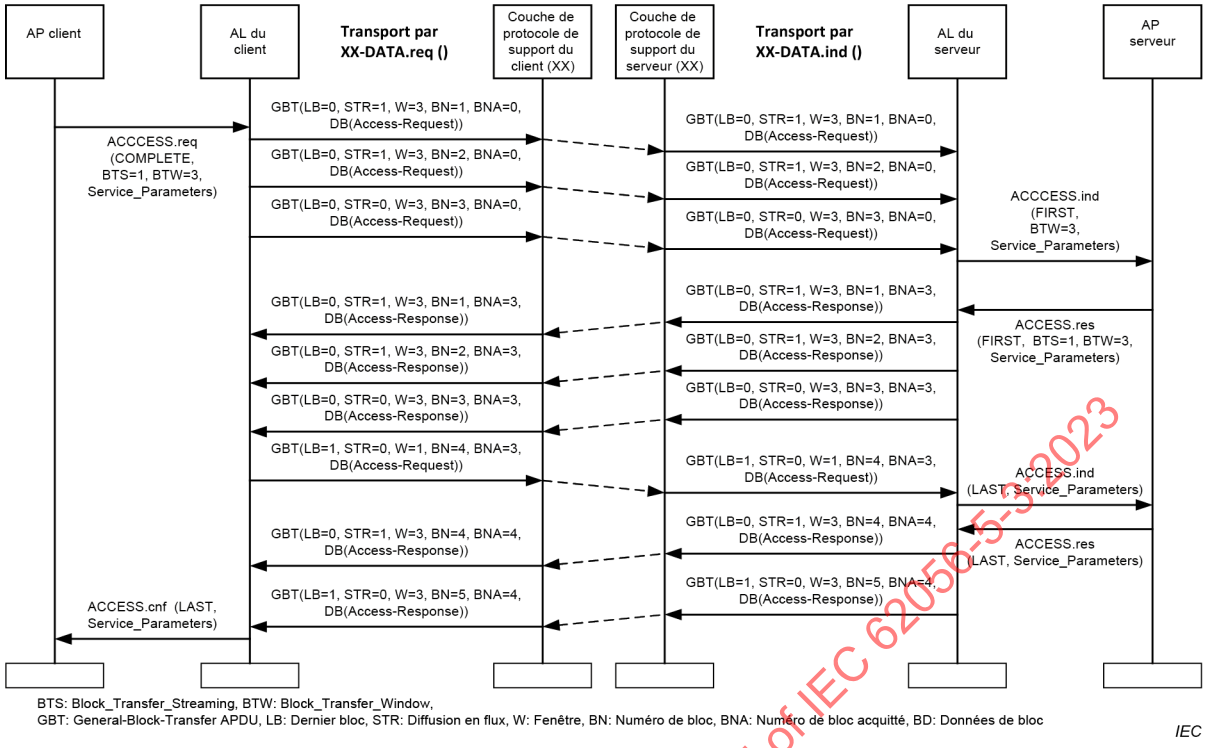



**Figure 51 – Service ACCESS avec demande et réponse longues**

### 7.3.7 Protocole du service DataNotification

Lorsque l'AP serveur appelle une primitive de service DataNotification.request, l'AL du serveur construit l'APDU DataNotification et l'envoie à la couche de protocole de support.

Lorsque l'AL du client reçoit cette APDU, il appelle la primitive de service DataNotification.indication.

Dans la situation a), non confirmée, réessayer en cas de défaillance de la couche de protocole de support (voir 7.3.2.2):

- l'opération Push est considérée comme étant défaillante lorsque l'AP serveur reçoit une primitive de service DataNotification.cnf avec Service_Class == Unconfirmed et Result == LOWER_LAYER_FAILED. Dans ce cas, l'AP serveur peut effectuer une nouvelle tentative après un retard de répétition;
- l'opération Push est considérée comme réussie lorsque l'AP serveur reçoit un DataNotification.cnf avec Service_Class == Unconfirmed et Result == CONFIRMED;

La Figure 52 représente le MSC du service DataNotification pour cette situation.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

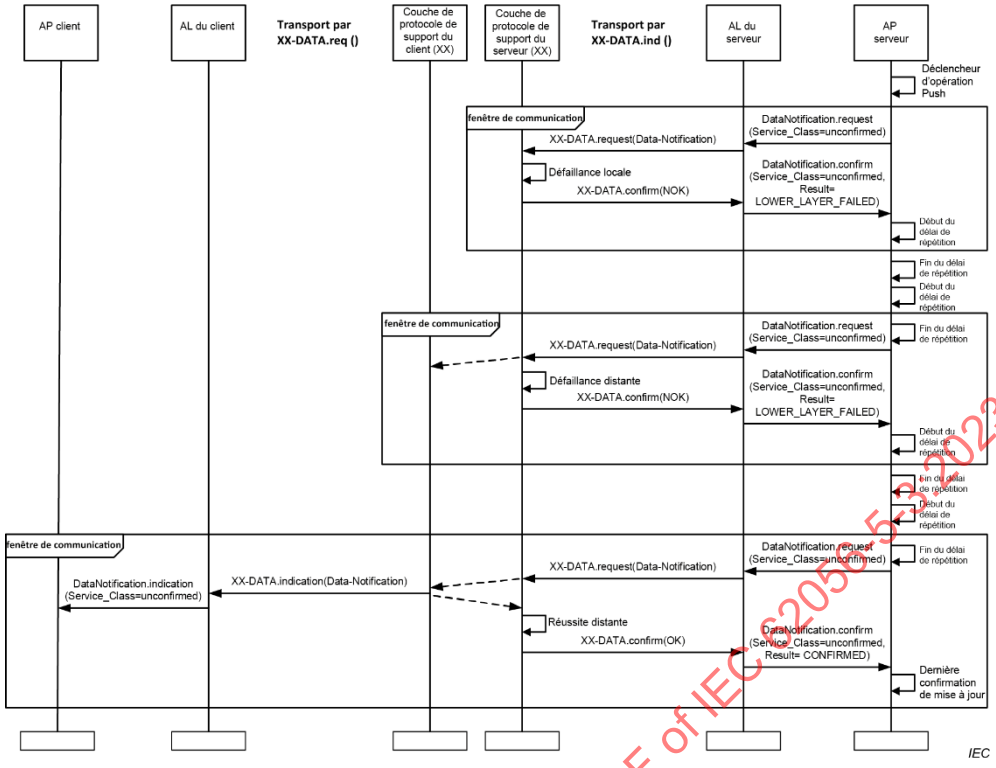



**Figure 52 – MSC pour le service DataNotification, situation a)**

Dans la situation b), non confirmée, réessayer en cas d'absence de confirmation de la couche de protocole de support (voir 7.3.2.2):

- l'opération Push est considérée comme étant défaillante si le retard de répétition expire avant qu'une confirmation ne soit reçue. Dans ce cas, le serveur peut effectuer une nouvelle tentative;
- l'opération Push est considérée comme réussie lorsque la couche de protocole de support du serveur informe l'AL que l'APDU DataNotification a été reçue avec succès par la couche de protocole de support distante. Dans ce cas, l'AL du serveur appelle une primitive de service DataNotification.cnf avec Service_Class == Unconfirmed et Result == CONFIRMED.

La Figure 53 représente le MSC du service DataNotification pour cette situation.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

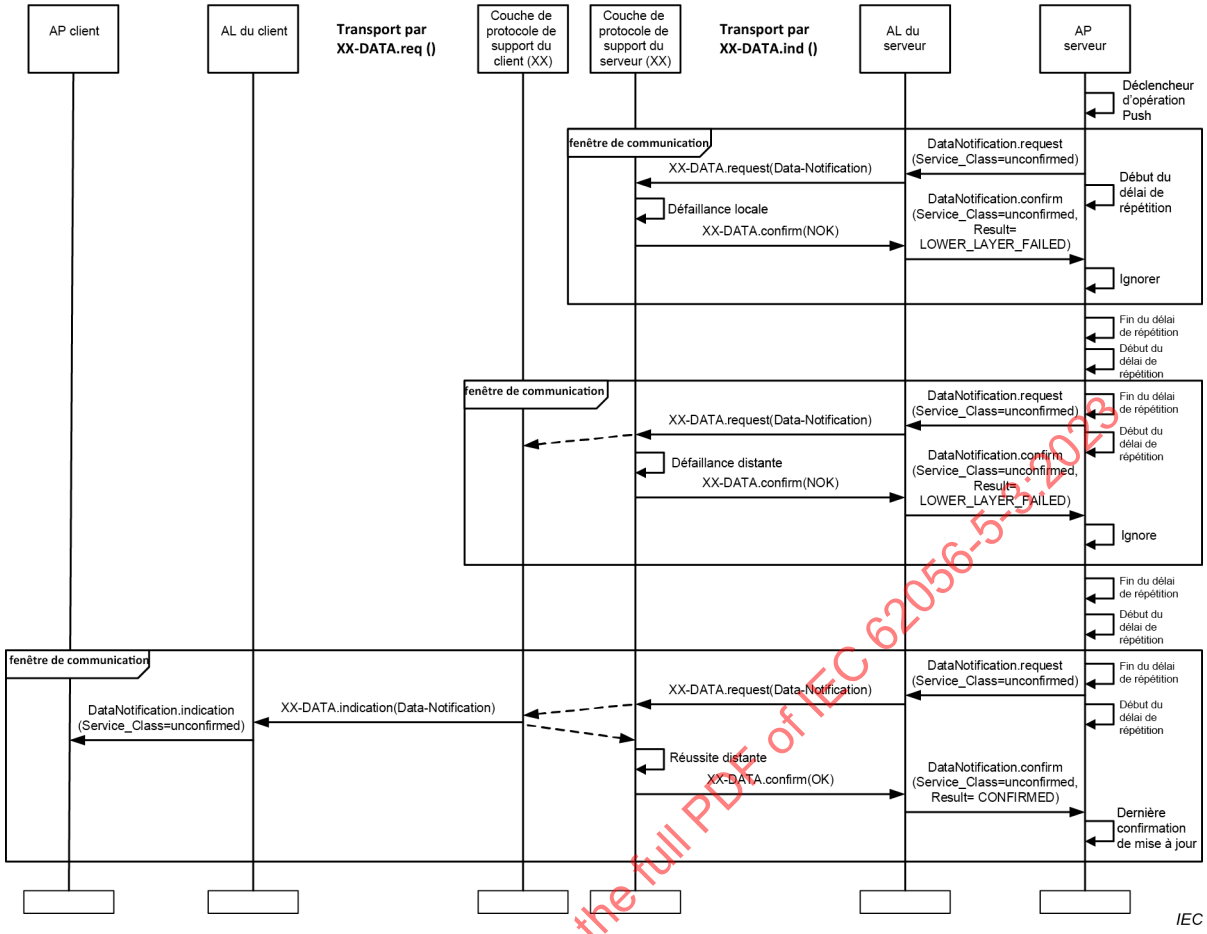



**Figure 53 – MSC pour le service DataNotification, situation b)**

Dans la situation c), confirmée, réessayer en cas d'absence de confirmation (voir 7.3.2.2):

- lorsque l'AL du client reçoit une APDU DataNotification appropriée, il appelle la primitive de service DataNotification.ind. Si l'APDU DataNotification ne peut pas être traitée pour une quelconque raison, elle est rejetée;
- l'AP client appelle la primitive de service DataNotification.response avec Result == Confirmed. L'AL du client génère ensuite l'APDU Data-Notification-Confirm et l'envoie au serveur;
- l'opération Push est considérée comme étant défaillante si le retard de répétition expire avant qu'une confirmation ne soit reçue. Dans ce cas, le serveur peut effectuer une nouvelle tentative;
- l'opération Push est considérée comme réussie lorsque la couche AL reçoit l'APDU Data-Notification-Confirm et appelle la primitive de service DataNotification.cnf avec Service_Class == Confirmed et Result == CONFIRMED.

La Figure 54 représente le MSC du service DataNotification pour cette situation.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

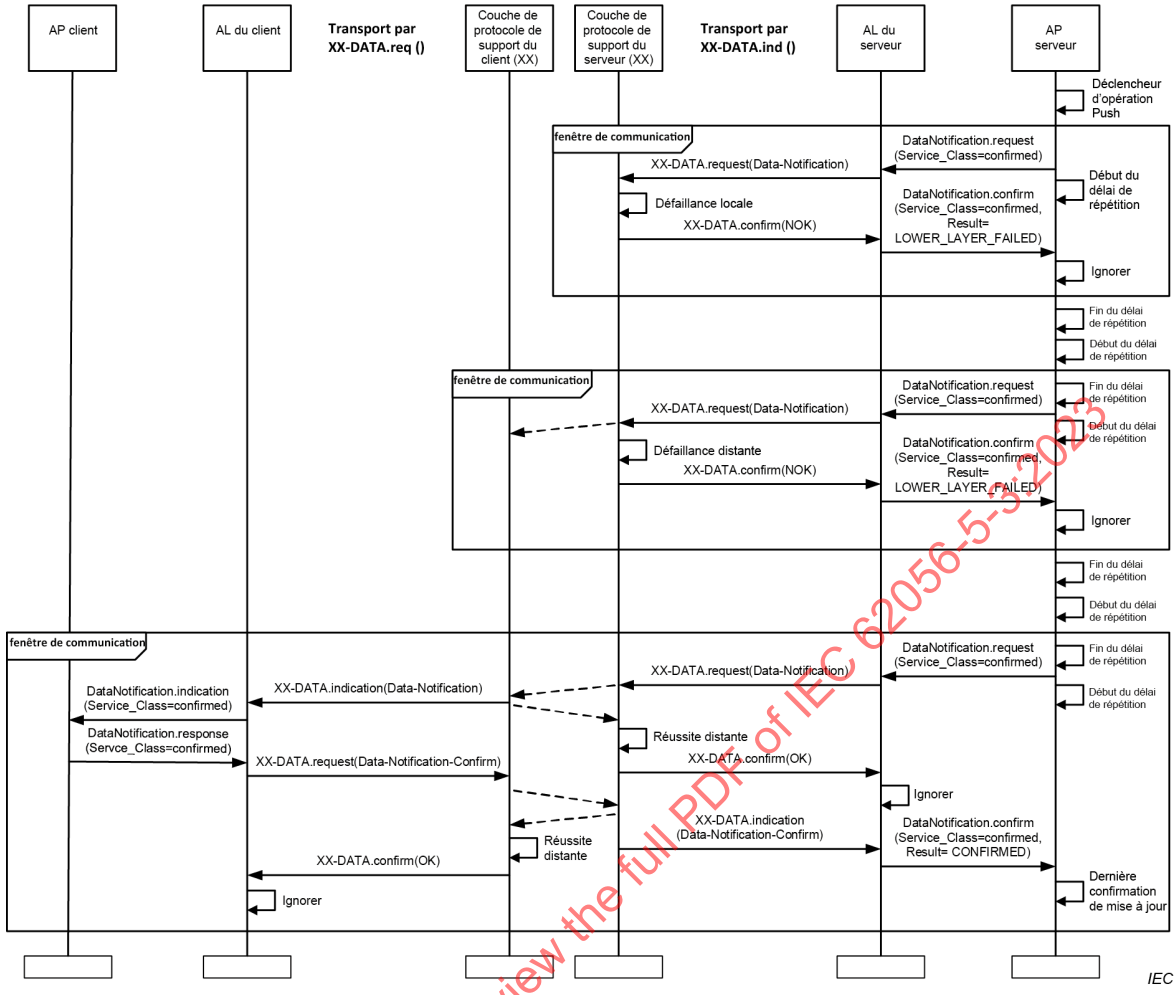



**Figure 54 – MSC pour le service DataNotification, situation c)**

### 7.3.8 Protocole du service EventNotification

À l'appel du service EventNotification.request, l'AL du serveur génère une APDU EventNotificationRequest. Les possibilités d'envoi de cette APDU dépendent du profil de communication et du statut de connexion des couches inférieures. Par conséquent, le protocole du service EventNotification est présenté plus en détail dans les parties de l'IEC 62056 qui décrivent les profils de communication.

Dans tous les cas, pour envoyer la ou les valeurs du ou des attributs au client, sans que le client les demande:

- le serveur utilise la primitive de service EventNotification.request;
- à l'appel de cette primitive, l'AL du serveur génère une APDU EventNotificationRequest;
- cette APDU est transmise au client par le service de couche de support à la première occasion. Le type de service et la disponibilité de cette première occasion dépendent du profil de communication utilisé;
- à réception de l'APDU EventNotificationRequest, l'AL du client génère une primitive EventNotification.indication sur l'AP client COSEM;

NOTE Côté client, il s'agit toujours de la primitive EventNotification.indication, indépendamment du schéma de référencement (LN ou SN) utilisé par le serveur.

- par défaut, les notifications d'événement sont envoyées à partir du dispositif logique de gestion (serveur) vers l'AP de gestion (client).

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 7.3.9 Protocole du service Read

Comme expliqué en 6.14, le service Read est utilisé lorsque le serveur applique le référencement SN, soit pour lire un ou des attributs d'objet d'interface COSEM, soit pour appeler une ou des méthodes lorsque des paramètres de retour sont attendus:

- dans le premier cas, les primitives de service GET.request sont mises en correspondance avec les primitives Read.request et les primitives Read.confirm sont mises en correspondance avec les primitives GET.confirm. La mise en correspondance et les APDU SN correspondantes sont présentées dans le Tableau 71;
- dans le deuxième cas, les primitives de service ACTION.request sont mises en correspondance avec les primitives Read.request et les primitives Read.response sont mises en correspondance avec les primitives ACTION.response. La mise en correspondance et les APDU SN correspondantes sont présentées dans le Tableau 72;

NOTE Dans les tableaux de mise en correspondance ci-dessous, la notation suivante est utilisée:

- pour les services LN, seuls les types de demandes et de réponses sont indiqués sans paramètres de service;
- pour les services SN, le nom de la primitive de service est suivi par les paramètres de service, entre parenthèses. Les éléments de nom de paramètre de service sont en majuscules et sont associés par un trait de soulignement afin de ne constituer qu'une seule entité. Les paramètres pouvant être répétés sont entre accolades. Les choix possibles pour le paramètre Variable_Access_Specification sont répertoriés après le symbole "=". Les différents choix sont séparés par une barre verticale "|";
- pour les APDU SN, le nom de l'APDU est suivi du symbole "=" et les champs sont entre parenthèses. Les éléments de nom de champ ne sont pas en majuscules et sont associés par un tiret afin de constituer une seule entité. Les champs pouvant être répétés sont entre accolades. Les différents choix sont séparés par une barre verticale "|".

**Tableau 71 – Mise en correspondance du service GET et du service Read**

| De la primitive GET.request de type | À la primitive Read.request | APDU SN |
|---|---|---|
| NORMAL | Read.request (Variable_Access_Specification)<br>Variable_Access_Specification = Variable_Name \| Parameterized_Access; | ReadRequest::= (variable-name \| parameterized-access) |
| NEXT | Read.request (Variable_Access_Specification)<br>Variable_Access_Specification = Block_Number_Access; | ReadRequest::= (block-number-access) |
| WITH-LIST | Read.request ({Variable_Access_Specification})<br>Variable_Access_Specification = Variable_Name \| Parameterized_Access; | ReadRequest::= ({variable-name \| parameterized-access}) |
| **À la primitive GET.confirm de type** | **APDU SN** | **De la primitive Read.response** |
| NORMAL | ReadResponse::= (data \| data-access-error) | Read.response (Data \| Data_Access_Error) |
| ONE-BLOCK | ReadResponse::= (data-block-result) | Read.response (Data_Block_Result) avec Last_Block = FALSE |
| LAST-BLOCK | ReadResponse::= (data-block-result) | Read.response (Data_Block_Result) avec Last_Block = TRUE |
| WITH-LIST | ReadResponse::= ({data \| data-access-error}) | Read.response ({Data \| Data_Access_Error}) |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 72 – Mise en correspondance du service ACTION et du service Read**

| De la primitive ACTION.request de type | À la primitive Read.request | APDU SN |
|---|---|---|
| NORMAL | Read.request<br>(Variable_Access_Specification)<br><br>Variable_Access_Specification =<br>Parameterized_Access;<br><br>avec<br>Variable_Name = référence de méthode,<br>Selector = 0,<br>Parameter = paramètre d'appel de méthode ou données NULL | ReadRequest::=<br>(parameterized-access) |
| NEXT | Read.request<br>(Variable_Access_Specification)<br><br>Variable_Access_Specification =<br>Block_Number_Access; | ReadRequest:: =<br>(block-number-access) |
| FIRST-BLOCK | Read.request<br>(Variable_Access_Specification)<br><br>Variable_Access_Specification =<br>Read_Data_Block_Access;<br><br>avec<br>Last_Block = FALSE,<br>Block_Number = 1,<br>Raw_Data = une partie des références de méthode et paramètre d'appel de méthode | ReadRequest::=<br>(read-data-block-access) |
| ONE-BLOCK | Read.request<br>(Variable_Access_Specification)<br><br>Variable_Access_Specification =<br>Read_Data_Block_Access;<br><br>avec<br>Last_Block = FALSE,<br>Block_Number = numéro suivant,<br>Raw_Data = voir ci-dessus | ReadRequest::=<br>(read-data-block-access) |
| LAST-BLOCK | Read.request<br>(Variable_Access_Specification)<br><br>Variable_Access_Specification =<br>Read_Data_Block_Access;<br><br>avec<br>Last_Block = TRUE,<br>Block_Number = numéro suivant,<br>Raw_Data = voir ci-dessus | ReadRequest::=<br>(read-data-block-access) |
| WITH-LIST | Read.request<br>({Variable_Access_Specification})<br><br>Variable_Access_Specification =<br>Parameterized_Access;<br><br>avec<br>Variable_Name = référence de méthode,<br>Selector = 0,<br>Parameter = paramètre d'appel de méthode ou données NULL | ReadRequest::=<br>({parameterized-access}) |
| WITH-LIST-AND-FIRST-BLOCK | Read.request<br>(Variable_Access_Specification)<br><br>Variable_Access_Specification =<br>Read_Data_Block_Access;<br><br>avec<br>Last_Block = FALSE,<br>Block_Number = 1,<br>Raw_Data = voir ci-dessus | ReadRequest::=<br>(read-data-block-access) |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| À la primitive ACTION.confirm | APDU SN | De la primitive Read.response |
|---|---|---|
| NORMAL | ReadResponse::=<br>(data I data-access-error) | Read.response (Read_Result)<br>Read_Result =<br>Data I Data_Access_Error; |
| ONE-BLOCK | ReadResponse::=<br>(data-block-result) | Read.response (Read_Result)<br>Read_Result = Data_Block_Result;<br>avec Last_Block = FALSE |
| LAST-BLOCK | ReadResponse::=<br>(data-block-result) | Read.response (Read_Result)<br>Read_result = Data_Block_Result;<br>avec Last_Block = TRUE |
| NEXT | ReadResponse::=<br>(block-number) | Read.confirm (Read_Result)<br>Read_Result = Block_Number; |
| WITH-LIST | ReadResponse::=<br>({data I data-access-error}) | Read.response ({Read_Result})<br>Read_Result =<br>Data I Data_Access_Error; |

La Figure 55 représente le MSC d'un service Read utilisé pour lire la valeur d'un attribut unique.

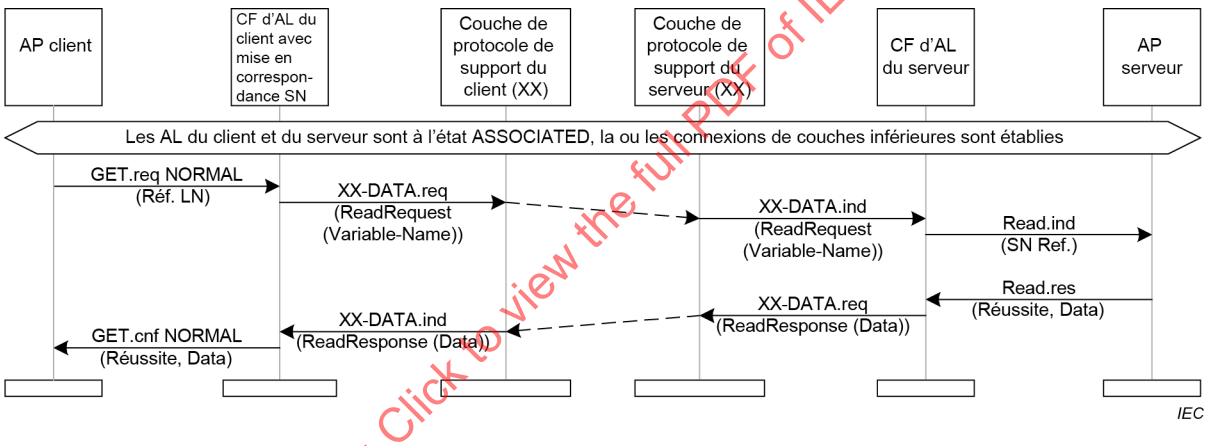



**Figure 55 – MSC du service Read utilisé pour lire un attribut**

La Figure 56 représente le MSC d'un service Read utilisé pour appeler une méthode unique.

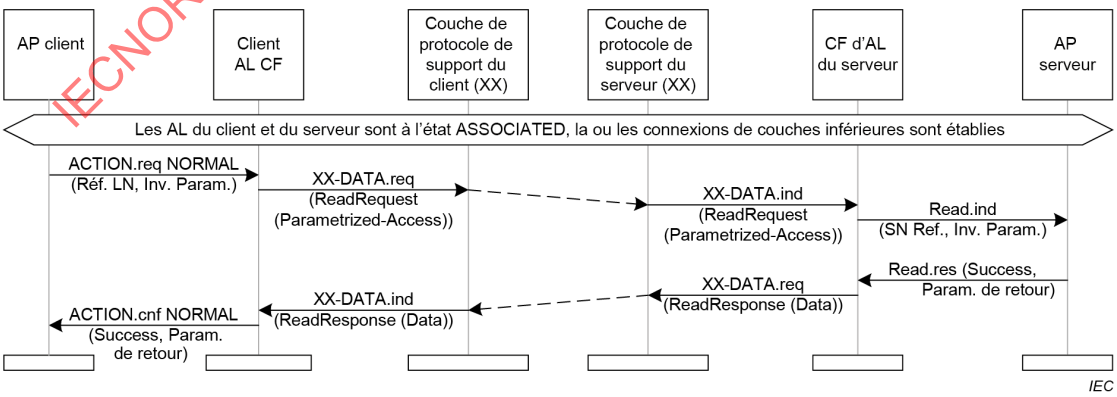



**Figure 56 – MSC du service Read utilisé pour appeler une méthode**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La Figure 57 représente le MSC du service Read utilisé pour lire un attribut unique, le résultat étant retourné en trois blocs à l'aide du mécanisme de transfert de blocs spécifique au service.

En variante, le mécanisme de transfert de bloc général peut être utilisé.

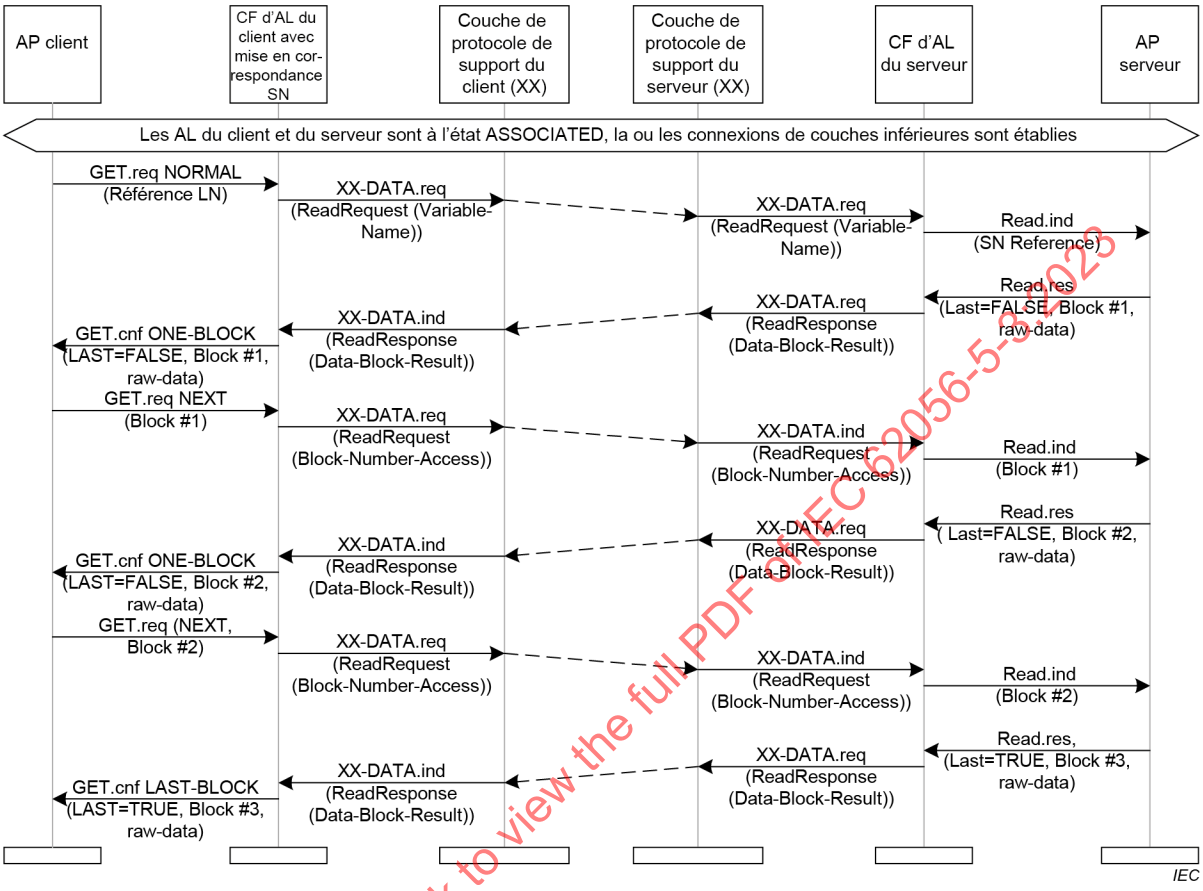



**Figure 57 – MSC du service Read utilisé pour lire un attribut, avec transfert de blocs**

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

Le processus de préparation et de transport de données longues est essentiellement le même que pour les services GET et ACTION:

- si le service Read est utilisé pour lire la valeur d'un ou de plusieurs attributs d'objet COSEM, l'élément Raw_Data du résultat Data_Block_Result contient une partie de la liste des résultats Read_Result;
- si le service Read est utilisé pour appeler une ou plusieurs méthodes d'objet COSEM et si de longs paramètres d'appel de méthodes sont à envoyer, l'élément Raw_Data du bloc Read_Data_Block_Access contient une partie de la ou des références de méthode et du ou des paramètres d'appel de méthode. Si de longues réponses aux appels de méthode sont retournées, l'élément Raw_Data du bloc Data_Block_Result contient une partie de la ou des réponses d'appel de méthode.

Si une erreur se produit, il convient que le serveur retourne une primitive de service Read.response avec l'élément Data_Access_Error contenant les informations de diagnostic appropriées; par exemple, data-block-number-invalid.

### 7.3.10 Protocole du service Write

Comme expliqué en 6.15, le service Write est utilisé lorsque le serveur utilise le référencement SN, soit pour écrire un ou des attributs d'objet COSEM, soit pour appeler une ou des méthodes lorsqu'aucun paramètre de retour n'est attendu:

- dans le premier cas, les primitives de service SET.request sont mises en correspondance avec les primitives Write.request et les primitives Write.confirm sont mises en correspondance avec les primitives SET.confirm. La mise en correspondance et les APDU SN correspondantes sont présentées dans le Tableau 73;
- dans le deuxième cas, les primitives de service ACTION.request sont mises en correspondance avec les primitives Write.request et les primitives Write.response sont mises en correspondance avec les primitives ACTION.confirm. La mise en correspondance et les APDU SN correspondantes sont présentées dans le Tableau 74.

**Tableau 73 – Mise en correspondance du service SET et du service Write**

| De la primitive SET.request de type | À la primitive Write.request | APDU SN |
|---|---|---|
| NORMAL | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Variable_Name I Parameterized_Access; | WriteRequest::= (variable-name I parameterized-access, data) |
| FIRST-BLOCK | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Write_Data_Block_Access;<br>avec<br>Last_Block = FALSE,<br>Block_Number = 1,<br>Data = raw-data, contenant la forme codée de la ou des références d'attribut et les données d'écriture. | WriteRequest::= (write-data-block-access, data) |
| ONE-BLOCK | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Write_Data_Block_Access;<br>avec<br>Last_Block = FALSE,<br>Block_Number = numéro suivant,<br>Data = voir ci-dessus | WriteRequest::= (write-data-block-access, data) |
| LAST-BLOCK | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Write_Data_Block_Access;<br>avec<br>Last_Block = TRUE,<br>Block_Number = numéro suivant,<br>Data = voir ci-dessus | WriteRequest::= (write-data-block-access, data) |
| WITH-LIST | Write.request ({Variable_Access_Specification}, {Data})<br>Variable_Access_Specification = Variable_Name I Parameterized_Access; | WriteRequest::= ({variable-name I parameterized-access}, {data}) |
| FIRST-BLOCK-WITH-LIST | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Write_Data_Block_Access;<br>avec<br>Last_Block = FALSE,<br>Block_Number = 1<br>Data = voir ci-dessus | WriteRequest::= (write-data-block-access, data) |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| De la primitive SET.request de type | À la primitive Write.request | APDU SN |
|---|---|---|
| **À la primitive SET.confirm de type** | **APDU SN** | **De la primitive Write.response** |
| NORMAL | WriteResponse::= (success I data-access-error) | Write.response (Write_Result)<br>Write_Result = Success I Data_Access_Error; |
| ACK-BLOCK | WriteResponse::= (block-number) | Write.response (Write_Result)<br>Write_Result = Block_Number; |
| LAST-BLOCK | WriteResponse::= (success I data-access-error) | Write.response (Write_Result)<br>Write_Result = Success I Data_Access_Error; |
| WITH-LIST | WriteResponse::= ({success I data-access-error}) | Write.response ({Write_Result})<br>Write_Result = Success I Data_Access_Error; |
| LAST-BLOCK-WITH-LIST | WriteResponse::= ({success I data-access-error}) | Write.response ({Write_Result})<br>Write_Result = Success I Data_Access_Error; |

**Tableau 74 – Mise en correspondance du service ACTION et du service Write**

| De la primitive ACTION.request de type | À la primitive Write.request | APDU SN |
|---|---|---|
| NORMAL | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Variable_Name;<br>Data = paramètres d'appel de méthode ou données NULL | WriteRequest::= (variable-name, data) |
| FIRST-BLOCK | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Write_Data_Block_Access;<br>avec<br>Last_Block = FALSE,<br>Block_Number = 1,<br>Data = raw-data, contenant la forme codée de la ou des références de méthode et les paramètres d'appel de méthode; | WriteRequest::= (write-data-block-access, data) |
| ONE-BLOCK | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Write_Data_Block_Access;<br>avec<br>Last_Block = FALSE,<br>Block_Number = numéro suivant,<br>Data = voir ci-dessus | WriteRequest::= (write-data-block-access, data) |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

| De la primitive ACTION.request de type | À la primitive Write.request | APDU SN |
|---|---|---|
| LAST-BLOCK | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Write_Data_Block_Access;<br>avec<br>Last_Block = TRUE,<br>Block_Number = numéro suivant,<br>Data = voir ci-dessus | WriteRequest::=<br>(write-data-block-access, data) |
| WITH-LIST | Write.request ({Variable_Access_Specification}, {Data})<br>Variable_Access_Specification = Variable_Name;<br>Data = paramètres d'appel de méthode ou données NULL | WriteRequest::=<br>({variable-name}, {data}) |
| WITH-LIST-AND-FIRST-BLOCK | Write.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Write_Data_Block_Access;<br>avec<br>Last_Block = FALSE,<br>Block_Number = 1,<br>Data = identique au premier bloc | WriteRequest::=<br>(write-data-block-access, data) |
| **À la primitive ACTION.confirm** | **APDU SN** | **De la primitive Write.response** |
| NORMAL | WriteResponse::=<br>(success I data-access-error) | Write.response (Write_Result)<br>Write_Result =<br>Success I Data_Access_Error; |
| NEXT | WriteResponse::= (block-number) | Write.response (Block_Number) |
| **À la primitive ACTION.confirm** | **APDU SN** | **De la primitive Write.response** |
| WITH-LIST | WriteResponse::=<br>({success I data-access-error}) | Write.response ({Write_Result})<br>Write_Result =<br>Success I Data_Access_Error; |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La Figure 58 représente le MSC d'un service Write utilisé pour écrire la valeur d'un seul attribut, en cas de réussite.

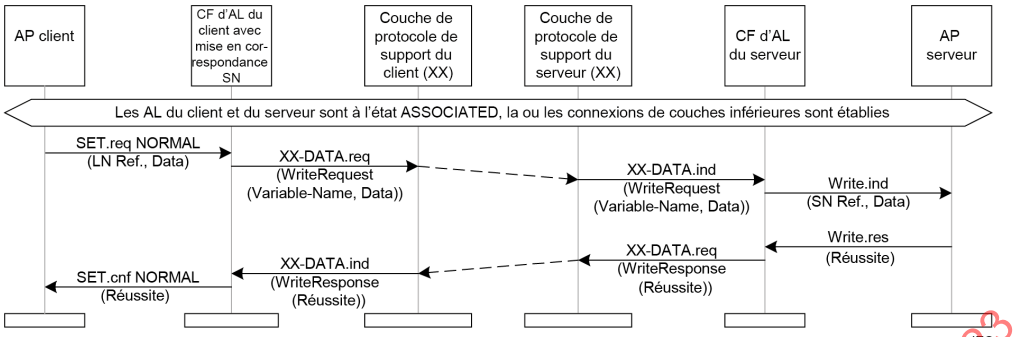



**Figure 58 – MSC du service Write utilisé pour écrire un attribut**

La Figure 59 représente le MSC d'un service Write utilisé pour appeler une méthode unique, en cas de réussite.

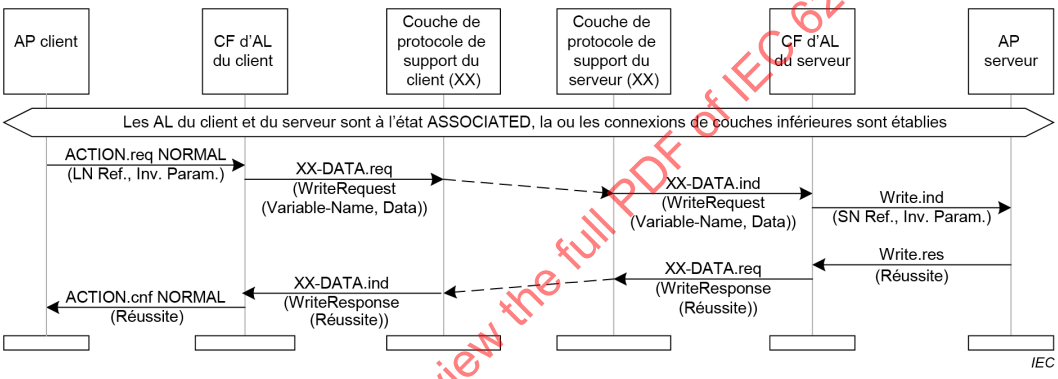



**Figure 59 – MSC du service Write utilisé pour appeler une méthode**

La Figure 60 représente le MSC du service Write utilisé pour écrire un attribut unique, le résultat étant retourné en trois blocs à l'aide du mécanisme de transfert de blocs spécifique au service.

En variante, le mécanisme de transfert de bloc général peut être utilisé.

Le processus de préparation et de transport de données longues est essentiellement le même que pour les services SET et ACTION.

Si une erreur se produit, il convient que le serveur retourne une primitive de service Write.response avec l'élément Data_Access_Error contenant les informations de diagnostic appropriées; par exemple, data-block-number-invalid.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

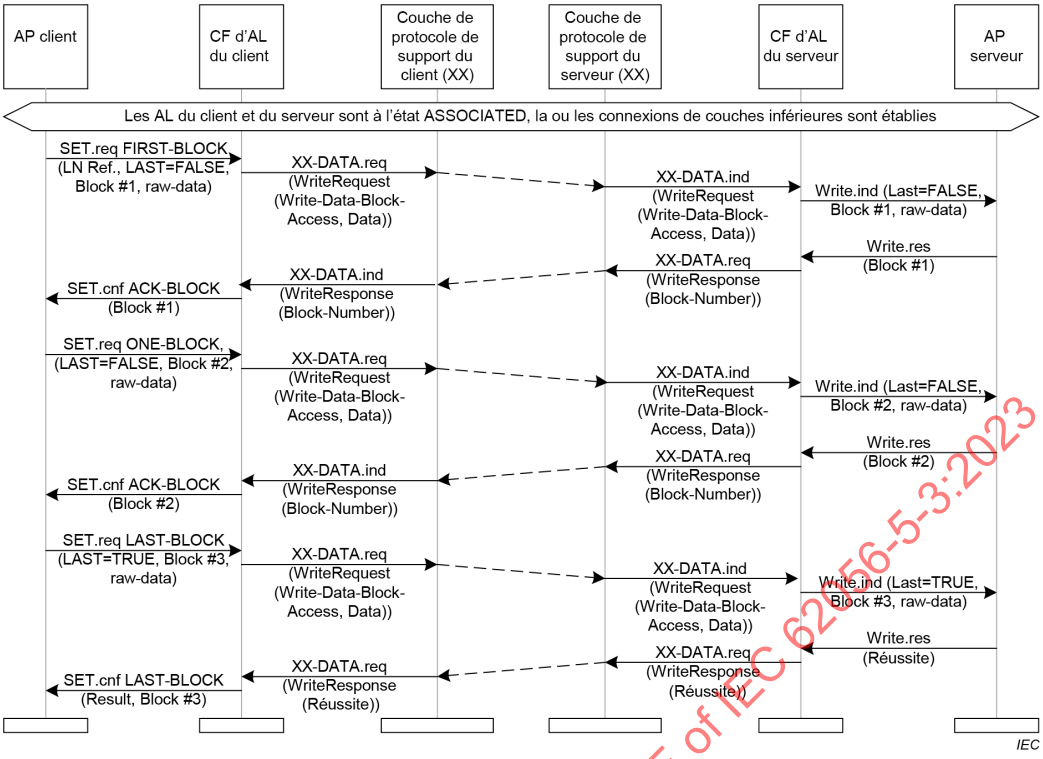



**Figure 60 – MSC du service Write utilisé pour écrire un attribut, avec transfert de blocs**

### 7.3.11 Protocole du service UnconfirmedWrite

Ce service peut être appelé uniquement si une AA a déjà été établie. En fonction du profil de communication, l'APDU correspondant à la demande peut être transportée à l'aide des services de données sans connexion (CL) ou orientés connexion (CO) de la couche de protocole de support.

Comme expliqué en 6.16, le service UnconfirmedWrite peut être utilisé soit pour écrire un ou des attributs d'objet COSEM, soit pour appeler une ou des méthodes lorsqu'aucun paramètre de retour n'est attendu:

- dans le premier cas, les primitives de service SET.request sont mises en correspondance avec les primitives UnconfirmedWrite.request. La mise en correspondance et les APDU SN correspondantes sont présentées dans le Tableau 75;
- dans le deuxième cas, les primitives de service ACTION.request sont mises en correspondance avec les primitives UnconfirmedWrite.request. La mise en correspondance et les APDU SN correspondantes sont présentées dans le Tableau 76.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Tableau 75 – Mise en correspondance du service SET et du service UnconfirmedWrite**

| De la primitive SET.request de type | À la primitive UnconfirmedWrite.request | APDU SN |
|---|---|---|
| NORMAL | UnconfirmedWrite.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Variable_Name I Parameterized_Access; | UnconfirmedWriteRequest::= (variable-name I parameterized-access, data) |
| WITH-LIST | UnconfirmedWrite.request ({Variable_Access_Specification}, {Data})<br>Variable_Acces_Specification = Variable_Name I Parameterized_Access; | UnconfirmedWriteRequest::= ({variable-name I parameterized-access}, {data}) |

**Tableau 76 – Mise en correspondance du service ACTION et du service UnconfirmedWrite**

| De la primitive ACTION.request de type | À la primitive UnconfirmedWrite.request | APDU SN |
|---|---|---|
| NORMAL | UnconfirmedWrite.request (Variable_Access_Specification, Data)<br>Variable_Access_Specification = Variable_Name;<br>Data = paramètres d'appel de méthode ou données NULL | UnconfirmedWriteRequest::= (variable-name, data) |
| WITH-LIST | UnconfirmedWrite.request ({Variable_Access_Specification}, {Data})<br>Variable_Access_Specification = Variable_Name;<br>Data = voir ci-dessus | UnconfirmedWriteRequest::= ({variable-name}, {data}) |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

La Figure 61 représente le MSC d'un service Write utilisé pour écrire la valeur d'un seul attribut, en cas de réussite.

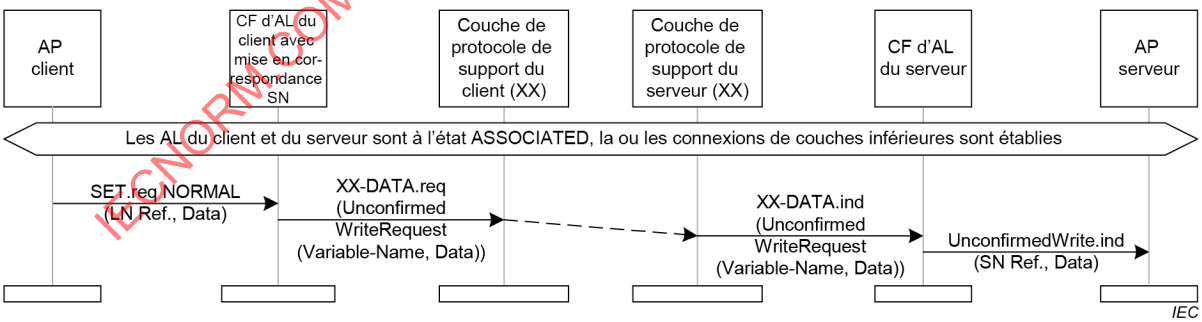



**Figure 61 – MSC du service UnconfirmedWrite utilisé pour écrire un attribut**

Le mécanisme de transfert de bloc général peut être utilisé lorsque les paramètres de service sont longs.

### 7.3.12 Protocole du service InformationReport

Le protocole du service InformationReport, spécifié en 6.17, est essentiellement identique à celui du service EventNotification (voir 7.3.8).

Contrairement au service EventNotification, le service InformationReport ne contient pas le paramètre facultatif Application_Addresses. Le rapport d'informations est donc toujours envoyé par le dispositif logique de gestion de serveur à l'AP de gestion du client.

À l'appel du service InformationReport.request, le AP serveur génère une APDU InformationReportRequest. Cette APDU est envoyée depuis le SAP du dispositif logique de gestion vers le SAP du dispositif de gestion du client, à l'aide de services de données des couches inférieures, de façon non sollicitée, à la première occasion.

Les possibilités d'envoi de cette APDU dépendent du profil de communication et du statut de connexion des couches inférieures. Le protocole du service InformationReport est décrit plus en détail à l'Annexe A.

Le service InformationReport peut contenir plusieurs noms d'attributs et leur contenu. En revanche, le service EventNotification spécifié en 6.11 ne contient qu'un seul attribut de référence. Par conséquent, lorsque l'APDU InformationReportRequest contient plusieurs attributs, elle doit être mise en correspondance avec plusieurs services EventNotification.ind, comme présenté dans le Tableau 77.

**Tableau 77 – Mise en correspondance des services EventNotification et InformationReport**

| EventNotification.ind (un ou plusieurs) | InformationReport.ind |
|---|---|
| Time (facultatif) | Current-time (facultatif) |
| COSEM_Class_Id,<br>COSEM_Object_Instance_Id,<br>COSEM_Object_Attribute_Id | Variable_Name {Variable_Name} |
| Attribute_Value | Data {Data} |

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

### 7.3.13 Protocole du mécanisme de transfert de bloc général

#### 7.3.13.1 Généralités

Le mécanisme de transfert de bloc général (GBT) peut être utilisé pour transporter toute APDU xDLMS lorsque les paramètres de service sont longs, c'est-à-dire que leur forme codée avec la charge utile de protection, le cas échéant, est plus longue que le paramètre Max Receive PDU Size de l'homologue négocié. Dans ce cas, l'AL utilise une ou plusieurs APDU xDLMS GBT pour transporter ces longues APDU.

Les appels de primitive de service peuvent être complets, incluant tous les paramètres de service, ou partiels, en incluant uniquement une partie des paramètres de service. L'utilisation d'appels de service complets ou partiels dépend de l'implémentation.

Après réception d'une primitive de service .request/.response du AP, l'AL:

- construit l'APDU qui transporte la primitive de service;
- lorsque le chiffrement est exigé, elle applique la protection telle qu'exigée par Security_Options et construit l'APDU chiffrée appropriée;
- lorsque l'APDU résultante est plus longue que la taille d'APDU maximale négociée, l'AL utilise alors le mécanisme GBT pour envoyer le message complet dans plusieurs APDU GBT.

Cependant, il n'y a aucune relation directe entre les appels partiels et les APDU GBT envoyées. L'AL peut appliquer la protection en utilisant des appels de service complets ou partiels.

Après réception des APDU GBT à partir d'une partie distante, l'AL:

- rassemble les champs de données de bloc des APDU GBT reçues;
- lorsque l'APDU complète résultante est chiffrée, elle vérifie et supprime la protection;
- appelle la primitive de service appropriée, en transmettant le paramètre supplémentaire Security_Status, et les paramètres General_Block_Transfer_Parameters et Protection_Element.

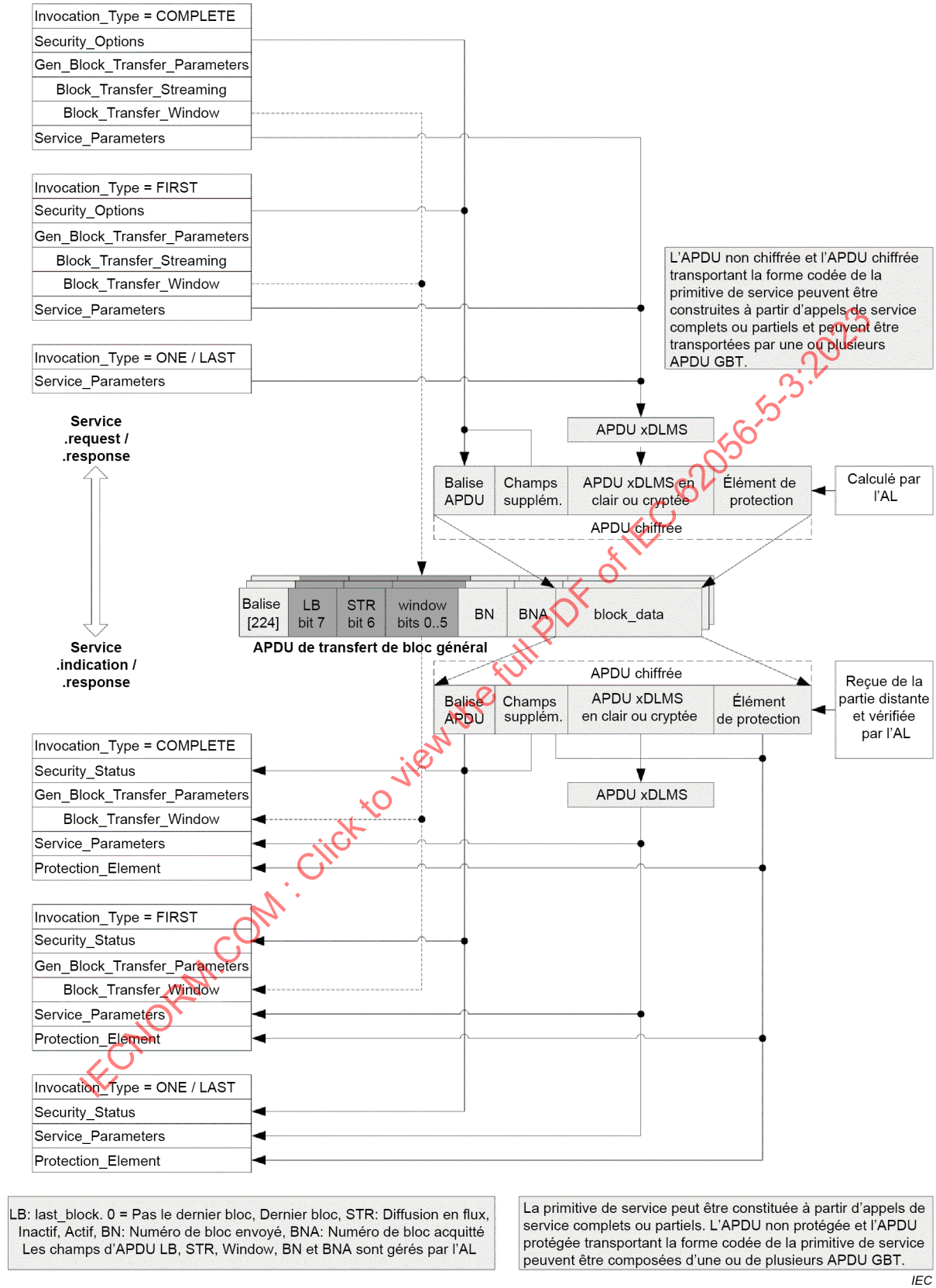
Cependant, il n'y a aucune relation directe entre les APDU GBT reçues et les appels de service partiels. L'AL peut vérifier et supprimer la protection en traitant les APDU GBT ou en traitant les APDU complètes assemblées.

Voir aussi Figure 36.

Un échange confirmé de messages peut démarrer avec ou sans l'utilisation de GBT. Cependant, si une partie envoie une demande ou une réponse à l'aide de GBT, l'autre partie doit faire de même. Les parties continuent alors à l'aide de GBT jusqu'à la fin, c'est-à-dire jusqu'à ce que la réponse complète soit reçue.

Le flux des blocs est géré par l'AL en tenant compte des paramètres GBT qui transitent du AP local à l'AL (voir 6.5) et des champs des APDU GBT (voir Figure 62) reçues de l'AL distante.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

NOTE L'application et la vérification/suppression de la protection cryptographique sur des APDU sont indépendantes du processus GBT. Elles sont incluses ici pour des raisons d'exhaustivité.

**Figure 62 – Appels de service partiels et APDU GBT**

Les différents types d'appels de service (COMPLETE, FIRST-PART, ONE-PART ou LAST-PART) et la relation entre ces appels, les paramètres de service et les champs des APDU chiffrées et des APDU GBT sont représentés à la Figure 62.

Le paramètre Block_Transfer_Streaming (BTS) est transmis par l'AP à l'AL, pour indiquer que l'AL peut envoyer des blocs par flux, c'est-à-dire sans attendre une confirmation de chaque bloc reçu par la partie distante. Ce paramètre n'est pas inclus dans l'APDU.

Le paramètre Block_Transfer_Window (BTW) indique la taille de la fenêtre de flux prise en charge, c'est-à-dire le nombre maximal de blocs qui peuvent être reçus. Le paramètre Block_Transfer_Window de l'autre partie peut être connu *a priori* par les parties. Cependant, la taille de la fenêtre est gérée par l'AL: elle peut utiliser une valeur inférieure, par exemple, pendant la récupération des blocs perdus.

NOTE 1 Cette relation est indiquée par une ligne en pointillés à la Figure 62 entre le paramètre Block_Transfer_Window et le champ de fenêtre de l'APDU.

Dans le cas de services non confirmés, le paramètre Block_Transfer_Streaming doit être défini sur FALSE et Block_Transfer_Window doit être défini sur 0. Ceci indique à l'AL qu'elle doit envoyer la forme codée de la primitive de service complète dans autant d'APDU GBT que nécessaire, sans attendre la confirmation des blocs envoyés.

L'utilisation des champs de l'APDU GBT est spécifiée ci-dessous:

- le bit last-block (LB) (dernier bloc) indique si le bloc est le dernier (LB = 1) ou non (LB = 0). Pour la gestion du champ du dernier bloc, voir 7.3.13.4.3.2;
- le bit streaming (flux) indique si le flux est en cours (STR = 1) ou terminé (STR = 0). Lorsque le flux est terminé, la partie distante doit confirmer les blocs reçus. Cependant, le client ne confirme pas la réception des blocs reçus dans le dernier flux. Les blocs reçus dans un GBT non confirmé ne sont pas non plus confirmés. Lorsque le paramètre Block_Transfer_Streaming a été défini sur FALSE, le bit streaming doit aussi être défini sur 0;
- le champ window indique le nombre de blocs qui peuvent être reçus par la partie émettrice de l'APDU. Sa valeur maximale est égale au paramètre Block_Transfer_Window transmis par l'AP à l'AL. À noter que l'AL peut utiliser une valeur inférieure pendant la récupération des blocs perdus. Dans le cas où les APDU GBT contiennent un service non confirmé (BTS = FALSE, BTW = 0; voir ci-dessus), la valeur de la fenêtre doit être 0, ce qui indique qu'aucune APDU GBT reçue ne doit être confirmée (et de ce fait aucun bloc perdu ne peut être récupéré);
- le champ block-number (BN) (numéro de bloc) indique le numéro du bloc envoyé. Le premier bloc envoyé doit avoir block-number = 1. Le numéro de bloc doit être augmenté à chaque envoi d'une APDU GBT, même si le champ des données de bloc (BD) est vide. Cependant, lors de la récupération des blocs perdus, un numéro de bloc peut être répété;
- le champ block-number-acknowledged (BNA) indique le numéro du bloc acquitté. Si aucun bloc n'est perdu, il doit être égal au numéro du dernier bloc reçu. Cependant, si un ou plusieurs blocs sont perdus, il doit être égal au numéro du bloc à hauteur duquel aucun bloc n'est manquant;
- le champ block-data (BD) contient une partie de l'APDU xDLMS qui est envoyée à l'aide du mécanisme GBT.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

#### 7.3.13.2 Procédure de GBT

##### 7.3.13.2.1 Vue d'ensemble

La procédure de GBT est représentée à la Figure 63. Elle peut être:

- confirmée, pour transporter des APDU correspondant aux primitives de service de tout service xDLMS confirmé;
- non confirmée, pour transporter une APDU correspondant à une primitive de service non confirmée du client vers le serveur ou pour transporter une APDU correspondant à une demande de service non sollicitée du serveur vers le client.

NOTE Le GBT est souvent utilisé avec les services DataNotification et Access.

La procédure de GBT est essentiellement la même côté client et côté serveur. Elle est appelée par l'AL lorsque:

a) une primitive de service (.request ou .response) est invoquée par l'AP et:
   - soit Invocation_Type = COMPLETE ou FIRST-PART avec les paramètres GBT BTS et BTW présents;
   - soit l'APDU codée et protégée par cryptographie à envoyer est trop longue pour s'adapter à la taille maximale d'APDU négociée;
b) une APDU GBT est reçue de la part de son homologue.

La procédure de GBT comprend trois sous-procédures:

1) *envoyer le flux d'APDU GBT* (voir 7.3.13.4);
2) *traiter l'APDU GBT* (voir 7.3.13.5);
3) *vérifier la RQ et combler les trous* (voir 7.3.13.6).

Ces sous-procédures sont appelées par l'AL comme décrit en 7.3.13.2.2 et en 7.3.13.2.3.

Le modèle de la procédure de GBT utilise une SQ et une RQ. Chaque bloc B dans les files d'attente a un numéro de bloc BN, un indicateur LastBlock LB et une charge utile BlockData BD (qui peut être vide). Les files d'attente sont ordonnées par BN.

La SQ est remplie par l'AL après traitement des primitives de service invoquées par l'AP, c'est-à-dire:

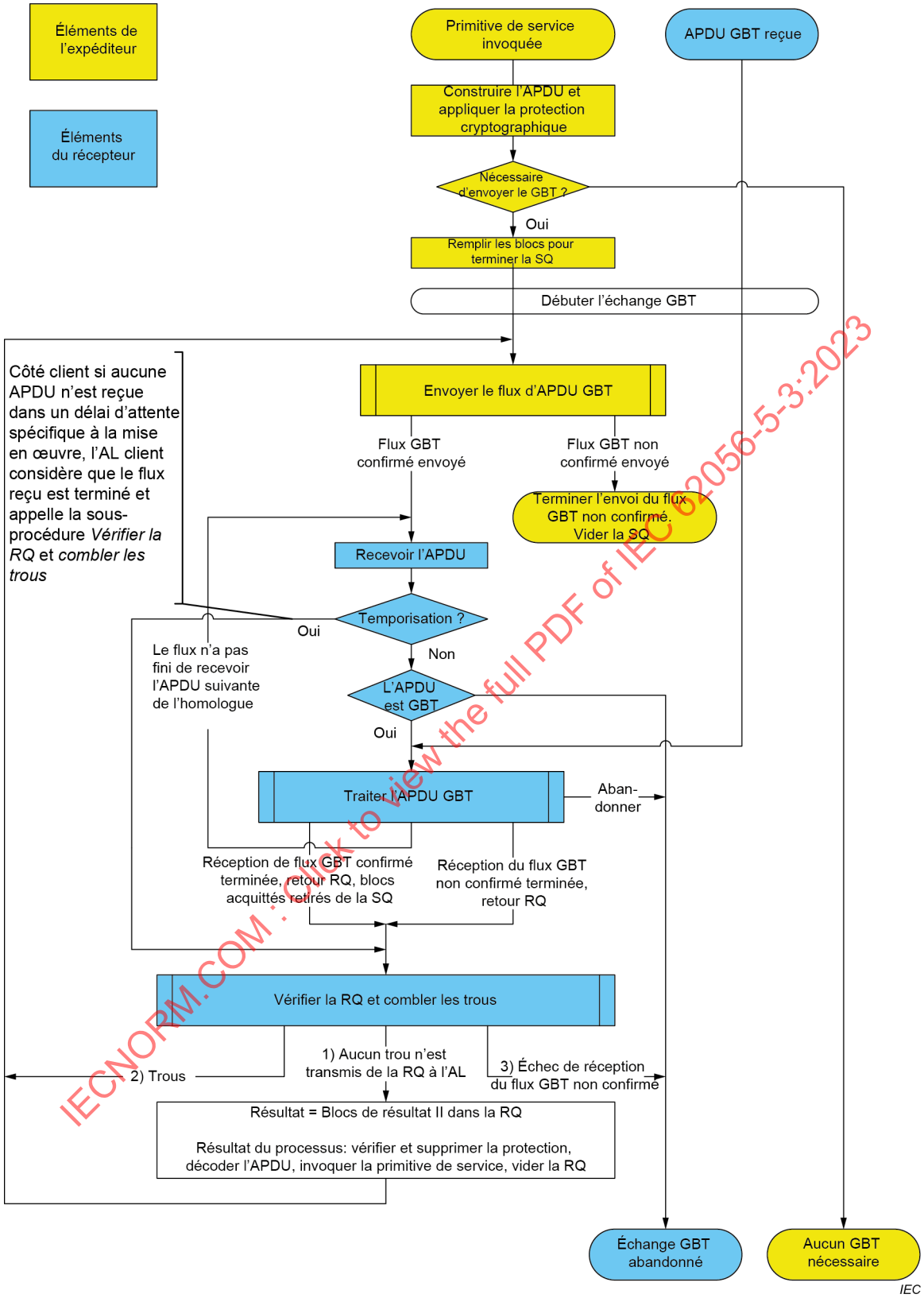
- construction de l'APDU;
- application d'une protection cryptographique telle que stipulée par les paramètres de protection;
- division de l'APDU protégée en blocs.

Pour toutes les primitives de service, des appels de service partiels peuvent être utilisés comme spécifié en 6.5 et en 0. Cependant, cela est sans conséquence pour le protocole GBT. Le service Invocation_Type peut être COMPLETE, FIRST-PART, ONE-PART ou LAST-PART. Chaque appel de service ajoute un ou plusieurs blocs à la SQ. Le numéro de bloc BN est incrémenté chaque fois qu'un nouveau bloc est ajouté.

Les blocs de la SQ sont envoyés par la sous-procédure *Envoyer le flux d'APDU GBT*.

La RQ est remplie par la sous-procédure *Traiter l'APDU GBT*. Les trous dans la RQ (en cas de procédure de GBT confirmé) sont comblés par la sous-procédure *Vérifier la RQ et combler les trous*.

IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023




IECNORM.COM : Click to view the full PDF of IEC 62056-5-3:2023

**Figure 63 – Procédure de GBT**