

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Electricity metering data exchange – The DLMS/COSEM suite –
Part 6-2: COSEM interface classes**

**Échange des données de comptage de l'électricité – La suite DLMS/COSEM –
Partie 6-2: Classes d'interfaces COSEM**



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2015 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 Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
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 corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue

The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub

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 also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 15 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

More than 60 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

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: csc@iec.ch.

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.

Catalogue IEC - webstore.iec.ch/catalogue

Application autonome pour consulter tous les renseignements bibliographiques sur les Normes internationales, Spécifications techniques, Rapports techniques et autres documents de l'IEC. Disponible pour PC, Mac OS, tablettes Android et iPad.

Recherche de publications IEC - www.iec.ch/searchpub

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 aussi une fois par mois par email.

Electropedia - www.electropedia.org

Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 15 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary

Plus de 60 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

Service Clients - webstore.iec.ch/csc

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Electricity metering data exchange – The DLMS/COSEM suite –
Part 6-2: COSEM interface classes**

**Échange des données de comptage de l'électricité – La suite DLMS/COSEM –
Partie 6-2: Classes d'interfaces COSEM**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

ICS 17.220; 35.110; 91.140.50

ISBN 978-2-8322-3041-1

**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.**

CONTENTS

FOREWORD.....	9
INTRODUCTION.....	12
1 Scope.....	14
2 Normative references.....	14
3 Terms, definitions and abbreviations	16
3.1 Terms and definitions related to the Image transfer process (see 5.3.6)	16
3.2 Terms and definitions related to the S-FSK PLC setup classes (see 5.8)	17
3.3 Terms and definitions related to the PRIME NB OFDM PLC setup ICs (see 5.10)	18
3.4 Terms and definitions related to ZigBee® (see 5.12)	20
3.5 Abbreviations	22
4 Basic principles	26
4.1 General.....	26
4.2 Referencing methods	27
4.3 Reserved base_names for special COSEM objects	27
4.4 Class description notation	27
4.5 Common data types	30
4.6 Data formats	31
4.6.1 Date and time formats	31
4.6.2 Floating point number formats	33
4.7 The COSEM server model.....	35
4.8 The COSEM logical device	36
4.8.1 General	36
4.8.2 COSEM logical device name (LDN)	36
4.8.3 The “association view” of the logical device	36
4.8.4 Mandatory contents of a COSEM logical device	37
4.8.5 Management logical device	37
4.9 Information security.....	37
5 The COSEM interface classes.....	38
5.1 Overview.....	38
5.2 Interface classes for parameters and measurement data.....	42
5.2.1 Data (class_id = 1, version = 0).....	42
5.2.2 Register (class_id = 3, version = 0)	42
5.2.3 Extended register (class_id = 4, version = 0)	46
5.2.4 Demand register (class_id = 5, version = 0)	47
5.2.5 Register activation (class_id = 6, version = 0).....	50
5.2.6 Profile generic (class_id = 7, version = 1)	52
5.2.7 Utility tables (class_id = 26, version = 0).....	57
5.2.8 Register table (class_id = 61, version = 0)	58
5.2.9 Status mapping (class_id = 63, version = 0).....	60
5.3 Interface classes for access control and management.....	61
5.3.1 Overview	61
5.3.2 Client user identification	61
5.3.3 Association SN (class_id = 12, version = 3)	62
5.3.4 Association LN (class_id = 15, version = 2)	66
5.3.5 SAP assignment (class_id = 17, version = 0)	72

5.3.6	Image transfer	72
5.3.7	Security setup (class_id = 64, version = 0).....	79
5.3.8	Push interface classes and objects	81
5.4	Interface classes for time- and event bound control.....	87
5.4.1	Clock (class_id = 8, version = 0).....	87
5.4.2	Script table (class_id = 9, version = 0)	90
5.4.3	Schedule (class_id = 10, version = 0)	91
5.4.4	Special days table (class_id = 11, version = 0)	94
5.4.5	Activity calendar (class_id = 20, version = 0)	95
5.4.6	Register monitor (class_id = 21, version = 0)	98
5.4.7	Single action schedule (class_id = 22, version = 0)	100
5.4.8	Disconnect control (class_id = 70, version = 0)	101
5.4.9	Limiter (class_id = 71, version = 0)	103
5.4.10	Parameter monitor (class_id = 65, version = 0)	106
5.4.11	Sensor manager interface class.....	107
5.5	Interface classes for setting up data exchange via local ports and modems	111
5.5.1	IEC local port setup (class_id = 19, version = 1)	111
5.5.2	IEC HDLC setup (class_id = 23, version = 1)	112
5.5.3	IEC twisted pair (1) setup (class_id = 24, version = 1)	114
5.5.4	Modem configuration (class_id = 27, version = 1)	117
5.5.5	Auto answer (class_id = 28, version = 2)	118
5.5.6	Auto connect (class_id = 29, version = 2)	121
5.5.7	GPRS modem setup (class_id = 45, version = 0)	123
5.5.8	GSM diagnostic (class_id = 47, version = 0)	124
5.6	Interface classes for setting up data exchange via M-Bus.....	126
5.6.1	M-Bus slave port setup (class_id = 25, version = 0)	126
5.6.2	M-Bus client (class_id = 72, version = 1)	127
5.6.3	Wireless Mode Q channel (class_id = 73, version = 1)	132
5.6.4	M-Bus master port setup (class_id = 74, version = 0)	133
5.7	Interface classes for setting up data exchange over the Internet.....	133
5.7.1	TCP-UDP setup (class_id = 41, version = 0)	133
5.7.2	IPv4 setup (class_id = 42, version = 0)	134
5.7.3	IPv6 setup (class_id = 48, version = 0)	137
5.7.4	MAC address setup (class_id = 43, version = 0)	140
5.7.5	PPP setup (class_id = 44, version = 0)	141
5.7.6	SMTP setup (class_id = 46, version = 0).....	145
5.8	Interface classes for setting up data exchange using S-FSK PLC	146
5.8.1	General	146
5.8.2	Overview	146
5.8.3	S-FSK Phy&MAC set-up (class_id = 50, version = 1)	149
5.8.4	S-FSK Active initiator (class_id = 51, version = 0).....	153
5.8.5	S-FSK MAC synchronization timeouts (class_id = 52, version = 0)	155
5.8.6	S-FSK MAC counters (class_id = 53, version = 0).....	157
5.8.7	IEC 61334-4-32 LLC setup (class_id = 55, version = 1).....	160
5.8.8	-FSK Reporting system list (class_id = 56, version = 0).....	161
5.9	Interface classes for setting up the LLC layer for ISO/IEC 8802-2.....	162
5.9.1	General	162
5.9.2	ISO/IEC 8802-2 LLC Type 1 setup (class_id = 57, version = 0)	162
5.9.3	ISO/IEC 8802-2 LLC Type 2 setup (class_id = 58, version = 0)	163

5.9.4	ISO/IEC 8802-2 LLC Type 3 setup (class_id = 59, version = 0)	164
5.10	Interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for PRIME networks	166
5.10.1	Overview	166
5.10.2	Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes	167
5.10.3	61334-4-32 LLC SSCS setup (class_id = 80, version = 0)	169
5.10.4	PRIME NB OFDM PLC Physical layer parameters	170
5.10.5	PRIME NB OFDM PLC Physical layer counters (class_id = 81, version = 0)	170
5.10.6	PRIME NB OFDM PLC MAC setup (class_id = 82, version = 0)	171
5.10.7	PRIME NB OFDM PLC MAC functional parameters (class_id = 83, version = 0)	172
5.10.8	PRIME NB OFDM PLC MAC counters (class_id = 84, version = 0)	174
5.10.9	PRIME NB OFDM PLC MAC network administration data (class_id = 85, version = 0)	175
5.10.10	PRIME NB OFDM PLC MAC address setup (class_id = 86, version = 0)	177
5.10.11	PRIME NB OFDM PLC Application identification (class_id = 86, version = 0)	177
5.11	Interface classes for setting up and managing the DLMS/COSEM narrowband OFDM PLC profile for G3-PLC networks	178
5.11.1	Overview	178
5.11.2	Mapping of G3-PLC PIB attributes to COSEM IC attributes	179
5.11.3	G3-PLC MAC layer counters (class_id = 90, version = 1)	180
5.11.4	G3-PLC MAC setup (class_id = 91, version = 1)	181
5.11.5	G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 1)	187
5.12	ZigBee® setup classes	192
5.12.1	Overview	192
5.12.2	ZigBee® SAS startup (class_id = 101, version = 0)	194
5.12.3	ZigBee® SAS join (class_id = 102, version = 0)	196
5.12.4	ZigBee® SAS APS fragmentation (class_id = 103, version = 0)	197
5.12.5	ZigBee® network control (class_id = 104, version = 0)	198
5.12.6	ZigBee® tunnel setup (class_id = 105, version = 0)	204
5.13	Maintenance of the interface classes	205
5.13.1	New versions of interface classes	205
5.13.2	New interface classes	205
5.13.3	Removal of interface classes	205
6	Relation to OBIS	206
6.1	General	206
6.2	Abstract COSEM objects	206
6.2.1	Use of value group C	206
6.2.2	Data of historical billing periods	207
6.2.3	Billing period values / reset counter entries	209
6.2.4	Other abstract general purpose OBIS codes	209
6.2.5	Clock objects (class_id = 8)	210
6.2.6	Modem configuration and related objects	210
6.2.7	Script table objects (class_id = 9)	210
6.2.8	Special days table objects (class_id = 11)	211
6.2.9	Schedule objects (class_id = 10)	211
6.2.10	Activity calendar objects (class_id = 20)	211

6.2.11	Register activation objects (class_id = 6)	212
6.2.12	Single action schedule objects (class_id = 22)	212
6.2.13	Register monitor objects (class_id = 21)	212
6.2.14	Parameter monitor objects (class_id = 65)	212
6.2.15	Limiter objects (class_id = 71)	212
6.2.16	IEC local port setup objects (class_id = 19)	213
6.2.17	Standard readout profile objects (class_id = 7)	213
6.2.18	IEC HDLC setup objects (class_id = 23)	213
6.2.19	IEC twisted pair (1) setup objects (class_id = 24)	213
6.2.20	Objects related to data exchange over M-Bus	214
6.2.21	Objects to set up data exchange over the Internet	215
6.2.22	Objects for setting up data exchange using S-FSK PLC	216
6.2.23	Objects for setting up the ISO/IEC 8802-2 LLC layer	216
6.2.24	Objects for data exchange using narrowband OFDM PLC for PRIME networks	217
6.2.25	Objects for data exchange using narrow-band OFDM PLC for G3-PLC networks	218
6.2.26	ZigBee® setup objects	218
6.2.27	Association objects (class_id = 12, 15)	218
6.2.28	SAP assignment object (class_id = 17)	218
6.2.29	COSEM logical device name object	219
6.2.30	Information security related objects	219
6.2.31	Image transfer objects (class_id = 18)	219
6.2.32	Utility table objects (class_id = 26)	219
6.2.33	Device ID objects	220
6.2.34	Metering point ID objects	220
6.2.35	Parameter changes and calibration objects	221
6.2.36	I/O control signal objects	221
6.2.37	Disconnect control objects (class_id = 70)	221
6.2.38	Status of internal control signals objects	221
6.2.39	Internal operating status objects	222
6.2.40	Battery entries objects	222
6.2.41	Power failure monitoring objects	222
6.2.42	Operating time objects	223
6.2.43	Environment related parameters objects	223
6.2.44	Status register objects	223
6.2.45	Event code objects	224
6.2.46	Communication port log parameter objects	224
6.2.47	Consumer message objects	224
6.2.48	Currently active tariff objects	224
6.2.49	Event counter objects	224
6.2.50	Meter tamper event related objects	225
6.2.51	Error register objects	225
6.2.52	Alarm register, Alarm filter and Alarm descriptor objects	226
6.2.53	General list objects	226
6.2.54	Event log objects	227
6.2.55	Inactive objects	227
6.3	Electricity related COSEM objects	227
6.3.1	Value group D definitions	227

6.3.2	Electricity ID numbers	227
6.3.3	Billing period values / reset counter entries.....	228
6.3.4	Other electricity related general purpose objects.....	228
6.3.5	Measurement algorithm	229
6.3.6	Metering point ID (electricity related)	231
6.3.7	Electricity related status objects	231
6.3.8	List objects – Electricity (class_id = 7)	231
6.3.9	Threshold values.....	232
6.3.10	Register monitor objects (class_id = 21)	232
6.4	Coding of OBIS identifications	233
7	Previous versions of interface classes.....	234
7.1	General.....	234
7.2	Profile generic (class_id = 7, version = 0)	234
7.3	Association SN (class_id = 12, version = 0)	237
7.4	Association SN (class_id = 12, version = 1)	239
7.5	Association SN (class_id = 12, version = 2)	242
7.6	Association LN (class_id = 15, version = 0).....	245
7.7	Association LN (class_id = 15, version = 1).....	250
7.8	IEC local port setup (class_id = 19, version = 0)	255
7.9	IEC HDLC setup, (class_id = 23, version = 0)	256
7.10	IEC twisted pair (1) setup (class_id = 24, version = 0).....	258
7.11	PSTN modem configuration (class_id = 27, version = 0).....	259
7.12	Auto answer (class_id = 28, version = 0)	261
7.13	PSTN auto dial (class_id = 29, version = 0)	262
7.14	Auto connect (class_id = 29, version = 1).....	263
7.15	S-FSK Phy&MAC setup (class_id = 50, version = 0).....	265
7.16	S-FSK IEC 61334-4-32 LLC setup (class_id = 55, version = 0)	268
7.17	M-Bus client (class_id = 72, version = 0).....	269
7.18	G3 NB OFDM PLC MAC layer counters (class_id = 90, version = 0)	274
7.19	G3 NB OFDM PLC MAC setup (class_id = 91, version = 0)	275
7.20	G3 NB OFDM PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 0).....	279
Annex A (informative)	Additional information on Auto answer and Auto connect ICs	285
Annex B (informative)	Additional information to M-Bus client (class_id = 72, version 1)	287
Annex C (informative)	Additional information on IPv6 setup class (class_id = 48, version = 0)	289
C.1	General.....	289
C.2	IPv6 addressing	289
C.3	IPv6 header format.....	291
C.4	IPv6 header extensions	292
C.4.1	Overview	292
C.4.2	Hop-by-Hop options	293
C.4.3	Destination options	293
C.4.4	Routing options.....	293
C.4.5	Fragment options	293
C.4.6	Security options	294
Annex D (informative)	Overview of the narrow-band OFDM PLC technology for PRIME networks	295

Annex E (informative) Overview of the narrow-band OFDM PLC technology for G3-PLC networks	296
Annex F (informative) Significant technical changes with respect to IEC 62056-6-2:2013	297
Bibliography	299
Index	301
Figure 1 – Meaning of the definitions concerning the Image	17
Figure 2 – An interface class and its instances	26
Figure 3 – The COSEM server model	35
Figure 4 – Combined metering device	36
Figure 5 – Overview of the interface classes – Part 1	38
Figure 6 – Overview of the interface classes – Part 2	39
Figure 7 – The time attributes when measuring sliding demand	47
Figure 8 – The attributes in the case of block demand	47
Figure 9 – The attributes in the case of sliding demand (number of periods = 3)	48
Figure 10 – Image transfer process flow chart	78
Figure 11 – COSEM model of push operation	82
Figure 12 – Push windows and delays	83
Figure 13 – The generalized time concept	88
Figure 14 – State diagram of the Disconnect control IC	101
Figure 15 – Definition of upper and lower thresholds	110
Figure 16 – Object model of DLMS/COSEM servers	147
Figure 17 – Object model of DLMS/COSEM servers	166
Figure 18 – Example of a ZigBee® network	193
Figure 19 – Data of historical billing periods – example with module 12, VZ = 5	208
Figure A.1 – Network connectivity example for a GSM/GPRS network	285
Figure B.1 – Encryption key status diagram	287
Figure C.1 – IPv6 address formats	290
Figure C.2 – IPv6 header format	291
Figure C.3 – Traffic class parameter format	291
Table 1 – Reserved base_names for SN referencing	27
Table 2 – Common data types	30
Table 3 – List of interface classes by class_id	40
Table 4 – Enumerated values for physical units	44
Table 5 – Examples for scaler_unit	46
Table 6 – Encoding of selective access parameters with data_index	87
Table 7 – Schedule	91
Table 8 – Special days table	92
Table 9 – Disconnect control IC – states and state transitions	102
Table 10 – Explicit presentation of threshold value arrays	111
Table 11 – Explicit presentation of action_sets	111
Table 12 – ADS address elements	116

Table 13 – Fatal error register	116
Table 14 – Mapping IEC 61334-4-512:2001 MIB variables to COSEM IC attributes / methods	148
Table 15 – MAC addresses in the S-FSK profile	153
Table 16 – Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes.....	167
Table 17 – Mapping of G3-PLC IB attributes to COSEM IC attributes	179
Table 18 – Use of ZigBee® setup COSEM interface classes	194
Table 19 – Use of value group C for abstract objects in the COSEM context	207
Table 20 – Representation of various values by appropriate ICs	227
Table 21 – Measuring algorithms – enumerated values.....	230
Table 22 – Threshold objects, electricity.....	232
Table 23 – Register monitor objects, electricity.....	233
Table B.1 – Encryption key is preset in the slave and cannot be changed	288
Table B.2 – Encryption key is preset in the slave and new key is set after installation.....	288
Table B.3 – Encryption key is not preset in the slave, but can be set, case a)	288
Table B.4 – Encryption key is not preset in the slave, but can be set, case b)	288
Table C.1 – IPv6 header vs. IPv6 IC.....	292
Table C.2 – Optional IPv6 header extensions vs. IPv6 IC.....	292

IECNORM.COM: Click to view the full PDF of IEC 62056-6-2:2016

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**ELECTRICITY METERING DATA EXCHANGE –
THE DLMS/COSEM SUITE –****Part 6-2: COSEM interface classes****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.

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this document may involve the use of a patent concerning the Image transfer procedure.

The IEC takes no position concerning the evidence, validity and scope of this patent right.

The holder of this patent right has assured the IEC that he/she is willing to negotiate licenses either free of charge or under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with the IEC. Information may be obtained from Itron, Inc., Liberty Lake, Washington, USA.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights other than those identified above. The IEC shall not be held responsible for identifying any or all such patent rights.

IEC (<http://patents.iec.ch>) maintains on-line databases of patents relevant to their standards. Users are encouraged to consult the databases for the most up to date information concerning patents.

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 62056-6-2 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:

DLMS¹ User Association
Zug/Switzerland
www.dlms.com

International Standard IEC 62056-6-2 has been prepared by IEC technical committee 13: Electrical energy measurement and control.

This second edition cancels and replaces the first edition of IEC 62056-6-2, published in 2013. It constitutes a technical revision.

The significant technical changes with respect to IEC 62056-6-2:2013 are listed in Annex F (informative).

The text of this standard is based on the following documents:

FDIS	Report on voting
13/1651A/FDIS	13/1659/RVD

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

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

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.

¹ Device Language Message Specification.

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "<http://webstore.iec.ch>" in the data related to the specific publication. At this date, the publication will be

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

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

INTRODUCTION

This second edition of IEC 62056-6-2 has been prepared by IEC TC13 WG14 with a significant contribution of the DLMS User Association, its D-type liaison partner.

This edition is in line with the DLMS UA Blue Book Edition 11.0. The main new features are the client user identification mechanism, the “Push setup”, the “Parameter monitor”, the “GSM diagnostic”, the “IPv6 setup”, the “Prime NB OFDM PLC setup”, the “G3-PLC setup” and the “ZigBee® setup”² interface classes.

In 2014, the DLMS UA has published Blue Book Edition 12.0 adding several new features regarding functionality, efficiency and security while keeping full backwards compatibility.

The intention of the DLMS UA is to bring also these latest developments to international standardization. Therefore, IEC TC13 WG14 launched a project to bring these new elements also to the IEC 62056 suite that will lead to Edition 3.0 of the standard.

Object modelling and data identification

Driven by the business needs of the energy market participants – generally in a liberalized, competitive environment – and by the desire to manage natural resources efficiently and to involve the consumers, the utility meter became part of an integrated metering, control and billing system. The meter is not any more a simple data recording device but it relies critically on communication capabilities. Ease of system integration, interoperability and data security are important requirements.

COSEM, the Companion Specification for Energy Metering, addresses these challenges by looking at the utility meter as part of a complex measurement and control system. The meter has to be able to convey measurement results from the metering points to the business processes which use them. It also has to be able to provide information to the consumer and manage consumption and eventually local generation.

COSEM achieves this by using object modelling techniques to model all functions of the meter, without making any assumptions about which functions need to be supported, how those functions are implemented and how the data are transported. The formal specification of COSEM interface classes forms a major part of COSEM.

To process and manage the information it is necessary to uniquely identify all data items in a manufacturer-independent way. The definition of OBIS, the Object Identification System is another essential part of COSEM. It is based on DIN 43863-3:1997, *Electricity meters – Part 3: Tariff metering device as additional equipment for electricity meters – EDIS – Energy Data Identification System*. The set of OBIS codes has been considerably extended over the years to meet new needs.

COSEM models the utility meter as a server application – see 4.7 – used by client applications that retrieve data from, provide control information to, and instigate known actions within the meter via controlled access to the COSEM objects. The clients act as agents for third parties i.e. the business processes of energy market participants.

The standardized COSEM interface classes form an extensible library. Manufacturers use elements of this library to design their products that meet a wide variety of requirements.

² ZigBee® is a trademark owned by ZigBee corporation. This information is given for the convenience of users of this document and does not constitute an endorsement by the IEC of the product named.

The server offers means to retrieve the functions supported, i.e. the COSEM objects instantiated. The objects can be organized to logical devices and application associations and to provide specific access rights to various clients.

The concept of the standardized interface class library provides different users and manufacturers with a maximum of diversity while ensuring interoperability.

IECNORM.COM: Click to view the full PDF of IEC 62056-6-2:2016

ELECTRICITY METERING DATA EXCHANGE – THE DLMS/COSEM SUITE –

Part 6-2: COSEM interface classes

1 Scope

This part of IEC 62056 specifies a model of a meter as it is seen through its communication interface(s). Generic building blocks are defined using object-oriented methods, in the form of interface classes to model meters from simple up to very complex functionality.

Annexes A to F (informative) provide additional information related to some interface classes.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61334-4-32:1996, *Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 32: Data link layer – Logical link control (LLC)*

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 61334-4-511:2000, *Distribution automation using distribution line carrier systems – Part 4-511: Data communication protocols – Systems management – CIASE protocol*

IEC 61334-4-512:2001, *Distribution automation using distribution line carrier systems – Part 4-512: Data communication protocols – System management using profile 61334-5-1 – Management Information Base (MIB)*

IEC 61334-5-1:2001, *Distribution automation using distribution line carrier systems – Part 5-1: Lower layer profiles – The spread frequency shift keying (S-FSK) profile*

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

IEC 62056-21:2002, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 21: Direct local data exchange*

IEC 62056-31:1999, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 31: Using local area networks on twisted pair with carrier signalling*

NOTE This Edition is referenced in the interface class “IEC twisted pair (1) setup” (class_id: 24, version: 0).

IEC 62056-3-1:2013, *Electricity metering data exchange – The DLMS/COSEM suite – Part 3-1: Use of local area networks on twisted pair with carrier signalling*

NOTE This Edition is referenced in the interface class “IEC twisted pair (1) setup” (class_id: 24, version: 1).

IEC 62056-46:2007, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 46: Data link layer using HDLC protocol*

IEC 62056-5-3:2016, *Electricity metering data exchange – The DLMS/COSEM suite – Part 5-3: DLMS/COSEM application layer*

IEC 62056-6-1:2015, *Electricity metering data exchange – The DLMS/COSEM suite – Part 6-1: Object identification system (OBIS)*

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/IEEE 60559:2011, *Information technology – Microprocessor Systems – Floating-Point arithmetic*

EN 13757-2:2004, *Communication system for and remote reading of meters – Part 2: Physical and link layer*

EN 13757-3:2004, *Communication systems for and remote reading of meters – Part 3: Dedicated application layer*

NOTE This standard is referenced in the “M-Bus client setup” interface class version 0.

EN 13757-3:2013, *Communication systems for and remote reading of meters – Part 3: Dedicated application layer*

NOTE This standard is referenced in the M-Bus client setup interface class version 1.

EN 13757-5:2015, *Communication systems for meters – Part 5: Wireless relaying*

IEEE 802.15.4:2006, *Standard for 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)*

NOTE This standard is also available as ISO/IEC/IEEE 8802-15-4:2010.

ITU-T G.9901:2014, *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS – Access Networks – In premises networks – Narrow-band orthogonal frequency division multiplexing power line communication transceivers – Power spectral density specification*

ITU-T G.9903 Amd. 1:2013, *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS – Access networks – In premises networks – Narrow-band orthogonal frequency division multiplexing power line communication transceivers for G3-PLC networks*

ITU-T G.9903:2014, *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS – Access networks – In premises networks – Narrow-band orthogonal frequency division multiplexing power line communication transceivers for G3-PLC networks*

ITU-T G.9904:2012, *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS – Access networks – In premises networks – Narrow-band orthogonal frequency division multiplexing power line communication transceivers for PRIME networks*

ETSI GSM 05.08, *Digital cellular telecommunications system (Phase 2+); Radio subsystem link control*

ANSI C12.19:1997, *IEEE 1377:1997, Utility industry end device data tables*

The following RFCs are available online from the Internet Engineering Task Force (IETF): <http://www.ietf.org/rfc/std-index.txt>, <http://www.ietf.org/rfc/>

IETF STD 51, *The Point-to-Point Protocol (PPP)*, 1994. (Also RFC 1661, RFC 1662)

RFC 791, *Internet Protocol* (Also: IETF STD 0005), 1981. RFC 1332, *The PPP Internet Protocol Control Protocol (IPCP)*, 1992, Updated by: RFC 3241. Obsoletes: RFC 1172

RFC 1570, *PPP LCP Extensions*, 1994

IETF STD 51 / RFC 1661, *The Point-to-Point Protocol (PPP)* (Also: IETF STD 0051), 1994, Updated by: RFC 2153, Obsoletes: RFC 1548

IETF STD 51 / RFC 1662, *PPP in HDLC-like Framing*, (Also: IETF STD 0051), 1994, Obsoletes: RFC 1549

RFC 1994, *PPP Challenge Handshake Authentication Protocol (CHAP)*, 1996. Obsoletes: RFC 1334

RFC 2433, *PPP CHAP Extension*, 1998

RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, 1998

RFC 2507, *IP Header Compression*, 1999

RFC 2759, *Microsoft PPP CHAP Extensions*, Version 2, 2000

RFC 3241 *Robust Header Compression (ROHC) over PPP*, 2002. Updates: RFC1332

RFC 3513, *Internet Protocol Version 6 (IPv6) Addressing Architecture*, 2003

RFC 3544, *IP Header Compression over PPP*, 2003

RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*, 2007

Point-to-Point (PPP) Protocol Field Assignments. Online database. Available from: <http://www.iana.org/assignments/ppp-numbers/ppp-numbers.xhtml>

3 Terms, definitions and abbreviations

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

3.1 Terms and definitions related to the Image transfer process (see 5.3.6)

3.1.1

Image

binary data of a specified size

Note 1 to entry: An Image can be seen as a container. It may consist of one or multiple elements (*image_to_activate*) which are transferred, verified and activated together.

3.1.2

ImageSize

size of the whole Image to be transferred

Note 1 to entry: ImageSize is expressed in octets.

3.1.3

ImageBlock

part of the Image of size ImageBlockSize

Note 1 to entry: The Image is transferred in ImageBlocks. Each block is identified by its ImageBlockNumber.

3.1.4

ImageBlockSize

size of ImageBlock expressed in octets

3.1.5

ImageBlockNumber

identifier of an ImageBlock. ImageBlocks are numbered sequentially, starting from 0.

The meaning of the definitions above is illustrated in Figure 1:

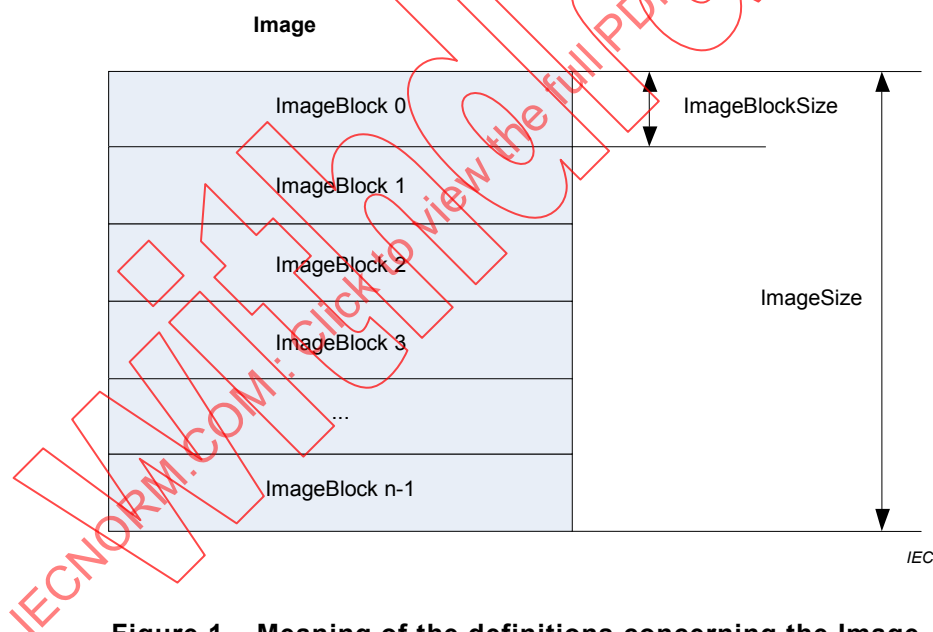


Figure 1 – Meaning of the definitions concerning the Image

3.2 Terms and definitions related to the S-FSK PLC setup classes (see 5.8)

3.2.1

initiator

user-element of a client System Management Application Entity (SMAE)

Note 1 to entry: The initiator uses the CIASE and xDLMS ASE and is identified by its system title.

[SOURCE: IEC 61334-4-511:2000, 3.8.1]

3.2.2

active initiator

initiator which issues or has last issued a CIASE Register request when the server is in the unconfigured state

[SOURCE: IEC 61334-4-511:2000, 3.9.1]

3.2.3

new system

server system which is in the unconfigured state: its MAC address equals "NEW-address"

[SOURCE: IEC 61334-4-511:2000, 3.9.3]

3.2.4

new system title

system-title of a new system

Note 1 to entry: This is the system title of a system, which is in the new state.

[SOURCE: IEC 61334-4-511:2000, 3.9.4]

3.2.5

registered system

server system which has an individual valid MAC address (therefore, different from "NEW Address", see IEC 61334-5-1:2001: Medium Access Control)

[SOURCE: IEC 61334-4-511:2000, 3.9.5]

3.2.6

reporting system

server system which issues a DiscoverReport

[SOURCE: IEC 61334-4-511:2000, 3.9.6, modified, to correct an error in IEC 61334-4-511:2000]

3.2.7

sub-slot

time needed to transmit two bytes by the physical layer

Note 1 to entry: Timeslots are divided to sub-slots in the RepeaterCall mode of the physical layer.

3.2.8

timeslot

time needed to transmit a physical frame

Note 1 to entry: As specified in IEC 61334-5-1, 3.3.1, a physical frame comprises 2 bytes preamble, 2 bytes start subframe delimiter, 38 bytes PSDU and 3 bytes pause.

3.3 Terms and definitions related to the PRIME NB OFDM PLC setup ICs (see 5.10)

Definitions related to the physical layer

3.3.1

base node

master node, which controls and manages the resources of a subnetwork

[SOURCE: ITU-T G.9904:2012, 3.2.1]

3.3.2

beacon slot

location of the beacon PDU within a frame

[SOURCE: ITU-T G.9904:2012, 3.2.2]

3.3.3**node**

any one element of a subnetwork, which is able to transmit to and receive from other subnetwork elements

[SOURCE: ITU-T G.9904:2012, 3.2.9]

3.3.4**registration**

process by which a service node is accepted as member of the subnetwork and allocated a LNID

[SOURCE: ITU-T G.9904:2012, 3.2.12]

3.3.5**service node**

any one node of a subnetwork, which is not a base node

[SOURCE: ITU-T G.9904:2012, 3.2.13]

3.3.6**subnetwork**

set of elements that can communicate by complying with this specification and share a single base node

[SOURCE: ITU-T G.9904:2012, 3.2.15]

Definitions related to the MAC layer**3.3.7****disconnected state <of a service node>**

initial functional state for all service nodes. When disconnected, a service node is not able to communicate data or switch other nodes' data; its main function is to search for a subnetwork within its reach and try to register on it

[SOURCE: ITU-T G.9904:2012, 8.1]

3.3.8**terminal state <of a service node>**

when in this functional state a service node is able to establish connections and communicate data, but it is not able to switch other nodes' data

[SOURCE: ITU-T G.9904:2012, 8.1]

3.3.9**switch state <of a service node>**

when in this functional state a service node is able to perform all Terminal functions. Additionally, it is able to forward data to and from other nodes in the same subnetwork. It is a branch point on the tree structure

[SOURCE: ITU-T G.9904:2012, 8.1]

3.3.10**promotion**

process by which a service node is qualified to switch (repeat, forward) data traffic from other nodes and act as a branch point on the subnetwork tree structure. A successful promotion

represents the transition between Terminal and Switch state. When a service node is in the Disconnected state, it cannot directly transition to Switch state

[SOURCE: ITU-T G.9904:2012, 8.1]

3.3.11

demotion

process by which a service node ceases to be a branch point on the subnetwork tree structure. A successful demotion represents the transition between Switch and Terminal state

[SOURCE: ITU-T G.9904:2012, 8.1]

3.4 Terms and definitions related to ZigBee® (see 5.12)

NOTE Terms marked with * are from the ZigBee® Specification.

3.4.1

CAD

Consumer Access Device; a ZigBee® gateway device that acts like an IHD within the ZigBee® network, but has an additional connection to a different network (i.e. WiFi)

3.4.2

IHD

in Home Display; a device that has a screen for the displaying of Energy information to the consumer

3.4.3

install code

a Hashed (via MMO) Pre-Configured Linked Key (PCLK) that is provided to a Trust Center via out-of-band communications

Note 1 to entry: A new device wishing to join the network would need to send this install code to the Trust Center, which would allow the Trust Center to execute the joining process, using this install code as part of the security information.

3.4.4

link key *

key that is shared exclusively between two, and only two, peer application-layer entities within a PAN

3.4.5

MAC address/IEEE address

these are used synonymously to represent the EUI-64 code allocated to the ZigBee® Radio

3.4.6

ZigBee®

specification for a suite of high level communication protocols used to create personal area networks built from small, low-power digital radios. ZigBee® is based on an IEEE 802.15 standard. Though low-powered, ZigBee® devices often transmit data over longer distances by passing data through intermediate devices to reach more distant ones, creating a mesh network

3.4.7

ZigBee® client

this is similar to the role of the DLMS/COSEM Client

Note 1 to entry: For a greater understanding of the interaction between the client and server the ZigBee® PRO specification should be read.

3.4.8**ZigBee® coordinator ***

an IEEE 802.15.4-2003 PAN coordinator that is the principal controller of an IEEE 802.15.4-2003-based network that is responsible for network formation. The PAN coordinator must be a full function device (FFD)

3.4.9**ZigBee® cluster**

set of message types related to a certain device function (e.g. metering, ballast control)

3.4.10**ZigBee® mirror**

device which echoes data being published by a battery operated ZigBee® device, allowing other network actors to obtain data while the battery operated device is unavailable due to power saving

3.4.11**ZigBee® PRO**

alternative name for the ZigBee® 2007 protocol. ZigBee® 2007, now the current stack release, contains two stack profiles, stack profile 1 (simply called ZigBee®), for home and light commercial use, and stack profile 2 (called ZigBee® PRO). ZigBee® PRO offers more features, such as multi-casting, many-to-one routing and high security with Symmetric-Key Key Exchange (SKKE), while ZigBee® (stack profile 1) offers a smaller footprint in RAM and flash. Both offer full mesh networking and work with all ZigBee® application profiles

3.4.12**ZigBee® router ***

an IEEE 802.15.4-2003 FFD participating in a ZigBee® network, which is not the ZigBee® coordinator but may act as an IEEE 802.15.4-2003 coordinator within its personal operating space, that is capable of routing messages between devices and supporting associations

3.4.13**ZigBee® server**

this is similar to the role of the DLMS/COSEM Server

Note 1 to entry: For a greater understanding of the interaction between the client and server the ZigBee® PRO specification should be read.

3.4.14**ZigBee® Trust Center ***

device trusted by devices within a ZigBee® network to distribute keys for the purpose of network and end-to-end application configuration management

3.5 Abbreviations

Abbreviation	Explanation
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Network
AA	Application Association
AARE	A-Associate Response – an APDU of the ACSE
AARQ	A-Associate Request – an APDU of the ACSE
ACSE	Association Control Service Element
ADP	Primary Station Address
ADS	Secondary Station Address
AGC	Automatic Gain Control
AL	Application layer
AP	Application process
APDU	Application Protocol Data Unit
APS	Application Support Sublayer (ZigBee® term)
ASE	Application Service Element
A-XDR	Adapted Extended Data Representation (IEC 61334-6)
base_name	The short_name corresponding to the first attribute ("logical_name") of a COSEM object
BCD	Binary Coded Decimal
BER	Bit Error Rate
CBCP	CallBack Control Protocol (PPP)
CC	Current Credit (S-FSK PLC profile)
CENELEC	European Committee for Electrotechnical Standardization
CHAP	Challenge Handshake Authentication Protocol
CIASE	Configuration Initiation Application Service Element (S-FSK PLC profile)
class_id	Interface class identification code
CLI	Calling Line Identity
COSEM	Companion Specification for Energy Metering
COSEM object	An instance of a COSEM interface class
CRC	Cyclic Redundancy Check
CSD	Circuit Switched Data
CSMA	Carrier Sense Multiple Access
CtoS	Client to Server challenge
CU	Currently Unused
DC	Delta credit (S-FSK PLC profile)
DHCP	Dynamic Host Control Protocol
DIB	Data Information Block (M-Bus)
DIF	Data Information Field (M-Bus)
DL	Data Link
DLMS	Device Language Message Specification
DLMS UA	DLMS User Association
DNS	Domain Name Server
DSCP	Differentiated Services Code Point
DSSID	Direct Switch ID
EAP	Extensible Authentication Protocol
EDGE	Enhanced Data rates for GSM Evolution

Abbreviation	Explanation
EUI-48	48-bit Extended Unique Identifier
EUI-64	64-bit Extended Unique Identifier
FCC	Federal Communications Commission
FFD	Full-Function Device
FIFO	First-In-First-Out
FTP	File Transfer Protocol
GCM	Galois/Counter Mode, an algorithm for authenticated encryption with associated data
GMT	Greenwich Mean Time. Replaced by Coordinated Universal Time (UTC).
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HAN	Home Area Network
HART	Highway Addressable Remote Transducer see http://www.hartcomm.org/ (in relation with the Sensor manager interface class)
HDLC	High-level Data Link Control
HES	Head End System
HHT	Hand Held Terminal
HLS	High Level Security Authentication
HSDPA	High-Speed Downlink Packet Access
IANA	Internet Assigned Numbers Authority
IB	Information Base
IC	Interface Class (COSEM)
IC	Initial credit (S-FSK PLC profile)
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IPCP	Internet Protocol Control Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO	International Organization for Standardization
ISP	Internet Service Provider
IT	Information Technology
ITU-T	International Telecommunication Union – Telecommunication
KEK	Key Encryption Key
LA	Local Area
LAC	Local Area Code
LAN	Local Area Network
LCID	Local Connection Identifier
LCP	Link Control Protocol
LDN	Logical Device Name
LLC	Logical Link Control (sublayer)
LLS	Low Level Security
LN	Logical Name
LNID	Local Node Identifier

Abbreviation	Explanation
LOADng	6LoWPAN Ad Hoc On-Demand Distance Vector Routing Next Generation (LOADng)
LQI	Link Quality Indicator (ZigBee ® term)
LSB	Least Significant Bit
LSID	Local Switch Identifier
LTE	Long Term Evolution (Wireless communication)
m	mandatory
MAC	Medium Access Control
M-Bus	Meter Bus
MD5	Message Digest Algorithm 5
MIB	Management Information Base (S-FSK PLC profile)
MID	Measuring Instruments Directive 2004/22/EC of the European Parliament and of the Council
MMO	Matyas-Meyer-Oseas hash (ZigBee ® term)
MPAN	(UK term) Meter Point Access Number – reference of the location of the Electricity meter on the electricity distribution network.
MPDU	MAC Protocol Data Unit
MSB	Most Significant Bit
MSDU	MAC Service Data Unit
MT	Mobile Termination
NB	Narrow-band
ND	Neighbour Discovery
NTP	Network Time Protocol
o	optional
OBIS	Object Identification System
OFDM	Orthogonal Frequency Division Multiplexing
OTA	Over the Air – Refers to Firmware Upgrade using ZigBee ®
PAN	Personal Area Network (Term used in relation to G3-PLC ¹⁾) and ZigBee ®
Pad	Padding
PAP	Password Authentication Protocol
PCLK	Pre-Configured Link Key (ZigBee® term)
PDU	Protocol Data Unit
PhL, PHY	Physical Layer
PIB	PLC Information Base
PIN	Personal Identity Number
PLC	Power Line Carrier
PLMN	Public Land Mobile Network
PNPDU	Promotion Needed PDU
POS	Personal Operating Space (ZigBee ®)
PPDU	Physical Protocol Data Unit
PPP	Point-to-Point Protocol
PSTN	Public Switched Telephone Network
REJ PDU	Reject Protocol Data Unit
RFC	Request for Comments; a document published by the Internet Engineering Task Force
RFD	Reduced Function Device
ROHC	Robust Header Compression

Abbreviation	Explanation
RREP	Route Reply
RREQ	Route Request
RRER	Route Error
RSSI	Receive Signal Strength Indication (ZigBee® term)
SAP	Service Access Point
SAS	Startup Attribute Set (ZigBee® term)
SCP	Shared Contention Period
SE	Smart Energy
SEP	Smart Energy Profile (ZigBee® term)
S-FSK	Spread – Frequency Shift Keying
SGERG88	Method for calculation of compressibility (Gas metering)
SHA	Secure Hash Algorithm
SID	Switch identifier
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SN	Short Name
SNA	Subnetwork Address
SSCS	Service Specific Convergence Layer
StoC	Server to Client Challenge
TAB	In the case of the EURIDIS profiles without DLMS and without DLMS/COSEM: data code. In the case of profiles using DLMS or DLMS/COSEM: value at which the equipment is programmed for Discovery
TABi	List of TAB field
TCC	Transmission Control Code (IPv4)
TCP	Transmission Control Protocol
TFTP	Trivial File Transfer Protocol
TOU	Time of use
TTL	Time To Live
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications System
UNC	Unconfigured (S-FSK PLC profile)
UTC	Coordinated Universal Time
VIB	Value Information Block (M-Bus)
VIF	Value Information Field (M-Bus)
VZ	Billing period counter (Form <i>Vorwertzähler</i> in German, see DIN 43863-3)
wake-up	trigger the meter to connect to the communication network to be available to a client (e.g. HES)
WAN	Wide Area Network
wM-Bus	Wireless M-Bus
ZTC	ZigBee® Trust Center
¹⁾ In the case of the G3-PLC technology, PAN may be defined as PLC Area Network.	

4 Basic principles

4.1 General

This Clause 4 describes the basic principles on which the COSEM interface classes (ICs) are built. It also gives a short overview on how interface objects – instantiations of the ICs – are used for communication purposes. Data collection systems and metering equipment from different vendors, following these specifications, can exchange data in an interoperable way.

For specification purposes, this standard uses the technique of object modelling.

An object is a collection of attributes and methods. Attributes represent the characteristics of an object. The value of an attribute may affect the behaviour of an object. The first attribute of any object is the *logical_name*. It is one part of the identification of the object. An object may offer a number of methods to either examine or modify the values of the attributes.

Objects that share common characteristics are generalized as an interface class, identified with a *class_id*. Within a specific IC, the common characteristics (attributes and methods) are described once for all objects. Instantiations of ICs are called COSEM interface objects.

Manufacturers may add proprietary methods and attributes to any object; see 4.2.

Figure 2 illustrates these terms by means of an example:

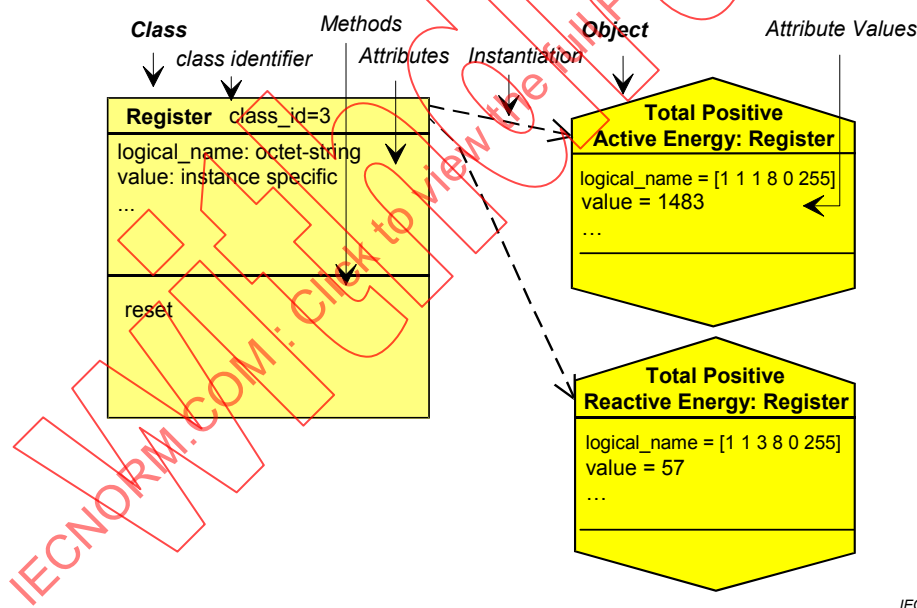


Figure 2 – An interface class and its instances

The IC “Register” is formed by combining the features necessary to model the behaviour of a generic register (containing measured or static information) as seen from the client (data collection system, hand held terminal). The contents of the register are identified by the attribute *logical_name*. The *logical_name* contains an OBIS identifier (see IEC 62056-6-1:2015). The actual (dynamic) content of the register is carried by its *value* attribute.

Defining a specific meter means defining several specific objects. In the example of Figure 2, the meter contains two registers; i.e. two specific instances of the IC “Register” are instantiated. Through the instantiation, one COSEM object becomes a “total, positive, active energy register” whereas the other becomes a “total, positive, reactive energy register”.

NOTE The COSEM interface objects (instances of COSEM ICs) represent the behaviour of the meter as seen from the "outside". Therefore, modifying the value of an attribute – for example resetting the *value* attribute of a register – is always initiated from the outside. Internally initiated changes of the attributes – for example updating the *value* attribute of a register – are not described in this model.

4.2 Referencing methods

Attributes and methods of COSEM objects can be referenced in two different ways:

Using logical names (LN referencing): In this case, the attributes and methods are referenced via the identifier of the COSEM object instance to which they belong.

The reference for an attribute is: class_id, value of the *logical_name* attribute, attribute_index.

The reference for a method is: class_id, value of the *logical_name* attribute, method_index, where:

- attribute_index is used as the identifier of the attribute required. Attribute indexes are specified in the definition of each IC. They are positive numbers starting with 1. Proprietary attributes may be added: these shall be identified with negative numbers;
- method_index is used as the identifier of the method required. Method indexes are specified in the definition of each IC. They are positive numbers starting with 1. Proprietary methods may be added: these shall be identified with negative numbers.

Using short names (SN referencing): This kind of referencing is intended for use in simple devices. In this case, each attribute and method of a COSEM object is identified with a 13-bit integer. The syntax for the short name is the same as the syntax of the name of a DLMS named variable. See IEC 61334-4-41:1996 and IEC 62056-5-3:2016, Clause 8.

4.3 Reserved base_names for special COSEM objects

In order to facilitate access to devices using SN referencing, some short_names are reserved as base_names for special COSEM objects. The range for reserved base_names is from 0xFA00 to 0xFFFF. The following specific base_names are defined, see Table 1.

Table 1 – Reserved base_names for SN referencing

Base_name (objectName)	COSEM object
0xFA00	Association SN
0xFB00	Script table (instantiation: Broadcast "Script table")
0xFC00	SAP assignment
0xFD00	"Data" or "Register" object containing the "COSEM logical device name" in the attribute "value"

4.4 Class description notation

This subclause describes the notation used to define the ICs.

A short text describes the functionality and application of the IC. A table gives an overview of the IC including the class name, the attributes, and the methods. Each attribute and method shall be described in detail. The template is shown below.

Class name		Cardinality	class_id, version			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	...	...				x + 0x...
3.	...	...				x + 0x...
Specific methods (if required)		m/o				
1.		...				
2.		...				
3.		...				

Class name	Describes the interface class (e.g. "Register", "Clock", "Profile generic"...).
	NOTE 1 Interface classes names are mentioned in quotation marks.
Cardinality	Specifies the number of instances of the IC within a logical device (see 4.8). <i>value</i> The IC shall be instantiated exactly "value" times. <i>min...max.</i> The IC shall be instantiated at least "min." times and at most "max." times. If min. is zero (0) then the IC is optional, otherwise (min. > 0) "min." instantiations of the IC are mandatory.
class_id	Identification code of the IC (range 0 to 65 535). The class_id of each object is retrieved together with the logical name by reading the <i>object_list</i> attribute of an "Association LN" / "Association SN" object. – class_id-s from 0 to 8 191 are reserved to be specified by the DLMS UA. – class_id-s from 8 192 to 32 767 are reserved for manufacturer specific ICs. – class_id-s from 32 768 to 65 535 are reserved for user group specific ICs. The DLMS UA reserves the right to assign ranges to individual manufacturers or user groups.
Version	Identification code of the version of the IC. The version of each object is retrieved together with the class_id and the logical name by reading the <i>object_list</i> attribute of an "Association LN" / "Association SN" object. Within one logical device, all instances of a certain IC shall be of the same version.
Attributes	Specifies the attributes that belong to the IC. <i>(dyn.)</i> Classifies an attribute that carries a process value, which is updated by the meter itself. <i>(static)</i> Classifies an attribute, which is not updated by the meter itself (for example configuration data). NOTE 2 There are some attributes which may be either static or dynamic depending on the application. In these cases this property is not indicated. NOTE 3 Attribute names use the underscore notation. When mentioned in the text they are in <i>italic</i> . Example: <i>logical_name</i>
logical_name	octet-string It is always the first attribute of an IC. It identifies the instantiation (COSEM object) of this IC. The value of the <i>logical_name</i> conforms to OBIS; see Clause 6 and IEC 62056-6-1:2015.

Data type	Defines the data type of an attribute; see 4.5.
Min.	Specifies if the attribute has a minimum value. x The attribute has a minimum value. <empty> The attribute has no minimum value.
Max.	Defines if the attribute has a maximum value. x The attribute has a maximum value. <empty> The attribute has no maximum value.
Def.	Specifies if the attribute has a default value. This is the value of the attribute after reset. x The attribute has a default value. <empty> The default value is not defined by the IC specification.
Short name	When Short Name (SN) referencing is used, each attribute and method of object instances has to be mapped to short names. The base_name x of each object instance is the DLMS named variable the logical name attribute is mapped to. It is selected in the implementation phase. The IC definition specifies the offsets for the other attributes and for the methods.
Specific method(s)	Provides a list of the specific methods that belong to the object. Method Name () The method has to be described in the subsection "Method description". NOTE 4 Method names use the underscore notation. When mentioned in the text they are in <i>italic</i> . Example: <i>add_object</i> .
m/o	Defines if the method is mandatory or optional. <i>m (mandatory)</i> The method is mandatory. <i>o (optional)</i> The method is optional.

Attribute description

Describes each attribute with its data type (if the data type is not simple), its data format and its properties (minimum, maximum and default values).

Method description

Describes each method and the invoked behaviour of the COSEM object(s) instantiated.

NOTE Services for accessing attributes or methods by the protocol are specified in IEC 62056-5-3:2016, 6.6 to 6.16.

Selective access

The xDLMS attribute-related services typically reference the entire attribute. However, for certain attributes selective access to just a part of the attribute may be provided. The part of the attribute is identified by specific selective access parameters. These are defined as part of the attribute specification.

4.5 Common data types

Table 2 contains the list of data types usable for attributes of COSEM objects.

Table 2 – Common data types

Type description	Tag ^a	Definition	Value range
-- simple data types			
null-data	[0]		
boolean	[3]	boolean	TRUE or FALSE
bit-string	[4]	An ordered sequence of boolean values	
double-long	[5]	Integer32	-2 147 483 648... 2 147 483 647
double-long-unsigned	[6]	Unsigned32	0...4 294 967 295
	[7]	Tag of the "floating-point" type in IEC 61334-4-41:1996, not usable in DLMS/COSEM. See tags [23] and [24]	
octet-string	[9]	An ordered sequence of octets (8 bit bytes)	
visible-string	[10]	An ordered sequence of ASCII characters	
	[11]	Tag of the "time" type in IEC 61334-4-41:1996, not usable in DLMS/COSEM. See tag [27]	
utf8-string	[12]	An ordered sequence of characters encoded as UTF-8	
bcd	[13]	binary coded decimal	
integer	[15]	Integer8	-128...127
long	[16]	Integer16	-32 768...32 767
unsigned	[17]	Unsigned8	0...255
long-unsigned	[18]	Unsigned16	0...65 535
long64	[20]	Integer64	- 2 ⁶³ ...2 ⁶³ -1
long64-unsigned	[21]	Unsigned64	0...2 ⁶⁴ -1
enum	[22]	The elements of the enumeration type are defined in the <i>Attribute description</i> or <i>Method description</i> section of a COSEM IC specification.	0...255
float32	[23]	OCTET STRING (SIZE(4))	For formatting, see 4.6.2.
float64	[24]	OCTET STRING (SIZE(8))	
date-time	[25]	OCTET STRING SIZE(12))	For formatting, see 4.6.1.
date	[26]	OCTET STRING (SIZE(5))	
time	[27]	OCTET STRING (SIZE(4))	
-- complex data types			
array	[1]	The elements of the array are defined in the <i>Attribute</i> or <i>Method description</i> section of a COSEM IC specification.	
structure	[2]	The elements of the structure are defined in the <i>Attribute</i> or <i>Method description</i> section of a COSEM IC specification.	
compact array	[19]	Provides an alternative, compact encoding of complex data.	
-- CHOICE		For some COSEM interface objects attributes, the data type may be chosen at instantiation, in the implementation phase of the COSEM server. The server always shall send back the data type and the value of each attribute, so that together with the logical name an unambiguous interpretation is ensured. The list of possible data types is defined in the "Attribute description" section of a COSEM IC specification.	

^a The tags are as defined in IEC 62056-5-3:2016, Clause 8.

4.6 Data formats

4.6.1 Date and time formats

Date and time information may be represented using the data type *octet-string*.

NOTE 1 In this case the encoding includes the tag of the data type *octet-string*, the length of the octet-string and the elements of *date*, *time* and *for date-time* as applicable.

Date and time information may be also represented using the data types *date*, *time* and *date-time*.

NOTE 2 In these cases, the encoding includes only the tag of the data types *date*, *time* or *date-time* as applicable and the elements of date, time or date-time.

NOTE 3 The (SIZE ()) specifications are applicable only when date, time or date-time are represented by the data types *date*, *time* or *date-time*.

<i>date</i>	OCTET STRING (SIZE(5)) <pre>{ year highbyte, year lowbyte, month, day of month, day of week }</pre> Where: year: interpreted as long-unsigned range 0...big 0xFFFF = not specified year highbyte and year lowbyte represent the 2 bytes of the long-unsigned month: interpreted as unsigned range 1..12, 0xFD, 0xFE, 0xFF 1 is January 0xFD = daylight_savings_end 0xFE = daylight_savings_begin 0xFF = not specified dayOfMonth: interpreted as unsigned range 1...31, 0xFD, 0xFE, 0xFF 0xFD = 2nd last day of month 0xFE = last day of month 0xE0 to 0xFC = reserved 0xFF = not specified dayOfWeek: interpreted as unsigned range 1...7, 0xFF 1 is Monday 0xFF = not specified For repetitive dates, the unused parts shall be set to "not specified".

The elements dayOfMonth and dayOfWeek shall be interpreted together:

- if last dayOfMonth is specified (0xFE) and dayOfWeek is wildcard, this specifies the last calendar day of the month;
- if last dayOfMonth is specified (0xFE) and an explicit dayOfWeek is specified (for example 7, Sunday) then it is the last occurrence of the weekday specified in the month, i.e. the last Sunday;
- if the year is not specified (0xFFFF), and dayOfMonth and dayOfWeek are both explicitly specified, this shall be interpreted as the dayOfWeek on, or following dayOfMonth;
- if the year and month are specified, and both the dayOfMonth and dayOfWeek are explicitly specified but the values are not consistent it shall be considered as an error.

Examples:

- 1) year = 0xFFFF, month = 0xFF, dayOfMonth = 0xFE, dayOfWeek = 0xFF: last day of the month in every year and month;
- 2) year = 0xFFFF, month = 0xFF, dayOfMonth = 0xFE, dayOfWeek = 0x07: last Sunday in every year and month;
- 3) year = 0xFFFF, month = 0x03, dayOfMonth = 0xFE, dayOfWeek = 0x07: last Sunday in March in every year;
- 4) year = 0xFFFF, month = 0x03, dayOfMonth = 0x01, dayOfWeek = 0x07: first Sunday in March in every year;
- 5) year = 0xFFFF, month = 0x03, dayOfMonth = 0x16, dayOfWeek = 0x05: fourth Friday in March in every year;
- 6) year = 0xFFFF, month = 0x0A, dayOfMonth = 0x16, dayOfWeek = 0x07: fourth Sunday in October in every year;
- 7) year = 0x07DE, month = 0x08, dayOfMonth = 0x13, (2014-08-13, Wednesday) dayOfWeek = 0x02 (Tuesday): error, as the dayOfMonth and dayOfWeek in the given year and month do not match.

time

OCTET STRING (SIZE(4))

{

hour,
minute,
second,
hundredths

}

Where:

hour: interpreted as unsigned range 0...23, 0xFF,
minute: interpreted as unsigned range 0...59, 0xFF,
second: interpreted as unsigned range 0...59, 0xFF,
hundredths: interpreted as unsigned range 0...99, 0xFF

For hour, minute, second and hundredths: 0xFF = not specified.

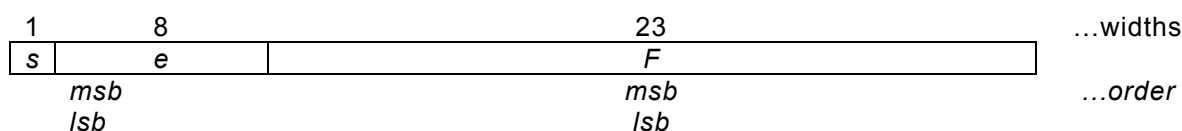
For repetitive times the unused parts shall be set to "not specified".

<i>date-time</i>	OCTET STRING (SIZE(12)) <pre>{ year highbyte, year lowbyte, month, day of month, day of week, hour, minute, second, hundredths of second, deviation highbyte, deviation lowbyte, clock status }</pre> The elements of <i>date</i> and <i>time</i> are encoded as defined above. Some may be set to “not specified” as defined above. In addition: deviation: interpreted as long range -720...+720 in minutes of local time to UTC 0x8000 = not specified Deviation highbyte and deviation lowbyte represent the 2 bytes of the long. clock_status interpreted as unsigned. The bits are defined as follows: bit 0 (LSB): invalid ^a value, bit 1: doubtful ^b value, bit 2: different clock base ^c, bit 3: invalid clock status ^d, bit 4: reserved, bit 5: reserved, bit 6: reserved, bit 7 (MSB): daylight saving active ^e 0xFF = not specified
^a Time could not be recovered after an incident. Detailed conditions are manufacturer specific (for example after the power to the clock has been interrupted). For a valid status, bit 0 shall not be set if bit 1 is set. ^b Time could be recovered after an incident but the value cannot be guaranteed. Detailed conditions are manufacturer specific. For a valid status, bit 1 shall not be set if bit 0 is set. ^c Bit is set if the basic timing information for the clock at the actual moment is taken from a timing source different from the source specified in clock_base. ^d This bit indicates that at least one bit of the clock status is invalid. Some bits may be correct. The exact meaning shall be explained in the manufacturer's documentation. ^e Flag set to true: the transmitted time contains the daylight saving deviation (summer time). Flag set to false: the transmitted time does not contain daylight saving deviation (normal time).	

4.6.2 Floating point number formats

Floating point number formats are defined in ISO/IEC/IEEE 60559:2011.

The single format is:



f is the fraction, it is 23 bits.

$$v = (-1)^s \cdot 2^{e-127} \cdot (1.f)$$

1	11	52	...
s	e	F	...
msb lsb		msb lsb	

f is the fraction, it is 52 bits.

$$v = (-1)^s \cdot 2^{e-1023} \cdot (1.f)$$

For details, see ISO/IEC/JEIE 60559:2011.

Floating-point numbers shall be represented as a fixed length octet-string, containing the 4 bytes (float32) of the single format or the 8 bytes (float64) of the double format floating-point number as specified above, most significant byte first.

Bit 31 Sign bit 0 0: + 1: -	Bits 30-23 Exponent field: 01111111 Decimal value of exponent field and exponent: $127 - 127 = 0$	Bits 22-0 Significand 1.000000000000000000000000 Decimal value of the significand: 1.0000000

NOTE The significand is the binary number 1 followed by the radix point followed by the binary bits of the fraction.

The encoding, including the tag of the data type is (all values are hexadecimal): 17 3F 80 00 00.

[illegible]

The encoding, including the tag of the data type is (all values are hexadecimal):
18 3F F0 00 00 00 00 00.

EXAMPLE 3 The decimal value “62056” represented in single floating-point format is:

Bit 31 Sign bit 0 0: + 1: -	Bits 30-23 Exponent field: 10001110 Decimal value of exponent field and exponent: 142 -127 = 15	Bits 22-0 Significand 1.111001001101000000000000 Decimal value of the significand: 1.8937988

The encoding, including the tag of the data type is (all values are hexadecimal): 17 47 72 68 00.

EXAMPLE 4 The decimal value “62056” represented in double floating-point format is:

[illegible]

The encoding, including the tag of the data type is (all values are hexadecimal): 18 40 EE 4D 00 00 00 00 00.

4.7 The COSEM server model

The COSEM server is structured into three hierarchical levels as shown in Figure 3:

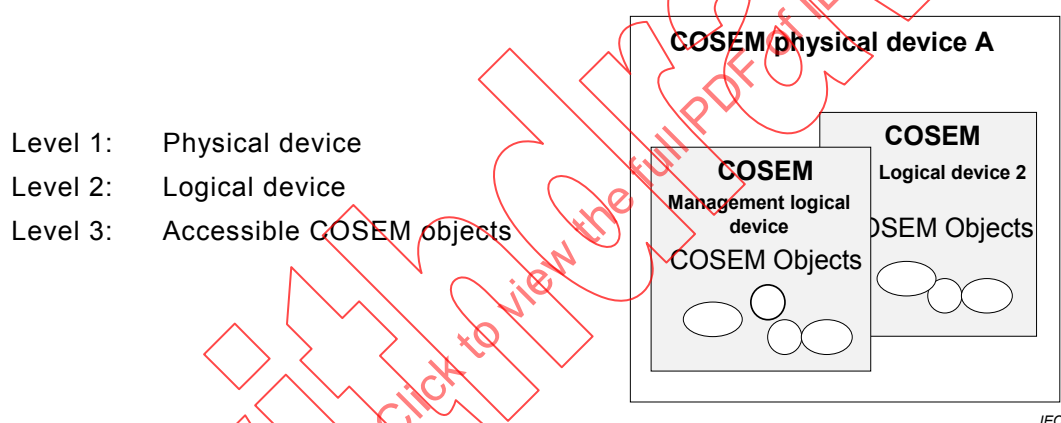


Figure 3 – The COSEM server model

The example in Figure 4 shows how a combined metering device can be structured using the COSEM server model.

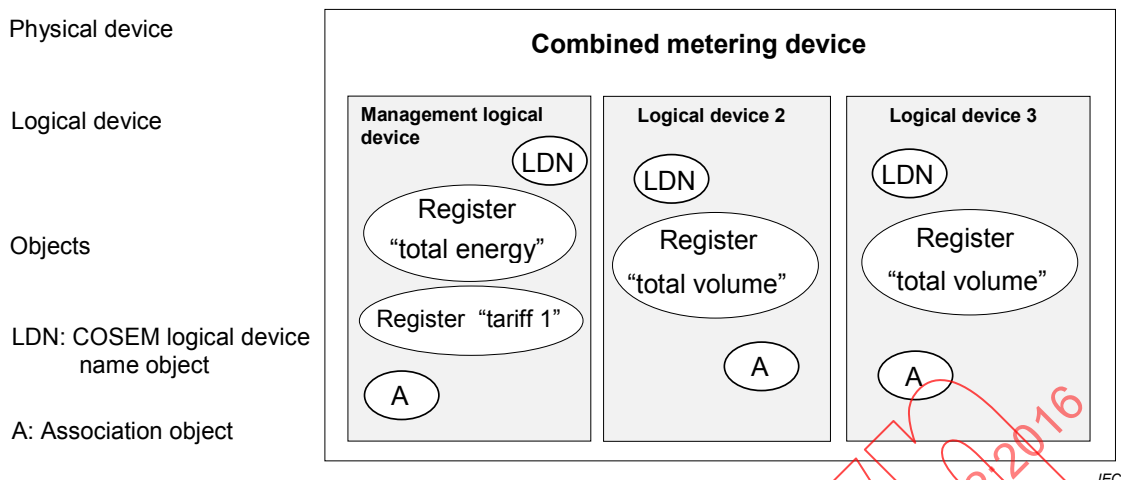


Figure 4 – Combined metering device

4.8 The COSEM logical device

4.8.1 General

The COSEM logical device contains a set of COSEM objects. Each physical device shall contain a "Management logical device".

The addressing of COSEM logical devices shall be provided by the addressing scheme of the lower layers of the protocol stack used.

4.8.2 COSEM logical device name (LDN)

The COSEM logical device can be identified by its unique COSEM LDN. This name can be retrieved from an instance of IC "SAP assignment" (see 5.3.5), or from a COSEM object "COSEM logical device name" (see 6.2.29).

The LDN is defined as an octet-string of up to 16 octets. The first three octets shall carry the manufacturer identifier³. The manufacturer shall ensure that the LDN, starting with the three octets identifying the manufacturer and followed by up to 13 octets, is unique.

4.8.3 The "association view" of the logical device

In order to access COSEM objects in the server, an application association (AA) shall first be established with a client. AAs identify the partners and characterize the context within which the associated applications will communicate. The major parts of this context are:

- the application context;
- the authentication mechanism;
- the xDLMS context.

AAs are modelled by special COSEM objects:

- instances of the IC "Association SN" – see 5.3.3 – are used with short name referencing;
- instances of the IC "Association LN" – see 5.3.4 – are used with logical name referencing;

³ Administered by the DLMS User Association, in cooperation with the FLAG Association.

Depending on the AA established between the client and the server, different access rights may be granted by the server. Access rights concern a set of COSEM objects – the visible objects – that can be accessed ('seen') within the given AA. In addition, access to attributes and methods of these COSEM objects may also be restricted within the AA (for example a certain type of client can only read a particular attribute of a COSEM object, but cannot write it). Access right may also stipulate required cryptographic protection.

The list of the visible COSEM objects – the "association view" – can be obtained by the client by reading the *object_list* attribute of the appropriate association object.

4.8.4 Mandatory contents of a COSEM logical device

The following objects shall be present in each COSEM logical device. They shall be accessible for GET/Read in all AAs with this logical device:

- COSEM logical device name object;
- current "Association" (LN or SN) object.

If the "SAP Assignment" object is present, then the COSEM logical device name object does not have to be present.

4.8.5 Management logical device

As specified in 4.8.1, the management logical device is a mandatory element of any physical device. It has a reserved address. It shall support an AA to a public client with the lowest level security (no security) authentication. Its role is to support revealing the internal structure of the physical device and to support notification of events in the server.

In addition to the "Association" object modelling the AA with the public client, the management logical device shall contain a "SAP assignment" object, giving its SAP and the SAP of all other logical devices within the physical device. The SAP assignment object shall be readable at least by the public client.

If there is only one logical device within the physical device, the "SAP assignment" object may be omitted.

4.9 Information security

DLMS/COSEM provides several information security features for accessing and transporting data:

- data access security controls access to the data held by a DLMS/COSEM server;
- data transport security allows the sending party to apply cryptographic protection to the xDLMS APDUs sent. This requires ciphered APDUs. The receiving party can remove or check this protection

For a description of these security mechanisms, see IEC 62056-5-3:2016, Clause 5.

Information security is provided on the DLMS/COSEM Application layer level and on the COSEM object level and it is supported / managed by the following objects:

- "Association SN", see 5.3.3;
- "Association LN", see 5.3.4; and
- "Security setup", see 5.3.7.

5 The COSEM interface classes

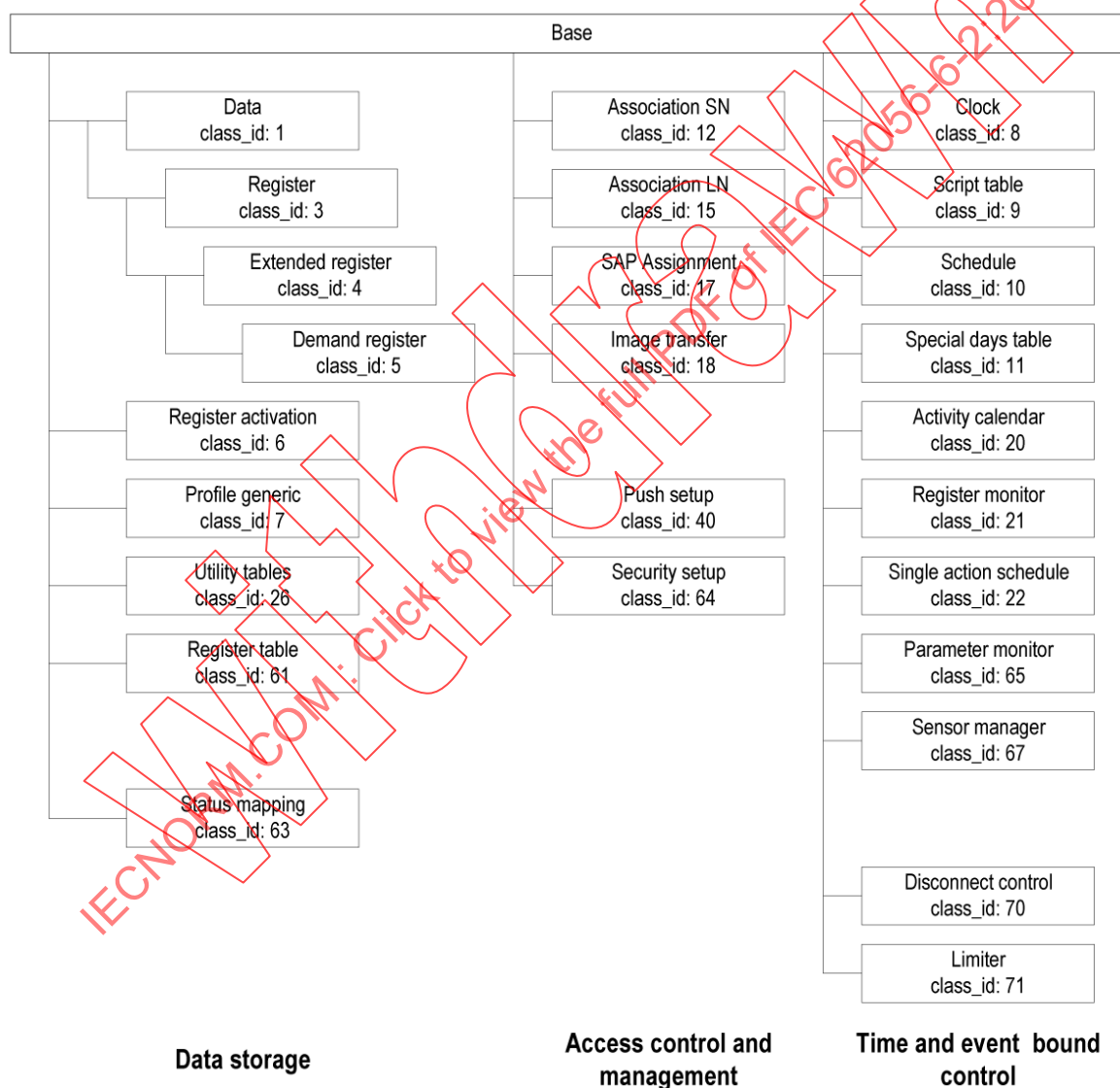
5.1 Overview

The ICs defined currently and the relations between them are shown in Figure 5 and Figure 6.

NOTE 1 The IC “base” itself is not specified explicitly. It contains only one attribute *logical_name*.

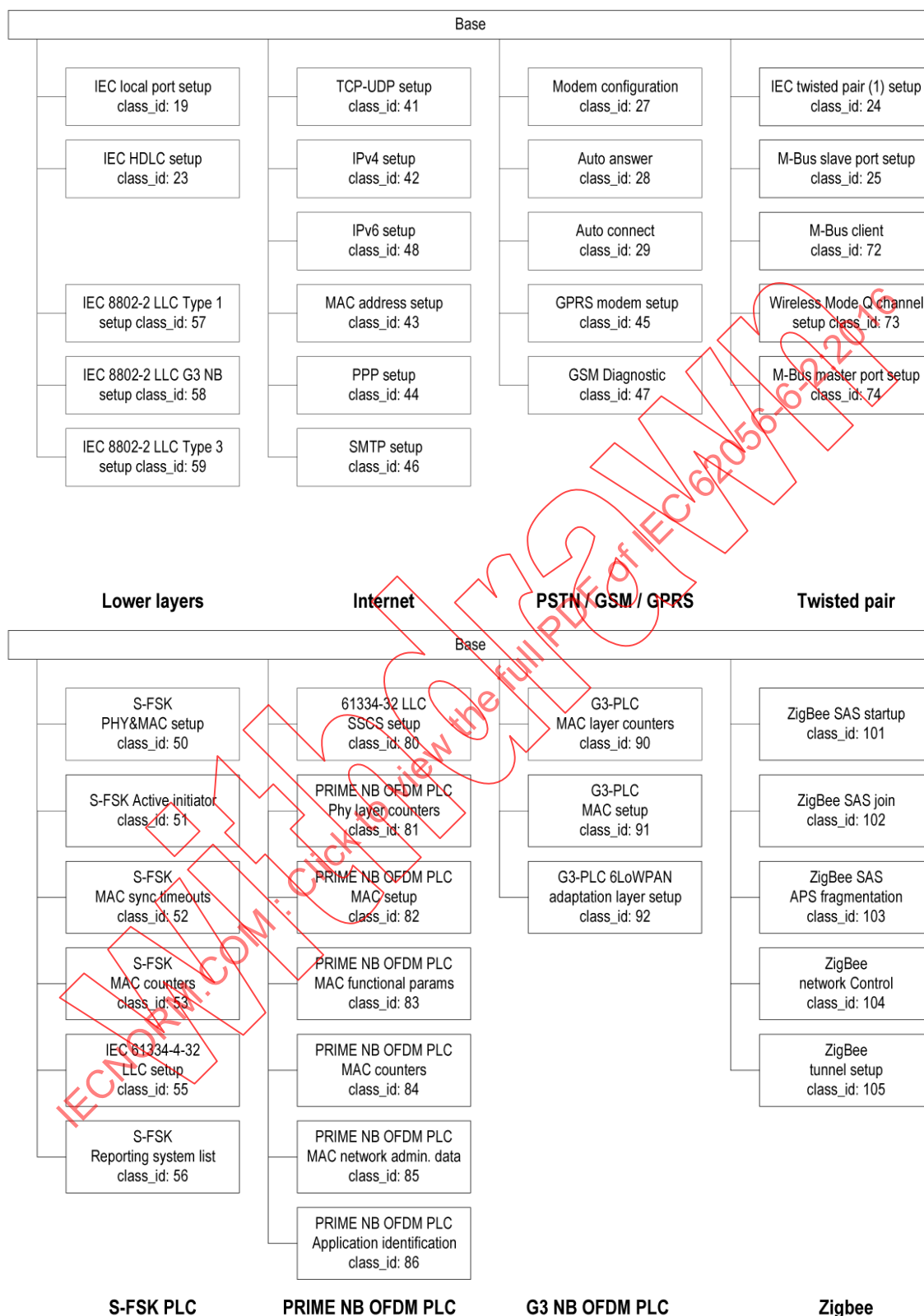
NOTE 2 In the description of the “Demand register”, “Clock” and “Profile generic” ICs, the 2nd attributes are labelled differently from that of the 2nd attribute of the “Data” IC, namely *current_average_value*, *time* and *buffer* vs. *value*. This is to emphasize the specific nature of the *value*.

NOTE 3 On these Figures the interface classes are presented in each group by increasing class_id. In the clauses specifying the various groups of interface classes, the new interface classes are put at the end of the relevant clause.



IEC

Figure 5 – Overview of the interface classes – Part 1



IEC

Figure 6 – Overview of the interface classes – Part 2

Table 3 lists the interface classes by class_id.

Table 3 – List of interface classes by class_id

Interface class name	class_id	version(s)	Clause
Data	1	0	5.2.1
Register	3	0	5.2.2
Extended register	4	0	5.2.3
Demand register	5	0	5.2.4
Register activation	6	0	5.2.5
Profile generic	7	1	5.2.6
		0	7.2
Clock	8	0	5.4.1
Script table	9	0	5.4.2
Schedule	10	0	5.4.3
Special days table	11	0	5.4.4
Association SN	12	3	5.3.3
		2	7.5
		1	7.4
		0	7.3
Association LN	15	2	5.3.4
		1	7.7
		0	7.6
SAP Assignment	17	0	5.3.5
Image transfer	18	0	5.3.6
IEC local port setup	19	1	5.5.1
		0	7.8
Activity calendar	20	0	5.4.5
Register monitor	21	0	5.4.6
Single action schedule	22	0	5.4.7
IEC HDLC setup	23	1	5.5.2
		0	7.9
IEC twisted pair (1) setup	24	1	5.5.3
		0	7.10
M-BUS slave port setup	25	0	5.6.1
Utility tables	26	0	5.2.7
Modem configuration PSTN modem configuration	27	1	5.5.4
		0	7.11
Auto answer	28	2	5.5.5
		1	–
		0	7.12
Auto connect PSTN Auto dial	29	2	5.5.6
		1	7.14
		0	7.13
Push setup	40	0	5.3.8.2
TCP-UDP setup	41	0	5.7.1
IPv4 setup	42	0	5.7.2

Interface class name	class_id	version(s)	Clause
MAC address setup (Ethernet setup)	43	0	5.7.4, 5.10.10
PPP setup	44	0	5.7.5
GPRS modem setup	45	0	5.5.7
SMTP setup	46	0	5.7.6
GSM diagnostic	47	0	5.5.8
IPv6 setup	48	0	5.7.3
S-FSK Phy&MAC setup	50	1 0	5.8.3 7.15
S-FSK Active initiator	51	0	5.8.4
S-FSK MAC synchronization timeouts	52	0	5.8.5
S-FSK MAC counters	53	0	5.8.6
IEC 61334-4-32 LLC setup	55	1	5.8.7
S-FSK IEC 61334-4-32 LLC setup		0	7.16
S-FSK Reporting system list	56	0	5.8.8
ISO/IEC 8802-2 LLC Type 1 setup	57	0	5.9.2
ISO/IEC 8802-2 LLC Type 2 setup	58	0	5.9.3
ISO/IEC 8802-2 LLC Type 3 setup	59	0	5.9.4
Register table	61	0	5.2.8
Status mapping	63	0	5.2.9
Security setup	64	0	5.3.7
Parameter monitor	65	0	5.4.10
Sensor manager	67	0	5.4.11
Disconnect control	70	0	5.4.8
Limiter	71	0	5.4.9
M-Bus client	72	1 0	5.6.2 7.17
Wireless Mode Q channel	73	0	5.6.3
M-Bus master port setup	74	0	5.6.4
61334-4-32 LLC SSCS setup	80	0	5.10.3
PRIME NB OFDM PLC Physical layer counters	81	0	5.10.5
PRIME NB OFDM PLC MAC setup	82	0	5.10.6
PRIME NB OFDM PLC MAC functional parameters	83	0	5.10.7
PRIME NB OFDM PLC MAC counters	84	0	5.10.8
PRIME NB OFDM PLC MAC network administration data	85	0	5.10.9
PRIME NB OFDM PLC Application identification	86	0	5.10.11
G3-PLC MAC layer counters	90	1	5.11.3
G3 NB OFDM PLC MAC layer counters		0	7.18
G3-PLC MAC setup	91	1	5.11.4
G3 NB OFDM PLC MAC setup		0	7.19
G3-PLC 6LoWPAN adaptation layer setup	92	1	5.11.5
G3 NB OFDM PLC 6LoWPAN adaptation layer setup		0	7.20
ZigBee® SAS startup	101	0	5.12.2
ZigBee® SAS join	102	0	5.12.3
ZigBee® SAS APS fragmentation	103	0	5.12.4

Interface class name	class_id	version(s)	Clause
ZigBee® network control	104	0	5.12.5
ZigBee® tunnel setup	105	0	5.12.6

5.2 Interface classes for parameters and measurement data

5.2.1 Data (class_id = 1, version = 0)

This IC allows modelling various data, such as configuration data and parameters. The data are identified by the attribute *logical_name*.

Data	0...n	class_id = 1, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. value	CHOICE				x + 0x08
Specific methods	m/o				

Attribute description

logical_name	Identifies the “Data” object instance. See clauses 6 and IEC 62056-6-1:2015.				
value	Contains the data. CHOICE <pre>{ -- simple data types null-data [0], boolean [3], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], utf8-string [12], bcd [13], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], enum [22], float32 [23], float64 [24], date-time [25], date [26], time [27], -- complex data types array [1], structure [2], compact-array [19] }</pre> The data type depends on the instantiation defined by the <i>logical_name</i> and possibly from the manufacturer. It has to be chosen so that together with the <i>logical_name</i>, an unambiguous interpretation is possible. Any simple and complex data types listed in 4.5 can be used, unless the choice is restricted in Clause 6.				

5.2.2 Register (class_id = 3, version = 0)

This IC allows modelling a process or a status value with its associated scaler and unit. “Register” objects know the nature of the process or status value. It is identified by the attribute *logical_name*.

Register	0...n	class_id = 3, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. value	CHOICE				x + 0x08
3. scaler_unit (static)	scal_unit_type				x + 0x10
Specific methods	m/o				
1. reset (data)	o				x + 0x28

Attribute description

logical_name Identifies the “Register” object instance. See Clause 6 and IEC 62056-6-1:2015.

value Contains the current process or status value.

CHOICE

{

 -- simple data types

 null-data [0],

 bit-string [4],

 double-long [5],

 double-long-unsigned [6],

 octet-string [9],

 visible-string [10],

 utf8-string [12],

 integer [15],

 long [16],

 unsigned [17],

 long-unsigned [18],

 long64 [20],

 long64-unsigned [21],

 enum [22],

 float32 [23],

 float64 [24],

}

The data type of the value depends on the instantiation defined by *logical_name* and possibly on the choice of the manufacturer. It has to be chosen so that, together with the *logical_name*, an unambiguous interpretation of the value is possible.

When, instead of a “Data” object, a “Register” object is used, (with the *scaler_unit* attribute not used or with *scaler* = 0, *unit* = 255) then the data types allowed for the *value* attribute of the “Data” IC are allowed.

scaler_unit Provides information on the unit and the scaler of the value.

scal_unit_type::= structure

{

 scaler,

 unit

}

scaler: integer

This is the exponent (to the base of 10) of the multiplication factor.

REMARK If the value is not numerical, then the scaler shall be set to 0.

unit: enum

Enumeration defining the physical unit; for details see Table 4 below.

Method description

reset (data) Forces a reset of the object. By invoking this method, the value is set to the default value. The default value is an instance specific constant.

data::= integer (0)

Table 4 – Enumerated values for physical units

unit::=enum	Unit	Quantity	Unit name	SI definition (comment)
(1)	a	time	year	
(2)	mo	time	month	
(3)	wk	time	week	7*24*60*60 s
(4)	d	time	day	24*60*60 s
(5)	h	time	hour	60*60 s
(6)	min.	time	min	60 s
(7)	s	time (<i>t</i>)	second	s
(8)	°	(phase) angle	degree	rad*180/π
(9)	°C	temperature (<i>T</i>)	degree-celsius	K-273.15
(10)	currency	(local) currency		
(11)	m	length (<i>l</i>)	metre	m
(12)	m/s	speed (<i>v</i>)	metre per second	m/s
(13)	m ³	volume (<i>V</i>) <i>r_V</i> , meter constant or pulse value (volume)	cubic metre	m ³
(14)	m ³	corrected volume	cubic metre	m ³
(15)	m ³ /h	volume flux	cubic metre per hour	m ³ /(60*60s)
(16)	m ³ /h	corrected volume flux	cubic metre per hour	m ³ /(60*60s)
(17)	m ³ /d	volume flux		m ³ /(24*60*60s)
(18)	m ³ /d	corrected volume flux		m ³ /(24*60*60s)
(19)	l	volume	litre	10 ⁻³ m ³
(20)	kg	mass (<i>m</i>)	kilogram	
(21)	N	force (<i>F</i>)	newton	
(22)	Nm	energy	newton meter	J = Nm = Ws
(23)	Pa	pressure (<i>p</i>)	pascal	N/m ²
(24)	bar	pressure (<i>p</i>)	bar	10 ⁵ N/m ²
(25)	J	energy	joule	J = Nm = Ws
(26)	J/h	thermal power	joule per hour	J/(60*60s)
(27)	W	active power (<i>P</i>)	watt	W = J/s
(28)	VA	apparent power (<i>S</i>)	volt-ampere	
(29)	var	reactive power (<i>Q</i>)	var	
(30)	Wh	active energy <i>r_W</i> , active energy meter constant or pulse value	watt-hour	W*(60*60s)
(31)	VAh	apparent energy <i>r_S</i> , apparent energy meter constant or pulse value	volt-ampere-hour	VA*(60*60s)
(32)	varh	reactive energy <i>r_B</i> , reactive energy meter constant or pulse value	var-hour	var*(60*60s)
(33)	A	current (<i>I</i>)	ampere	A
(34)	C	electrical charge (<i>Q</i>)	coulomb	C = As
(35)	V	voltage (<i>U</i>)	volt	V
(36)	V/m	electric field strength (<i>E</i>)	volt per metre	V/m
(37)	F	capacitance (<i>C</i>)	farad	C/V = As/V
(38)	Ω	resistance (<i>R</i>)	ohm	Ω = V/A
(39)	Ωm ² /m	resistivity (<i>ρ</i>)		Ωm
(40)	Wb	magnetic flux (<i>Φ</i>)	weber	Wb = Vs
(41)	T	magnetic flux density (<i>B</i>)	tesla	Wb/m ²

unit::=enum	Unit	Quantity	Unit name	SI definition (comment)
(42)	A/m	magnetic field strength (H)	ampere per metre	A/m
(43)	H	inductance (L)	henry	H = Wb/A
(44)	Hz	frequency (f, ω)	hertz	1/s
(45)	1/(Wh)	R_W , active energy meter constant or pulse value		
(46)	1/(varh)	R_B , reactive energy meter constant or pulse value		
(47)	1/(VAh)	R_S , apparent energy meter constant or pulse value		
(48)	V ² h	volt-squared hour, r_{U2h} , volt-squared hour meter constant or pulse value	volt-squared-hours	V ² (60*60s)
(49)	A ² h	ampere-squared hour, r_{I2h} , ampere-squared hour meter constant or pulse value	ampere-squared-hours	A ² (60*60s)
(50)	kg/s	mass flux	kilogram per second	kg/s
(51)	S, mho	conductance	siemens	1/Ω
(52)	K	temperature (T)	kelvin	
(53)	1/(V ² h)	R_{U2h} , volt-squared hour meter constant or pulse value		
(54)	1/(A ² h)	R_{I2h} , ampere-squared hour meter constant or pulse value		
(55)	1/m ³	R_V , meter constant or pulse value (volume)		
(56)		percentage	%	
(57)	Ah	ampere-hours	Ampere-hour	
...				
(60)	Wh/m ³	energy per volume	3,6*10 ³ J/m ³	
(61)	J/m ³	calorific value, wobbe		
(62)	Mol %	molar fraction of gas composition	mole percent	(Basic gas composition unit)
(63)	g/m ³	mass density, quantity of material		(Gas analysis, accompanying elements)
(64)	Pa s	dynamic viscosity	pascal second	(Characteristic of gas stream)
(65)	J/kg	Specific energy NOTE The amount of energy per unit of mass of a substance	Joule / kilogram	$m^2 \cdot kg \cdot s^{-2} / kg$ $= m^2 \cdot s^{-2}$
....				
(70)	dBm	Signal strength, dB milliwatt (e.g. of GSM radio systems)		
(71)	dbuV	Signal strength, dB microvolt		
(72)	dB	Logarithmic unit that expresses the ratio between two values of a physical quantity		
...				
(253)		reserved		
(254)	other	other unit		
(255)	count	no unit, unitless, count		

Some examples are shown in Table 5 below.

Table 5 – Examples for scaler_unit

Value	Scaler	Unit	Data
263788	-3	m ³	263,788 m ³
593	3	Wh	593 kWh
3467	-1	V	346,7
3467	0	V	3 467 V
3467	1	V	34 670 V

5.2.3 Extended register (class_id = 4, version = 0)

This IC allows modelling a process value with its associated scaler, unit, status and capture time information. “Extended register” objects know the nature of the process value. It is described by the attribute *logical_name*.

Extended register		0...n	class_id = 4, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	value (dyn.)	CHOICE				x + 0x08
3.	scaler_unit (static)	scal_unit_type				x + 0x10
4.	status (dyn.)	CHOICE				x + 0x18
5.	capture_time (dyn.)	octet-string				x + 0x20
Specific methods		m/o				
1.	reset (data)	o				x + 0x38

Attribute description

logical_name Identifies the “Extended register” object instance. See 6.3.1.

value See the specification of the IC “Register” in 5.2.2.

scaler_unit See the specification of the IC “Register” in 5.2.2.

status Provides “Extended register” specific status information. The meaning of the elements of the status shall be provided for each object instance.

CHOICE

-- simple data types

null-data [0],
 bit-string [4],
 double-long-unsigned [6],
 octet-string [9],
 visible-string [10],
 utf8-string [12],
 unsigned [17],
 long-unsigned [18],
 long64-unsigned [21]
 }

The data type and the encoding depend on the instantiation and possibly on the choice of the manufacturer.

For the interpretation, extra information from the manufacturer may be necessary.

Def. Depending on the status type definition.

capture_time Provides an “Extended register” specific date and time information showing when the value of the attribute *value* has been captured.
octet-string, formatted as specified in 4.6.1 for *date-time*.

Method description

reset (data) This method forces a reset of the object. By invoking this method, the attribute *value* is set to the default value. The default value is an instance specific constant.
The attribute *capture_time* is set to the time of the reset execution.
data::= integer (0)

5.2.4 Demand register (class_id = 5, version = 0)

This IC allows modelling a demand value with its associated scalar, unit, status and time information. A “Demand register” object measures and computes a *current_average_value* periodically and it stores a *last_average_value*. The time interval T over which the demand is calculated is defined by specifying *number_of_periods* and *period*. See Figure 7.

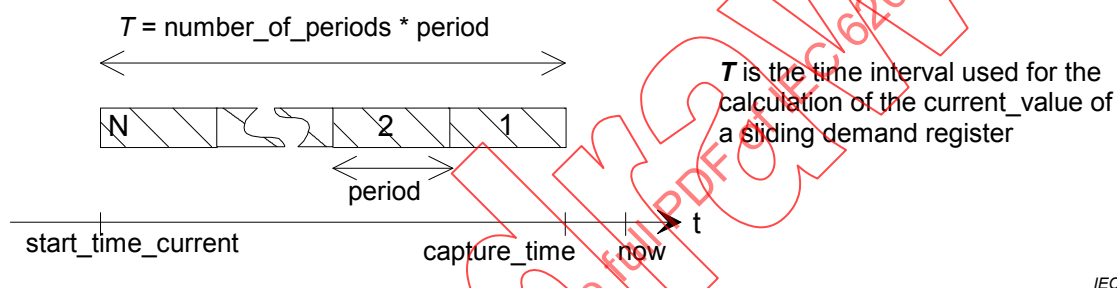


Figure 7 – The time attributes when measuring sliding demand

The demand register delivers two types of demand: *current_average_value* and *last_average_value* (see Figure 8 and Figure 9).

“Demand register” objects know the nature the of process value, which is described by the attribute *logical_name*.

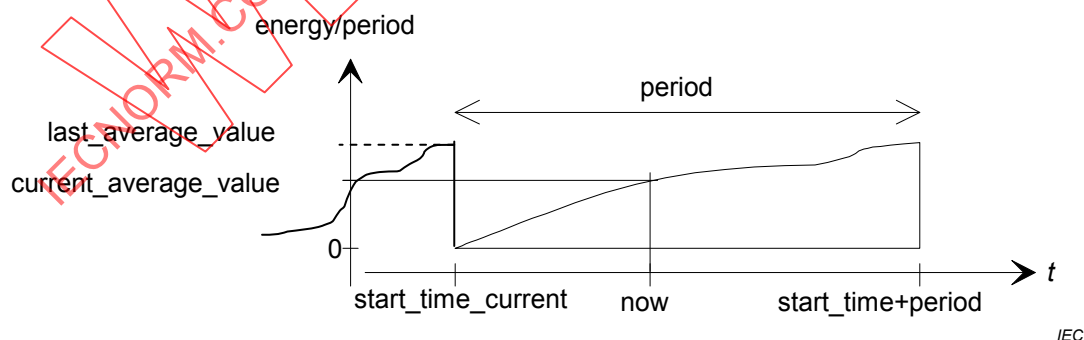
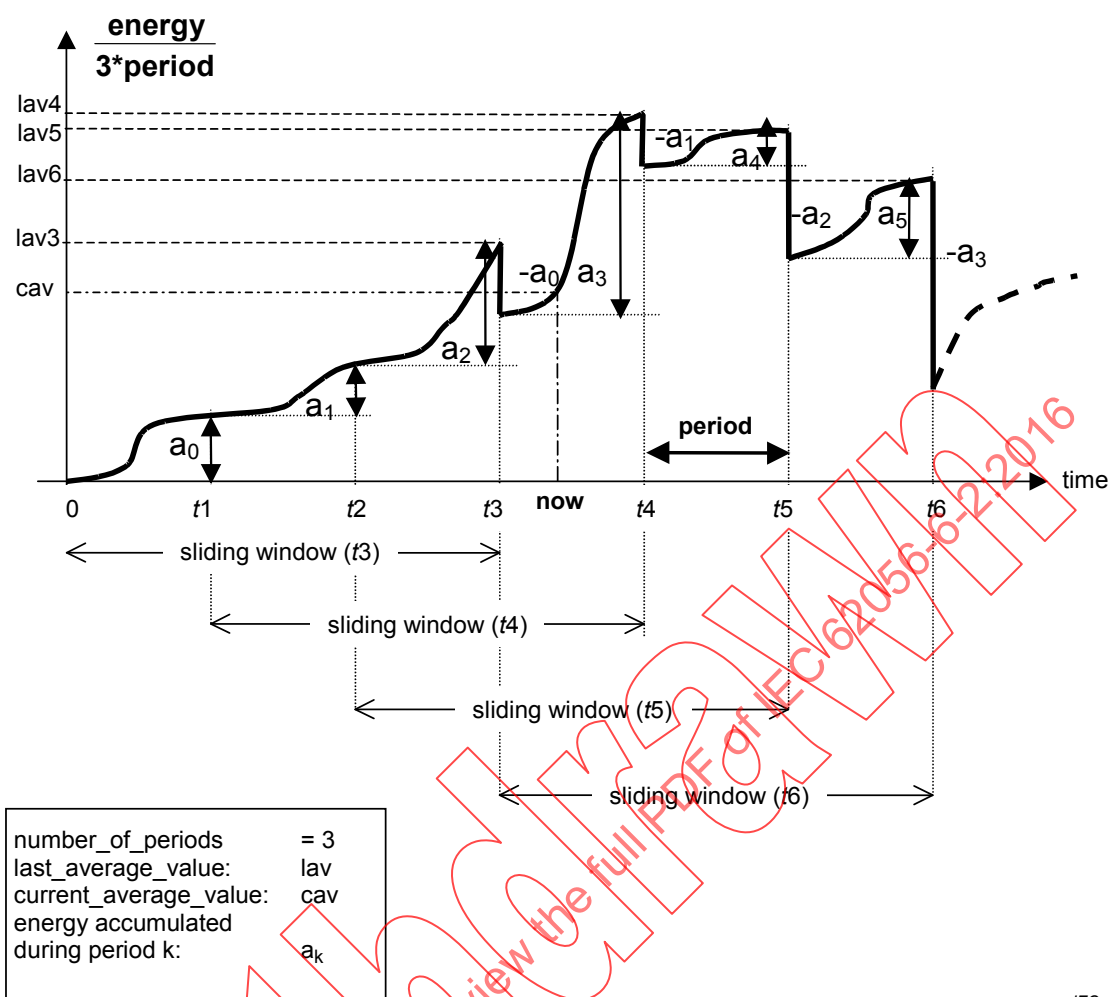


Figure 8 – The attributes in the case of block demand



IEC

Figure 9 – The attributes in the case of sliding demand (number of periods = 3)

Demand register		0...n	class_id = 5, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. current_average_value	(dyn.)	CHOICE			0	x + 0x08
3. last_average_value	(dyn.)	CHOICE			0	x + 0x10
4. scaler_unit	(static)	scal_unit_type				x + 0x18
5. status	(dyn.)	CHOICE				x + 0x20
6. capture_time	(dyn.)	octet-string				x + 0x28
7. start_time_current	(dyn.)	octet-string				x + 0x30
8. period	(static)	double-long-unsigned	1			x + 0x38
9. number_of_periods	(static)	long-unsigned	1		1	x + 0x40
Specific methods		m/o				
1. reset (data)		o				x + 0x48
2. next_period (data)		o				x + 0x50

Attribute description	
logical_name	Identifies the “Demand register” object instance. See 6.3.1 and IEC 62056-6-1:2015.
current_average_value	Provides the current value (running demand) of the energy accumulated since <i>start_time</i>, divided by <i>number_of_periods*period</i>. The data type of the value depends on the instantiation defined by <i>logical_name</i> and possibly on the choice of the manufacturer. The type has to be chosen so that – together with the <i>logical_name</i> – unambiguous interpretation of the value is possible. If a quantity other than energy is measured, other calculation methods may apply (for example for calculating average values of voltage or current). CHOICE For data types, see “Register” in 5.2.2, value attribute.
last_average_value	Provides the value of the energy accumulated (over the last <i>number_of_periods*period</i>) divided by <i>number_of_periods*period</i>. The energy of the current (not terminated) period is not considered by the calculation. If a quantity other than energy is measured, other calculation methods may apply (for example for calculating average values of voltage or current). CHOICE For data types, see “Register” IC, value attribute.
scaler_unit	See the specification of IC “Register” in 5.2.2.
status	Provides “Demand register” specific status information. The data type and the encoding depend on the instantiation and possibly on the choice of the manufacturer. For the interpretation, extra information from the manufacturer may be necessary. CHOICE For data types, see “Extended register” IC in 5.2.3, status attribute. Def. Depending on the status type definition.
capture_time	Provides the date and time when the <i>last_average_value</i> has been calculated. octet-string, formatted as specified in 4.6.1 for <i>date-time</i>.
start_time_current	Provides the date and time when the measurement of the <i>current_average_value</i> has been started. octet-string, formatted as specified in 4.6.1 for <i>date-time</i>.
period	Period is the interval between two successive updates of the <i>last_average_value</i>. (<i>number_of_periods*period</i> is the denominator for the calculation of the demand). double-long-unsigned Measuring period in seconds The behaviour of the meter after writing a new value to this attribute shall be specified by the manufacturer.
number_of_periods	The number of periods used to calculate the <i>last_average_value</i>. <i>number_of_periods</i> >= 1 – <i>number_of_periods</i> > 1 indicates that the <i>last_average_value</i> represents “sliding demand”, – <i>number_of_periods</i> = 1 indicates that the <i>last_average_value</i> represents “block demand”. The behaviour of the meter after writing a new value to this attribute shall be specified by the manufacturer.

Method description

reset (data) This method forces a reset of the object. Activating this method provokes the following actions:

- the current period is terminated;
- the *current_average_value* and the *last_average_value* are set to their default values;
- the *capture_time* and the *start_time_current* are set to the time of the execution of *reset (data)*.

data::= integer (0)

next_period (data) This method is used to trigger the regular termination (and restart) of a period. Closes (terminates) the current measuring period. Updates *capture_time* and *start_time* and copies *current_average_value* to *last_average_value*, sets *current_average_value* to its default value. Starts the next measuring period.

REMARK The old *last_average_value* (and *capture_time*) can be read during the time "period". The old *current_average_value* is not available any more at the interface.

data::= integer (0)

5.2.5 Register activation (class_id = 6, version = 0)

This IC allows modelling the handling of different tariffication structures. To each "Register activation" object, groups of "Register", "Extended register" or "Demand register" objects, modelling different kind of quantities (for example active energy, active demand, reactive energy, etc.) are assigned. Subgroups of these registers, defined by the *activation masks* define different tariff structures (for example day tariff, night tariff). One of these activation masks, the *active_mask*, defines which subset of the registers, assigned to the "Register activation" object instance is active. Registers not included in the *register_assignment* attribute of any "Register activation" object are always enabled by default.

Register activation		0...n	class_id = 6, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	register_assignment (static)	array				x + 0x08
3.	mask_list (static)	array				x + 0x10
4.	active_mask (dyn.)	octet-string				x+ 0x18
Specific methods		m/o				
1.	add_register (data)	o				x + 0x30
2.	add_mask (data)	o				x + 0x38
3.	delete_mask (data)	o				x + 0x40

Attribute description	
logical_name	Identifies the “Register activation” object instance. See 6.2.11.
register_assignment	Specifies an ordered list of COSEM objects assigned to the “Register activation” object. The list may contain different kinds of COSEM objects, for example instances of “Register”, “Extended register” or “Demand register”. array object_definition object_definition::= structure { class_id: long-unsigned, logical_name: octet-string }
mask_list	Specifies a list of register activation masks. Each entry (mask) is identified by its mask_name and contains an array of indices referring to the registers assigned to the mask (the first object in register_assignment is referenced by index 1, the second object by index 2,...). array register_act_mask register_act_mask::= structure { mask_name: octet-string, index_list: index_array } mask_name has to be uniquely defined within the object index_array::= array of unsigned
active_mask	Defines the currently active mask. The mask is defined by its mask_name (see mask_list). The <i>active_mask</i> defines the registers currently enabled; all other registers listed in the <i>register_assignment</i> are disabled.
Method description	
add_register (data)	Adds one more registers to the attribute <i>register_assignment</i>. The new register is added at the end of the array; i.e. the newly added register has the highest index. The indices of the existing registers are not modified. data::= structure { class_id: long-unsigned, logical_name: octet-string }
add_mask (data)	Adds another mask to the attribute <i>mask_list</i>. If there exists already a mask with the same name, the existing mask will be overwritten by the new mask. data::= register_act_mask (see above)
delete_mask (data)	Deletes a mask from the attribute <i>mask_list</i>. The mask is defined by its mask name. data::= octet-string (mask_name)

5.2.6 Profile generic (class_id = 7, version = 1)

This IC provides a generalized concept allowing to store, sort and access data groups or data series, called *capture objects*. Capture objects are appropriate attributes or elements of (an) attribute(s) of COSEM objects. The capture objects are collected periodically or occasionally.

A profile has a *buffer* to store the captured data. To retrieve only a part of the buffer, either a value range or an entry range may be specified, asking to retrieve all entries that fall within the range specified.

The list of *capture objects* defines the values to be stored in the *buffer* (using auto capture or the method *capture*). The list is defined statically to ensure homogenous buffer entries (all entries have the same size and structure). If the list of capture objects is modified, the *buffer* is cleared. If the buffer is captured by other “Profile generic” objects, their *buffer* is cleared as well, to guarantee the homogeneity of their *buffer* entries.

The *buffer* may be defined as sorted by one of the *capture objects*, e.g. the clock, or the entries are stacked in a “last in first out” order. For example, it is very easy to build a “maximum demand register” with a one entry deep sorted profile capturing and sorted by a “Demand register” *last_average_value* attribute. It is just as simple to define a profile retaining the three largest values of some period.

The size of profile data is determined by three parameters:

- the number of entries filled (*entries_in_use*). This will be zero after clearing the profile;
- the maximum number of entries to retain (*profile_entries*). If all entries are filled and a capture () request occurs, the least important entry (according to the requested sorting method) will get lost. This maximum number of entries may be specified. Upon changing it, the buffer will be adjusted;
- the physical limit for the buffer. This limit typically depends on the objects to capture. The object will reject an attempt of setting the maximum number of entries that is larger than physically possible.

Profile generic		0...n	class_id = 7, version = 1			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. buffer	(dyn.)	compact-array or array				x + 0x08
3. capture_objects	(static)	array				x + 0x10
4. capture_period	(static)	double-long-unsigned				x + 0x18
5. sort_method	(static)	enum				x + 0x20
6. sort_object	(static)	capture_object_definition				x + 0x28
7. entries_in_use	(dyn.)	double-long-unsigned	0		0	x + 0x30
8. profile_entries	(static)	double-long-unsigned	1		1	x + 0x38
Specific methods		m/o				
1. reset (data)		o				x + 0x58
2. capture (data)		o				x + 0x60
3. reserved from previous versions		o				
4. reserved from previous versions		o				

Attribute description

logical_name Identifies the “Profile generic” object instance. For examples, see 6.2.17, 6.2.19, 6.2.20, 6.2.33, 6.2.36, 6.2.38, 6.2.49, 6.2.51 6.2.52, 6.3.2 etc.

buffer Contains a sequence of entries. Each entry contains values of the captured objects.

compact-array or array entry

entry ::= structure

```
{
  CHOICE
  {
    -- simple data types
    null-data           [0],
    boolean             [3],
    bit-string          [4],
    double-long         [5],
    double-long-unsigned [6],
    octet-string        [9],
    visible-string      [10],
    utf8-string         [12],
    bcd                 [13],
    integer             [15],
    long                [16],
    unsigned            [17],
    long-unsigned       [18],
    long64              [20],
    long64-unsigned     [21],
    enum                [22],
    float32             [23],
    float64             [24],
    date-time           [25],
    date                [26],
    time                [27],
    -- complex data types
    array               [1],
    structure           [2],
    compact-array       [19]
  }
}
```

The number and the order of the elements of the structure holding the entries is the same as in the definition of the *capture_objects*. The *buffer* is filled by auto captures or by subsequent calls of the method *capture*. The sequence of the entries within the array is ordered according to the *sort_method* specified.

Default: The *buffer* is empty after reset.

REMARK 1 Reading the entire *buffer* delivers only those entries, which are “in use”.

REMARK 2 The value of a captured object may be replaced by “null-data” if it can be unambiguously recovered from the previous value (for example for time: if it can be calculated from the previous value and *capture_period*; or for a value: if it is equal to the previous value).

selective access (see 4.4) to the attribute buffer may be available (optional). The selective access parameters are as defined below.

capture_objects	Specifies the list of capture objects that are assigned to the object instance. Upon a call of the <i>capture</i> (data) method or automatically in defined intervals, the selected attributes are copied into the <i>buffer</i> of the profile. array capture_object_definition capture_object_definition ::= structure <pre> { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer, data_index: long-unsigned } </pre> - where attribute_index is a pointer to the attribute within the object identified by its class_id and logical_name. Attribute_index 1 refers to the 1st attribute (i.e. the logical_name), attribute_index 2 to the 2nd, etc.); attribute_index 0 refers to all public attributes; - where data_index is a pointer selecting a specific element of the attribute. The first element in the attribute structure is identified by data_index 1. If the attribute is not a structure, then the data_index has no meaning. If the capture object is the buffer of a profile, then the data_index identifies the captured object of the buffer (i.e. the column) of the inner profile; data_index 0: references the whole attribute.
capture_period	>= 1: Automatic capturing assumed. Specifies the capturing period in seconds. 0: No automatic capturing; capturing is triggered externally or capture events occur asynchronously.
sort_method	If the profile is unsorted, it works as a “first in first out” buffer (it is hence sorted by capturing, and not necessarily by the time maintained in the “Clock” object). If the <i>buffer</i> is full, the next call to <i>capture</i> () will push out the first (oldest) entry of the <i>buffer</i> to make space for the new entry. If the profile is sorted, a call to <i>capture</i> () will store the new entry at the appropriate position in the buffer, moving all following entries and probably losing the least interesting entry. If the new entry would enter the <i>buffer</i> after the last entry and if the <i>buffer</i> is already full, the new entry will not be retained at all. enum: (1) fifo (first in first out), (2) lifo (last in first out), (3) largest, (4) smallest, (5) nearest_to_zero, (6) farthest_from_zero Def. fifo
sort_object	If the profile is sorted, this attribute specifies the register or clock that the ordering is based upon. capture_object_definition See above. Def. no object to sort by (only possible with sort_method fifo or lifo) NOTE 1 If the sort_method is FIFO or LIFO, then all elements of the capture_object_definition specifying the sort object can be zero.

entries_in_use	Counts the number of entries stored in the buffer. After a call of the <i>reset</i> () method, the buffer does not contain any entries, and this value is zero. Upon each subsequent call of the <i>capture</i> () method, this value will be incremented up to the maximum number of entries that will get stored (see <i>profile_entries</i>).
	double-long-unsigned 0...profile_entries
	Def. 0
profile_entries	Specifies how many entries shall be retained in the buffer.
	double-long-unsigned 1...(limited by physical size)
	Def. 1

Parameters for selective access to the buffer attribute

Access selector	Access parameter	Comment
1	range_descriptor	Only buffer elements corresponding to the range_descriptor shall be returned in the response.
2	entry_descriptor	Only buffer elements corresponding to the entry_descriptor shall be returned in the response.

range_descriptor::= structure

```

{
    restricting_    capture_object_definition    Defines the capture_object restricting the
    object:        range of entries to be retrieved. Only
                  simple data types are allowed.

    from_value:    Oldest or smallest entry to retrieve

                  CHOICE
                  {
                      -- simple data types
                      double-long                [5],
                      double-long-unsigned       [6],
                      octet-string               [9],
                      visible-string              [10],
                      utf8-string                 [12],
                      integer                     [15],
                      long                        [16],
                      unsigned                    [17],
                      long-unsigned               [18],
                      long64                     [20],
                      long64-unsigned             [21],
                      float32                     [23],
                      float64                     [24],
                      date-time                   [25],
                      date                        [26],
                      time                        [27]
                  }

    to_value:      CHOICE                        Newest or largest entry to retrieve
                  {see above}

    selected_      array
    values:         capture_object_definiti
                   on

    }

```

List of columns to retrieve. If the array is empty (has no entries), all captured data are returned. Otherwise, only the columns specified in the array are returned. The type *capture_object_definition* is specified above (*capture_objects*).

entry_descriptor::= structure

```
{
    from_entry:          double-long-unsigned  first entry to retrieve,
    to_entry:            double-long-unsigned  last entry to retrieve
                                to_entry == 0: highest possible entry,
    from_selected_value: long-unsigned        index of first value to retrieve,
    to_selected_value:   long-unsigned        index of last value to retrieve
                                to_selected_value == 0: highest possible selected_value
}
```

NOTE 2 from_entry and to_entry identify the lines, from_selected_value to_selected_value identify the columns of the buffer to be retrieved.

NOTE 3 Numbering of entries and selected values starts from 1.

Method description

reset (data) Clears the *buffer*. It has no valid entries afterwards; *entries_in_use* is zero after this call. This call does not trigger any additional operations on the capture objects. Specifically, it does not reset any attributes captured.

data::= integer (0)

capture (data) Copies the values of the objects to capture into the *buffer* by reading each capture object. Depending on the *sort_method* and the actual state of the *buffer* this produces a new entry or a replacement for the less significant entry. As long as not all entries are already used, the *entries_in_use* attribute will be incremented.

This call does not trigger any additional operations within the capture objects such as *capture ()* or *reset ()*.

Note, that if more than one attribute of an object need to be captured, they have to be defined one by one on the list of *capture_objects*. If the *attribute_index* = 0, all attributes are captured.

data::= integer (0)

Behaviour of the object after modification of certain attributes

Any modification of one of the *capture_objects* describing the static structure of the *buffer* will automatically call a *reset ()* and this call will propagate to all other profiles capturing this profile.

If writing to *profile_entries* is attempted with a value too large for the buffer, it will be rejected.

Restrictions

When defining the *capture_objects*, circular reference to the profile shall be avoided.

Profile used to define a subset of preferred readout values

By setting *profile_entries* to 1, a “Profile generic” object can be used to define a set of preferred readout values. See also 6.2.17. Setting *capture_period* to 1 ensures that the values are updated every second.

5.2.7 Utility tables (class_id = 26, version = 0)

This IC allows encapsulating ANSI C12.19 table data. Each “table” is represented by an instance of this IC, identified by its logical name.

Utility tables	0...n	class_id = 26, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. table_ID (static)	long-unsigned				x + 0x08
3. length	double-long-unsigned				x + 0x10
4. buffer	octet-string				x + 0x18
Specific methods	m/o				

Attribute description

logical_name	Identifies the “Utility tables” object instance. See 6.2.32.
table_ID	Table number. This table number is as specified in the ANSI standard and may be either a standard table or a manufacturer’s table.
length	Number of octets in table buffer.
buffer	Contents of the table. <i>Selective access</i> (see 4.4) to the attribute <i>buffer</i> may be available (optional). The selective access parameters are defined below.

Parameters for selective access to the buffer attribute

Access selector	Parameter	Comment
1	offset_access	Access to table by offset and count using offset_selector for parameter data.
2	index_access	Access to table by element id and number of elements using index_selector for parameter data.

offset_selector ::= structure

{

Offset: double-long-unsigned

Offset in octets to the start of access area, relative to the start of the table.

Count: long-unsigned

Number of octets requested or transferred

}

index_selector ::= structure

{

Index: array long-unsigned

Sequence of indices to identify elements within the table’s hierarchy.

Count: long- Number of elements requested or transferred. Values of count
 unsigned greater than 1 return up to that many elements. A value of zero,
 when given in the context of a request, refers to the entire sub-tree
} of the hierarchy starting at the selection point.

5.2.8 Register table (class_id = 61, version = 0)

This IC allows to group homogenous entries, identical attributes of multiple objects, which are all instances of the same IC, and in their *logical_name* (OBIS code) the value in value groups A to D and F is identical. The possible values in value group E are defined in IEC 62056-6-1:2015 in a tabular form: the table header defines the common part of the OBIS code and each table cell defines one possible value of value group E. A “Register table” object may capture attributes of some or all of those objects.

NOTE 1 Some examples are the “Extended phase angle measurement” table, see IEC 62056-6-1:2015, Table 16 or the “UNIPED voltage dip quantities” table, see IEC 62056-6-1:2015, Table 18.

NOTE 2 If more complex functionality is needed, the “Profile generic” IC can be used.

Register table		0...n	class_id = 61, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	table_cell_values (dyn.)	compact-array or array				x + 0x08
3.	table_cell_definition (static)	structure				x + 0x10
4.	scaler_unit (static)	scaler_unit_type				x + 0x18
Specific methods		m/o				
1.	reset (data)	o				x + 0x28
2.	capture (data)	o				x + 0x30

Attribute description

logical_name	Identifies the “Register table” object instance. When the format of the <i>logical_name</i> is A.B.C.D.255.F; the values A to D and F define the common part of the logical name of the objects, the attributes of which are captured. Only one attribute of the objects concerned can be captured (for example the <i>value</i> attribute). When the format of the <i>logical_name</i> is A.B.98.10.x.255, several instances of the “Register table” IC can be used to capture different attributes of the objects concerned. The value group E numbers the instances. See IEC 62056-6-1:2015, 6.4.
table_cell_values	Holds the value of the attributes captured, as they would be returned to a GET or Read .request to the individual attributes. compact array or array table_cell_entry

table_cell_
values
cont'd

table_cell_entry::= CHOICE
{
 -- simple data types
 null-data [0],
 bit-string [4],
 double-long [5],
 double-long-unsigned [6],
 octet-string [9],
 visible-string [10],
 utf8-string [12],
 bcd [13],
 integer [15],
 long [16],
 unsigned [17],
 long-unsigned [18],
 long64 [20],
 long64-unsigned [21],
 float32 [23],
 float64 [24],
 -- complex data types
 structure [2]
}

If the captured attribute is attribute_0, redundant values may be replaced by "null-data", if their value can be unambiguously recovered (for example *scaler_unit*).

table_cell_
definition

Specifies the list of attributes captured in the register table.

structure
{
 class_id: long-unsigned,
 logical_name: octet-string,
 group_E_values: array unsigned,
 attribute_index: integer
}

where:

- class_id defines the common class_id of the objects the attributes of which are captured;
- logical_name contains the common logical name of the objects, with E = 255 (wildcard);
- group_E_values contain the list of cell identifiers, of type *unsigned*, as defined in the respective table of IEC 62056-6-1:2015;
- attribute_index is a pointer to the attribute within the object. attribute_index 0 refers to all public attributes.

If the *logical_name* of the "Register table" object is in the format A.B.C.D.255.F and the defined attribute of all objects identified in the respective table in IEC 62056-6-1:2015 are captured, then attribute 3 may not be accessible. In this case:

- the class_id shall be 1 "Data", 3 "Register" or 4 "Extended register";
 - the *logical_name* of the objects to be captured is defined by the *logical_name* of the "Register table" object and the respective table in IEC 62056-6-1:2015;
 - the attribute index shall be 2 (value).
-

scaler_unit See the description of IC “Register”.

In the case when “value” attributes of “Register” or “Extended register” objects are captured, the *scaler_unit* shall be common for all objects and this attribute shall hold a copy.

If other attributes or ICs are captured, the *scaler_unit* attribute has no meaning and shall be inaccessible.

Method description

reset (data) Clears the *table_cell_values*. It has no effect on the attributes captured.

data::= integer (0)

capture (data) Copies the values of the attributes into the *table_cell_values*. If the attribute_index = 0, all attributes are captured.

Behaviour of the object after modification of the table_cell_definition attribute

Any modification to this attribute will automatically call the *reset (data)* method and this will propagate to all profiles capturing this object.

If writing to *table_cell_definition* is attempted with a value too large the buffer holding the *table_cell_values* attribute, it will be rejected.

5.2.9 Status mapping (class_id = 63, version = 0)

This IC allows modelling the mapping of bits in a status word to entries in a reference table.

Status mapping	0..n	class_id = 63, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. status_word (dyn.)	CHOICE				x + 0x08
3. mapping_table (static)	structure				x + 0x10
Specific methods	m/o				

Attribute description

logical_name Identifies the “Status mapping” object instances. See 6.2.36, 6.2.38, 6.2.44 and 6.3.7.

status_word Contains the current value of the status word.

CHOICE

```
{
    bit-string [4],
    double-long-unsigned [6],
    octet-string [9],
    visible-string, [10],
    utf8-string [12],
    unsigned [17],
    long-unsigned [18],
    long64-unsigned [21]
}
```

The size of the *status_word* is n*8 bits, the maximum size is 65 536 bits.

NOTE Manufacturers may choose any of the types listed above. However, the status word is always interpreted as a bit-string.

mapping_
table Contains the mapping of the *status_word* to the positions in the reference table.

structure

```

{
    ref_table_id:          unsigned,
    ref_table_mapping:     CHOICE
    {
        long-unsigned      [18],
        array long-unsigned [1]
    }
}

```

Where:

- *ref_table_id* identifies the reference status table;
 - if the “long-unsigned” choice is taken, the value points to an entry in the reference table. This entry is mapped to the leading bit of the status word. The next entry is mapped to the next bit and so on. The last entry that is mapped to the trailing bit is determined by the length of the status word;
 - if the “array” choice is taken, the elements of the array point to entries in the reference status table. The order of the elements in the array corresponds to the position in the status word. The first element in the array maps the table entry referenced to the leading bit and the last element to the trailing bit.
-

5.3 Interface classes for access control and management

5.3.1 Overview

Interface classes in this category model the logical structure of the DLMS/COSEM server, allow configuring and managing access to its resources, updating the firmware and managing security:

- the “Association SN” class – see 5.3.3 – and the “Association LN” class – see 5.3.4 – model AAs. Their instances, the Association objects provide the list of objects accessible in each AA, manage and control access rights to their attributes and methods. They also manage the authentication of the communicating partners;
- the “SAP Assignment” class – see 5.3.5 – models the logical structure of the server;
- the “Image transfer” class – see 5.3.6 – models the firmware update process;
- the “Security setup” class – see 5.3.7 – models the elements of the security context. “Security setup” objects are referenced from the “Association” objects and allow configuring security suites and security policies and managing security material;
- the “Push setup” class – see 5.3.8 – models the push operation of the server.

5.3.2 Client user identification

This new feature enables the server to distinguish between different users from the client side and to log their activities accessing the meter.

Each AA established between a client and a server can be used by several users on the client side. The properties of the AA are configured in the server, using the “Association” and the “Security setup” objects. All users of an AA on the client side use these same properties.

NOTE 1 The security keys are known by the client and the server but they need not be known by the users of the client.

The list of users – identified by their *user_id* and *user_name* – is known both by the client and the server. In the server it is held by the *user_list* attribute of the “Association” objects.

NOTE 2 The way a client authenticates a user to log into a client system is outside the scope of this specification.

During AA establishment, the *user_id* – belonging to the *user_name* – is carried by the calling-AE-invocation-id field of the AARQ APDU. If the *user_id* provided is on the *user_list*, the AA can be established – provided that all other conditions are met – and the *current_user* attribute is updated. The value of this attribute can be logged.

If the server does not “know” the user, the AA shall not be established. The server may silently discard the request to establish the AA or it may send back an appropriate error message.

The user identification process is optional: if the *user_list* is empty – i.e. it is an array of 0 elements – the function is disabled.

5.3.3 Association SN (class_id = 12, version = 3)

NOTE 1 This new version 3 of the “Association SN” IC supports the client user identification process, see 5.3.2.

COSEM logical devices able to establish AAs within a COSEM context using SN referencing, model the AAs using instances of the “Association SN” IC. A COSEM logical device may have one instance of this IC for each AA the device is able to support.

The **short_name** of the “Association SN” object itself is fixed within the COSEM context. See 4.3.

Association SN		0...n	class_id = 12, version = 3			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	object_list (static)	objlist_type				x + 0x08
3.	access_rights_list (static)	access_rights_type				x + 0x10
4.	security_setup_reference (static)	octet-string				x + 0x18
5.	user_list (static)	array				x + 0x20
6.	current_user	structure				x + 0x28
Specific methods		m/o				
1.	reserved from previous versions	o				
2.	reserved from previous versions	o				
3.	read_by_logicalname (data)	o				x + 0x30
4.	reserved from previous versions	o				
5.	change_secret (data)	o				x + 0x40
6.	reserved from previous versions	o				
7.	reserved from previous versions					
8.	reply_to_HLS_authentication (data)	o				x + 0x58
9.	add_user (data)	o				x + 0x60
10.	remove_user (data)	o				x + 0x68

Attribute description

logical_name Identifies the “Association SN” object instance. See 6.2.27.

object_list Contains the list of all objects with their *base_name* (*short_name*), *class_id*, *version* and *logical_name*. The *base_name* is the DLMS *objectName* of the first attribute (*logical_name*).

objlist_type::= array *objlist_element*

objlist_element::= structure

```
{
    base_name:      long,
    class_id:       long-unsigned,
    version:        unsigned,
    logical_name:   octet-string
}
```

selective access (see 4.4) to the attribute *object_list* may be available. The access selector values and their parameters are as defined below.

access_rights_list Contains the access rights to attributes and methods.

The link between the *object_list* and the *access_rights_list* is the *base_name*, present in both the *objlist_element* structure and the *access_right_element* structure. Therefore, the *base_names* on the two lists shall be the same. The number – and preferably, the order – of the elements in the array of *objlist_element* and the array of *access_right_element* shall also be the same.

access_rights_type::= array *access_rights_element*

access_rights_element::= structure

```
{
    base_name:      long,
    attribute_access: attribute_access_descriptor,
    method_access:  method_access_descriptor
}
```

attribute_access_descriptor::= array *attribute_access_item*

attribute_access_item::= structure

```
{
    attribute_id:    integer,
    access_mode:    enum:
        (0) no_access,
        (1) read_only,
        (2) write_only,
        (3) read_and_write,
        (4) authenticated_read_only,
        (5) authenticated_write_only,
        (6) authenticated_read_and_write
}
```

access_selectors: CHOICE

```
{
    null-data            [0],
    array integer        [1]
}
```

}

access_rights_list cont'd	method_access_descriptor ::= array method_access_item method_access_item ::= structure <pre> { method_id: integer, access_mode: enum: (0) no_access, (1) access, (2) authenticated_access } </pre> <i>selective access</i> (see 4.4) to the attribute <i>access_rights_list</i> may be available (optional). The access selector values and their parameters are as defined below.
security_setup_reference	References the “Security setup” object by its <i>logical_name</i>. The referenced object manages security for a given “Association SN” object instance.
user_list	Contains the list of users allowed to use the AA managed by the given instance of the “Association SN” IC. array user_list_entry user_list_entry ::= structure <pre> { user_id: unsigned, user_name: visible-string } </pre> Where: - user_id is the identifier of the user (this value is carried by the calling-AE-invocation-id field of the AARQ); - user_name is the name of the user. If the user_list attribute is empty – i.e. it is an array of 0 elements – any user can use the AA, i.e. the calling-AE-invocation-id field of the AARQ is ignored. If the user_list attribute is not empty then only the users in the list can establish the AA, i.e. the calling-AE-invocation-id field of the AARQ shall be present and its value shall match one of user_ids in the user_list or else, the AA is not established.
current_user	Holds the identifier of the current user. current_user ::= user_list_entry (see above) If the user_list is empty, then current_user shall be a structure {user_id: unsigned 0, user_name: visible string of 0 elements}

Parameters for selective access to the *object_list* and *access_rights_list* attribute

Access selector value	Parameter	Available with attribute	Comment
1	class_id: long-unsigned	2	Delivers the subset of the object_list for a specific class_id. For the response: data::= objlist_type
2	structure { class_id: long-unsigned, logical_name: octet-string }	2	Delivers the entry of the object_list for a specific class_id and <i>logical_name</i> . For the response: data::= objlist_element
3	base_name: long	2, 3	In the case of attribute 2, delivers the entry of the object_list for a specific base_name. For the response: data::= objlist_element In the case of attribute 3, delivers the entry of the access_rights_list for a specific base_name. For the response: data::=access_rights_element

Method description**read_by_logicalname
(data)**

Reads attributes for selected objects. The objects are specified by their class_id and their *logical_name*. With this method, the parameterized access feature can also be used.

data::= array attribute_identification

attribute_identification::= structure
{
 class_id: long-unsigned,
 logical_name: octet-string,
 attribute_index: integer
}

where attribute_index is a pointer (i.e. offset) to the attribute within the object.

attribute_index 0 delivers all attributes; attribute_index 1 delivers the first attribute (i.e. *logical_name*), etc.).

For the response: data is according to the type of the attribute.

NOTE 2 If at least one attribute has no read access right under the current association, then a *read_by_logicalname* () to attribute index 0 reveals the error message "scope-of-access-violated", see IEC 62056-5-3:2016, Clause 8.

**change_secret
(data)**

Changes the LLS or HLS secret (for example password).

data::= octet-string new secret

NOTE 3 The structure of the "new secret" depends on the security mechanism implemented. The "new secret" may contain additional check bits and it may be encrypted.

NOTE 4 In the case of HLS with GMAC, the (HLS_) secret is held by the "Security setup" object referenced in attribute.

reply_to_HLS_authentication (data)	The remote invocation of this method delivers to the server the result of the secret processing by the client of the server's challenge to the client, f(StoC), as the data service parameter of the Read.request primitive invoked with parameterised access. data::= octet-string client's response to the challenge If the authentication is accepted, then the response (Read.confirm primitive) contains Result == OK and the result of the secret processing by the server of the client's challenge to the server, f(CtoS) in the data service parameter of the Read.response service. data::= octet-string server's response to the challenge If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back.
add_user (data)	Adds a user to the user_list. data::= user_list_entry (see above)
remove_user (data)	Removes a user from the user_list. data::= user_list_entry (see above)

5.3.4 Association LN (class_id = 15, version = 2)

NOTE 1 This new version 2 of the "Association LN" IC supports the client user identification process, see 5.3.2.

COSEM logical devices able to establish AAs within a COSEM context using LN referencing, model the AAs through instances of the "Association LN" IC. A COSEM logical device has one instance of this IC for each AA the device is able to support.

Association LN		0...MaxNbOfAss.	class_id = 15, version = 2			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	object_list (static)	object_list_type				x + 0x08
3.	associated_partners_id	associated_partners				x + 0x10
4.	application_context_name	context_name_type				x + 0x18
5.	xDLMS_context_info	xDLMS_context_type				x + 0x20
6.	authentication_mechanism_name	mechanism_name_type				x + 0x28
7.	secret	octet-string				x + 0x30
8.	association_status	enum				x + 0x38
9.	security_setup_reference (static)	octet-string				x + 0x40
10.	user_list (static)	array				x + 0x48
11.	current_user	structure				x + 0x50
Specific methods		m/o				
1.	reply_to_HLS_authentication (data)	o				x + 0x60
2.	change_HLS_secret (data)	o				x + 0x68
3.	add_object (data)	o				x + 0x70
4.	remove_object (data)	o				x + 0x78
5.	add_user (data)	o				x + 0x80
6.	remove_user (data)	o				x + 0x88

Attribute description

logical_name	Identifies the “Association LN” object instance. See 6.2.27.
object_list	Contains the list of visible COSEM objects with their class_id, version, <i>logical_name</i> and the access rights to their attributes and methods within the given AA. object_list_type ::= array object_list_element object_list_element ::= structure <pre> { class_id: long-unsigned, version: unsigned, logical_name: octet-string, access_rights: access_right } access_right ::= structure { attribute_access: attribute_access_descriptor, method_access: method_access_descriptor } attribute_access_descriptor ::= array attribute_access_item attribute_access_item ::= structure { attribute_id: integer, access_mode: enum: (0) no_access, (1) read_only, (2) write_only, (3) read_and_write, (4) authenticated_read_only, (5) authenticated_write_only, (6) authenticated_read_and_write access_selectors: CHOICE { null-data [0], array integer [1] } } method_access_descriptor ::= array method_access_item method_access_item ::= structure { method_id: integer, access_mode: enum: (0) no_access, (1) access, (2) authenticated_access } </pre>

object_list (continued)	Where: - the <code>attribute_access_descriptor</code> and the <code>method_access_descriptor</code> elements always contain all implemented attributes or methods; - <code>access_selectors</code> contain a list of the supported selector values. <i>selective access</i> (see 4.4) to the attribute <i>object_list</i> may be available (optional). The selective access parameters are as defined below.
associated_partners_id	Contains the identifiers of the COSEM client and server (logical device) APs within the physical devices hosting these APs, which belong to the AA modelled by the “Association LN” object. <code>associated_partners_type</code>::= structure <pre> { client_SAP: integer, server_SAP: long-unsigned } </pre> The range for the <code>client_SAP</code> is 0...0x7F. The range for the <code>server_SAP</code> is 0x0000...0x3FFF. The SAPs shall be in the range allowed by the data type and the media.
application_context_name	In the COSEM environment, it is intended that an application context pre-exists and is referenced by its name during the establishment of an AA. This attribute contains the name of the application context for that AA. <code>context_name_type</code>::= CHOICE <pre> { context_name_structure [2], octet-string [9] } </pre> The application context name is specified as OBJECT IDENTIFIER in IEC 62056-5-3:2016, 7.2.2.2. When the <code>context_name_type</code> is encoded as a structure, it includes the arc labels of the OBJECT IDENTIFIER. <code>context_name_structure</code>::= structure <pre> { joint_iso_ctt_element: unsigned, country_element: unsigned, country_name_element: long-unsigned, identified_organization_element: unsigned, DLMS_UA_element: unsigned, application_context_element: unsigned, context_id_element: unsigned } </pre> Example 1: In the case of <code>context_id(1)</code> the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 01 11 01 (all values are hexadecimal). When the <code>context_name_type</code> is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See IEC 62056-5-3:2016, Clause C.4. Example 2: In the case of <code>context_id(1)</code> the A-XDR encoding is: 09 07 60 85 74 05 08 01 01 (all values are hexadecimal).

**xDLMS_
context_info**

Contains all the necessary information on the xDLMS context for the given AA.

xDLMS_context_type ::= structure

```
{
    conformance:                bit-string,
    max_receive_pdu_size:        long-unsigned,
    max_send_pdu_size:           long-unsigned,
    dlms_version_number:         unsigned,
    quality_of_service:          integer,
    cyphering_info:              octet-string
}
```

Where:

- the conformance element contains the xDLMS conformance block supported by the server. The length of the bit-string is 24 bits.
- the max_receive_pdu_size element contains the maximum length for an xDLMS APDU, expressed in bytes that the client may send. This is the same as the server-max-receive-pdu-size parameter of the xDLMS initiateResponse APDU;
- the max_send_pdu_size element, in an active AA contains the maximum length for an xDLMS APDU, expressed in bytes that the server may send. This is the same as the client-max-receive-pdu-size parameter of the xDLMS initiateRequest APDU;
- the dlms_version_number element contains the DLMS version number supported by the server;
- the quality_of_service element is not used;
- the cyphering_info element – in an active AA – contains the dedicated key parameter of the xDLMS initiateRequest APDU. See IEC 62056-5-3:2016, Clause 8.

**authentication_
mechanism_
name**

Contains the name of the authentication mechanism for the AA.

mechanism_name_type ::= CHOICE

```
{
    mechanism_name_structure    [2],
    octet-string                 [9]
}
```

The authentication mechanism name is specified as an OBJECT IDENTIFIER in IEC 62056-5-3:2016, 7.2.2.3.

When the mechanism_name_type is encoded as a structure, it includes the arc labels of the OBJECT IDENTIFIER.

mechanism_name_structure ::= structure

```
{
    joint_iso_ctt_element:        unsigned,
    country_element:              unsigned,
    country_name_element:         long-unsigned,
    identified_organization_element: unsigned,
    DLMS_UA_element:             unsigned,
    authentication_mechanism_name_element: unsigned,
    mechanism_id_element:         unsigned
}
```

authentication_mechanism_name cont'd	Example 3: In the case of mechanism_id(1) the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 02 11 01 (all values are hexadecimal); When the mechanism_name_type is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See IEC 62056-5-3:2016, Clause C.4. EXAMPLE 4: In the case of mechanism_id(1) the A-XDR encoding is: 09 07 60 85 74 05 08 02 01 (all values are hexadecimal). No mechanism_name is required when no authentication is used.
secret	Contains the secret for the LLS or HLS authentication process. NOTE 2 In the case of HLS with GMAC, the (HLS_)secret is held by the "Security setup" object referenced in attribute 9, security_setup_reference.
association_status	Indicates the current status of the association, which is modelled by the object. enum: (0) non-associated, (1) association-pending, (2) associated
security_setup_reference	References a "Security setup" object by its logical name. The referenced object manages security for a given "Association LN" object instance.
user_list	Contains the list of users allowed to use the AA managed by the given instance of the "Association LN" IC. array user_list_entry user_list_entry::=structure { user_id: unsigned, user_name: visible-string } Where: – user_id is the identifier of the user (this value is carried by the calling-AE-invocation-id field of the AARQ); – user_name is the name of the user. If the user_list attribute is empty – i.e. it is an array of 0 elements – any user can use the AA, i.e. the calling-AE-invocation-id field of the AARQ is ignored. If the user_list attribute is not empty then only the users in the list can establish the AA, i.e. the calling-AE-invocation-id field of the AARQ shall be present and its value shall match one of user_ids in the user_list or else the AA is not established.
current_user	Holds the identifier of the current user. current_user::= user_list_entry (see above) If the user_list is empty, then current_user shall be a structure {user_id: unsigned 0, user_name: visible string of 0 elements}

Parameters for selective access to the **object_list** attribute

- If no selective access is requested, (no Access_Selection_Parameters parameter is present in the GET.request (.indication) service primitive for the object_list attribute) the corresponding .response (.confirmation) service shall contain all object_list_elements of the object_list attribute.
- When selective access is requested to the object_list attribute (the Access_Selection_Parameter parameter is present), the response shall contain a 'filtered' list of object_list_elements, as follows:

Access selector	Access parameter	Comment
1	NULL	All information excluding the <i>access_rights</i> shall be included in the response.
2	<i>class_list</i>	Access by <i>class_id</i> . In this case, only those <i>object_list_elements</i> of the <i>object_list</i> shall be included in the response, which have a <i>class_id</i> equal to one of the <i>class_id</i> -s of the <i>class_list</i> . No <i>access_right</i> information is included. <i>class_list</i> ::= array <i>class_id</i> <i>class_id</i> : long-unsigned
3	<i>object_id_list</i>	Access by object. The full information record of object instances on the <i>object_id_list</i> shall be returned. <i>object_id_list</i> ::= array <i>object_id</i> <i>object_id</i> ::= structure { <i>class_id</i> : long-unsigned, <i>logical_name</i> : octet-string }
4	<i>object_id</i>	The full information record of the required COSEM object instance shall be returned. <i>object_id</i> ::= structure See above.

Method description

reply_to_HLS_authentication (data)

The remote invocation of this method delivers to the server the result of the secret processing by the client of the server's challenge to the client, *f*(StoC), as the *data* service parameter of the ACTION.request primitive invoked:

data::= octet-string client's response to the challenge

If the authentication is accepted, then the response (ACTION.confirm primitive) contains Result == OK and the result of the secret processing by the server of the client's challenge to the server, *f*(CtoS) in the *data* service parameter of the response service.

data::= octet-string server's response to the challenge

If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back.

change_HLS_secret (data)

Changes the HLS secret (for example encryption key).

data::= octet-string new HLS secret

The structure of the "new secret" depends on the security mechanism implemented. The "new secret" may contain additional check bits and it may be encrypted.

add_object (data)

Adds the referenced object to the *object_list*.

data::= object_list_element (see above)

remove_object (data)

Removes the referenced object from the *object_list*.

data::= object_list_element (see above)

add_user (data)

Adds a user to the *user_list*.

data::= user_list_entry (see above)

remove_user (data)	Removes a user from the <i>user_list</i> . data::= user_list_entry (see above)

5.3.5 SAP assignment (class_id = 17, version = 0)

This IC allows modelling the logical structure of physical devices, by providing information on the assignment of the logical devices to their SAP-s. See IEC 62056-5-3:2016, Annex A.

SAP assignment		0...1	class_id = 17, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	SAP_assignment_list (static)	asslist_type			0	x + 0x08
Specific methods		m/o				
1.	connect_logical_device (data)	o				x + 0x20

Attribute description

logical_name Identifies the “SAP assignment” objects instance. See 6.2.28.

SAP_assignment_list Contains the list of all logical devices and their SAP addresses within the physical device.

```

asslist_type::= array          asslist_element
asslist_element::= structure
{
    SAP: long-unsigned,
    logical_device_name: CHOICE
    {
        octet-string [9],
        visible-string [10],
        utf8-string [12]
    }
}

```

REMARK: The actual addressing is performed by the supporting communication layers.

Method description

connect_logical_device (data) Connects a logical device to a SAP. Connecting to SAP 0 will disconnect the device. More than one device cannot be connected to one SAP (exception SAP 0).

data::= asslist_element

5.3.6 Image transfer

5.3.6.1 General

Instances of the Image transfer IC model the process of transferring binary files, called Images to COSEM servers.

NOTE This specification includes some improvements and precisions to the text. The main changes are:

- The description of the Image transfer process is given as an example only. The text and the flow chart are updated;
- Data exchange between the client and a conceptual Image server is out of Scope;
- A clear distinction is made between the Image transferred and the Images to activate;
- Steps 1 and 6 are optional now;
- Size of the *image_transferred_blocks_status* bit-string may be dynamic;
- It is specified now that setting the value of the *image_transfer_enabled* attribute to FALSE disables the image transfer process;

- It is specified now that re-initiating the image transfer process resets the whole process;
- Some precisions have been added to the effect of invoking the methods;
- See also the highlighted parts of the text.

5.3.6.2 The steps of the image transfer process

The Image transfer usually takes place in several steps:

- Step 1: (Optional): Get ImageBlockSize;
- Step 2: Client initiates Image transfer;
- Step 3: Client transfers ImageBlocks;
- Step 4: Client checks completeness of the Image;
- Step 5: Server verifies the Image (Initiated by the client or on its own);
- Step 6 (Optional): Client checks the information on the images to activate;
- Step 7: Server activates the Image(s) (Initiated by the client or on its own).

For an example with more detailed explanations, see 5.3.6.4.

5.3.6.3 Image transfer (class_id = 18, version = 0)

Image transfer		0...n	class_id = 18, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	image_block_size (static)	double-long-unsigned				x + 0x08
3.	image_transferred_blocks_status (dyn.)	bit-string				x + 0x10
4.	image_first_not_transferred_block_number (dyn.)	double-long-unsigned				x + 0x18
5.	image_transfer_enabled (static)	boolean				x + 0x20
6.	image_transfer_status (dyn.)	enum				x + 0x28
7.	image_to_activate_info (dyn.)	array				x + 0x30
Specific methods		m/o				
1.	image_transfer_initiate	m				x + 0x40
2.	image_block_transfer	m				x + 0x48
3.	image_verify	m				x + 0x50
4.	image_activate	m				x + 0x58

Attribute description

logical_name	Identifies the “Image transfer” object instance. See 6.2.31.
image_block_size	Holds the ImageBlockSize, expressed in octets, which can be handled by the server. ImageBlockSize shall not exceed the ServerMaxReceivePduSize negotiated.

NOTE 1 *image_block_size* is a property of the server.

image_transferred_blocks_status	Provides information about the transfer status of each ImageBlock. Each bit in the bit-string provides information about one individual ImageBlock: 0 = Not transferred, 1 = Transferred NOTE 2 The size of the attribute may be dynamic, i.e. it may grow upon reception of new ImageBlocks.						
image_first_not_transferred_block_number	Provides the ImageBlockNumber of the first ImageBlock not transferred. Once the Image is complete, the value returned should be equal to or above the number of blocks calculated from the Image size and the ImageBlockSize.						
image_transfer_enabled	Controls the enabling of the Image transfer process. boolean: FALSE = Disabled, TRUE = Enabled The image transfer methods can be invoked successfully only if the value of this attribute is TRUE. Setting the value of this attribute to FALSE disables all methods (invoking them fails).						
image_transfer_status	Holds the status of the Image transfer process. enum: (0) Image transfer not initiated, (1) Image transfer initiated, (2) Image verification initiated, (3) Image verification successful, (4) Image verification failed, (5) Image activation initiated, (6) Image activation successful, (7) Image activation failed						
image_to_activate_info	Provides information on the Image(s) ready for activation. It is generated as the result of the Image verification process. The client may check this information before activating the Images. array image_to_activate_info_element image_to_activate_info_element::= structure <table> <tr> <td>image_to_activate_size:</td><td>double-long-unsigned,</td></tr> <tr> <td>image_to_activate_identification:</td><td>octet-string,</td></tr> <tr> <td>image_to_activate_signature:</td><td>octet-string</td></tr> </table> Where: – image_to_activate_size is the size of the Image to be activated, expressed in octets; – image_to_activate_identification is the identification of the Image to be activated, and may contain information like manufacturer, device type, version information, etc.; – image_to_activate_signature is the signature of the Image to be activated. NOTE 3 The algorithm to generate the signature is out of the Scope of this specification.	image_to_activate_size:	double-long-unsigned,	image_to_activate_identification:	octet-string,	image_to_activate_signature:	octet-string
image_to_activate_size:	double-long-unsigned,						
image_to_activate_identification:	octet-string,						
image_to_activate_signature:	octet-string						

Method description	
image_transfer_initiate (data)	Initializes the Image transfer process. data::= structure <pre> { image_identifier: octet-string, image_size: double-long-unsigned } </pre> Where: - image_identifier identifies the Image to be transferred; - image_size holds the ImageSize, expressed in octets. NOTE 4 The <i>image_identifier</i> identifies the Image to be transferred (container) but it is not necessarily linked to its content, i.e. the Images which will be activated. That information can be retrieved from the <i>image_to_activate</i> attribute after verification of the Image transferred. After a successful invocation of the method the <i>image_transfer_status</i> attribute is set to (1) and the <i>image_first_not_transferred_block_number</i> is set to 0. Any subsequent invocation of the method resets the whole Image transfer process and all ImageBlocks need to be transferred again.
image_block_transfer (data)	Transfers one block of the Image to the server. data::= structure <pre> { image_block_number: double-long-unsigned, image_block_value: octet-string } </pre> After a successful invocation of the method the corresponding bit in the <i>image_transferred_blocks_status</i> attribute is set to 1 and the <i>image_first_not_transferred_block_number</i> attribute is updated.
image_verify (data)	Verifies the integrity of the Image before activation. data::= integer (0) The result of the invocation of this method may be success, temporary_failure or other_reason. If it is not success, then the result of the verification can be found by retrieving the value of the <i>image_transfer_status</i> attribute. In the case of success, the <i>image_to_activate_info</i> attribute holds the information about the images to activate. NOTE 5 xDLMS services Action-Result / Data-Access-Result codes are specified in IEC 62056-5-3:2016, Clause 8.
image_activate (data)	Activates the Image. data::= integer (0) If the Image transferred has not been verified before, then this is done as part of the Image activation. The result of the invocation of this method may be success, temporary-failure or other-reason. If it is not success, then the result of the activation can be learned by retrieving the value of the <i>image_transfer_status</i> attribute. NOTE 6 xDLMS services Action-Result / Data-Access-Result codes are specified in IEC 62056-5-3:2016, Clause 8.

5.3.6.4 Example for a standard Image transfer process

In this subclause an example of a usual Image transfer process from a client prospective is given including a flow chart, see Figure 10. Note that it is not mandatory to follow the process; depending on the use case some steps may be omitted or others may be necessary.

Precondition: The Image transfer has to be enabled: *image_transfer_enabled* = TRUE.

Setting *image_transfer_enabled* any time to FALSE disables all methods (invoking those methods fails). The value of the status attributes is not defined.

Step 1 (Optional): Get ImageBlockSize

If the client does not know the size of the image blocks the image transfer target server can handle, it shall read the *image_block_size* attribute of the relevant “Image transfer” object of each server the image has to be transferred to, before starting the process. The client can transfer then ImageBlocks of the right size.

If ImageBlocks are sent using broadcast to a group of COSEM servers the ImageBlockSize shall be the same in each member of the group.

Step 2: Client initiates Image transfer

The client initiates the Image transfer process individually or using broadcast in all servers by invoking the *image_transfer_initiate* method. The method invocation parameter holds the identifier and the size of the Image to be transferred. The server shall make the memory space – necessary to accommodate the Image – available.

After a successful initiation, the value of the *image_transfer_status* attribute is (1). The *image_transferred_blocks_status* attribute shall be reset, the value of the *image_first_not_transferred_block_number* attribute shall be set to 0 and the value of the *image_to_activate_info* attribute should be reset. The image transfer process is initiated and the COSEM server is prepared to accept ImageBlocks.

Step 3: Client transfers ImageBlocks

The client transfers ImageBlocks to (a group of) server(s) by invoking the *image_block_transfer* method individually or using broadcast. The method invocation parameters include the ImageBlockNumber and one ImageBlock. ImageBlocks are accepted only by those COSEM servers, in which the Image transfer process has been successfully initiated. Other servers silently discard any ImageBlocks received.

Step 4: Client checks completeness of the Image

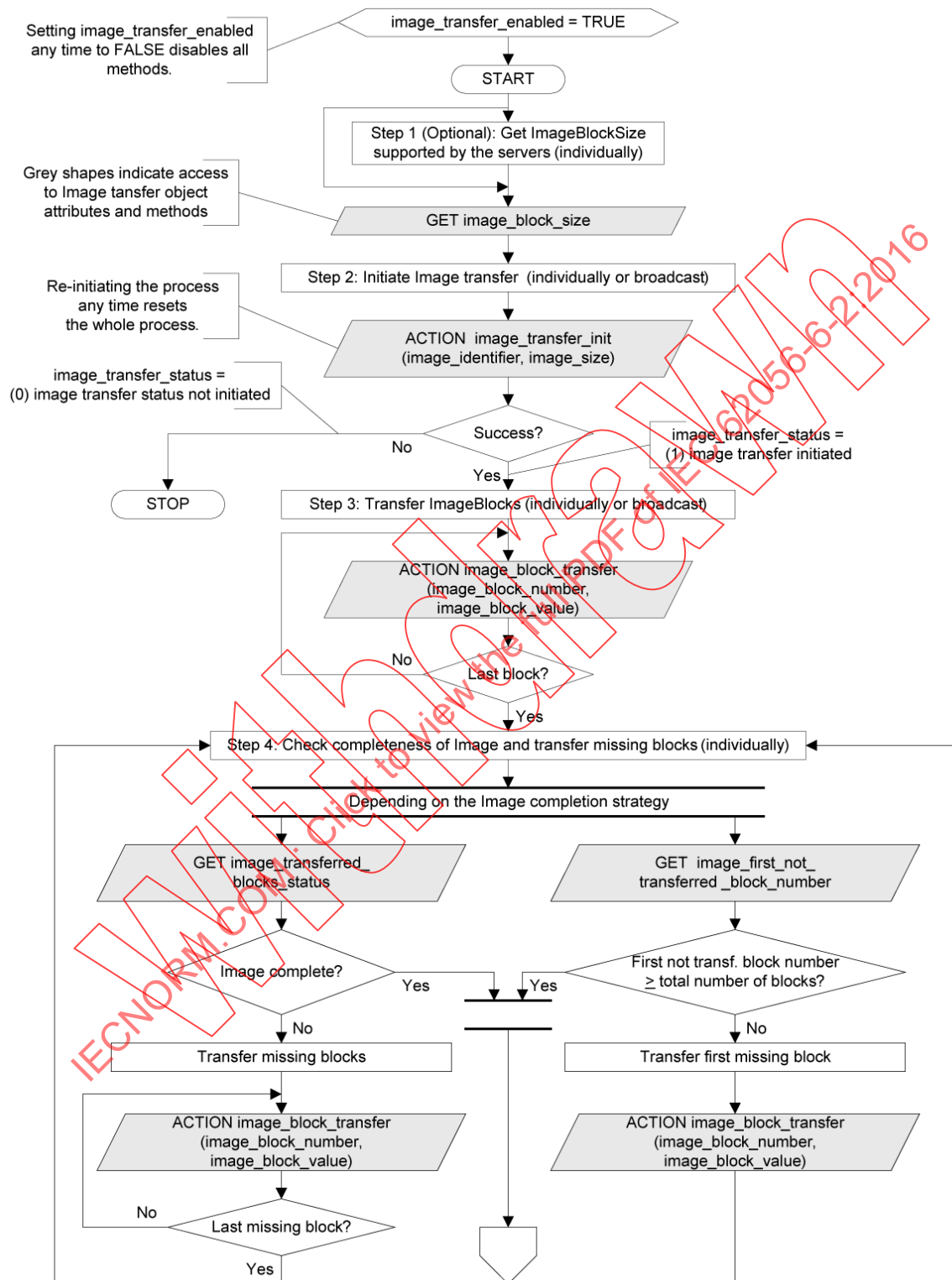
The client checks – with each server individually – the completeness of the Image transferred. If the Image is not complete, it transfers the ImageBlocks not (yet) transferred. This is an iterative process, continued until the whole Image is successfully transferred.

To identify and transfer the ImageBlocks not transferred, two mechanisms are available.

- the client may retrieve the status of each ImageBlock: either not transferred or transferred. This is performed by retrieving the value of the *image_transferred_blocks_status* attribute. The client transfers then the ImageBlocks not (yet) transferred;
- alternatively, the client may retrieve the ImageBlockNumber of the first block not transferred. This is performed by retrieving the value of the *image_first_not_transferred_block_number* attribute. The client transfers then this ImageBlock not (yet) transferred;

- after this, the client checks again he completeness of the Image.

NOTE 1 The two mechanisms can be freely combined.



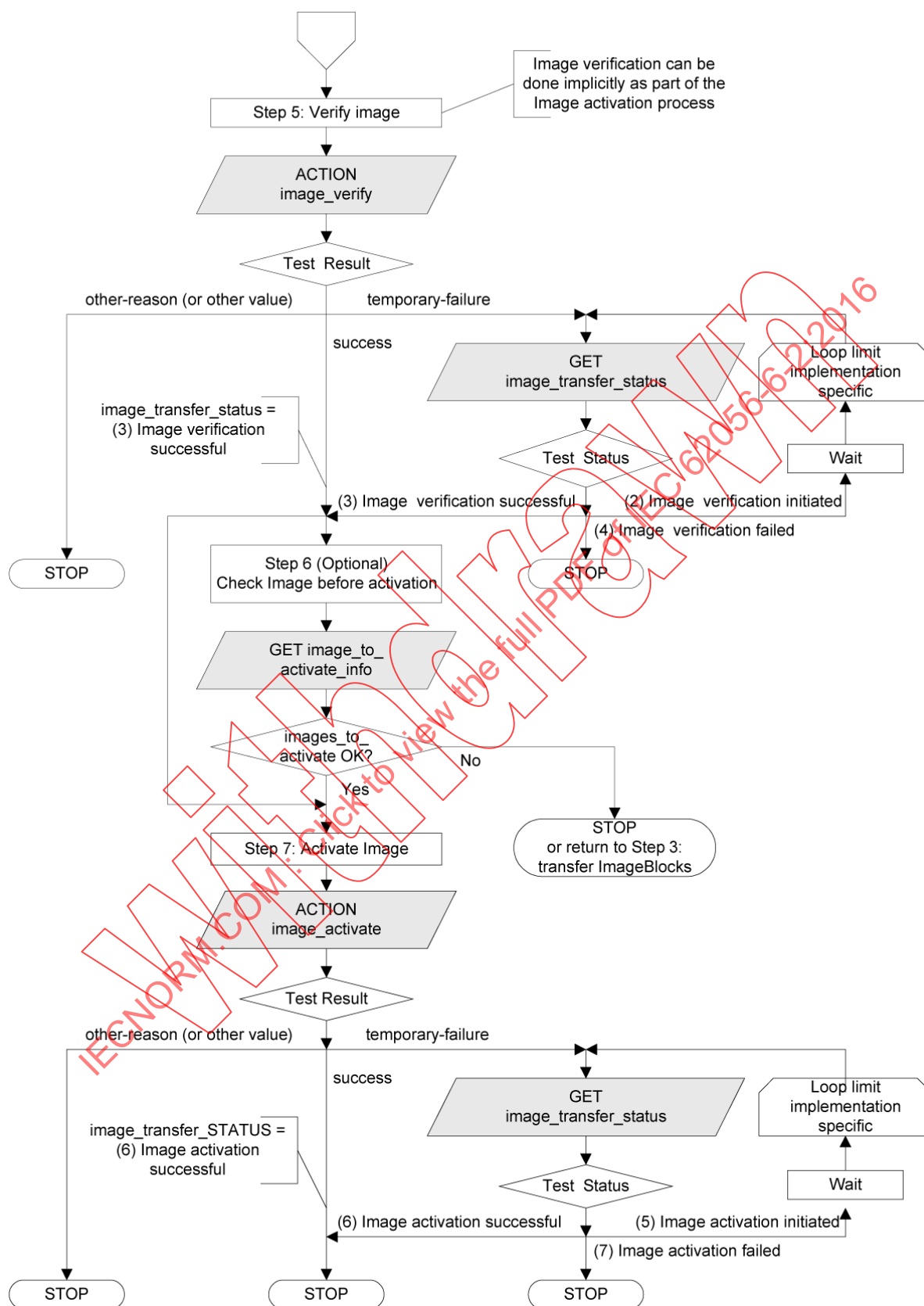


Figure 10 – Image transfer process flow chart

Step 5: Server verifies the Image

The Image is verified by the server. This can be initiated by invoking the *image_verify* method by the client or it may be initiated also by the server. The result can be:

- success, if the verification could be completed;
- temporary-failure, if the verification has not been completed;
- other-reason, if the verification failed.

NOTE 2 The conditions of verifying the Image are out of the scope of this standard.

Step 6 (Optional): Client checks the information on the images to activate

The result of the Image verification may be checked by the client by retrieving the value of the *image_transfer_status* attribute. The value of this attribute is updated as a result of the Image verification.

An Image transferred may hold one or several Images to be activated. For each Image to be activated, this attribute holds the parameters: {image_to_activate_size, image_to_activate_identification, image_to_activate_signature}.

If this information is not what is expected, the client can restart transferring the image.

Otherwise it goes to the next step, activation of the Image.

Step 7: Server activates the Image(s)

The Image is activated by the server. This can be initiated by invoking the *image_activate* method by the client or it may be initiated also by the server.

If the activation is done without a previous verification, then verification is done implicitly as part of the activation. The result of the invocation of the *Image_activate* method can be:

- success, if the Image activation has been successfully started;
- temporary-failure, if the verification/activation has not been completed;
- other-reason, if the activation failed.

In the case of success, the server performs the activation of the new Image(s). During this process, it is not accessible. After the Image(s) has (have) been activated, the result of may be checked by the client by retrieving the value of the *image_transfer_status* attribute or by reading the contents of the appropriate COSEM objects holding the identifier, version and digital signature of the active firmware.

5.3.7 Security setup (class_id = 64, version = 0)

Instances of this IC contain the necessary information on the security policy applicable and the security suite in use within a particular AA, between two systems identified by their client system title and server system title respectively. They also contain methods to increase the level of security and to transfer the global keys. See IEC 62056-5-3:2016, Clause 5.

Security setup		0...n	class_id = 64, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	security_policy (static)	enum	0	3	0	x + 0x08
3.	security_suite (static)	enum	0	0	0	x + 0x10
4.	client_system_title (dyn.)	octet-string				x + 0x18
5.	server_system_title (static)	octet-string				x + 0x20
Specific methods		m/o				
1.	security_activate	o				x + 0x28
2.	global_key_transfer	o				x + 0x30

Attribute description

logical_name	Identifies the “Security setup” object instance. See 6.2.30.
security_policy	Enforces authentication and/or encryption algorithm provided with security_suite. enum: (0) nothing, (1) all messages to be authenticated, (2) all messages to be encrypted, (3) all messages to be authenticated and encrypted (4)...(15) reserved
security_suite	Specifies authentication, encryption and key transport algorithm. enum: (0) AES-GCM-128 for authenticated encryption and AES-128 for key wrapping (1)...(15) reserved
client_system_title	Carries the (current) client system title: – in the S-FSK PLC environment, the active initiator sends its system title using the CIASE protocol; NOTE 1 It is also held by the <i>active_initiator</i> attribute of the S-FSK Active initiator object; see 5.8.4; – during confirmed or unconfirmed AA establishment, it is carried by the calling-AP-title field of the AARQ APDU; – If a client system title has already been sent during a registration process, like in the case of the S-FSK PLC profile, the client system title carried by the AARQ APDU should be the same. If not, the AA shall be rejected and appropriate diagnostic information shall be sent. – in a pre-established AA, it can be written by the client using an unsecured SET / Write service.
server_system_title	Carries the server system title. – in the S-FSK PLC environment, the server sends its system title during the discover process, using the CIASE protocol; – during confirmed AA establishment, it is carried by the responding-AP-title field of the AARE APDU. This attribute shall be read only.

Method description	
security_activate (data)	Activates and strengthens the security policy: enum: (0) nothing, (1) all messages to be authenticated, (2) all messages to be encrypted, (3) all messages to be authenticated and encrypted The new security policy applies as soon as the method invocation has been confirmed with success. NOTE 2 The security policy can only be strengthened.
global_key_transfer (data)	Updates one or more global keys. The <i>data</i> parameter includes wrapped key data. Key data include the key identifiers and the keys themselves. array key_data key_data ::= structure <pre>{ key_id: enum: (0) global unicast encryption key, (1) global broadcast encryption key, (2) authentication key key_wrapped: octet-string }</pre> The key wrapping algorithm is as specified by the security suite. The KEK is the master key. The new key(s) are valid as soon as the method invocation has been confirmed with success.

5.3.8 Push interface classes and objects

5.3.8.1 Overview

There are several occasions on which DLMS messages can be ‘pushed’ to a destination without being explicitly requested, e.g.

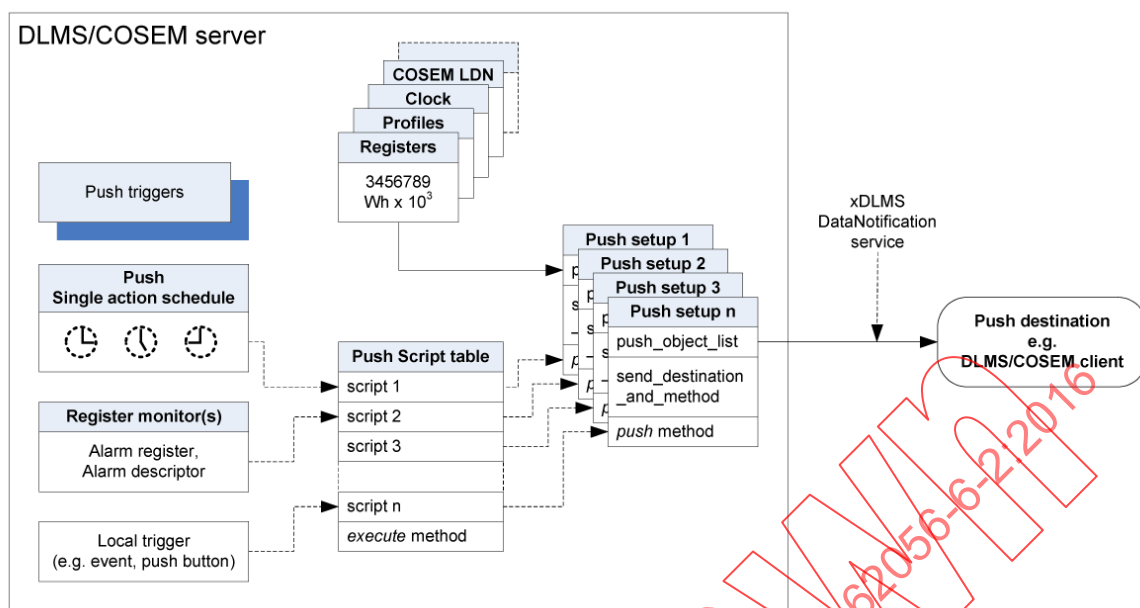
- if a scheduled time is reached;
- if a value being locally monitored exceeds a threshold;
- on triggering by a local event (e.g. power-up/down, push button pressed, meter cover opened).

The DLMS/COSEM push mechanism follows the publish/subscribe pattern:

Publish/subscribe is a messaging pattern where senders (publishers) of messages do not program the messages to be sent directly to specific receivers (subscribers). Rather, published messages are characterized into classes, without knowledge of what, if any, subscribers there may be. Subscribers express interest in one or more classes, and only receive messages that are of interest, without knowledge of what, if any, publishers there are. [Wikipedia]

In DLMS/COSEM, publishing is modelled by the *object_list* attribute of “Association” objects providing the list of COSEM objects and their attributes accessible in a given AA. Subscription is modelled by writing the appropriate attributes of “Push setup” objects. The required data are sent – upon the trigger specified – using the xDLMS DataNotification service.

The COSEM model of the Push operation is shown in Figure 11.



IEC

Figure 11 – COSEM model of push operation

The core element of modelling the push operation is the “Push setup” IC. The *push_object_list* attribute contains a list of references to COSEM object attributes to be pushed.

The various triggers (e.g. schedulers, monitors, local triggers, etc.) call a script entry in a Push “Script table” object (new instances of the “Script table” IC) which invokes then the *push* method of the related “Push setup” object. The destination, the communication media, the protocol, the encoding, the timing as well as any retries of the push operation are determined by the other attributes of the “Push setup” object.

Each trigger can cause the data to be sent to a dedicated destination. Therefore, for every trigger or a group of triggers an individual “Push setup” object is available defining the content and the destination of the push message as well as the communication medium used.

For the purposes of pushing data when an alarm occurs, Alarm monitor objects (new instances of the “Register monitor” IC) are available.

The alarms are held by *Alarm register* or *Alarm descriptor* objects, see 6.2.52.

The structure of the *Alarm descriptor* as well as the alarm conditions needs to be defined in a project specific companion specification.

Alarm descriptor objects are monitored by Alarm “Register monitor” objects, see 6.2.13. The *actions* attribute provides the link to the *action_up* and maybe the *action_down* scripts in the Push “Script table” object – see 6.2.7 – which invoke then the *push* method of the desired “Push setup” object. When an alarm occurs, the predefined set of data – that may include among other data the *Alarm register* and *Alarm descriptor* objects – are pushed.

The push data is sent – when the conditions for pushing are met – using the unsolicited, non-client/server type xDLMS service, the DataNotification service. When the data pushed is long, it can be sent in blocks.

The push process takes place within the application context of the AA in which the “Push setup” object is visible. The security context is determined by the “Security setup” object referenced from the “Association” object.

NOTE Details are subject to project specific companion specifications.

All information necessary to process the data received by the client shall be either:

- retrieved by the client e.g. by reading the *push_object_list attribute*; or
- shall be part of the data pushed; or
- shall be predefined in the client application.

5.3.8.2 Push setup (class_id = 40, version = 0)

The "Push setup" IC contains a list of references to COSEM object attributes to be pushed. It also contains the push destination and method as well as the communication time windows and the handling of retries.

The push takes place upon invoking the *push* method, triggered by a Push "Single action schedule" object, by an alarm "Register monitor" object, by a dedicated internal event or externally. After the push operation has been triggered, it is executed according to the settings made in the given "Push setup" object. Depending on the communication window settings, the push is executed immediately or as soon as a communication window becomes active, after a random delay. If the push was not successful, retries are made. Push windows, delays and retries are shown in Figure 12.

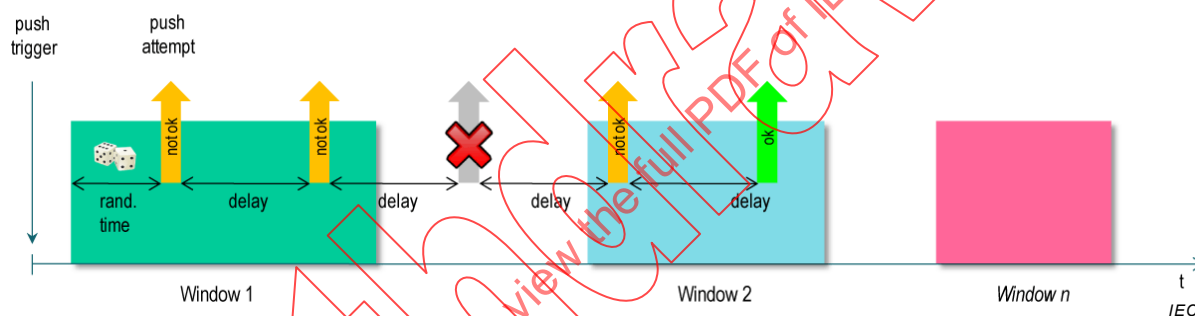


Figure 12 – Push windows and delays

Push setup		0...n	class_id = 40, version = 0			
Attribute (s)		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. push_object_list	(static)	array				x + 0x08
3. send_destination_and_method	(static)	structure				x + 0x10
4. communication_window	(static)	array				x + 0x18
5. randomisation_start_interval	(static)	long-unsigned				x + 0x20
6. number_of_retries	(static)	unsigned				x + 0x28
7. repetition_delay	(static)	long-unsigned				x + 0x30
Specific methods		m/o				
1. push (data)		m				
						x + 0x38

Attribute description

logical_name Identifies the “Push setup” object instance. See 6.2.21.

push_object_list Defines the list of attributes to be pushed.
Upon invocation of the push (data) method the items are sent to the destination defined in the send_destination_and_method attribute.

array object_definition

```
object_definition ::= structure
{
    class_id:      long-unsigned,
    logical_name:  octet-string,
    attribute_index: integer,
    data_index:    long-unsigned
}
```

Where:

attribute_index is a pointer to the attribute within the object, identified by class_id and logical_name: attribute_index 1 refers to the 1st attribute (i.e. the logical_name), attribute_index 2 to the 2nd attribute etc.; attribute_index 0 refers to all public attributes;

data_index is a pointer selecting one or several specific elements of an attribute with a complex data type (structure or array).

data_index:	MS-Byte	LS-Byte
	Upper nibble	Lower nibble

If the data_type of the attribute is simple, then data_index has no meaning.

If the attribute is a structure or an array, then data_index points to an element in the structure or array. The first element in the complex attribute is identified by data_index 1.

When the attribute is the *buffer* of a “Profile generic” object, the data_index carries selective access parameters.

- 0x0000 = identifies the whole attribute;
- 0x0001 to 0x0FFF = identifies one element in the complex attribute;
- 0x1000 to 0xFFFF = selective access to the array holding the *buffer* of a “Profile generic” object. The data-index selects entries within a number of last (recent) time periods, or a number of last (recent) entries, as well as the columns in the array.

The encoding is specified in Table 6.

NOTE 1 If the push_object_list array is empty, the push operation is disabled.

NOTE 2 The push_object_list attribute itself can be pushed as well to clearly identify the data pushed.

NOTE 3 Similarly to the case of the “Profile generic” IC, all attributes included in the push_object_list attribute are pushed regardless of the access rights to them. Therefore, writing of the push_object_list attribute should be restricted to clients with appropriate access rights.

**send_
destination_
and_method**

Contains the destination address (e.g. phone number, email address, IP address) where the data specified by the *push_object_list* has to be sent, as well as the sending method.

send_destination_and_method ::= structure

```
{
    transport_service:    transport_service_type,
    destination:          octet-string,
    message:              message_type
}
```

Where:

- the transport_service element defines the type of service used to push the data:

transport_service_type ::= enum:

(0)	TCP,
(1)	UDP,
(2)	reserved for FTP,
(3)	reserved for SMTP,
(4)	SMS,
(5)	HDLC
(6)	reserved for M-Bus
(7)	reserved for ZigBee®
(200...255)	manufacturer specific

- the destination element contains the target address where the data has to be sent. The elements of the target address depend on the transport service used.

Each “Push setup” object instance specifies a single destination. If it is required to push data to several destinations, several “Push setup” objects have to be instantiated.

- the message_type element identifies the encoding of the xDLMS APDU used.

message_type ::= enum:

(0)	A-XDR encoded xDLMS APDU,
(1)	XML encoded xDLMS APDU,
(128...255)	manufacturer specific

All other transport_service_type and message_type values are reserved for future use.

**communication_
window**

Defines the time points when the communication window(s) for the push become(s) active (start_time) and inactive (end_time). See Figure 12.

array window_element

window_element ::= structure

```
{
    start_time: octet-string,
    end_time: octet-string
}
```

start_time and end_time are formatted as specified in 4.6.1 for *date-time* including wildcards.

If the end of a communication window is reached an already started push operation is completed.

If no communication windows are defined (array [0]) the push operation is always possible.

randomisation_start_interval	To avoid simultaneous push operations by a lot of devices at exactly the same point in time, a randomisation interval – in seconds – can be defined. This means that the push operation is not started immediately after the invocation of the <i>push</i> method but randomly delayed within the interval. The <i>randomisation_start_interval</i> attribute defines the maximum value which can be obtained from a randomizing algorithm. The resulting value is used to delay the first push operation. It is not used anymore in case of retries. If no communication window is active when invoking the <i>push</i> method, the delay starts at the beginning of the next communication window. If the <i>push</i> method is invoked during an active communication window, the push operation is still delayed. The <i>randomisation_start_interval</i> is only active for the first push attempt. If no <i>communication_window</i> is defined, the <i>randomisation_start_interval</i> is active for every push attempt. If the <i>randomisation_start_interval</i> is set to 0 no delay is active. If the randomisation time is longer than the first communication window, the first push attempt will be made during any later window.
number_of_retries	Defines the maximum number of retries in the case of unsuccessful or skipped push attempts. After a successful push operation no further push attempts are made until the push operation is triggered again. A push is treated as successful if a lower layer transmission confirmation has been received. The conditions of detecting an unsuccessful push attempt may depend on the communication profile used and on the implementation and it is therefore out of the Scope of this Technical Specification.
repetition_delay	The time delay, expressed in seconds until the next push attempt is started after an unsuccessful push. NOTE 4 The repetition delay itself is not influenced by the communication window. But a retry only can be made if a communication window is active at that time. Otherwise it is handled like an unsuccessful push attempt. NOTE 5 The push data is not stored in an intermediate buffer. In the case of retries, the current values of the attributes may change with every push attempt.
Method description	
push (data)	Activates the push process leading to the elaboration and the sending of the push data taking into account the values of the attributes defined in the given instance of this IC. data::= integer (0) The push data returned shall be a structure. The number of elements in this structure shall be the same as number of elements in the array of the <i>push_object_list</i> attribute.

**Table 6 – Encoding of selective access parameters
with data_index**

	MS_Byte upper nibble: Selects the time periods or entries.
0xF	Last complete number of months: Selective access to the <i>buffer</i> resulting in all entries of the last complete number of months and the first entry at midnight of the current month.
0xE	Last complete number of days: Selective access to the <i>buffer</i> resulting in all entries of the last complete number of days and the first entry at midnight of today.
0xD	Last complete number of hours: Selective access to the <i>buffer</i> resulting in all entries of the last complete number of hours and the first entry of the current hour.
0xC	Last complete number of minutes: Selective access to the <i>buffer</i> resulting in all entries of the last complete number of minutes and the first entry of the current minute.
0xB	Last number of seconds: Selective access to the <i>buffer</i> resulting in all entries of the last number of seconds.
0xA	Last complete number of months including the current month: Same as 0xF above but all entries up to now are retrieved.
0x9	Last complete number of days including the current day: Same as 0xE above but all entries up to now are retrieved.
0x8	Last complete number of hours including the current hour: Same as 0xD above but all entries up to now are retrieved.
0x7	Last complete number of minutes including the current minute: This is the same as 0xC above but all entries up to now are retrieved.
0x6...0x2	Reserved
0x1	Last number of entries
0x0	Whole attribute or a single element in an attribute with complex data type is selected (MS-Byte lower nibble 0x0...0xF, LS-Byte 0x00...0xFF).
	MS-Byte lower nibble: Defines the number of columns of a "Profile generic" <i>buffer</i> selected.
0x0	All columns
0x1 to 0xF	Number of columns, starting from column 1.
0x00...0xFF	LS-Byte: - in the case of selective access by time period, (i.e. number of month, days, hours, etc.) defines the number of recent complete time periods (1 to 255), - in the case of selective access by entry defines the number of recent entries.
Example 1)	0xE401: The entries for the last complete day are selected. The first 4 columns are included.
Example 2)	0xA300: The entries for zero last complete months and the current month are selected. The first 3 columns are included.
Example 3)	0x800C: The entries for the last complete 12 hours are selected. All columns are included.
Example 4)	0x1080: The last 128 entries are selected. All columns are included.

5.4 Interface classes for time- and event bound control

5.4.1 Clock (class_id = 8, version = 0)

This IC models the device clock, managing all information related to date and time including deviations of the local time to a generalized time reference (UTC) due to time zones and daylight saving time schemes. The IC also offers various methods to adjust the clock.

The *date* information includes the elements year, month, day of month and day of week. The *time* information includes the elements hour, minutes, seconds, hundredths of seconds, and the deviation of the local time from UTC. The daylight saving time function modifies the deviation of local time to UTC depending on the attributes; see Figure 13. The start and end point of that function is normally set once. An internal algorithm calculates the real switch point depending on these settings.

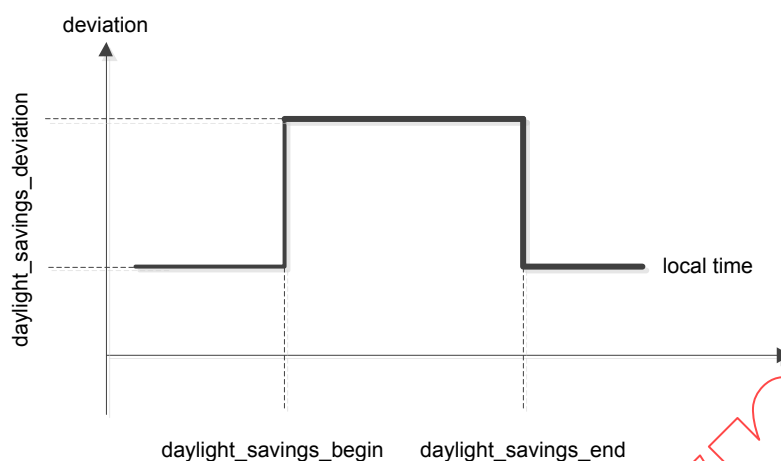


Figure 13 – The generalized time concept

Clock		0...n	class_id = 8, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. time	(dyn.)	octet-string				x + 0x08
3. time_zone	(static)	long		± 720		x + 0x10
4. status	(dyn.)	unsigned				x + 0x18
5. daylight_savings_begin	(static)	octet-string				x + 0x20
6. daylight_savings_end	(static)	octet-string				x + 0x28
7. daylight_savings_deviation	(static)	integer		± 120		x + 0x30
8. daylight_savings_enabled	(static)	boolean				x + 0x38
9. clock_base	(static)	enum				x + 0x40
Specific methods		m/o				
1. adjust_to_quarter (data)		o				x + 0x60
2. adjust_to_measuring_period (data)		o				x + 0x68
3. adjust_to_minute (data)		o				x + 0x70
4. adjust_to_preset_time (data)		o				x + 0x78
5. preset_adjusting_time (data)		o				x + 0x80
6. shift_time (data)		o				x + 0x88

Attribute description

logical_name	Identifies the “Clock” object instance. See 6.2.5.
time	Contains the meter’s local date and time, its deviation to UTC and the status. octet-string, formatted as specified in 4.6.1 for <i>date-time</i>. When this attribute is set, all the fields in the octet-string representing <i>date-time</i> shall be evaluated and the local date and time in the meter shall be set according to the rules defined in 4.6.1. Only specified fields of the <i>date-time</i> are changed. EXAMPLE For setting the <i>date</i> without changing the <i>time</i>, all time-relevant octets in the octet string representing <i>date-time</i> shall be set to “not specified”.

time_zone	The deviation of local, normal time to UTC in minutes. The value depends on the geographical location of the meter.
status	The clock_status maintained by the meter. The clock_status element indicated in the <i>time</i> attribute is equal to the value of this attribute. unsigned, formatted as specified in 4.6.1 for <i>clock_status</i>
daylight_savings_begin	Defines the local switch date and time when the local time starts to deviate from the normal time. For generic definitions, wildcards are allowed. octet-string, formatted as specified in 4.6.1 for <i>date-time</i> .
daylight_savings_end	Defines the local switch date and time when the local time ends to deviate from the local normal time. octet-string, formatted as specified in 4.6.1 for <i>date-time</i> .
daylight_savings_deviation	Contains the number of minutes by which the deviation in generalized time shall be corrected at daylight savings begin. integer: Deviation in the range of ± 120 min
daylight_savings_enabled	boolean: TRUE = DST enabled, FALSE = DST disabled NOTE This is a parameter to enable and disable the daylight savings time feature. The current status is indicated in the status attribute.
clock_base	Defines where the basic timing information comes from. enum: (0) not defined, (1) internal crystal, (2) mains frequency 50 Hz, (3) mains frequency 60 Hz, (4) GPS (global positioning system), (5) radio controlled
Method description	
adjust_to_quarter (data)	Sets the meter's time to the nearest (+/-) quarter of an hour value (*:00, *:15, *:30, *:45). data::= integer (0)
adjust_to_measuring_period (data)	Sets the meter's time to the nearest (+/-) starting point of a measuring period. data::= integer (0)
adjust_to_minute (data)	Sets the meter's time to the nearest minute. If second_counter < 30 s, second_counter is set to 0. If second_counter ≥ 30 s, second_counter is set to 0, and minute_counter and all depending clock values are incremented if necessary. data::= integer (0)
adjust_to_preset_time (data)	This method is used in conjunction with the preset_adjusting_time method. If the meter's time lies between validity_interval_start and validity_interval_end, then time is set to preset_time. data::= integer (0)

preset_adjusting_time (data) Presets the time to a new value (preset_time) and defines a validity_interval within which the new time can be activated.

data::= structure

```
{
    preset_time:          octet-string,
    validity_interval_start: octet-string,
    validity_interval_end:  octet-string
}
```

all octet-strings formatted as specified in 4.6.1 for *date-time*.

shift_time (data) Shifts the time by *n* ($-900 \leq n \leq 900$) s.

data::= long

5.4.2 Script table (class_id = 9, version = 0)

This IC allows modelling the triggering of a series of actions by executing scripts using the *execute* (data) method.

“Script table” objects contain a table of script entries. Each entry consists of a script identifier and a series of action specifications. An action specification activates a method or modifies an attribute of a COSEM object within the logical device.

A certain script may be activated by other COSEM objects within the same logical device or from the outside.

If two scripts have to be executed at the same time instance, then the one with the smaller index is executed first.

Script table		0...n	class_id = 9, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	scripts (static)	array				x + 0x08
Specific methods		m/o				
1.	execute(data)	m				x + 0x20

Attribute description

logical_name Identifies the “Script table” object instance. See 6.2.7.

scripts Specifies the different scripts, i.e. the lists of actions.

array script

script::= structure

```
{
    script_identifier: long-unsigned,
    actions:          array action_specification
}
```

The script_identifier 0 is reserved. If specified with an *execute* method, it results in a null script (no actions to perform).

scripts
cont'd

```

action_specification ::= structure
{
    service_id:          enum,
    class_id:            long-unsigned,
    logical_name:        octet-string,
    index:               integer,
    parameter:           service specific
}

```

Where:

- the service_id element defines which action to be applied to the referenced object:
 - (1) write attribute,
 - (2) execute specific method
- the index element defines (with service_id 1) which attribute of the selected object is affected; or (with service_id 2) which specific method is to be executed. The first attribute (logical_name) has index 1, the first specific method has index 1 as well.

NOTE 1 The action_specification is limited to activate methods that do not produce any response (from the server to the client).

NOTE 2 A “dummy” action specification with all elements 0 means that the action is not configured.

Method description**execute**
(data)

Executes the script specified in parameter data.

data ::= long-unsigned

If data matches one of the script_identifiers in the script table, then the corresponding action_specification is executed.

5.4.3 Schedule (class_id = 10, version = 0)

This IC, together with the IC “Special days”, allows modelling time- and date-driven activities within a device. Table 7 provides an overview and show the interactions between the two ICs. See also Table 8.

Table 7 – Schedule

Index	enable	action (script)	Switch time	validity_ window	exec_weekdays							exec_specdays					date range	
					Mo	Tu	We	Th	Fr	Sa	Su	S1	S2	...	S8	S9	begin_ date	end_ date
120	Yes	xxxx:yy	06:00	0xFFFF	x	x	x	x	x	x							xx-04-01	xx-09-30
121	Yes	xxxx:yy	22:00	15	x	x	x	x	x								xx-04-01	xx-09-30
122	Yes	xxxx:yy	12:00	0						x							xx-04-01	xx-09-30
200	No	xxxx:yy	06:30		x	x	x	x	x	x							xx-04-01	xx-09-30
201	No	xxxx:yy	21:30		x	x	x	x	x								xx-04-01	xx-09-30
202	No	xxxx:yy	11:00							x							xx-04-01	xx-09-30

Table 8 – Special days table

Index	special_day_date	day_id
12	xx-12-24	S1
33	xx-12-25	S3
77	97-03-31	S3

Schedule	0...n	class_id = 10, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. entries (static)	array				x + 0x08
Specific methods	m/o				
1. enable/disable (data)	o				x + 0x20
2. insert (data)	o				x + 0x28
3. delete (data)	o				x + 0x30

Attribute description

logical_name Identifies the “Schedule” object instance. See 6.2.9.

entries Specifies the scripts to be executed at given times. There is only one script that can be executed per entry.

array schedule_table_entry

```

schedule_table_entry := structure
{
    index: long-unsigned,
    enable: boolean,
    script_logical_name: octet-string,
    script_selector: long-unsigned,
    switch_time: octet-string,
    validity_window: long-unsigned,
    exec_weekdays: bit-string,
    exec_specdays: bit-string,
    begin_date: octet-string,
    end_date: octet-string
}

```

Where:

- script_logical_name: defines the logical name of the “Script table” object;
- script_selector: defines the script_identifier of the script to be executed;
- switch_time accepts wildcards to define repetitive entries. The format of the octet-string follows the rules set in 4.1.6.1 for time;
- validity_window defines a period in minutes, in which an entry shall be processed after power fail. (time between defined switch_time and actual power_up) 0xFFFF: the script is processed any time;
- exec_weekdays defines the days of the week on which the entry is valid;
- exec_specdays performs the link to the IC “Special days table”, day_id;
- begin_date and end_date define the date period in which the entry is valid (wildcards are allowed). The format follows the rules set in 4.6.1 for date.

Method description

enable/disable (data) Sets the disabled bit of range A entries to true and then enables the entries of range B.

```
data := structure
{
    firstIndexA:    long-unsigned,
    lastIndexA:     long-unsigned,
    firstIndexB:    long-unsigned,
    lastIndexB:     long-unsigned
}
```

firstIndexA	first index of the range that is disabled,

lastIndexA	last index of the range that is disabled,

firstIndexB	first index of the range that is enabled,

lastIndexB	last index of the range that is enabled,

firstIndexA/B < lastIndexA/B:	all entries of the range A/B are disabled / enabled,

firstIndexA/B = lastIndexA/B:	one entry is disabled/enabled,

firstIndexA/B > lastIndexA/B:	nothing disabled/enabled,

firstIndexA/B and lastIndexA/B > 9999:	no entry is disabled/enabled

insert (data) Inserts a new entry in the table. If the index of the entry exists already, the existing entry is overwritten by the new entry.

```
entry: schedule_table_entry
data := corresponding to entry
```

delete (data) Deletes a range of entries in the table.

```
data := structure
{
    firstIndex: long-unsigned,
    lastIndex:  long-unsigned
}
```

firstIndex:	first index of the range that is deleted,

lastIndex:	last index of the range that is deleted,

firstIndex < lastIndex:	all entries of the range A/B are deleted,

firstIndex = lastIndex:	one entry is deleted,

firstIndex > lastIndex:	nothing deleted

Remarks concerning “inconsistencies” in the table entries:

If the same script should be executed several times at a specific time instance, then it is executed only once.

If different scripts should be executed at the same time instance, then the execution order is according to the “index”. The script with the lowest “index” is executed first.

Recovery after power failure

After a power failure, the whole schedule is processed to execute all the necessary scripts that would get lost during a power failure. For this, the entries that were not executed during the power failure have to be detected. Depending on the validity window they are executed in the correct order (as they would have been executed in normal operation).

Handling of time changes

There are four different "actions" of time changes:

- a) time setting forward (by writing to the attribute *time* of the "Clock" object);
- b) time setting backward (by writing to the attribute *time* of the "Clock" object);
- c) time synchronization (using the method *adjust_to_quarter* of the "Clock" object);
- d) daylight saving action.

All these four actions need a different handling executed by the schedule in interaction with the time setting activity.

Time setting forward

This is handled the same way as a power failure. All entries missed are executed depending on the validity_window. A (manufacturer specific defined) short time setting can be handled like time synchronization.

Time setting backward

This results in a repetition of those entries that are activated during the repeated time. A (manufacturer specific defined) short time setting can be handled like time synchronization.

Time synchronization

Time synchronization is used to correct small deviations between a master clock and the local clock. The algorithm is manufacturer specific. It shall guarantee that no entry of the schedule gets lost, or is executed twice. The validity_window attribute has no effect, because all entries have to be executed in normal operation.

Daylight saving

If the clock is put forward, then all scripts, which fall into the forwarding interval (and would therefore get lost) are executed. If the clock is put back, re-execution of the scripts, which fall into the backwarding interval, is suppressed.

5.4.4 Special days table (class_id = 11, version = 0)

This IC allows defining special dates. On such dates, a special switching behaviour overrides the normal one. The IC works in conjunction with the class "Schedule" or "Activity calendar". The linking data item is *day_id*.

Special days table		0...n	class_id = 11, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	entries (static)	array				x + 0x08
Specific methods		m/o				
1.	insert (data)	o				x + 0x10
2.	delete (data)	o				x + 0x18

Attribute description

logical_name	Identifies the “Special days table” object instance. See 6.2.8.
entries	Specifies a special day identifier for a given date. The date may have wildcards for repeating special days like Christmas. array spec_day_entry spec_day_entry::= structure <pre>{ index: long-unsigned, specialday_date: octet-string, day_id: unsigned }</pre> Where: – specialday_date formatting follows the rules set in 4.6.1 for <i>date</i>; – the range of the day_id shall match the length of the bit-string exec_specdays in the related object of IC “Schedule”.

Method description

insert (data)	Inserts a new entry in the table. entry::= spec_day_entry If a special day with the same index or with the same date as an already defined day is inserted, the old entry will be overwritten.
delete (data)	Deletes an entry in the table. index Index of the entry that shall be deleted. data::= long-unsigned

5.4.5 Activity calendar (class_id = 20, version = 0)

This IC allows modelling the handling of various tariff structures in the meter. The IC provides a list of scheduled actions, following the classical way of calendar based schedules by defining seasons, weeks, etc.

An “Activity calendar” object may coexist with the more general “Schedule” object and it can even overlap with it. If actions in a “Schedule” object are scheduled for the same activation time as in an “Activity calendar” object, the actions triggered by the “Schedule” object are executed first.

After a power failure, only the “last action” missed from the object “Activity calendar” is executed (delayed). This is to ensure proper tariffication after power up. If a “Schedule” object is present, then the missed “last action” of the “Activity calendar” shall be executed at the correct time within the sequence of actions requested by the “Schedule” object.

The “Activity calendar” object defines the activation of certain scripts, which can perform different activities inside the logical device. The interface to the IC “Script table” is the same as for the IC “Schedule” (see 5.4.3).

If an instance of the IC “Special days table” (see 5.4.4) is available, relevant entries there take precedence over the “Activity calendar” object driven selection of a day profile. The day profile referenced in the “Special days table” activates the day_schedule of the day_profile_table in the “Activity calendar” object by referencing through the day_id.

Activity calendar		0...n	class_id = 20, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	calendar_name_active (static)	octet-string				x + 0x08
3.	season_profile_active (static)	array				x + 0x10
4.	week_profile_table_active (static)	array				x + 0x18
5.	day_profile_table_active (static)	array				x + 0x20
6.	calendar_name_passive (static)	octet-string				x + 0x28
7.	season_profile_passive (static)	array				x + 0x30
8.	week_profile_table_passive (static)	array				x + 0x38
9.	day_profile_table_passive (static)	array				x + 0x40
10.	activate_passive_calendar_time (static)	octet-string				x + 0x48
Specific methods		m/o				
1.	activate_passive_calendar (data)	o				x + 0x50

Attribute description

Attributes called ..._active are currently active. Attributes called ..._passive will be activated by the specific method activate_passive_calendar.

logical_name Identifies the “Activity calendar” object instance. See 6.2.10.

calendar_name Typically contains an identifier, which is descriptive to the set of scripts activated by the object.

season_profile Contains a list of seasons defined by their starting date and a specific week_profile to be executed. The list is sorted according to season_start (in increasing order);

array season

season ::= structure

```

{
    season_profile_name:    octet-string,
    season_start:           octet-string,
    week_name:              octet-string
}

```

season_profile cont'd	Where: – season_profile_name is a user defined name identifying the current season_profile; – season_start defines the starting time of the season, formatted as specified in 4.6.1 for <i>date-time</i>. In season_start, wildcards are allowed. If all fields are wildcards, the season will never start. When using wildcards, special care has to be taken to avoid conflicting parametrization, i.e. that the season_start of two different seasons is the same; NOTE The current season is terminated by the season_start of the next season. – week_name defines the week_profile active in this season.

week_profile_table	Contains an array of week_profiles to be used in the different seasons. For each week_profile, the day_profile for every day of a week is identified. array week_profile <pre> week_profile ::= structure { week_profile_name: octet-string, monday: day_id, tuesday: day_id, wednesday: day_id, thursday: day_id, friday: day_id, saturday: day_id, sunday: day_id } day_id: unsigned </pre> Where: – week_profile_name is a user defined name identifying the current week_profile; – Monday defines the day_profile valid on Monday; – ... – Sunday defines the day_profile valid on Sunday.

day_profile_table	Contains an array of day_profiles, identified by their day_id. For each day_profile, a list of scheduled actions is defined by a script to be executed and the corresponding activation time (start_time). The list is sorted according to start_time. array day_profile <pre> day_profile ::= structure { day_id: unsigned, day_schedule: array day_profile_action } day_profile_action ::= structure { start_time: octet-string, script_logical_name: octet-string, script_selector: long-unsigned } </pre>

day_profile_table cont'd	Where: – day_id is a user defined identifier, identifying the current day_profile; – start_time: defines the time when the script is to be executed (no wildcards); the format follows the rules set in 4.6.1 for <i>time</i>; – script_logical_name: defines the <i>logical_name</i> of the “Script table” object; – script_selector: defines the script_identifier of the script to be executed.
activate_passive_calendar_time	Defines the time when the object itself calls the specific method <i>activate_passive_calendar</i> . A definition with "not specified" notation in all fields of the attribute will deactivate this automatism. Partial "not specified" notation in just some fields of date and time are not allowed. octet-string, formatted as specified in 4.6.1 for <i>date-time</i> .

Method description

activate_passive_calendar (data)	This method copies all attributes called ..._passive to the corresponding attributes called ..._active. data::= integer (0)

5.4.6 Register monitor (class_id = 21, version = 0)

This IC allows modelling the function of monitoring of values modelled by “Data”, “Register”, “Extended register” or “Demand register” objects. It allows specifying thresholds, the value monitored, and a set of scripts (see 5.4.2) that are executed when the value monitored crosses a threshold.

The IC “Register monitor” requires an instantiation of the IC “Script table” in the same logical device.

Register monitor		0...n	class_id = 21, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	thresholds (static)	array				x + 0x08
3.	monitored_value (static)	value_definition				x + 0x10
4.	actions (static)	array			0	x + 0x18
Specific methods		m/o				

Attribute description

logical_name	Identifies the “Register monitor” object instance. See 6.2.13 and 6.3.10.
thresholds	Provides the threshold values to which the attribute of the referenced register is compared. array threshold threshold: The threshold is of the same type as the monitored attribute of the referenced object.

monitored_value	Defines which attribute of an object is to be monitored. Only values with simple data types are allowed. <pre>value_definition ::= structure { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer }</pre>
actions	Defines the scripts to be executed when the monitored attribute of the referenced object crosses the corresponding threshold. The attribute <i>actions</i> has exactly the same number of elements as the attribute <i>thresholds</i>. The ordering of the action_items corresponds to the ordering of the thresholds (see above). <pre>array action_set action_set ::= structure { action_up: action_item, action_down: action_item }</pre> Where: – action_up defines the action when the attribute value of the monitored register crosses the threshold in the upwards direction; – action_down defines the action when the attribute value of the monitored register crosses the threshold in the downwards direction. <pre>action_item ::= structure { script_logical_name: octet-string, script_selector: long-unsigned }</pre>

5.4.7 Single action schedule (class_id = 22, version = 0)

This IC allows modelling the execution of periodic actions within a meter. Such actions are not necessarily linked to tariffication (see “Activity calendar” or “Schedule”).

Single action schedule		0...n	class_id = 22, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	executed_script (static)	script				x + 0x08
3.	type (static)	enum				x + 0x10
4.	execution_time (static)	array				x + 0x18
Specific methods		m/o				

Attribute description

logical_name Identifies the “Single action schedule” object instance. See 6.2.12.

executed_script Contains the logical name of the “Script table” object and the script selector of the script to be executed.

script::= structure

```
{
    script_logical_name: octet-string,
    script_selector: long-unsigned
}
```

Script_logical_name and script_selector define the script to be executed.

type enum:

- (1) size of execution_time = 1; wildcard in *date* allowed,
- (2) size of execution_time = *n*; all *time* values are the same, wildcards in *date* not allowed,
- (3) size of execution_time = *n*; all *time* values are the same, wildcards in *date* are allowed,
- (4) size of execution_time = *n*; *time* values may be different, wildcards in *date* not allowed,
- (5) size of execution_time = *n*; *time* values may be different, wildcards in *date* are allowed

execution_time Specifies the time and the date when the script is executed.

array execution_time_date

execution_time_date::= structure

```
{
    time: octet-string,
    date: octet-string
}
```

The two octet-strings contain *time* and *date*, in this order; *time* and *date* are formatted as specified in 4.6.1. Hundredths of seconds shall be zero.

5.4.8 Disconnect control (class_id = 70, version = 0)

Instances of the “Disconnect control” IC manage an internal or external disconnect unit of the meter (e.g. electricity breaker, gas valve) in order to connect or disconnect – partly or entirely – the premises of the consumer to / from the supply. The state diagram and the possible state transitions are shown in Figure 14.

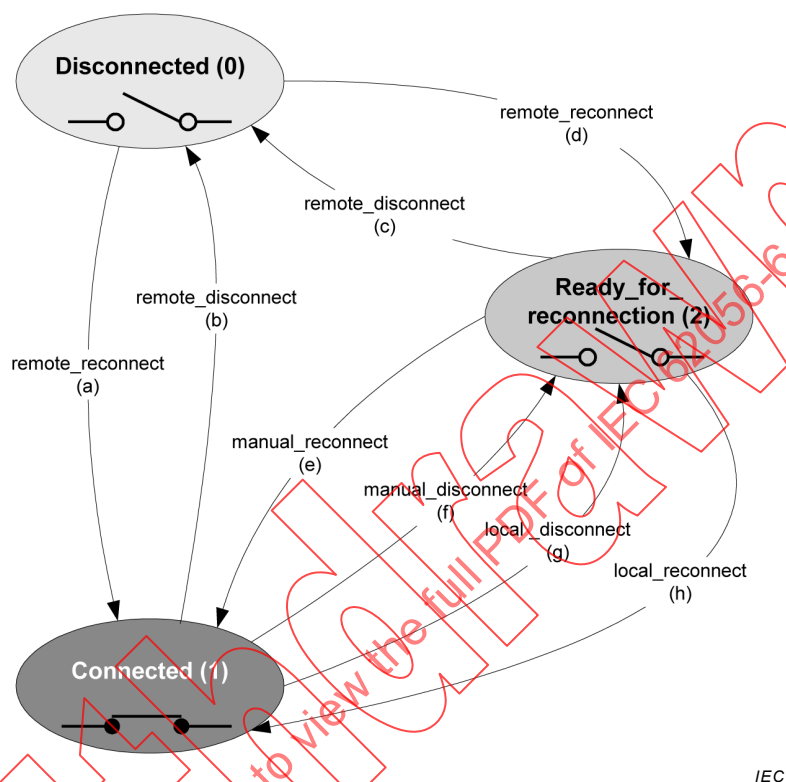


Figure 14 – State diagram of the Disconnect control IC

Disconnect and reconnect can be requested:

- Remotely, via a communication channel: *remote_disconnect*, *remote_reconnect*;
- Manually, using e.g. a push button: *manual_disconnect*, *manual_reconnect*;
- Locally, by a function of the meter, e.g. limiter, prepayment: *local_disconnect*, *local_reconnect*.

The states and state transitions of the Disconnect control IC are shown in Table 9. The possible state transitions depend on the control mode. The Disconnect control object does not feature a memory, i.e. any commands are executed immediately.

To define the behaviour of the disconnect control object for each trigger, the control mode shall be set.

Table 9 – Disconnect control IC – states and state transitions

States		
State number	State name	State description
0	Disconnected	The output_state is set to FALSE and the consumer is disconnected.
1	Connected	The output_state is set to TRUE and the consumer is connected.
2	Ready_for_reconnection	The output_state is set to FALSE and the consumer is disconnected.
State transitions		
Transition	Transition name	State description
a	remote_reconnect	Moves the “Disconnect control” object from the Disconnected (0) state directly to the Connected (1) state without manual intervention.
b	remote_disconnect	Moves the “Disconnect control” object from the Connected (1) state to the Disconnected (0) state.
c	remote_disconnect	Moves the “Disconnect control” object from the Ready_for_reconnection (2) state to the Disconnected (0) state.
d	remote_reconnect	Moves the “Disconnect control” object from the Disconnected (0) state to the Ready_for_reconnection (2) state. From this state, it is possible to move to the Connected (2) state via the manual_reconnect transition (e) or local_reconnect transition (h).
e	manual_reconnect	Moves the “Disconnect control” object from the Ready_for_connection (2) state to the Connected (1) state.
f	manual_disconnect	Moves the “Disconnect control” object from the Connected (1) state to the Ready_for_connection (2) state. From this state, it is possible to move back to the Connected (2) state via the manual_reconnect transition (e) or local_reconnect transition (h).
g	local_disconnect	Moves the “Disconnect control” object from the Connected (1) state to the Ready_for_connection (2) state. From this state, it is possible to move back to the Connected (2) state via the manual_reconnect transition (e) or local_reconnect transition (h). NOTE 1 Transitions f) and g) are essentially the same, but their trigger is different.
h	local_reconnect	Moves the “Disconnect control” object from the Ready_for_connection (2) state to the Connected (1) state NOTE 2 Transitions e) and h) are essentially the same, but their trigger is different.

Disconnect control		0...n	class_id = 70, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	output_state (dyn.)	boolean				x + 0x08
3.	control_state (dyn.)	enum				x + 0x10
4.	control_mode (static)	enum				x + 0x18
Specific methods		m/o				
1.	remote_disconnect()	m			x + 0x20	
2.	remote_reconnect()	m			x + 0x28	

Attribute description

logical_name Identifies the “Disconnect control” object instance. See 6.2.20 and 6.2.37.

output_state Shows the actual physical state of the device connection the supply.
 boolean: TRUE = (BOOLEAN 1) Connected,
 FALSE = (BOOLEAN 0) Disconnected

NOTE 1 In electricity metering, the supply is connected when the disconnecter device is closed.

NOTE 2 In gas and water metering, the supply is connected when the valve is open.

control_state Shows the internal state of the disconnect control object.
 enum: (0) Disconnected,
 (1) Connected,
 (2) Ready_for_reconnection

control_mode Configures the behaviour of the disconnect control object for all triggers, i.e. the possible state transitions.

control_ mode	Disconnection				Reconnection			
	Remote		Manual	Local	Remote		Manual	Local
enum:	(b)	(c)	(f)	(g)	(a)	(d)	(e)	(h)
(0)	–	–	–	–	–	–	–	–
(1)	x	x	x	x	–	x	x	–
(2)	x	x	x	x	x	–	x	–
(3)	x	x	–	x	–	x	x	–
(4)	x	x	–	x	x	–	x	–
(5)	x	x	x	x	–	x	x	x
(6)	x	x	–	x	–	x	x	X

NOTE 3 In Mode (0) the disconnect control object is always in 'connected' state.

NOTE 4 Local disconnection is always possible unless the corresponding trigger is inhibited.

Method description

remote_disconnect (data) Forces the “Disconnect control” object into 'disconnected' state if remote disconnection is enabled (control mode > 0).
 data::= integer (0)

remote_reconnect (data) Forces the “Disconnect control” object into the 'ready_for_reconnection' state if a direct remote reconnection is disabled (control_mode = 1, 3, 5, 6).

Forces the “Disconnect control” object into the 'connected' state if a direct remote reconnection is enabled (control_mode = 2, 4).

data::= integer (0)

5.4.9 Limiter (class_id = 71, version = 0)

Instances of the “Limiter” IC allow defining a set of actions that are executed when the value of a *value* attribute of a monitored object “Data”, “Register”, “Extended Register”, “Demand Register”, etc., crosses the threshold value for at least minimal duration time.

The threshold value can be normal or emergency threshold. The *emergency threshold* is activated via the *emergency_profile* defined by *emergency profile id*, *emergency activation time*, and *emergency duration*. The *emergency profile id* element is matched to an *emergency profile group id*: this mechanism enables the activation of the emergency threshold only for a specific emergency group.

Limiter		0...n	class_id = 71, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. monitored_value	(static)	value_definition				x + 0x08
3. threshold_active	(dyn.)	threshold				x + 0x10
4. threshold_normal	(static)	threshold				x + 0x18
5. threshold_emergency	(static)	threshold				x + 0x20
6. min_over_threshold_duration	(static)	double-long-unsigned				x + 0x28
7. min_under_threshold_duration	(static)	double-long-unsigned				x + 0x30
8. emergency_profile	(static)	emergency_profile				x + 0x38
9. emergency_profile_group_id_list	(static)	array				x + 0x40
10. emergency_profile_active	(dyn.)	boolean				x + 0x48
11. actions	(static)	action				x + 0x50
Specific methods		m/o				

Attribute description

logical_name	Identifies the “Limiter” object instance. See 6.2.15.
monitored_value	Defines an attribute of an object to be monitored. Only attributes with simple data types are allowed. value_definition::= structure <pre> { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer }</pre>
threshold_active	Provides the active threshold value to which the attribute monitored is compared. threshold: The threshold is of the same type as the attribute monitored
threshold_normal	Provides the threshold value to which the attribute monitored is compared when in normal operation. threshold: see above
threshold_emergency	Provides the threshold value to which the attribute monitored is compared when an emergency profile is active. threshold: see above
min_over_threshold_duration	Defines minimal over threshold duration in seconds required to execute the over threshold action.
min_under_threshold_duration	Defines minimal under threshold duration in seconds required to execute the under threshold action.

emergency_profile	An <i>emergency_profile</i> is defined by three elements: <i>emergency_profile_id</i>, <i>emergency_activation_time</i>, <i>emergency_duration</i>. An emergency profile is activated if the <i>emergency_profile_id</i> element matches one of the elements on the <i>emergency_profile_group_id_list</i>, and time matches the <i>emergency_activation_time</i> and <i>emergency_duration</i> element: <i>emergency_profile</i>::= structure <pre>{ emergency_profile_id: long-unsigned, emergency_activation_time: octet-string, emergency_duration: double-long-unsigned }</pre> Where: – the <i>emergency_activation_time</i> element defines the date and time when the <i>emergency_profile</i> is activated. The octet-string is encoded as specified in 4.6.1 for <i>date-time</i>, – the <i>emergency_duration</i> element defines the duration in seconds, for which the <i>emergency_profile</i> is activated. When an emergency profile is active, the <i>emergency_profile_active</i> attribute is set to TRUE.
emergency_profile_group_id_list	Defines a list of group ids of the emergency profile. The emergency profile can be activated only if <i>emergency_profile_id</i> element of the <i>emergency_profile</i> attribute matches one of the elements on the <i>emergency_profile_group_id_list</i>: array <i>emergency_profile_group_id</i> <i>emergency_profile_group_id</i>::= long-unsigned
emergency_profile_active	Indicates that the <i>emergency_profile</i> is active.
actions	Defines the scripts to be executed when the monitored value crosses the threshold for minimal duration time. <i>action</i>::= structure <pre>{ action_over_threshold: action_item, action_under_threshold: action_item }</pre> Where: – <i>action_over_threshold</i> defines the action when the value of the attribute monitored crosses the threshold in upwards direction and remains over threshold for minimal over threshold duration time; – <i>action_under_threshold</i> defines the action when the value of the attribute monitored crosses the threshold in the downwards direction and remains under threshold for minimal under threshold duration time. <i>action_item</i>::= structure <pre>{ script_logical_name: octet-string, script_selector: long-unsigned }</pre>

5.4.10 Parameter monitor (class_id = 65, version = 0)

Instances of the “Parameter monitor” IC monitor a list of COSEM object attributes holding parameters.

The parameters can be changed as usual. If the value of an attribute changes and this attribute is present in the *parameter_list* attribute, the identifier and the value of that attribute is automatically captured to the *changed_parameter* attribute. The time when the change of the parameter occurred is captured in the *capture_time* attribute. These attributes may be captured then by a “Profile generic” object. In this way, a log of all parameter changes can be built. For the OBIS code of the Parameter monitor log objects, see IEC 62056-6-1:2015, Clause 6.5.

NOTE 1 In the case of simultaneous or quasi simultaneous parameter changes the order of capturing and logging the changed parameters has to be managed by the application.

Several “Parameter monitor” objects and corresponding “Profile generic” objects can be instantiated to manage a number of parameter groups. The link between the “Parameter monitor” object and the corresponding “Profile generic” object is via the *capture_object* attribute of the “Profile generic” object.

NOTE 2 As the various parameters may be of different type and length, the entries in the profile column holding the parameters will be also of different type and length. This can be managed for example by capturing different kind of parameters into different Parameter list “Profile generic” objects and parameter logs.

NOTE 3 The “Profile generic” object holding the parameter change log may capture other suitable object attributes, like the *time* attribute of the “Clock” object, and any other relevant values.

Parameter monitor	0...n	class_id = 65, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. changed_parameter	structure				x + 0x08
3. capture_time	date-time				x + 0x10
4. parameter_list	array				x + 0x18
Specific methods	<i>m/o</i>				
1. add_parameter (data)	o				x + 0x20
2. delete_parameter (data)	o				x + 0x28

Attribute description

logical_name	Identifies the “Parameter monitor” object instance. See 6.2.14.
changed_parameter	Holds the identifier and the value of the most recently changed parameter. <pre> structure { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer, attribute_value: CHOICE -- CHOICE as specified in the “Data” interface class } </pre>
capture_time	Provides data and time information showing when the value of the <i>changed_parameter</i> attribute has been captured. <i>date-time</i> is formatted as specified in 4.6.1.

parameter_list	Holds the list of parameters managed by a given instance of the “Parameter monitor” IC. <code>parameter_list ::= array parameter_list_element</code> <code>parameter_list_element ::= structure</code> <pre> { class_id: long-unsigned, logical_name: octet_string, attribute_index: integer } </pre> NOTE 4 The list of parameters monitored may be changed by using the <i>add_parameter</i> or <i>delete_parameter</i> methods or writing this attribute.
Method description	
add_parameter (data)	Adds one parameter to the <i>parameter_list</i>. <code>data ::= parameter_list_element</code> NOTE 5 A parameter can be logged as soon as it is added to the list. Adding an element to the list does not affect the <i>buffer</i> of the “Profile generic” object capturing the <i>changed_parameter</i> attribute.
delete_parameter (data)	Deletes one parameter from the <i>parameter_list</i>. <code>data ::= parameter_list_element</code> NOTE 6 When a parameter is deleted from the parameter list, its changes will not be logged any more. Removing an element from the list does not affect the <i>buffer</i> of the “Profile generic” object capturing the <i>changed_parameter</i> attribute.

5.4.11 Sensor manager interface class

5.4.11.1 General

NOTE This interface class is used mainly in metering energy types other than electricity.

Most measuring instruments under the scope of the MID operate with dedicated sensors (transducers and transmitters) connected to the processing unit. These sensors have to be permanently supervised concerning their functioning and limits to fulfil the metrological requirements for subsequent calculation of monetary values.

In addition, the measured values have to be monitored. These values may be related to a physical quantity – raw values of voltage, current, resistance, frequency, digital output – provided by the sensor, and the measured quantities resulting from the processing of the information provided by the sensor.

It is necessary to monitor and often to log the relevant values in order to obtain diagnostic information that allows:

- the identification of the sensor device;
- the connection and the sealing status of the sensor;
- the configuration of the sensors;
- the monitoring of the operation of the sensors;
- the monitoring of the result of the processing.

The “Sensor manager” interface class allows managing detailed information related to a sensor by a single object.

For simpler sensors / devices, already existing COSEM objects – identifying the sensors, holding measurement values and monitoring those measurement values – can be used.

5.4.11.2 Sensor manager (class_id = 67, version = 0)

Instances of the “Sensor manager” IC manage complex information related to sensors. They also allow monitoring the raw data and the processed value, derived by processing the raw-data using appropriate algorithms as required by the particular application. This IC includes a number of functions:

- nameplate data of the sensor and site information (attributes 2 to 6);
- an “Extended register” function for the *raw-value* (attributes 7 to 10);
- a “Register monitor” function for the *raw-value* (attributes 11-12);

NOTE 1 Not every raw data (e.g. the voltage output of a pressure sensor) has its own OBIS code / object. This is the reason to include raw data in the Sensor manager class.

- a “Register monitor” function for the *processed_value* (attributes 13 to 15).

NOTE 2 Not all “modules” are necessarily present. The attributes not used are possibly not implemented or not accessible.

Sensor manager		0...n	class_id = 67, version = 0			
Attribute (s)		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. serial_number	(dyn.)	octet-string				x + 0x08
3. metrological_identification	(dyn.)	octet-string				x + 0x10
4. output_type	(dyn.)	enum				x + 0x18
5. adjustment_method	(dyn.)	octet-string				x + 0x20
6. sealing_method	(dyn.)	enum				x + 0x28
7. raw_value	(dyn.)	CHOICE				x + 0x30
8. scaler_unit	(dyn.)	structure				x + 0x38
9. status	(dyn.)	CHOICE				x + 0x40
10. capture_time	(dyn.)	date-time				x + 0x48
11. raw_value_thresholds	(dyn.)	array				x + 0x50
12. raw_value_actions	(dyn.)	array				x + 0x58
13. processed_value	(dyn.)	processed_value_definition				x + 0x60
14. processed_value_thresholds	(dyn.)	array				x + 0x68
15. processed_value_actions	(dyn.)	array				x + 0x70
Specific methods		m/o				
1. reset (data)		o				x + 0x80

Attribute description

logical_name Identifies the “Sensor manager” object instance. See DLMS UA 1000-1, the “Blue Book” Ed. 11.0: 2013 6.4.4.

serial_number Identifies the sensor (->site information).

NOTE 3 For simple sensors, the serial number may be held by “Data” objects with appropriate OBIS codes if this is the only information required. See Gas related general purpose objects, Configuration objects in DLMS UA 1000-1, the “Blue Book” Ed. 11.0: 2013 Table 77.

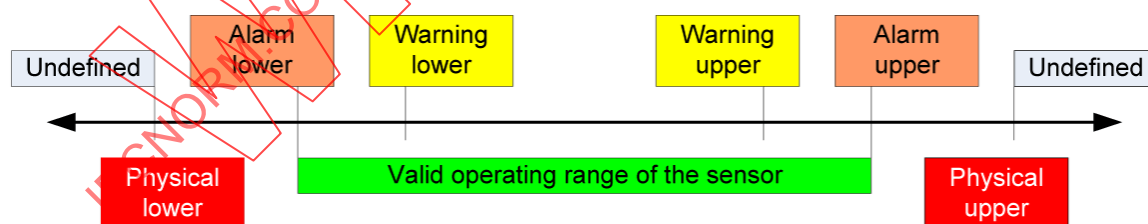
metrological_identification Describes metrologically relevant information of the sensor, e.g. metrological identifier, calibration date.

output_type	describes the physical output of the sensor (->site information) enum: (0) not specified, (1) 4–20 mA, (2) 0–20 mA (3) 0–5 V, (4) 0–10 V, (5) Pt100, (6) Pt500, (7) Pt1000, (8) HART 128 manufacturer specific All other values are reserved.
adjustment_method	Describes the sensor adjustment method, e.g. by 3-Measuring-point-equation.
sealing_method	Type of seals applied to the sensor. enum: (0) none, (1) mechanical (e.g. wire or protective sticker); (2) electronic (e.g. contact), (3) software (e.g. password protection)
raw_value	Physical value from the sensor (e.g. voltage from a pressure to voltage converter). For the possible data type choices, see the “Register” IC in 5.2.2.
scaler_unit	The scaler and the unit of the <i>raw_value</i> . For the definition of <i>scaler_unit</i> , see the “Register” IC in 5.2.2.
status	Status of last <i>raw_value</i> captured (for the definition of status, see the “Extended register” IC. The status keeps information like: - sensor failure, - sensor activated / inactivated. The meaning of the elements of the <i>status</i> shall be provided for each “Sensor manager” object instance.
capture_time	Provides a “Sensor manager” object specific date and time information showing when the value of the attribute <i>raw_value</i> has been captured. <i>date_time</i> is formatted as specified in 4.6.1. For the possible data type choices, see the “Extended register” interface class in 5.2.3.
raw_value_thresholds	Provides the threshold values to which the <i>raw_value</i> attribute is compared. array threshold threshold: The threshold is of the same type as the raw-data.
raw_value_actions	Defines the scripts to be executed when the <i>raw_value</i> crosses the corresponding threshold. The attribute <i>raw_value_actions</i> has exactly the same number of elements as the attribute <i>raw_value_thresholds</i> . The ordering of the <i>action_items</i> corresponds to the ordering of the thresholds. For the specification of actions, see the <i>Register monitor</i> IC in 5.4.6.

processed_value	References the attribute holding the processed value of the raw data provided by the sensor. processed_value_definition::= structure <pre> { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer } </pre> Note that more than one “Sensor manager” objects and processed value objects may belong to the same sensor.
processed_value_thresholds	Provides the threshold values to which the processed value is compared. array threshold threshold: The threshold is of the same type as the value processed. (referenced by the <i>processed-value</i> attribute).
processed_value_actions	Defines the scripts to be executed when the processed value crosses the corresponding threshold. The attribute <i>processed_value_actions</i> has exactly the same number of elements as the attribute <i>processed_value_thresholds</i>. The ordering of the <i>action_items</i> corresponds to the ordering of the thresholds. For the specification of actions, see the <i>Register monitor</i> IC in 5.4.6.
Method description	
reset (data)	This method resets the <i>raw_value</i> to the default value. The default value is an instance specific constant. The attribute <i>capture_time</i> is set to the time of the reset execution. data::= integer (0)

5.4.11.3 Example for absolute pressure sensor

Figure 15 illustrates the definition of relevant upper and lower thresholds.



IEC

Figure 15 – Definition of upper and lower thresholds

Table 10 and Table 11 show examples of the various thresholds and the actions performed when the thresholds are crossed.

Table 10 – Explicit presentation of threshold value arrays

Threshold	Physical lower	Physical upper	Alarm lower	Alarm upper	Warning lower	Warning upper
Value	1,0	5,5	1,2	5,0	1,4	4,8
scaler_unit	1, Volt	1, Volt	1, bar	1, bar	1, bar	1, bar

Table 11 – Explicit presentation of action_sets

action_set	Physical lower	Physical upper	Alarm lower	Alarm upper	Warning lower	Warning upper
action_up	clr_phy_alarm_bit	set_phy_alarm_bit	clear_alarm_bit	set_alarm_bit	clear_warn_bit	set_warn_bit
action_down	set_phy_alarm_bit	clr_phy_alarm_bit	set_alarm_bit	clear_alarm_bit	set_warn_bit	clear_warn_bit

5.5 Interface classes for setting up data exchange via local ports and modems

5.5.1 IEC local port setup (class_id = 19, version = 1)

This IC allows modelling the configuration of communication ports using the protocols specified in IEC 62056-21:2002. Several ports can be configured.

IEC local port setup		0...n	class_id = 19, version = 1			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	default_mode (static)	enum				x + 0x08
3.	default_baud (static)	enum				x + 0x10
4.	prop_baud (static)	enum				x + 0x18
5.	response_time (static)	enum				x + 0x20
6.	device_addr (static)	octet-string				x + 0x28
7.	pass_p1 (static)	octet-string				x + 0x30
8.	pass_p2 (static)	octet-string				x + 0x38
9.	pass_w5 (static)	octet-string				x + 0x40
Specific methods		m/o				

Attribute description

logical_name Identifies the “IEC local port setup” object instance. See 6.2.16.

default_mode Defines the protocol used by the meter on the port.
enum:
(0) protocol according to IEC 62056-21:2002 (modes A...E),
(1) protocol according to IEC 62056-46:2007. Using this enumeration value all other attributes of this IC are not applicable,
(2) protocol not specified. Using this enumeration value, attribute 4), *prop_baud* is used for setting the communication speed on the port. All other attributes are not applicable.

default_baud	Defines the baud rate for the opening sequence. enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud
prop_baud	Defines the baud rate to be proposed by the meter. enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud
response_time	Defines the minimum time between the reception of a request (end of request telegram) and the transmission of the response (begin of response telegram). enum: (0) 20 ms, (1) 200 ms
device_addr	Device address according to IEC 62056-21:2002.
pass_p1	Password 1 according to IEC 62056-21:2002.
pass_p2	Password 2 according to IEC 62056-21:2002.
pass_w5	Password W5 reserved for national applications.

5.5.2 IEC HDLC setup (class_id = 23, version = 1)

This IC allows modelling and configuring communication channels according to IEC 62056-46:2007.

Several communication channels can be configured.

IEC HDLC setup		0...n	class_id = 23, version = 1			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. comm_speed	(static)	enum	0	9	5	x + 0x08
3. window_size_transmit	(static)	unsigned	1	7	1	x + 0x10
4. window_size_receive	(static)	unsigned	1	7	1	x + 0x18
5. max_info_field_length_transmit	(static)	long-unsigned	32	2030	128	x + 0x20
6. max_info_field_length_receive	(static)	long-unsigned	32	2030	128	x + 0x28
7. inter_octet_time_out	(static)	long-unsigned	20	6000	25	x + 0x30
8. inactivity_time_out	(static)	long-unsigned	0		120	x + 0x38
9. device_address	(static)	long-unsigned	0x0010	0x3FF D		x + 0x40
Specific methods		m/o				

NOTE 1 The maximum value of the attributes *max_info_field_length_transmit* and *max_info_field_length_receive* has been increased from 128 to 2030 for efficiency reasons.

NOTE 2 A *max_info_field_length_receive* of 128 bytes is needed to ensure a minimal performance.

NOTE 3 The maximum value of the *inter-octet-time-out* attribute has been increased from 1000 ms to 6000 ms in order to allow using communication media, where long delays may occur. The default value has been changed to 25 ms to align with 6.4.4.3.4 of IEC 62056-46:2007.

Attribute description											
logical_name	Identifies the “IEC HDLC setup” object instance. See 6.2.18.										
comm_speed	The communication speed supported by the corresponding port. enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud This communication speed can be overridden if the HDLC mode of a device is entered through a special mode of another protocol.										
window_size_transmit	The maximum number of frames that a device or system can transmit before it needs to receive an acknowledgement from a corresponding station. During logon, other values can be negotiated.										
window_size_receive	The maximum number of frames that a device or system can receive before it needs to transmit an acknowledgement to the corresponding station. During logon, other values can be negotiated.										
max_info_length_transmit	The maximum information field length that a device can transmit. During logon, a smaller value can be negotiated.										
max_info_length_receive	The maximum information field length that a device can receive. During logon, a smaller value can be negotiated.										
inter_octet_time_out	Defines the time, expressed in milliseconds, over which, when no character is received from the primary station, the device will treat the already received data as a complete frame.										
inactivity_time_out	Defines the time, expressed in seconds over which, when no frame is received from the primary station, the device will process a disconnection. When this value is set to 0, this means that the <i>inactivity_time_out</i> is not operational.										
device_address	Contains the physical device address of a device. In the case of one byte addressing: <table> <tr> <td>0x00</td><td>NO_STATION Address,</td></tr> <tr> <td>0x01...0x0F</td><td>Reserved for future use,</td></tr> <tr> <td>0x10...0x7D</td><td>Usable address space,</td></tr> <tr> <td>0x7E</td><td>‘CALLING’ device address,</td></tr> <tr> <td>0x7F</td><td>Broadcast address</td></tr> </table>	0x00	NO_STATION Address,	0x01...0x0F	Reserved for future use,	0x10...0x7D	Usable address space,	0x7E	‘CALLING’ device address,	0x7F	Broadcast address
0x00	NO_STATION Address,										
0x01...0x0F	Reserved for future use,										
0x10...0x7D	Usable address space,										
0x7E	‘CALLING’ device address,										
0x7F	Broadcast address										

device_address	In the case of two byte addressing:	
cont'd	0x0000	NO_STATION address,
	0x0001...0x000F	Reserved for future use,
	0x0010...0x3FFD	Usable address space,
	0x3FFE	'CALLING' physical device address,
	0x3FFF	Broadcast address

5.5.3 IEC twisted pair (1) setup (class_id = 24, version = 1)

5.5.3.1 Overview

The communication medium *twisted pair with carrier signalling* is widely used in metering. The main advantages of using this medium are the ease of installation and the reliability of communications due to carrier signalling. This medium can be used:

- between Local Network Access Points (LNAPs) and metering end devices (M interface);
- between Local Network Access Points (LNAPs) and Neighbourhood Network Access Points (NNAPs); and
- for direct connection between a HHU and the metering end device.

IEC 62056-3-1:2013 specifies three communication profiles using the medium twisted pair with carrier signalling:

- without DLMS;
- with DLMS; and
- with DLMS/COSEM.

IEC 62056-31:1999 supports only the first two profiles.

The new, DLMS/COSEM profile introduces a Support Manager Layer entity performing the initialisation of the bus, discovery management, alarm management and communication speed negotiation. It also allows higher baud rates up to 9 600 Bd. The Transport Layer supports segmentation and reassembly.

The IC "IEC Twisted pair (1) set up" (class_id = 24, version = 0) supports the first two communication profiles specified in IEC 62056-31:1999.

The new version 1 supports the DLMS/COSEM profile. With its introduction, the use of version 0 is deprecated.

The use of the communication profiles specified in IEC 62056-3-1:2013 requires using the registration services provided by the Euridis Association: www.euridis.org.

The following COSEM interface objects are necessary to set up data exchange over the medium *Twisted pair with carrier signalling*:

- "IEC Twisted pair (1) setup": class_id = 24, version = 1;
- "MAC address": class_id = 43, version = 0;
- "Data": class_id = 1, version = 0.

For OBIS codes, see clause 6.2.19.

5.5.3.2 IEC twisted pair (1) setup (class_id = 24, version = 1)

Instances of this IC allow setting up data exchange over the medium *twisted pair with carrier signalling* as specified in IEC 62056-3-1:2013. Several communication channels can be configured.

IEC twisted pair (1) setup		0...n	class_id = 24, version = 1			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	mode (static)	enum	0		1	x + 0x08
3.	comm_speed (static)	enum	(2)	(7)	(2)	x + 0x10
4.	primary_address_list (static)	primary_address_list_type				x + 0x18
5.	tabi_list (static)	tabi_list_type				x + 0x20
Specific methods		m/o				

Attribute description

logical_name Identifies the “IEC twisted pair setup” object instance. See 6.2.19.

mode This attribute specifies the working mode of this interface
enum:
(0) inactive. The interface ignores all frames received,
(1) always active,
(2) to (127) reserved,
(128) to manufacturer specific.
(250)

comm_speed Holds the communication speed supported by the port.

enum:
(2) 1 200 baud,
(3) 2 400 baud,
(4) 4 800 baud,
(5) 9 600 baud,
(6) 19 200 baud,
(7) 38 400 baud

NOTE IEC 62056-3-1:2013 supports baud rates from 1 200 to 9 600.

primary_address_list Holds the list of Primary Station Addresses (ADP) for which each logical device of the real equipment (the secondary station) has been programmed. See IEC 62056-3-1:2013, 5.2.4.

primary_address_list_type::= array unsigned

tabi_list Represents the list of the TAB(i) for which the real equipment (the secondary station) has been programmed in the case of forgotten station call (see IEC 62056-3-1:2013).

When using IEC 62056-3-1 profile with DLMS/COSEM, the *tabi_list* attribute is made of only one element of value 0, the value used for the discovery process.

tabi_list_type::= array tabi_element

tabi_element: integer

5.5.3.3 MAC address

Secondary stations – typically metering end devices – hold a secondary station address (ADS). The length of the ADS shall be 6 octets and consists of the elements shown in Table 12. This address shall be worldwide unique and it shall assigned by the manufacturer to the secondary station. The ADS assigned is valid through the lifetime of the secondary station.

Table 12 – ADS address elements

Elements of ADS	Description	Length (byte)	Type	Range
manufacturer_id	Assigned upon request by the Euridis Association to the manufacturer. It shall be used in all devices using the <i>Twisted pair with carrier signalling</i> medium and made by this manufacturer.	1	BCD	0-99
year_of_manufacture	Holds the year of manufacturing the device (the lower two numbers only).	1		0-99
equipment_id	Related to the type of equipment concerned and provides information on the functionality available. Like the manufacturer_id, it is assigned by the Euridis Association.	1		0-99
device_serial_number	The manufacturing number of the device assigned by the manufacturer. It should have the same value as the value held by the object Device ID 1, see IEC 62056-6-1:2015, Table 7..	3		000001-999999 000000 is reserved
EXAMPLE 031267123456: 03: ITRON International; 12: Year 2012; 67: Smart meter LINKY Single phase 123456: Serial number beginning each year from 000001.				

5.5.3.4 Fatal error register

Each device implementing the DLMS/COSEM communication profile specified in IEC 62056-3-1:2013 shall provide an error register holding the result of the last communication with the primary station. The structure of the fatal error register shall be as specified in Table 13.

Table 13 – Fatal error register

Ref	Name	Description
Bit 0	EP-3F	Transmission error. The time out TOE is elapsed without the byte being sent, leading to a non-ability to send the remaining part of the frame.
Bit 1	EP-4F	Reception error. The number of bytes received is higher than the maximum expected.
Bit 2	EP-5F	Expiry of TARSO wake-up while receiving an RSO frame. Not relevant for secondary station (server); concerns the primary station (client) only.
Bit 3	EL-1F	Alarm indication received during an association. Not relevant. Concern the primary station (client) only.
Bit 4	EL-2F	Incorrect response from the Secondary Station after MaxRetry repeated transmissions of a request.
Bit 5	EA-1F	Incorrect TAB from the server. Not relevant for secondary station (server); concerns the primary station (client) only.
Bit 6	EA-2F	Authentication error on the data received from the server. Not relevant for secondary station (server); concerns the primary station (client) only.
Bit 7	EA-3F	Authentication error detected by the secondary station.

5.5.4 Modem configuration (class_id = 27, version = 1)

This IC allows modelling the configuration and initialisation of modems used for data transfer from/to a device. Several modems can be configured.

Modem configuration	0...n	class_id = 27, version = 1			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. comm_speed (static)	enum	0	9	5	x + 0x08
3. initialization_string (static)	array				x + 0x10
4. modem_profile (static)	array				x + 0x18
Specific methods	m/o				

Attribute description

logical_name Identifies the “Modem configuration” object instance. See 6.2.6.

comm_speed The communication speed between the device and the modem, not necessarily the communication speed on the WAN.

enum:

- (0) 300 baud,
- (1) 600 baud,
- (2) 1 200 baud,
- (3) 2 400 baud,
- (4) 4 800 baud,
- (5) 9 600 baud,
- (6) 19 200 baud,
- (7) 38 400 baud,
- (8) 57 600 baud,
- (9) 115 200 baud

initialization_string Contains all the necessary initialization commands to be sent to the modem in order to configure it properly. This may include the configuration of special modem features.

array initialization_string_element

initialization_string_element ::= structure

```

{
    request:          octet-string,
    response:         octet-string,
    delay_after_response: long-unsigned
}

```

If the array contains more than one initialization_string_element, the requests are sent in a sequence. The next request is sent after the expected response matching the previous request and waiting a delay_after_response time [ms], to allow the modem to execute the request.

NOTE It is assumed that the modem is pre-configured so that it accepts the initialization_string. If no initialization is needed, the initialization string is empty.

modem_profile Defines the mapping from Hayes standard commands/responses to modem specific strings.

array modem_profile_element

modem_profile_element: octet-string

modem_profile cont'd	The modem_profile array shall contain the corresponding strings for the modem used in following order: Element 0: OK, Element 1: CONNECT, Element 2: RING, Element 3: NO CARRIER, Element 4: ERROR, Element 5: CONNECT 1 200, Element 6: NO DIAL TONE, Element 7: BUSY, Element 8: NO ANSWER, Element 9: CONNECT 600, Element 10: CONNECT 2 400, Element 11: CONNECT 4 800, Element 12: CONNECT 9 600, Element 13: CONNECT 14 400, Element 14: CONNECT 28 800, Element 15: CONNECT 33 600, Element 16: CONNECT 56 000

5.5.5 Auto answer (class_id = 28, version = 2)

NOTE 1 Version 1 of the Auto answer class was an interim version.

Version 0 of the Auto answer class models how the device handles incoming calls to request the connection of the modem.

In version 2, new capabilities are added to manage wake-up requests that may be in the form of a wake-up call or a wake-up message e.g. an (empty) SMS message. After a successful wake-up request, the device connects to the network. See also Annex A.

For both functions, additional security is provided by adding the possibility of checking the calling number: calls or messages are accepted only from a pre-defined list of callers. This feature requires the presence of a calling line identification (CLI) service in the communication network used.

NOTE 2 The wake-up process is fully decoupled from AL services, i.e. a wake-up message cannot contain any xDLMS service requests. This is to avoid creating a backdoor. xDLMS messages may be exchanged in SMS messages once the wake-up process is completed.

Auto answer		0...n	class_id = 28, version = 2			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	mode (static)	enum				x + 0x08
3.	listening_window (static)	array				x + 0x10
4.	status (dyn.)	enum				x + 0x18
5.	number_of_calls (static)	unsigned				x + 0x20
6.	number_of_rings (static)	nr_rings_type				x + 0x28
7.	list_of_allowed_callers (static)	array				x + 0x30
Specific methods		m/o				

Attribute description	
logical_name	Identifies the “Auto answer” object instance. See 6.2.6.
mode	Defines the working mode of the line when the device is auto answering. enum: (0) line dedicated to the device, (1) shared line management with a limited number of calls allowed. Once the number of calls is reached, the window status becomes inactive until the next start date, whatever the result of the call, (2) shared line management with a limited number of successful calls allowed. Once the number of successful communications is reached, the window status becomes inactive until the next start date, (3) currently no modem connected, (200...255) manufacturer specific modes
listening_window	Defines the time points when the communication window(s) become active (start_time) and inactive (end_time). The start_time implicitly defines the period. EXAMPLE When the day of month is not specified (equal to 0xFF) this means that we have a daily listening window management. Daily, monthly ...window management can be defined. array window_element <pre> window_element ::= structure { start_time: octet-string, end_time: octet-string } </pre> start_time and end_time are formatted as specified in 4.6.1 for <i>date-time</i>.
status	Here the status of the window is defined. enum: (0) Inactive: the device will manage no new incoming call. This status is automatically reset to Active when the next listening window starts, (1) Active: the device can answer to the next incoming call, (2) Locked: This value can be set automatically by the device or by a specific client when this client has completed its reading session and wants to give the line back to the customer before the end of the window duration. This status is automatically reset to Active when the next listening window starts.
number_of_calls	This number is the reference used in modes 1 and 2. When set to 0, this means there is no limit.

number_of_rings	Defines the number of rings before the meter connects the modem. Two cases are distinguished: The number of rings within the window defined by the attribute <i>listening_window</i> and the number of rings outside the <i>listening_window</i>. <pre>nr_rings_type ::= structure { nr_rings_in_window: unsigned (0 = no connection in the window), nr_rings_out_of_window: unsigned (0 = no connection outside the window) }</pre> If the number of rings inside and outside the window is the same, the modem always connects regardless of the settings of the <i>listening_window</i>.
list_of_allowed_callers	Contains an – optional – list of calling numbers which further limits the connectivity of the modem based on the calling number. It also controls the acceptance of wake-up calls or wake-up messages (e.g. SMS) from a calling number. This requires the presence of a calling line identification (CLI) service in the communication network used. <code>list_of_allowed_callers</code> ::= array <code>list_of_allowed_callers_element</code> <pre>list_of_allowed_callers_element ::= structure { caller_id: octet-string, call_type: enum }</pre> – the <code>caller_id</code> element holds a calling number from which calls or messages (e.g. SMS) are accepted. The wild-card characters '?' and '*' are supported. With '?' any single character matches, with '*' any character string matches. '*' can only be used at the beginning or at the end of a number, but neither in between nor alone. Example 1: "+994193500" = only calls from "+994193500" are accepted. Example 2: "+9941935?????" = calls from all numbers in the range of "+99419350000" to "+99419359999" are accepted. Example 3: "7777*" = calls from all numbers starting with "7777" are accepted. Example 4: "*9000" = calls from all numbers ending with "9000" are accepted. – the <code>call_type</code> element defines the purpose of the call, i.e. if it is a standard CSD call or a wake-up call / wake-up message.

list_of_allowed_callers

(continued)

enum:

(0) = normal CSD call; the modem only connects if the calling number matches an entry in the list. This is tested in addition to all other attributes, e.g. *number_of_rings*, *listening_windows*, etc.

(1) = wake-up request; calls or messages from this calling number are handled as wake-up requests. The wake-up request is processed immediately regardless of all other attributes like *number_of_rings* and *listening_window* (except if the calling number is also present in the list of normal CSD calls, see note 3 below).

The received message shall be completely empty; otherwise it is not treated as a wake-up message.

If the message contains a valid xDLMS APDU from a client in a pre-established AA, the corresponding xDLMS service is executed instead of processing the wake-up request.

If the message is not empty but does not contain any valid xDLMS APDU from a client in a pre-established Application Association then the server does not react.

For a call of type (1) the modem *does not* connect – independently of the outcome of the wake-up process.

For a call of both type (1) and type (0): see Note 3.

NOTE 3 If the same calling number is defined as initiator of a normal CSD call (type (0)) and as initiator of a wake-up request (type (1)) the following rule applies for the incoming calls: the wake-up request is only processed if the caller disconnects the line before the *number_of_rings* criterion (depending on *listening_window*) is reached. If the *number_of_rings* criterion is met then a modem connection is established.

The *number_of_rings* parameter shall be set large enough to allow the initiator of the call to control the behaviour of the receiver of the call without any knowledge of the time instances of the rings at the receiver's side.

NOTE 4 If the *list_of_allowed_callers* is empty (= array [0]) the auto answer function operates in type (0) = normal CSD call and the modem connects independently of the calling number.

5.5.6 Auto connect (class_id = 29, version = 2)

Version 1 of the “Auto connect” class models how the device performs auto dialling or sends messages using various services.

In version 2 new capabilities are added to model the connection of the device to a communication network. Network connection may be permanent, within a time window or on invocation of the connect method.

Auto connect		0...n	class_id = 29, version = 2			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	mode (static)	enum				x + 0x08
3.	repetitions (static)	unsigned				x + 0x10
4.	repetition_delay (static)	long-unsigned				x + 0x18
5.	calling_window (static)	array				x + 0x20
6.	destination_list (static)	array				x + 0x28
Specific methods		m/o				
1.	connect (data)	o				x + 0x30

Attribute description	
logical_name	Identifies the “Auto connect” object instance. See 6.2.6.
mode	Controls the auto connect functionality in terms of the timing, the message type and the infrastructure to be used. Modes (1) to (3) are dedicated to CSD services. Modes (4) to (6) are dedicated to sending specific messages using a specific infrastructure. Modes (101) to (104) apply to packet switched network connections only (e.g. GPRS). enum: (0) no auto connect; the device never connects, (1) auto dialling allowed anytime, the values defined in the <i>calling_window</i> are ignored, (2) auto dialling allowed within the validity time of the <i>calling_window</i>, (3) “regular” auto dialling allowed within the validity time of the <i>calling_window</i>; “alarm” initiated auto dialling allowed anytime, (4) SMS sending via Public Land Mobile Network (PLMN), (5) SMS sending via PSTN, (6) email sending, (7...99) reserved, (101) the device is permanently connected to the communication network, (102) the device is permanently connected to the communication network within the validity time of the calling window. The device is disconnected outside the calling window. No connection possible outside the calling window, (103) the device is permanently connected to the communication network within the validity time of the calling window. The device is disconnected outside the calling window but it connects to the communication network as soon as the connect method is invoked, (104) the device is usually disconnected. It connects to the communication network as soon as the connect method is invoked, (105...199) reserved, (200...255) manufacturer specific modes.
repetitions	The maximum number of retries in case of unsuccessful connection attempts.
repetition_delay	The time delay, expressed in seconds until an unsuccessful connection attempt can be repeated. repetition_delay = 0 means delay is not specified

calling_window	Contains the time points when the window becomes active (start_time), and inactive (end_time). The start_time implicitly defines the period. EXAMPLE When the day of month is not specified (equal to 0xFF) this means that the calling window is managed on a daily basis. Daily, monthly ...window management can be defined. <pre> array window_element window_element ::= structure { start_time: octet-string, end_time: octet-string } </pre> start_time and end_time are formatted as specified in 4.6.1 for <i>date-time</i>.
destination_list	Contains the list of destinations (for example phone numbers, email addresses or their combinations) where the message(s) have to be sent under certain conditions. The conditions and their link to the elements of the array are not defined here. <pre> array destination destination ::= octet-string </pre>

Method description

connect (data)	Initiates the connection process to the communication network according to the rules defined via the mode attribute. data ::= integer (0)

5.5.7 GPRS modem setup (class_id = 45, version = 0)

This IC allows setting up GPRS modems, by handling all data necessary data for modem management.

GPRS modem setup	0...n	class_id = 45, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. APN (static)	octet-string				x + 0x08
3. PIN_code (static)	long-unsigned				x + 0x10
4. quality_of_service (static)	structure				x + 0x18
Specific methods	<i>m/o</i>				

Attribute description

logical_name	Identifies the “GPRS modem setup” object instance. See 6.2.21.
APN	Defines the access point name of the network.
PIN_code	Holds the personal identification number.

quality_of_service Specifies the quality of service parameters. It is a structure of 2 elements:

- the first element defines the default or minimum characteristics of the network concerned. These parameters have to be set to best effort value;
- the second element defines the requested parameters.

quality_of_service ::= structure

```
{
    default:      qos_element,
    requested:    qos_element
}
```

qos_element ::= structure

```
{
    precedence:    unsigned,
    delay:         unsigned,
    reliability:   unsigned,
    peak throughput: unsigned,
    mean throughput: unsigned
}
```

5.5.8 GSM diagnostic (class_id = 47, version = 0)

The GSM/GPRS network is undergoing constant changes in terms of registration status, signal quality, etc. It is necessary to monitor and log the relevant parameters in order to obtain diagnostic information that allows identifying communication problems in the network.

An instance of the “GSM diagnostic” class stores parameters of the GSM/GPRS network necessary for analysing the operation of the network.

A GSM diagnostic “Profile generic” object is also available to capture the attributes of the GSM diagnostic object, see IEC 62056-6-1:2015, 6.5.

GSM diagnostic		0...n	class_id = 47, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	operator (dyn.)	visible-string				x + 0x08
3.	status (dyn.)	enum	0	255	0	x + 0x10
4.	cs_attachment (dyn.)	enum	0	255	0	x + 0x18
5.	ps_status (dyn.)	enum	0	255	0	x + 0x20
6.	cell_info (dyn.)	cell_info_type				x + 0x30
7.	adjacent_cells (dyn.)	array				x + 0x38
8.	capture_time (dyn.)	date-time				x + 0x40
Specific methods		m/o				

Attribute description

logical_name Identifies the “GSM Diagnostic” object instance. For logical names, see 6.2.21.

operator Holds the name of the network operator e.g. “YourNetOp”

status	Indicates the registration status of the modem. enum: (0) not registered, (1) registered, home network, (2) not registered, but MT is currently searching a new operator to register to, (3) registration denied, (4) unknown, (5) registered, roaming (6) ... (255) reserved
cs_attachment	Indicates the current circuit switched status. enum: (7) inactive, (8) incoming call, (9) active, (10) ... (255) reserved
ps_status	The <i>ps_status</i> value field indicates the packet switched status of the modem. enum: (0) inactive, (1) GPRS, (2) EDGE, (3) UMTS, (4) HSDPA, (5) ... (255) reserved
cell_info	Represents the cell information: cell_info_type := structure { cell_ID: long-unsigned, location_ID: long-unsigned, signal_quality: unsigned, ber: unsigned } Where: - cell_ID: Two-byte cell ID in hexadecimal format; - location_ID: Two-byte location area code (LAC) in hexadecimal format (e.g. "00C3" equals 195 in decimal); - signal_quality: Represents the signal quality: (0) –113 dBm or less, (1) –111 dBm, (2...30) –109...–53 dBm, (31) –51 or greater, (99) not known or not detectable; - ber: Bit error (BER) measurement in percent: (0...7) as RXQUAL_n values specified in ETSI GSM 05.08 8.2.4. (99) not known or not detectable.

adjacent_cells array adjacent_cell_info

```
adjacent_cell_info:= structure
{
    cell_ID:          long-unsigned,
    signal_quality:   unsigned
}
```

Where:

- cell_ID: Two-byte cell ID in hexadecimal format;
- signal_quality: Represents the signal quality:
 - (0) –113 dBm or less,
 - (1) –111 dBm,
 - (2...30) –109...–53 dBm,
 - (31) –51 or greater,
 - (99) not known or not detectable

capture_time Holds the date and time when the data have been last captured.
date-time is formatted as specified in 4.6.1.

5.6 Interface classes for setting up data exchange via M-Bus

5.6.1 M-Bus slave port setup (class_id = 25, version = 0)

NOTE The name of this IC has been changed from “M-BUS port setup” to “M-Bus slave port setup”, to indicate that it serves to set up data exchange when a COSEM server communicates with a COSEM client using wired M-Bus.

This IC allows modelling and configuring communication channels according to EN 13757-2:2004. Several communication channels can be configured.

M-Bus slave port setup		0...n	class_id = 25, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	default_baud (static)	enum	0	5	0	x + 0x08
3.	avail_baud (static)	enum	0	7		x + 0x10
4.	addr_state (static)	enum				x + 0x18
5.	bus_address (static)	unsigned				x + 0x20
Specific methods		m/o				

Attribute description

logical_name Identifies the “M-Bus slave port setup” object instance. See 6.2.20.

default_baud Defines the baud rate for the opening sequence.

- enum:
- (0) 300 baud,
 - (3) 2 400 baud,
 - (5) 9 600 baud

avail_baud	Defines the baud rates that can be negotiated after startup. enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud
addr_state	Defines whether or not the device has been assigned an address since last power up of the device. enum: (0) Not assigned an address yet, (1) Assigned an address either by manual setting, or by automated method.
bus_address	The currently assigned address on the bus for the device. NOTE If no bus address is assigned, the value is 0.

5.6.2 M-Bus client (class_id = 72, version = 1)

Instances of the “M-Bus client” allow setting up M-Bus slave devices using wired M-Bus and to exchange data with them. Each “M-Bus client” object controls one M-Bus slave device. For details on the M-Bus dedicated application layer, see EN 13757-3:2013.

NOTE Version 1 of the “M-Bus client” IC is in line with EN 13757-3:2013.

The M-Bus client device may have one or more physical M-Bus interfaces, which can be configured using instances of the “M-Bus master port setup” IC, see 5.6.4.

An M-Bus slave device is identified with its Primary Address, Identification Number, Manufacturer ID, etc., as defined in EN 13757-3:2013 Clause 5, Variable Data Send and Variable Data respond. These parameters are carried by the respective attributes of the M-Bus client IC.

Values to be captured from an M-Bus slave device are identified by the *capture_definition* attribute, containing a list of data identifiers (DIB, VIB) for the M-Bus slave device. The data are captured periodically or on an appropriate trigger. Each data element is stored in an M-Bus value object, of IC “Extended register”. M-Bus value objects may be captured in M-Bus “Profile generic” objects, eventually along with other, non M-Bus specific objects.

Using the methods of “M-Bus client” objects, M-Bus slave devices can be installed and de-installed.

It is also possible to send data to M-Bus slave devices and to perform operations like resetting alarms, synchronizing the clock, transferring an encryption key, etc.

Configuration field as defined in EN 13757-3:2013, 5.12 provides information about the encryption mode and number of encrypted bytes.

Encryption key status provides information if encryption key has been set, transferred to M-Bus slave device and is in use with M-Bus slave device.

M-Bus client		0...n	class_id = 72, version = 1			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	mbus_port_reference (static)	octet-string				x + 0x08
3.	capture_definition (static)	array			empty	x + 0x10
4.	capture_period (static)	double-long-unsigned			0	x + 0x18
5.	primary_address	unsigned			0	x + 0x20
6.	identification_number (dyn.)	double-long-unsigned			0	x + 0x28
7.	manufacturer_id (dyn.)	long-unsigned			0	x + 0x30
8.	version (dyn.)	unsigned			0	x + 0x38
9.	device_type (dyn.)	unsigned			0	x + 0x40
10.	access_number (dyn.)	unsigned			0	x + 0x48
11.	status (dyn.)	unsigned			0	x + 0x50
12.	alarm (dyn.)	unsigned			0	x + 0x58
13.	configuration (dyn.)	long-unsigned			0	x + 0x60
14.	encryption_key_status (dyn.)	enum			0	x + 0x68
Specific methods		m/o				
1.	slave_install (data)	o				x + 0x70
2.	slave_deinstall (data)	o				x + 0x78
3.	capture (data)	o				x + 0x80
4.	reset_alarm (data)	o				x + 0x88
5.	synchronize_clock (data)	o				x + 0x90
6.	data_send (data)	o				x + 0x98
7.	set_encryption_key (data)	o				x + 0xA0
8.	transfer_key (data)	o				x + 0xA8

Attribute description

logical_name Identifies the “M-Bus client” object instance. See 6.2.20.

mbus_port_reference Provides reference to an “M-Bus master port setup” object, used to configure an M-Bus port, each interface allowing to exchange data with one or more M-Bus slave devices.

capture_definition Provides the *capture_definition* for M-Bus slave devices.

NOTE 1 This attribute can be pre-configured or written as part of the installation procedure.

array capture_definition_element

```
capture_definition_element ::= structure
{
    data_information_block:    octet-string,
    value_information_block:   octet-string
}
```

NOTE 2 The elements *data_information_block* and *value_information_block* correspond to Data Information Block (DIB) and Value Information Block (VIB) described in EN 13757-3:2013 sub-clause 6.2 and Clause 7 respectively.

capture_period	>= 1: Automatic capturing assumed. Specifies the capture period in seconds. 0: No automatic capturing: capturing is triggered externally or capture events occur asynchronously.
primary_address	Carries the primary address of the M-Bus slave device. The range is 0...250. Each M-bus device is bound to a channel of the M-Bus master. However, there is no direct link between the primary address and the channel number. NOTE 3 The specification of the B field of the OBIS codes limits the range to 1...64 within one logical device. See IEC 62056-6-1:2015, 5.2. If the slave device is already configured and thus, its primary address is different from 0, then this value shall be written to the <i>primary_address</i> attribute. From this moment, the data exchange with the M-Bus slave device is possible. Otherwise, the <i>slave_install</i> method shall be used; see below. NOTE 4 The <i>primary_address</i> attribute cannot be used to store a desired primary address for an unconfigured slave device. If the <i>primary_address</i> attribute is set, this means that the M-Bus client can immediately operate with this primary address, which is not the case with an unconfigured slave device.
identification_number	Carries the Identification Number element of the data header as specified in EN 13757-3:2013, 5.5. This attribute, together with attributes 7, 8 and 9 are filled with the values found in the first message received after installation. If in subsequent messages these values are not the same, the message is discarded.
manufacturer_id	Carries the Manufacturer Identification element of the data header as specified in EN 13757-3:2013, 5.6.
version	Carries the Version element of the data header as specified in EN 13757-3:2013, 5.7.
device_type	Carries the Device type identification element of the data header as specified in EN 13757-3:2013, 5.8, Table 6.
access_number	Carries the Access Number element of the data header as specified in EN 13757-3:2013, 5.9.
status	Carries the Status byte element of the data header as specified in EN 13757-3:2013, 5.10, Table 7 and 8. It is updated with every readout of the M-Bus slave device.
alarm	Carries the Alarm state specified in EN 13757-3:2013, Annex D. It is updated with every readout of the M-Bus slave device.
configuration	Carries the Configuration field (previously: Signature field) as specified in EN 13757-3:2013, 5.12. It contains information about the encryption mode and the number of encrypted bytes. It is updated with every readout of the M-Bus slave device.

encryption_key_status	Provides information on the status of the encryption key. See also Annex B. enum: (0) no encryption key, (1) encryption_key set, (2) encryption_key transferred, (3) encryption_key set and transferred, (4) encryption_key in use. See also Annex B.

Method description

slave_install (data)	Installs a slave device, which is yet unconfigured (its primary address is 0). data::= unsigned This method can be successfully invoked only if the current value of the <i>primary_address</i> attribute is 0. The following actions are performed: – the M-Bus address 0 is checked for presence of a new device; – if no uninstalled M-Bus slave is found, the method invocation fails; – if the <i>slave_install</i> method is invoked with "0" as a parameter, then the primary address is assigned automatically. This is done by checking the <i>primary_address</i> attribute of all M-Bus client objects in the DLMS/COSEM device and then selecting the first unused number. The <i>primary_address</i> attribute is set to this address and it is then transferred to the M-Bus slave device; – if the <i>slave_install</i> method is invoked with a primary address (other than 0) as a parameter, then the <i>primary_address</i> attribute is set to this value and it is then transferred to the M-Bus slave device. NOTE 5 Unconfigured slave devices are configured with primary address as specified in EN 13757-3:2013, Annex E.5.
slave_deinstall (data)	De-installs the slave device. The main purpose of this service is to de-install the M-Bus slave device and to prepare the master for the installation of a new device. The following actions are performed: – the M-Bus address is set to 0 in the M-Bus slave device; – the encryption key transferred previously to the M-Bus slave device is destroyed; the default key is not affected; – the <i>encryption_key_status</i> is set to (0): no encryption_key; – the attribute <i>primary_address</i> is also set to 0. NOTE 6 A new M-Bus slave can be installed only once the value of the <i>primary_address</i> attribute is 0. data::= unsigned (0)
capture (data)	Captures values – as specified by the <i>capture_definition</i> attribute – from the M-Bus slave device. data::= integer (0)
reset_alarm (data)	Resets alarm state of the M-Bus slave device. data::= integer (0)
synchronize_clock (data)	Synchronize the clock of the M-Bus slave device with that of the M-Bus client device. data::= integer (0)

data_send (data)	Sends data to the M-Bus slave device. data::= array data_definition_element data_definition_element::= structure <pre> { data_information_block: octet-string, value_information_block: octet-string, data: CHOICE { -- simple data types null-data [0], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], utf8-string [12], bcd [13], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], float32 [23], float64 [24] } </pre>
set_encryption_ key (data)	Sets the encryption key in the M-Bus client and enables encrypted communication with the M-Bus slave device. data::= octet-string (encryption_key) After the installation of the M-Bus slave, the M-Bus client holds an empty encryption key. With this, encryption of M-Bus frames is disabled. Encryption can be disabled by invoking the <i>set_encryption_key</i> method with an octet_string of zero length. Changing the encryption key requires two steps: – first, the key is sent to the M-Bus slave, encrypted with the default key, using the <i>transfer_key</i> method; – second, the key is set in the M-Bus master using the <i>set_encryption_key</i> method.

transfer_key (data)	Transfers an encryption key to the M-Bus slave device. data::= octet-string (encrypted_key) Before encrypted M-Bus frames can be used, an operational encryption key has to be sent to the M-Bus slave, by invoking the <i>transfer_key method</i>. The method invocation parameter is the operational encryption key encrypted with the default key of the M-Bus slave device. The M-Bus frame sent is not encrypted. After the execution, the encryption is enabled and all further telegrams are encrypted. Each M-Bus slave device shall be delivered with a default encryption key. A new encryption key can be set in the M-Bus client by invoking the <i>set_encryption_key</i> method with the new encryption key as a parameter. With further invocations of the <i>transfer_key method</i>, new encryption keys can be sent to the M-Bus slave. The method invocation parameter is the new encryption key encrypted with the default key. The M-Bus frame is encrypted. When an M-Bus slave is de-installed, the encryption key is destroyed but the default key is not affected. Encryption remains disabled until a new encryption key is transferred.

5.6.3 Wireless Mode Q channel (class_id = 73, version = 1)

Instances of this IC define the operational parameters for communication using the mode Q interfaces. See also EN 13757-5:2015.

Wireless Mode Q channel		0...n	class_id = 73, version = 1			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. addr_state	(static)	enum				x + 0x08
3. device_address	(static)	octet-string				x + 0x10
4. address_mask	(static)	octet-string				x + 0x18
Specific methods		m/o				

Attribute description

logical_name	Identifies the “Wireless Mode Q channel” object instance. See 6.2.20.
addr_state	Defines whether or not the device has been assigned an address since last power up of the device. enum: (0) not assigned an address yet, (1) assigned an address either by manual setting or by automated method.
device_address	The currently assigned address of the device on the network.
address_mask	The group address the device will respond to when short form addressing is used.

5.6.4 M-Bus master port setup (class_id = 74, version = 0)

Instances of this IC define the operational parameters for communication using the EN 13757-2 interfaces if the device acts as an M-bus master.

M-Bus master port setup		0...n	class_id = 74, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. comm_speed	(static)	enum	0	7	3	x + 0x08
Specific methods		m/o				

Attribute description

logical_name	Identifies the “M-Bus master port setup” object instance. See 6.2.20					
comm_speed	The communication speed supported by the port					
	enum:	(0)	300 baud,			
		(1)	600 baud,			
		(2)	1 200 baud,			
		(3)	2 400 baud,			
		(4)	4 800 baud,			
		(5)	9 600 baud,			
		(6)	19 200 baud,			
		(7)	38 400 baud			

5.7 Interface classes for setting up data exchange over the Internet

5.7.1 TCP-UDP setup (class_id = 41, version = 0)

This IC allows modelling the setup of the TCP or UDP sub-layer of the COSEM TCP or UDP based transport layer of a TCP-UDP/IP based communication profile.

In TCP-UDP/IP based communication profiles, all AAs between a physical device hosting one or more COSEM client application processes and a physical device hosting one or more COSEM server APs rely on a single TCP or UDP connection. The TCP or UDP entity is wrapped in the COSEM TCP-UDP based transport layer. Within a physical device, each AP – client AP or server logical device – is bound to a Wrapper Port (WPort). The binding is done with the help of the SAP Assignment object. See 5.3.5.

On the other hand, a COSEM TCP or UDP based transport layer may be capable to support more than one TCP or UDP connections, between a physical device and several peer physical devices hosting COSEM APs.

When a COSEM physical device supports various data link layers – for example Ethernet and PPP – an instance of the TCP-UDP setup object is necessary for each of them.

TCP-UDP setup		0...n	class_id = 41, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	TCP-UDP_port (static)	long-unsigned				x + 0x08
3.	IP_reference (static)	octet-string				x + 0x10
4.	MSS (static)	long-unsigned	40	65...535	576	x + 0x18
5.	nb_of_sim_conn (static)	unsigned	1			x + 0x20
6.	inactivity_time_out (static)	long-unsigned			180	x + 0x28
Specific methods		m/o				

Attribute description

logical_name	Identifies the “TCP-UDP setup” object instance. See 6.2.21.						
TCP-UDP_port	Holds the TCP-UDP port number on which the physical device is listening for the DLMS/COSEM application. For DLMS/COSEM, the following port numbers have been registered by the IANA. See http://www.iana.org/assignments/port-numbers <table><tr><td>dlms/cosem</td><td>4059/TCP</td><td>DLMS/COSEM</td></tr><tr><td>dlms/cosem</td><td>4059/UDP</td><td>DLMS/COSEM</td></tr></table>	dlms/cosem	4059/TCP	DLMS/COSEM	dlms/cosem	4059/UDP	DLMS/COSEM
dlms/cosem	4059/TCP	DLMS/COSEM					
dlms/cosem	4059/UDP	DLMS/COSEM					
IP_reference	References an IP setup object by its logical name. The referenced object contains information about the IP Address settings of the IP layer supporting the TCP-UDP layer.						
MSS	With the help of the Maximum Segment Size (MSS) option, a TCP can indicate the maximum receive segment size to its partner. Note, that: – this option shall only be sent in the initial connection request (i.e. in segments with the SYN control bit sent); – if this option is not present, conventionally MSS is considered as its default value, 576; – MSS is not negotiable; its value is indicated by this attribute.						
nb_of_sim_conn	The maximum number of simultaneous connections the COSEM TCP-UDP based transport layer is able to support.						
inactivity_time_out	Defines the time, expressed in seconds over which, if no frame is received from the COSEM client, the inactive TCP connection shall be aborted. When this value is set to 0, this means that the <i>inactivity_time_out</i> is not operational. In other words, a TCP connection, once established, in normal conditions – no power failure, etc. – will never be aborted by the COSEM server. Note, that all actions related to the management of the inactivity time-out function – measuring the inactivity time, aborting the TCP connection if the time-out is over, etc. – are managed inside the TCP-UDP layer implementation.						

5.7.2 IPv4 setup (class_id = 42, version = 0)

NOTE 1 Compared to earlier editions of this standard, this specification provides improvements in presenting the attributes. As this does not constitute technical changes, the version of the IC remains 0.

This IC allows modelling the setup of the IPv4 layer, handling all information related to the IP Address settings associated to a given device and to a lower layer connection on which these settings are used.

There shall be an instance of this IC in a device for each different network interface implemented. For example, if a device has two interfaces (using the TCP-UDP/IPv4 profile on both of them), there shall be two instances of the IPv4 setup IC in that device: one for each of these interfaces.

IPv4 setup		0...n	class_id = 42, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	DL_reference (static)	octet-string				x + 0x08
3.	IP_address	double-long-unsigned				x + 0x10
4.	multicast_IP_address	array				x + 0x18
5.	IP_options	array				x + 0x20
6.	subnet_mask	double-long-unsigned				x + 0x28
7.	gateway_IP_address	double-long-unsigned				x + 0x30
8.	use_DHCP_flag (static)	boolean				x + 0x38
9.	primary_DNS_address	double-long-unsigned				x + 0x40
10.	secondary_DNS_address	double-long-unsigned				x + 0x48
Specific methods		m/o				
1.	add_mc_IP_address (data)	o				x + 0x60
2.	delete_mc_IP_address (data)	o				x + 0x68
3.	get_nbof_mc_IP_addresses (data)	o				x + 0x70

Attribute description

logical_name Identifies the "IPv4 setup" object instance. See 6.2.21.

DL_reference References a Data link layer (e.g. Ethernet or PPP) setup object by its logical name.

The referenced object contains information about the specific settings of the data link layer supporting the IP layer.

IP_address Carries the value of the IP address (IPv4) of this physical device on the network to which the device is connected via the referenced lower layer interface. It can be either (static) or (dynamic). In the latter case, dynamic IP address assignment (for example DHCP) is used.

If no IP address is assigned, the value is 0.

EXAMPLE The IPv4 address 192.168.0.1 (in dotted decimal notation) corresponds to C0A80001 (hexa) which gives 3232235521 (double-long-unsigned).

multicast_IP_address Contains an array of IP addresses. IP addresses in this array shall fall into the multicast group address range ("Class D" addresses, including IP addresses in the range of 224.0.0.0 to 239.255.255.255). When a device receives an IP datagram with one of these IP addresses in the destination IP address field, it shall consider that this datagram is addressed to it.

multicast_IP_address::= array double-long-unsigned

IP_options

Contains the necessary parameters to support the selected IP options – for example Datagram time-stamping or security services (IPSec).

IP_options ::= array IP_options_element

IP_options_element ::= structure

```
{
    IP_Option_Type:      unsigned,
    IP_Option_Length:    unsigned,
    IP_Option_Data:      octet-string
}
```

NOTE 2 In all cases, as specified in RFC 791, the IP_Option_Length field includes the total length of all three fields: IP_Option_Type, IP_Option_Length and IP_Option_Data.

Allowed IP_Option_Types:

- Security: IP_Option_Type = 0x82, IP_Option_Length = 11

If this option is present, the device shall be allowed to send security, compartmentation, handling restrictions and TCC (closed user group) parameters within its IP Datagrams. The IP_Option_Data shall contain the value of the Security, Compartments, Handling Restrictions and Transmission Control Code values, as specified in RFC 791.

- Loose Source and Record Route: IP_Option_Type = 0x83

If this option is present, the device shall supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information. The IP_Option_Length and IP_Option_Data values are specified in RFC 791.

- Strict Source and Record Route: IP_Option_Type = 0x89

If this option is present, the device shall supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information. The IP_Option_Length and IP_Option_Data values are specified in RFC 791.

- Record Route: IP_Option_Type = 0x07

If this option is present, the device shall as well:

- send originated IP Datagrams with that option, providing means to record the route of these Datagrams;
- as a router, send routed IP Datagrams with the route option adjusted according to this option.

The IP_Option_Length and IP_Option_Data values are specified in RFC 791.

- Internet Timestamp: IP_Option_Type = 0x44

If this option is present, the device shall as well:

- send originated IP Datagrams with that option, providing means to time-stamp the datagram in the route to its destination;
- as a router, send routed IP Datagrams with the time-stamp option adjusted according to this option.

The IP_Option_Length and IP_Option_Data values are specified in RFC 791.

subnet_mask	Contains the subnet mask. When sub-networking is used in a network segment, each device concerned shall behave conforming to the sub-networking rules. In order to do that, the device, besides of its IP address, needs also to know, how the IP address is structured within this sub-networked segment. The <i>subnet_mask</i> attribute carries this information. With IPv4, the <i>subnet_mask</i> is a 32 bits word, expressed exactly in the same format as an IP Address (for example 255.255.255.0), but has another meaning: the '0' bits of the <i>subnet_mask</i> indicate the portion of the IP Address which is still used as Device_ID on a sub-networked IP Network 4.
gateway_IP_address	Contains the IP Address of the gateway device. In most IP implementations, there is code in the module that handles outgoing datagrams to decide if a datagram can be sent directly to the destination on the local network or if it shall be sent to a gateway. In order to be able to send non-local datagrams to the gateway, the device shall know the IP address of the gateway device assigned to the given network segment. If no IP address is assigned, the value is 0.
use_DHCP_flag	When this flag is set to TRUE, the device uses DHCP (Dynamic Host Configuration Protocol) to dynamically determine the <i>IP_address</i> , <i>subnet_mask</i> and <i>gateway_IP_address</i> parameters. On the other hand, when this flag is set to FALSE, the <i>IP_address</i> , <i>subnet_mask</i> and <i>gateway_IP_address</i> parameters shall be locally set.
primary_DNS_address	The IP Address of the primary Domain Name Server (DNS). If no IP address is assigned, the value is 0.
secondary_DNS_address	The IP Address of the secondary Domain Name Server (DNS). If no IP address is assigned, the value is 0.
Method description	
add_mc_IP_address (IP_Address)	Adds one multicast IP address to the <i>multicast_IP_address</i> array. IP_Address::= double-long-unsigned
delete_mc_IP_address (IP_Address)	Deletes one IP Address from the <i>multicast_IP_address</i> array. The IP Address to be deleted is identified by its value. IP_Address::= double-long-unsigned
get_nbof_mc_IP_addresses (data)	Returns the number of IP Addresses contained in the <i>multicast_IP_address</i> array. data::= unsigned

5.7.3 IPv6 setup (class_id = 48, version = 0)

NOTE 1 See also Annex C.

The IPv6 setup IC allows modelling the setup of the IPv6 layer, handling all information related to the IPv6 address settings associated to a given device and to a lower layer connection on which these settings are used.

There shall be an instance of this IC in a device for each different network interface implemented. For example, if a device has two interfaces (using the UDP/IP and/or TCP/IP profile on both of them), there shall be two instances of the IPv6 setup IC in that device: one for each of these interfaces.

IPv6 setup		0...n	class_id = 48, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	DL_reference (static)	octet-string				x + 0x08
3.	address_config_mode (static)	enum			0	x + 0x10
4.	unicast_IPv6_addresses	array				x + 0x18
5.	multicast_IPv6_addresses (static)	array				x + 0x20
6.	gateway_IPv6_addresses (static)	array			0	x + 0x28
7.	primary_DNS_address (static)	octet-string			0	x + 0x30
8.	secondary_DNS_address (static)	octet-string			0	x + 0x38
9.	traffic_class (static)	unsigned	0	63	0	x + 0x40
10.	neighbor_discovery_setup (static)	array				x + 0x48
Specific methods		m/o				
1.	add_IPv6_address (data)	o				x + 0x60
2.	remove_IPv6_address (data)	o				x + 0x68

Attribute description

logical_name Identifies the “IPv6 setup” object instance. See 6.2.21.

DL_reference References a Data link layer setup object by its logical name. The referenced object contains information about the specific settings of the data link layer supporting the IPv6 layer.

address_config_mode Defines the IPv6 address configuration mode
enum: (0) Auto-configuration (default),
(1) DHCPv6,
(2) Manual,
(3) ND (Neighbour Discovery)

NOTE 2 The *address_config_mode* is common for all IPv6 addresses managed by an instance of the IPv6 setup class.

unicast_IPv6_addresses Carries unicast IPv6 address(es) assigned to the related interface of the physical device on the network (unique local unicast, link local unicast and / or global unicast addresses). An IPv6 address can be either (static) or (dynamic) or both.

unicast_IPv6_addresses ::= array octet-string

The format of each unicast IPv6 address shall be as specified in RFC 3513.

If no unicast IPv6 address is assigned to the interface, an array of zero elements is present (default).

To reset all unicast IPv6 address(es) configured, an array of zero elements shall be written.

multicast_IPv6_addresses	Contains an array of IPv6 addresses used for multicast. multicast_IPv6_addresses::= array octet-string The format of each multicast IPv6 address shall be as specified in RFC 3513 . If no multicast IPv6 address is assigned to the interface, an array of zero elements is present (default). To reset all multicast IPv6 address(es) configured, an array of zero elements shall be written.						
gateway_IPv6_addresses	Contains the IPv6 addresses of the IPv6 gateway device. gateway_IPv6_addresses::= array octet-string The format of each gateway IPv6 address shall be as specified in RFC 3513. If no gateway IPv6 address is assigned to the interface, an array of zero elements is present (default). To reset all gateway IPv6 address(es) configured, an array of zero elements shall be written.						
primary_DNS_address	Contains the IPv6 address of the primary Domain Name Server (DNS). If no IPv6 address is assigned, the length of the octet-string shall be 0.						
secondary_DNS_address	Contains the IPv6 address of the secondary Domain Name Server (DNS). If no IPv6 address is assigned, the length of the octet-string shall be 0.						
traffic_class	Contains the traffic class element of the IPv6 header. The 6 most-significant bits are used for DSCP (Differentiated Services Codepoint), which is used to classify packets as specified in RFC 2474:1998, Clause 3.						
neighbor_discovery_setup	Contains the configuration to be used for both routers and hosts to support the Neighbor Discovery protocol for IPv6 (RFC 4861). array neighbour_discovery_setup neighbor_discovery_setup::= structure <pre> { RS_max_retry: unsigned, RS_retry_wait_time: long-unsigned, RA_send_period: double-long-unsigned } </pre> Where: <table> <tr> <td>RS_max_retry</td><td> Gives the maximum number of router solicitation retries to be performed by a node if the expected router advertisement has not been received. Range: 1-255 Default value: 3 </td></tr> <tr> <td>RS_retry_wait_time</td><td> Gives the waiting time in milliseconds between two successive router solicitation retries. Range: 0-65 535 Default value: 10 000 </td></tr> <tr> <td>RA_send_period</td><td> Gives the router advertisement transmission period in seconds. Range: 0-4 294 967 295 </td></tr> </table>	RS_max_retry	Gives the maximum number of router solicitation retries to be performed by a node if the expected router advertisement has not been received. Range: 1-255 Default value: 3	RS_retry_wait_time	Gives the waiting time in milliseconds between two successive router solicitation retries. Range: 0-65 535 Default value: 10 000	RA_send_period	Gives the router advertisement transmission period in seconds. Range: 0-4 294 967 295
RS_max_retry	Gives the maximum number of router solicitation retries to be performed by a node if the expected router advertisement has not been received. Range: 1-255 Default value: 3						
RS_retry_wait_time	Gives the waiting time in milliseconds between two successive router solicitation retries. Range: 0-65 535 Default value: 10 000						
RA_send_period	Gives the router advertisement transmission period in seconds. Range: 0-4 294 967 295						

Method description

add_IPv6_address (data) Adds one IPv6 address for the physical interface to the IPv6 address array.

```
data ::= structure
{
    IPv6_address_type:  enum,
    IPv6_address:       octet_string
}
```

Where IPv6_address_type defines the type of IPv6 address to add:

```
enum:
    (0) unicast,
    (1) multicast,
    (2) gateway
```

IPv6_address specifies the IPv6 address to add. The new address will be automatically added at the end of the list. If an address has to be added to a specific position, this can be done by writing the whole array.

remove_IPv6_address (data) Removes one IPv6 address for the physical interface to the IPv6 address array.

```
data ::= structure
{
    IPv6_address_type:  enum,
    IPv6_address:       octet-string
}
```

Where IPv6_address_type defines the type of IPv6 address to remove:

```
enum:
    (0) unicast,
    (1) multicast,
    (2) gateway
```

IPv6_address specifies the IPv6 address to remove.

5.7.4 MAC address setup (class_id = 43, version = 0)

NOTE 1 The name and the use of this interface class has been changed from “Ethernet setup” to “MAC address setup” to allow a more general use, without changing the version.

Instances of this IC hold the MAC address of the physical device (or, more generally, a device or software.) There shall be an instance of this IC for each network interface of a physical device.

NOTE 2 In the case of the three-layer HDLC based communication profile, the MAC address (lower HDLC address) is carried by an IEC HDLC setup object.

NOTE 3 In the case of the S-FSK PLC communication profile, the MAC address is carried by a S-FSK Phy&MAC setup object.

MAC address setup	0...n	class_id = 43, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. MAC_address	octet-string				x + 0x08
Specific methods	m/o				

Attribute description

logical_name Identifies the “MAC address setup” object instance. See 6.2.19, 6.2.21 and 6.2.24.

mac_address Holds the MAC address.

5.7.5 PPP setup (class_id = 44, version = 0)

NOTE 1 Compared to earlier editions of this standard, this specification provides improvements in presenting the attributes. As this does not constitute technical changes, the version of the IC remains 0.

This IC allows modelling the setup of interfaces using the PPP protocol, by handling all information related to PPP settings associated to a given physical device and to a lower layer connection on which these settings are used. There shall be an instance of this IC for each network interface of a physical device.

PPP setup		0...n	class_id = 44, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	PHY_reference (static)	octet-string				x + 0x08
3.	LCP_options (static)	LCP_options_type				x + 0x10
4.	IPCP_options (static)	IPCP_options_type				x + 0x18
5.	PPP_authentication (static)	PPP_auth_type				x + 0x20
Specific methods		m/o				

Attribute description

logical_name Identifies the “PPP setup” object instance. See 6.2.21.

PHY_reference References another object by its *logical_name*. The object referenced contains information about the specific physical layer interface, supporting the PPP layer.

LCP_options Contains the necessary parameters to support the selected LCP configuration options.

LCP_options_type ::= array LCP_options_type_element

LCP_options_type_element ::= structure

```

{
    LCP_Option_Type:      unsigned,
    LCP_Option_Length:    unsigned,
    LCP_Option_Data:      CHOICE
    {
        structure         [2] -- for Callback-data
        boolean            [3] -- for ProtF-Compr and AdCtr-Compr,
        double-long-unsigned [6] -- for ACCM and Mag-Num,
        unsigned           [17] -- for FCS-Alternatives,
        long-unsigned       [18] -- for MRU and Auth-Prot
    }
}

```

NOTE 2 In all cases, as specified in IETF STD 51 / RFC 1661, the LCP_Option_Length field includes the total length of all three fields: LCP_Option_Type, LCP_Option_Length and LCP_Option_Data.

NOTE 3 For assigned values see *Point-to-Point (PPP) Protocol Field Assignments*, available at: <http://www.iana.org/assignments/ppp-numbers/ppp-numbers.xml>

LCP_options	The supported LCP_Option_Types are the following:
(continued)	– Maximum-Receive-Unit (MRU), LCP_Option_Type = 1. See IETF STD 51 / RFC 1661. This configuration option may be sent to inform the peer that the implementation can receive larger packets, or to request that peer send smaller packets. The default value is 1 500 octets; – Async-Control-Character-Map (ACCM), LCP_Option_Type = 2. See IETF STD 51 / RFC 1662. This configuration option provides a method to negotiate the use of control character transparency on asynchronous links; – Authentication-Protocol, LCP_Option_Type = 3. See IETF STD 51 / RFC 1661. This configuration option provides a method to negotiate the use of a specific protocol for authentication. By default, authentication is not required. The value indicates the authentication protocol used on the given PPP link. Possible values are: 0x0000 – No authentication protocol is used, 0xc023 – The PAP protocol is used, 0xc223 – The CHAP protocol is used, 0xc227 – The EAP protocol is used. – Magic-Number, LCP_Option_Type = 5. See IETF STD 51 / RFC 1661. This configuration option provides a method to detect looped-back links and other data link layer anomalies; – Protocol-Field-Compression (PFC), LCP_Option_Type = 7. See IETF STD 51 / RFC 1661. This configuration option provides a method to negotiate the compression of the PPP protocol fields; – Address-and-Control-Field-Compression (ACFC), LCP_Option_Type = 8. See IETF STD 51 / RFC 1661. This configuration option provides a method to negotiate the compression of the data link layer address and control fields; – FCS-Alternatives, LCP_Option_Type = 9. See RFC 1570. This configuration option provides a method for an implementation to specify another FCS format to be sent by the peer, or to negotiate away the FCS altogether. The value of the FCS-Alter (FCS Alternatives) options field identifies the FCS used. This field is one octet, and is comprised of the "logical or" of the following values: - bit 1 Null FCS, - bit 2 CCITT 16-bit FCS, - bit 4 CCITT 32-bit FCS. – Callback, LCP_Option_Type = 13. See RFC 1570. This configuration option provides a method for an implementation to request a dial-up peer to call back. This provides enhanced security by ensuring that the remote site can connect only from a single location as defined by the callback number.

LCP_options

(continued)

callback_data ::= structure

```

{
  callback_active:          boolean,    // default: false,
  callback_data_length:     unsigned,
  callback_operation:       unsigned,
  callback_message:         octet-string
}

```

Where:

- the callback_active field indicates whether the callback option is active on this PPP link;
- the callback_operation field indicates the contents of the Message field;

callback_operation: unsigned

- (0) Location determined by user authentication,
- (1) Dialing string,
- (2) Location identifier,
- (3) E.164 number,
- (4) X.500 distinguished name,
- (5) Unassigned,
- (6) Location is determined during CBCP negotiation.

The callback_message field is zero or more octets, and its general contents are determined by the callback_operation field. The actual format of the information is site or application specific.

IPCP_options

Contains the necessary parameters for the IP Control Protocol – the Network Control Protocol module of the PPP – that allow the negotiation of desirable Internet Protocol parameters. For details on IPCP, please refer to RFC 1332.

IPCP_options_type ::= array IPCP_options_type_element

IPCP_options_type_element ::= structure

```

{
  IPCP_Option_Type          unsigned,
  IPCP_Option_Length        unsigned,
  IPCP_Option_Data          CHOICE
{
  array                      [1] -- for Pref-Peer-IP,
  -- each IP address is of type double-long-unsigned
  boolean                   [3] -- for GAO and USIP,
  double-long-unsigned       [6] -- for Pref-Local-IP,
  long-unsigned              [18] -- for IP-Comp-Prot
}
}

```

NOTE 4 In all cases, as specified in RFC 1332, the IPCP_Option_Length field includes the total length of all three fields: IPCP_Option_Type, IPCP_Option_Length and IPCP_Option_Data.

IPCP_options
(continued)

The supported IPCP_Option_Types are the following:

- IP-Compression-Protocol (IP-Comp-Prot) IPCP_Option_Type = 2. See RFC 1332.

This configuration option provides a way to negotiate the use of a specific compression protocol. By default, compression is not enabled. Possible values are:

- 0x0000 – No IP Compression is used (default),
- 0x002d – Van Jacobson Compressed TCP/IP (RFC 1332),
- 0x0003 – Robust Header Compression (ROHC) (RFC 3241),
- 0x0061 – IP Header Compression (RFC 2507, RFC 3544)

- Preferred-Local-IP-Address (Pref-Local-I), IPCP_Option_Type = 3. See RFC 1332.

This configuration option provides a way to negotiate the IP address to be used on the local end of the link. It allows the sender of the Configure-Request to state, which IP-address is desired, or to request that the peer provide the information. The peer can provide this information by NAK-ing the option, and returning a valid IP-Address;

NOTE 5 In RFC 1332 the name of option Type 3 is "IP-Address".

NOTE 6 Option types 20, 21 and 22 have been specified for the purposes of DLMS/COSEM; they are not specified in RFC 1332.

- Preferred-Peer-IP-Addresses (Pref-Peer-IP), IPCP_Option_Type = 20.

This configuration option provides a way to negotiate the IP Address to be used on the remote end of the link. When the Grant-Access-Only-to-Pref-Peer-on-List (GAO) parameter is set to be TRUE, the device shall accept PPP connection only with a remote device having one of the IP Addresses on this list. When the Use-Static-IP-Pool (USIP) parameter is set to TRUE, the COSEM Server device shall try to assign one of these IP Addresses to the remote device;

- Grant-Access-Only-to-Pref-Peer-on-List (GAO), IPCP_Option_Type = 21.

This configuration option indicates whether the device can accept PPP connection only with peer devices with IP Address on the above list or not. Its default value is FALSE;

- Use-Static-IP-Pool (USIP), IPCP_Option_Type = 22.

This configuration option indicates whether the device should try to assign one of the IP Addresses of the Preferred-Peer-IP-Addresses to the remote end device during the IP Address negotiation phase or not.

PPP_authentication Contains the parameters required by the PPP authentication procedure used.

```
PPP_auth_type::= CHOICE
{
    null-data    [0] -- used when no authentication is required,
    structure    [2] -- PAP_login or CHAP_algorithm or EAP_params
}
```

```
PAP_login::= structure
{
    user-name:    octet-string,
    PAP-password: octet-string
}
```

```
CHAP_algorithm::= structure
{
    user-name:    octet-string,
    algorithm_id: unsigned
}
```

Possible values for CHAP-algorithm-id parameter today are as follows:

0x05: CHAP with MD5 (default), see RFC 1994.

0x06 – SHA-1,

0x80 – MS-CHAP, see RFC 2433

0x81 – MS-CHAP-2, see RFC 2759.

NOTE 7 New values can be used as become assigned.

When CHAP is used, a “secret” is also required to verify the “challenge” sent by the client. This “secret” is not accessible in the PPP setup object.

NOTE 8 The PPP Challenge Handshake Authentication Protocol (CHAP) is specified in RFC 1994.

```
EAP_params::= structure
{
    md5_challenge (Type 4):                boolean,
    one_time_password (OTP, Type 5):        boolean,
    generic_token_card (GTC, Type 6):       boolean
}
```

NOTE 9 The Extensible Authentication Protocol (EAP) is specified in RFC 3748.

5.7.6 SMTP setup (class_id = 46, version = 0)

This IC allows setting up data exchange using the SMTP protocol.

SMTP modem setup		0...n				class_id = 46, version = 0			
Attributes		Data type		Min.	Max.	Def.	Short name		
1.	logical_name (static)	octet-string					x		
2.	server_port (static)	long-unsigned				25	x + 0x08		
3.	user_name (static)	octet-string					x + 0x10		
4.	login_password (static)	octet-string					x + 0x18		
5.	server_address (static)	octet-string					x + 0x20		
6.	sender_address (static)	octet-string					x + 0x28		
Specific methods		m/o							

Attribute description	
logical_name	Identifies the “SMTP setup” object instance. See 6.2.21.
server_port	Defines the value of the TCP-UDP port related to this protocol. By default, this value is the SMTP port numberID assigned by IANA: - smtp 25/tcp, smtp 25/udp Simple Mail Transfer
user_name	Defines the user name to be used for the login to the SMTP server.
login_password	Password to be used for login. When the string is void, this means that there is no authentication.
server_address	Defines the server address as an octet string. This server address can be a name, which shall be resolvable by the primary DNS or the secondary DNS. In the case when it is directly the IP address of the server, which is specified here, it shall be a string in dotted format. EXAMPLE: 163.187.45.87.
sender_address	Defines the sender address as an octet string. This sender address can be a name. In the case when it is directly the IP address of the sender, which is specified here, it will be a string in dotted format.

5.8 Interface classes for setting up data exchange using S-FSK PLC

5.8.1 General

This subclause specifies COSEM interface classes to set up and manage the protocol layers of DLMS/COSEM S-FSK PLC communication profile.

- the S-FSK Physical layer and the MAC sub-layer as defined in IEC 61334-5-1:2001 and IEC 61334-4-512:2001;
- the LLC sub-layer as specified in IEC 61334-4-32:1996.

The MIB variables / logical link parameters specified in IEC 61334-4-512:2001, IEC 61334-5-1:2001, IEC 61334-4-32:1996 and ISO/IEC 8802-2:1998 respectively have been mapped to attributes and/or methods of COSEM ICs. The specification of these elements has been taken from the above standards and the text has been adapted to the DLMS/COSEM environment.

NOTE IEC 61334-4-512:2001 also specifies some management variables to be used on the Client side. However, the Client side object model is not covered in this standard.

For definitions related to S-FSK PLC profile see 3.2.

5.8.2 Overview

COSEM objects for setting up the S-FSK PLC channel and the LLC layer, if implemented, shall be located in the Management Logical Device of COSEM servers.

Figure 16 shows an example with a COSEM physical device comprising three logical devices. Each logical device shall contain a Logical Device Name (LDN) object. Each logical device contains one or more Association objects, one for each client supported.

NOTE As in this example there is more than one logical device, the mandatory Management Logical Device contains a SAP Assignment object instead of a Logical Device Name object.

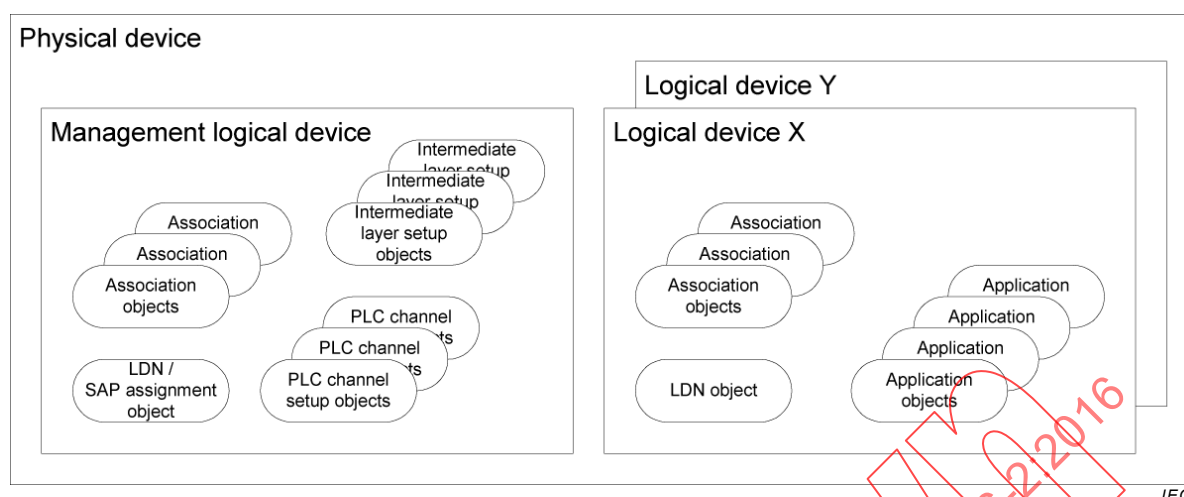


Figure 16 – Object model of DLMS/COSEM servers

The management logical device contains the setup objects of the physical and MAC layers of the PLC channel, as well as setup objects for the intermediate layer(s). It may contain further application objects.

The other logical devices, in addition to the Association and Logical Device Name objects mentioned above, contain further application objects, holding parameters and measurement values.

IEC 61334-4-512:2001 uses DLMS named variables to model the MIB objects and specifies their DLMS name in the range 8...184. For compatibility with existing implementations, the short names 8...400 [sic] are reserved for devices using the IEC 61334-5-1:2001 S-FSK PLC profiles without COSEM. Therefore, when mapping the attributes and methods of the COSEM objects specified in this standard to DLMS named variables (SN mapping) this range shall not be used.

Table 14 shows the mapping of MIB variables to attributes and/or methods of COSEM ICs.

Note that on the one hand, not all MIB variables specified in IEC 61334-4-512:2001 have been mapped to attributes and methods of COSEM ICs. On the other hand, some new management variables are specified in this standard.

Table 14 – Mapping IEC 61334-4-512:2001 MIB variables to COSEM IC attributes / methods

Name	Reference (unless otherwise indicated)	Interface class	class_id / attribute / method
S-FSK Physical layer management			
delta-electrical-phase	variable 1	S-FSK Phy&MAC set-up (class_id = 50, version = 1)	50 / Attr. 3
max-receiving-gain	variable 2		50 / Attr. 4
max-transmitting-gain	–		50 / Attr. 5
search-initiator-threshold	–		50 / Attr. 6
frequencies	–		50 / Attr. 7
transmission-speed	–		50 / Attr. 15
MAC layer management			
mac-address	variable 3	S-FSK Phy&MAC set-up (class_id = 50, version = 1)	50 / Attr. 8
mac-group-addresses	variable 4		50 / Attr. 9
repeater	variable 5		50 / Attr. 10
repeater-status	–		50 / Attr. 11
search-initiator time-out	–	S-FSK MAC synchronization timeouts (class_id = 52, version = 0)	52 / Attr. 2
synchronization-confirmation- time-out	variable 6		52 / Attr. 3
time-out-not-addressed	variable 7		52 / Attr. 4
time-out-frame-not-OK	variable 8		52 / Attr. 5
min-delta-credit	variable 9		50 / Attr. 12
initiator-mac-address	IEC 61334-5-1:2001 4.3.7.6	S-FSK Phy&MAC set-up (class_id = 50, version = 1)	50 / Attr. 13
synchronization-locked	variable 10		50 / Attr. 14
IEC 61334-4-32 LLC layer management			
max-frame-length	IEC 61334-4-32:1996 5.1.4	IEC 61334-4-32 LLC setup (class_id = 55, version = 1)	55 / Attr. 2
reply-status-list	variable 11		55 / Attr. 3
broadcast-list	variable 12	–	–
L-SAP-list	variable 13	NOTE In DLMS/COSEM, L-SAPs of logical devices are held by a SAP Assignment object	
ACSE management			
application-context-list	variable 14	NOTE In DLMS/COSEM the Association objects play a similar role.	
Application management			
active-initiator	variable 15	S-FSK Active initiator (class_id = 51, version = 0)	51 / Attr. 2
MIB system objects			
reporting-system-list	variable 16	-FSK Reporting system list (class_id = 56, version = 0)	56 / Attr. 2

Name	Reference (unless otherwise indicated)	Interface class	class_id / attribute / method
Other MIB objects			
reset-NEW-not-synchronized	variable 17	S-FSK Active initiator (class_id = 51, version = 0)	51 / Method 1
new-synchronization	IEC 61334-5-1:2001 4.3.7.6	–	
initiator-electrical-phase	variable 18		50 / Attr. 2
broadcast-frames-counter	variable 19	S-FSK MAC counters (class_id = 53, version = 0)	53 / Attr. 4
repetitions-counter	variable 20		53 / Attr. 5
transmissions-counter	variable 21		53 / Attr. 6
CRC-OK-frames-counter	variable 22		53 / Attr. 7
CRC-NOK-frames-counter	–		53 / Attr. 8
synchronization-register	variable 23		53 / Attr. 2
desynchronization-listing	variable 24		53 / Attr. 3

5.8.3 S-FSK Phy&MAC set-up (class_id = 50, version = 1)

NOTE 1 The use of version 0 of this interface class is deprecated.

An instance of the “S-FSK Phy&MAC set-up” class stores the data necessary to set up and manage the physical and the MAC layer of the PLC S-FSK lower layer profile.

S-FSK Phy&MAC setup		0...n	class_id = 50, version = 1			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. initiator_electrical_phase	(static)	enum	0	3		x + 0x08
3. delta_electrical_phase	(dyn.)	enum	0	6		x + 0x10
4. max_receiving_gain	(static)	unsigned				x + 0x18
5. max_transmitting_gain	(static)	unsigned				x + 0x20
6. search_initiator_threshold	(static)	unsigned			98	x + 0x28
7. frequencies	(static)	frequencies_type				x + 0x30
8. mac_address	(dyn.)	long-unsigned			FFE	x + 0x38
9. mac_group_addresses	(static)	array				x + 0x40
10. repeater	(static)	enum				x + 0x48
11. repeater_status	(dyn.)	boolean				x + 0x50
12. min_delta_credit	(dyn.)	unsigned				x + 0x58
13. initiator_mac_address	(dyn.)	long-unsigned				x + 0x60
14. synchronization_locked	(dyn.)	boolean				x + 0x68
15. transmission_speed	(static)	enum	0	6	3	x + 0x70
Specific methods		<i>m/o</i>				

Attribute description	
logical_name	Identifies the “S-FSK Phy&MAC setup” object instance. See 6.2.22.
initiator_electrical_phase	Holds the MIB variable <i>initiator-electrical-phase</i> (variable 18) specified in IEC 61334-4-512:2001, 5.8. It is written by the client system to indicate the phase to which it is connected. enum: (0) Not defined (default), (1) Phase 1, (2) Phase 2, (3) Phase 3 NOTE 2 This enumeration is different from that of IEC 61334-4-512.
delta_electrical_phase	Holds the MIB variable <i>delta-electrical-phase</i> (variable 1) specified in IEC 61334-4-512:2001, 5.2 and IEC 61334-5-1:2001, 3.5.5.3. It indicates the phase difference between the client's connecting phase and the server's connecting phase. The following values are predefined: enum: (0) Not defined: the server is temporarily not able to determine the phase difference, (1) The server system is connected to the same phase as the client system. The phase difference between the server's connecting phase and the client's connecting phase is equal to: (2) 60 degrees, (3) 120 degrees, (4) 180 degrees, (5) -120 degrees, (6) -60 degrees
max_receiving_gain	Holds the MIB variable <i>max-receiving-gain</i> (variable 2) specified in IEC 61334-4-512:2001, 5.2 and IEC 61334-5-1:2001, 3.5.5.3. Corresponds to the maximum allowed gain bound to be used by the server system in the receiving mode. The default unit is dB. NOTE 3 In IEC 61334-4-512:2001, no units are specified. The possible values of the gain may depend on the hardware. Therefore, after writing a value to this attribute, the value should be read back to know the actual value.
max_transmitting_gain	Holds the value of the <i>max-transmitting-gain</i>. Corresponds to the maximum attenuation bound to be used by the server system in the transmitting mode. The default unit is dB. The possible values of the gain may depend on the hardware. Therefore, after writing a value to this attribute, the value should be read back to know the actual value.
search_initiator_threshold	This attribute is used in the intelligent search initiator process. If the value of the initiator signal is above the value of this attribute, a fast synchronization process is possible. The default value is 98 dBμV.

frequencies	Contains frequencies required for S-FSK modulation. <pre>frequencies_type ::= structure { mark_frequency: double-long-unsigned, space_frequency: double-long-unsigned }</pre> The default unit is Hz.
mac_address	Holds the MIB variable <i>mac-address</i> (variable 3) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. NOTE 4 MAC addresses are expressed on 12 bits. Contains the value of the address of the physical attachment (MAC address) associated to the local system. In the unconfigured state, the MAC address is "NEW-address". This attribute is locally written by the CIASE when the system is registered (with a Register service). The value is used in each outgoing or incoming frame. The default value is "NEW-address". This attribute is set to NEW: – by the MAC sub-layer, once the time-out-not-addressed delay is exceeded; – when a client system "resets" the server system. See the 5.8.4. When this attribute is set to NEW: – the system loses its synchronization (function of the MAC-sublayer); – the <i>mac_group_address</i> attribute is reset (array of 0 elements); – the system automatically releases all AAs which can be released. NOTE 5 The second item is not present in IEC 61334-4-512:2001. The predefined MAC addresses are shown in Table 15.
mac_group_addresses	Holds the MIB variable <i>mac-group-address</i> (variable 4) specified in IEC 61334-4-512:2001 5.3 and in IEC 61334-5-1:2001 4.3.7.6. Contains a set of MAC group addresses used for broadcast purposes. <pre>array mac-address mac-address ::= long-unsigned</pre> The ALL-configured-address, ALL-physical-address and NO-BODY addresses are not included in this list. These ones are internal predefined values. This attribute shall be written by the initiator using DLMS services to declare specific MAC group addresses on a server system. This attribute is locally read by the MAC sublayer when checking the destination address field of a MAC frame not recognized as an individual address or as one of the three predefined values (ALL-configured-address, ALL-physical-address and NO-BODY).
repeater	Holds the MIB variable <i>repeater</i> (variable 5) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. It specifies whether the server system effectively repeats all frames or not. <pre>enum: (0) never repeater, (1) always repeater, (2) dynamic repeater</pre>

repeater cont'd	If the <i>repeater</i> variable is equal to 0, the server system should never repeat the frames. If it is set to 1, the server system is a repeater: it has to repeat all frames received without error and with a current credit greater than zero. If it is set to 2, then the repeater status can be dynamically changed by the server itself. NOTE 6 The value 2 value is not specified in IEC 61334-4-512. This attribute is internally read by the MAC sub-layer each time a frame is received. The default value shall be specified in project specific companion specifications.
repeater_status	Holds the current <i>repeater status</i> of the device. boolean (0) FALSE = no repeater, : (1) TRUE = repeater
min_delta_credit	Holds the MIB variable <i>min-delta-credit</i> (variable 9) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. NOTE 7 Only the three least significant bits are used. The Delta Credit (DC) is the subtraction of the Initial Credit (IC) and Current Credit (CC) fields of a correct received MAC frame. The delta-credit minimum value of a correct received MAC frame, directed to a server system, is held by this variable. The default value is set to the maximal initial credit (see IEC 61334-5-1:2001, 4.2.3.1 for further explanations on the credit and the value of MAX_INITIAL_CREDIT). A client system can reinitialise this variable by setting its value to the maximal initial credit.
initiator_mac_address	Holds the MIB variable <i>initiator-mac-address</i> specified in IEC 61334-5-1:2001, 4.3.7.6. Its value is either the MAC address of the active-initiator or the NO-BODY address, depending on the value of the <i>synchronization_locked</i> attribute (see below). See also IEC 61334-5-1:2001, 3.5.3, 4.1.6.3 and 4.1.7.2.
synchronization_locked	Holds the MIB variable <i>synchronization-locked</i> (variable 10) specified in IEC 61334-4-512:2001, 5.3. Controls the synchronization locked / unlocked state. See IEC 61334-5-1:2001 for more details. If the value of this attribute is equal to TRUE, the system is in the synchronization-locked state. In this state, the <i>initiator-mac-address</i> is always equal to the MAC address field of the active-initiator MIB object. See attribute 2 of the S-FSK Active initiator IC in 5.8.4. If the value of this attribute is equal to FALSE, the system is in the synchronization-unlocked state. In this state, the <i>initiator_mac_address</i> attribute is always set to the NO-BODY value: a value change in the MAC address field of the active-initiator MIB object does not affect the content of the <i>initiator_mac_address</i> attribute which remains at the NO-BODY value. The default value of this variable shall be specified in the implementation specifications. NOTE 8 In the synchronization-unlocked state, the server synchronizes on any valid frame. In the synchronization locked state, the server only synchronizes on frames issued or directed to the client system the MAC address of which is equal to the value of the <i>initiator_mac_address</i> attribute.

transmission_speed	The transmission speed supported by the physical device. See also IEC 61334-5-1:2001, 3.2.2.	
enum:	50Hz	60 Hz
(0)	300 baud	360 baud
(1)	600 baud	720 baud
(2)	1 200 baud	1 440 baud
(3) -- default	2 400 baud	2 880 baud
(4)	4 800 baud	5 760 baud
(5)	7 200 baud	8 640 baud
(6)	9 600 baud	11 520 baud

Table 15 – MAC addresses in the S-FSK profile

Address	Value
NO-BODY	000
Local MAC	001...FIMA-1
Initiator	FIMA...LIMA
MAC group address	LIMA + 1...FFB
All configured	FFC
NEW	FFE
All Physical	FFF
NOTE MAC addresses are expressed on 12 bits. These addresses are specified in IEC 61334-5-1:2001 4.2.3.2, 4.3.7.5.1, 4.3.7.5.2 and 4.3.7.5.3.	
FIMA = First Initiator MAC address; C00	
LIMA = Last Initiator MAC address; DFF	

5.8.4 S-FSK Active initiator (class_id = 51, version = 0)

An instance of the "S-FSK Active initiator" IC stores the data of the active initiator. The active initiator is the client system, which has last registered the server system with a CIASE Register request. See IEC 61334-4-511:2000, 7.2.

S-FSK Active initiator		0...n				class_id = 51, version = 0			
Attributes		Data type		Min.	Max.	Def.	Short name		
1.	logical_name (static)	octet-string					x		
2.	active_initiator (dyn.)	initiator_descriptor					x + 0x08		
Specific methods		m/o							
1.	reset_NEW_not_synchronized						x + 0x10		

Attribute description	
logical_name	Identifies the “S-FSK Active initiator” object instance. See 6.2.22.
active_initiator	Holds the MIB variable <i>active-initiator</i> (variable 15) specified in IEC 61334-4-512:2001, 5.6. Contains the identifiers of the active initiator, which has last registered the system with a Register request. See IEC 61334-4-511:2000, 7.2. The Initiator system is identified with its System Title, MAC address and L-SAP selector: initiator_descriptor ::= structure <pre> { system_title: octet-string, MAC_address: long-unsigned, L_SAP_selector: unsigned } </pre> The size and the structure of the system title may be specified in system specifications. When the system title is used as part of the initialisation vector of cryptographic algorithms, then the size shall meet the requirements applicable for the initialisation vector. The MAC_address element is used to update the <i>initiator-mac-address</i> MAC management variable when the system is configured in the synchronization-locked state. See the specification of the <i>initiator_mac_address</i> and the <i>synchronization_locked</i> attributes of the S-FSK Phy&MAC setup IC in 5.8.3. As long as the server is not registered by an active initiator, the L_SAP_selector field is set to 0 and the system_title field is equal to an octet string of 0s. The default value of the initiator-descriptor is: system_title = octet-string of 0s, MAC_address = NO-BODY and L_SAP_selector = 0. The value of this attribute can be updated by the invocation of the <i>reset_NEW_not_synchronized</i> method or by the CIASE Register service.
Method description	
reset_NEW_not_synchronized (data)	Holds the MIB variable <i>reset-NEW-not-synchronized</i> (variable 17) specified in IEC 61334-4-512:2001, 5.8. Allows a client system to “reset” the server system. The submitted value corresponds to a client MAC address. The writing is refused if: - the value does not correspond to a valid client MAC address or the predefined NO-BODY address; - the submitted value is different from the NO-BODY address and the <i>synchronization_locked</i> attribute is not equal to TRUE. For the description of the Intelligent Search Initiator process, see IEC 62056-8-3:2013, 10.7. When this method is invoked, the following actions are performed: - the system returns to the unconfigured state (UNC: MAC-address equals NEW-address). This transition automatically causes the synchronization lost (function of the MAC sub layer); - the system changes the value of the <i>active_initiator</i> attribute: the MAC_address is set to the submitted value, the L-SAP_selector is set to the value 0 and the system_title is set to an octet-string of 0s; - all AAs that can be released are released.

5.8.5 S-FSK MAC synchronization timeouts (class_id = 52, version = 0)

An instance of the “S-FSK MAC synchronization timeouts” IC stores the timeouts related to the synchronization process.

S-FSK MAC synchronization timeouts		0...n	class_id = 52, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	search_initiator_timeout (static)	long-unsigned				x + 0x08
3.	synchronization_confirmation_timeout (static)	long-unsigned				x + 0x10
4.	time_out_not_addressed (static)	long-unsigned				x + 0x18
5.	time_out_frame_not_OK (static)	long-unsigned				x + 0x20
Specific methods		m/o				

Attribute description

logical_name Identifies the “S-FSK synchronization timeouts” object instance. See 6.2.22.

search_initiator_timeout This timeout supports the intelligent search initiator function.

It defines the value of the time, expressed in seconds, during which the server system is searching for the initiator with the strongest signal.

During this timeout, all initiators, which may be heard by the servers, are expected to talk.

After the expiry of this timeout, the server will accept a Register request from the initiator having provided the strongest signal and it will be locked to that initiator.

If the value of the timeout is equal to 0, this means that the feature is not used.

The timeout is started at the beginning of the Search Initiator Phase, when the server receives the first frame with a valid initiator MAC address. The timeout is restarted when the Search Initiator Phase is over and the server locks on the initiator. During the Check Initiator Phase, it is restarted on the reception of each valid frame.

NOTE A Fast synchronization may be performed if the level of signal is good enough (Level of initiator signal $\geq$ Search-Initiator-Threshold) and one of the MAC addresses (Source or Destination) is an Initiator MAC address. This means that the module (the meter) is next to a DC or next to a module that is already locked on that DC. The module locks in this case on that initiator.

synchronization_confirmation_timeout	Holds the MIB variable <i>synchronization-confirmation-timeout</i> (variable 6) specified in IEC 61334-4-512:2001, 5.3 and IEC 61334-5-1:2001, 4.3.7.6. Defines the value of the time, expressed in seconds, after which a server system which just gets frame synchronized (detection of a data path equal to AAAA54C7 hex) will automatically lose its frame synchronization if the MAC sublayer does not identify a valid MAC frame. The timeout starts after the reception of the first four bytes of a physical frame. The value of this variable can be modified by a client system. This time-out ensures a fast desynchronization of a system, which has synchronized on a wrong physical frame. See IEC 61334-5-1:2001, 3.5.3 for more details. The default value of this variable should be specified in the implementation specifications. A value equal to 0 is equivalent to cancel the use of the related <i>synchronization_confirmation_timeout</i> counter.
time_out_not_addressed	Holds the MIB variable <i>time-out-not-addressed</i> (variable 7) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. Defines the time, in minutes, after which a server system that has not been individually addressed: – returns to the non configured state (UNC: MAC-address equals NEW-address): this transition automatically involves the loss of the synchronization (function of the MAC sub layer) and releasing all AAs that can be released; – loses its active initiator: the MAC address of the active-initiator is set to NO-BODY, the LSAP selector is set to the value 00 and the System Title is set to an octet-string of 0s. Because broadcast addresses are not individual system addresses, the timer associated with the <i>time-out-not-addressed</i> delay ensures that a forgotten system will sooner or later return to the unconfigured state. It will be then discovered again. A forgotten system is a system, which has not been individually addressed for more than the "<i>time-out-not-addressed</i>" amount of time. The default value of this variable should be specified in the implementation specifications. A value equal to 0 is equivalent to cancel the use of the related time-out-not-addressed counter.
time_out_frame_not_OK	Holds the MIB variable <i>time-out-frame-not-OK</i> (variable 8), specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. Defines the time, in seconds, after which a server system that has not received a properly formed MAC frame (incorrect NS field, inconsistent number of received sub frames, false Cyclic Redundancy Code checking) loses its frame synchronization. The default value of this variable shall be specified in the implementation specifications. A value equal to 0 is equivalent to cancel the use of the related <i>time-out-frame-not-OK</i> counter.

5.8.6 S-FSK MAC counters (class_id = 53, version = 0)

An instance of the “S-FSK MAC counters” IC stores counters related to the frame exchange, transmission and repetition phases.

S-FSK MAC counters		0...n	class_id = 53, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	synchronization_register (dyn.)	array				x + 0x08
3.	desynchronization_listing (dyn.)	structure				x + 0x10
4.	broadcast_frames_counter (dyn.)	array				x + 0x18
5.	repetitions_counter (dyn.)	double-long-unsigned			0	x + 0x20
6.	transmissions_counter (dyn.)	double-long-unsigned			0	x + 0x28
7.	CRC_OK_frames_counter (dyn.)	double-long-unsigned			0	x + 0x30
8.	CRC_NOK_frames_counter (dyn.)	double-long-unsigned				x + 0x38
Specific methods		m/o				
1.	reset (data)					x + 0x50

Attribute description

logical_name Identifies the “S-FSK MAC counters” object instance. See 6.2.22.

synchronization_register Holds the MIB variable *synchronization-register* (variable 23), specified in IEC 61334-4-512:2001, 5.8.

array synchronization_couples

synchronization_couples ::= structure

```
{
    mac_address:                long-unsigned,
    synchronizations_counter:    double-long-unsigned
}
```

This variable counts the number of synchronization processes performed by the system. Processes that lead to a synchronization loss due to the detection of a wrong initiator are registered. The other processes that lead to a synchronization loss (time-out, management writing) **are not registered**.

This variable provides a balance sheet of the different systems on which the server system is "potentially" able to synchronize.

A synchronization process is initialized when the Management Application Entity (connection manager) receives a MA_Sync.indication (Synchronization State = SYNCHRO_FOUND) primitive from the MAC Sublayer Entity. This process is registered in the synchronization-register variable only if the MA_Sync.indication (Synchronization State = SYNCHRO_FOUND) primitive is followed by one of the three primitives:

synchronization_
register
(continued)

- 1) MA_Data.indication (DA, SA, MSDU) primitive;
- 2) MA_Sync.indication (Synchronization State = SYNCHRO_CONF, SA, DA);
- 3) MA_Sync.indication (Synchronization State = SYNCHRO_LOSS, Synchro Loss Cause = wrong_initiator, SA, DA).

NOTE The third primitive is only generated if the server system is configured in a synchronization-locked state. See 5.8.3.

Processes which lead to the generation of MA_Sync.indication (Synchronization State = SYNCHRO_LOSS) primitives indicating synchronization loss due to:

- the physical layer;
- the time-out-not-addressed counter;
- setting the mac_address attribute of the S-FSK Phy&MAC setup object to NEW; see 5.8.3; or
- invoking the reset_NEW_not_synchronized method of the S-FSK Active initiator object; see 5.8.4 (this is known as Management Writing)

are not taken into account in this variable.

For details on the MA_Sync.indication service primitive, see IEC 61334-5-1:2001, 4.1.7.1.

If the synchronization process ends with one of the three primitives listed above, the synchronization-register variable is updated by taking into account the SA and DA fields of the primitive.

The updating of the *synchronization-register* variable is carried out as follows:

First, the Management Entity checks the SA and DA fields.

- If one of these fields corresponds to a client MAC address (CMA) the Entity:
 - checks if the client MAC address (CMA) appears in one of the couples contained in the synchronization-register variable;
 - if it appears, the related synchronizations-counter subfield is incremented;
 - if it does not appear, a new (mac-address, synchronizations-counter) couple is added. This couple is initialized to the (CMA, 1) value.

If none of the SA and DA fields correspond to a client MAC address, it is supposed that the system found its synchronization reference on a DiscoverReport type frame. In that case, the mac-address which should be registered in the synchronization-register variable is the predefined NEW value (FFE). The updating of the synchronization-register variable is carried out in the same way as it is done for a normal client MAC address (CMA).

When a *synchronizations-counter* field reaches the maximum value, it automatically returns to 0 on the next increment.

The maximum number of synchronization couples {mac-address, synchronizations-counter} contained in this variable should be specified in the implementation specifications. When this maximum is reached, the updating of the variable follows a First-In-First-Out (FIFO) mechanism: only the newest source MAC addresses are memorized.

The default value of this variable is an empty array.

desynchronization_listing Holds the MIB variable *desynchronization-listing* (variable 24), specified in IEC 61334-4-512:2001, 5.8.

```

structure
{
    nb_physical_layer_desynchronization:
                                                double_long_unsigned,
    nb_time_out_not_addressed_desynchronization:
                                                double_long_unsigned,
    nb_timeout_frame_not_OK_desynchronization:
                                                double_long_unsigned,
    nb_write_request_desynchronization:
                                                double_long_unsigned,
    nb_wrong_initiator_desynchronization:
                                                double_long_unsigned
}

```

This variable counts the number of desynchronizations that occurred depending on their cause. On reception of synchronization loss notification, the Management Entity updates this attribute by incrementing the counter related to the cause of the desynchronization.

When one of the counters reaches the maximum value, it automatically returns to 0 on the next increment.

The default value of this variable is a structure with all elements equal to 0.

broadcast_frames_counter Holds the MIB variable *broadcast-frames-counter* (variable 19) specified in IEC 61334-4-512:2001, 5.8.

array broadcast-couples

```

broadcast-couples::= structure
{
    source-mac-address: long-unsigned,
    frames-counter:     double-long-unsigned
}

```

It counts the broadcast frames received by the server system and issued from a client system (source-mac-address = any valid client-mac-address, destination-mac address = ALL-physical). The number of frames is classified according to the origin of the transmitter. The counter is incremented even if the LLC-destination-address is not valid on the server system. When the frames-counter field reaches its maximum value, it automatically returns to 0 on the next increment.

The maximum number of broadcast-couples {source-mac-address, frames-counter} contained in this variable should be specified in the implementation specifications. When this maximum is reached, the updating of the variable follows a First-In-First-Out (FIFO) mechanism: only the newest source MAC addresses are memorized.

repetitions_counter	Holds the MIB variable <i>repetitions-counter</i> (variable 20) specified in IEC 61334-4-512:2001, 5.8. Counts the number of repetition phases. The repetition phases following a transmission are not counted. If the MAC sub-layer is configured in the no-repeater mode, this variable is not updated. The repetitions counter measures the activity of the system as a repeater. A received frame repeated five times (from CC=4 to CC=0) is counted only once in the repetitions counter since it corresponds to one repetition phase. The counter is incremented at the beginning of each repetition phase. When the repetitions counter reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0.
transmissions_counter	Holds the MIB variable <i>transmissions-counter</i> (variable 21) specified in IEC 61334-4-512:2001, 5.8. Counts the number of transmission phases. A transmission phase is characterized by the transmission and the repetition of a frame. A repetition phase, which follows the reception of a frame, is not counted. The transmission counter is incremented at the beginning of each transmission phase. A client system can write this variable to update the counter. When the transmissions counter reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0.
CRC_OK_frames_counter	Holds the MIB variable <i>CRC-OK-frames-counter</i> (variable 22) specified in IEC 61334-4-512:2001, 5.8 Counts the number of frames received with a correct Frame Check Sequence Field. When the CRC OK frames counter field reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0.
CRC_NOK_frames_counter	Counts the number of frames received with an incorrect Frame Check Sequence Field. When the CRC NOK frames counter field reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0.
Method description	
reset (data)	Clears all counters. data::= integer (0)

5.8.7 IEC 61334-4-32 LLC setup (class_id = 55, version = 1)

An instance of the “IEC 61334-4-32 LLC setup” IC holds parameters necessary to set up and manage the LLC layer as specified in IEC 61334-4-32.

IEC 61334-4-32 LLC setup		0...n				class_id = 55, version = 1			
Attributes		Data type		Min.	Max.	Def.	Short name		
1.	logical_name	(static)	octet-string				x		
2.	max_frame_length	(static)	long-unsigned				x + 0x08		
3.	reply_status_list	(dyn.)	array				x + 0x10		
Specific methods		m/o							

Attribute description

logical_name	Identifies the “IEC 61334-4-32 LLC setup” object instance. See 6.2.22.
max_frame_length	Holds the length of the LLC frame in bytes. See IEC 61334-4-32:1996, 5.1.4. For the S-FSK PLC profile, the minimum / default / maximum values are 26 / 134 / 242 bytes respectively. See IEC 61334-5-1:2001, 4.2.2. NOTE For other lower layer profiles, see the corresponding values in the relevant specification.
reply_status_list	Holds the MIB variable <i>reply-status-list</i> (variable 11) specified in IEC 61334-4-512:2001, 5.4. Lists the L-SAPs that have a not empty RDR (Reply Data on Request) buffer, which has not already been read. The length of a waiting L-SDU is specified in number of sub frames (different from zero). The variable is locally generated by the LLC sub layer. array reply_status reply_status ::= structure <pre> { L_SAP_selector: unsigned, length_of_waiting_L_SDU: unsigned } </pre> length_of_waiting_L_SDU in the case of the S-FSK profile is in number of sub-frames; valid values are 1 to 7.

5.8.8 -FSK Reporting system list (class_id = 56, version = 0)

An instance of the “S-FSK Reporting system list” IC holds the list of reporting systems.

S-FSK Reporting system list		0...n				class_id = 56, version = 0			
Attributes		Data type		Min.	Max.	Def.	Short name		
1.	logical_name	(static)	octet-string				x		
2.	reporting_system_list	(dyn.)	array				x + 0x08		
Specific methods		m/o							

Attribute description

logical_name	Identifies the “S-FSK Reporting system list” object instance. See 6.2.22.
reporting_system_list	Holds the MIB variable <i>reporting-system-list</i> (variable 16) specified in IEC 61334-4-512:2001, 5.7. array system-title system-title ::= octet-string Contains the system-titles of the server systems which have made a DiscoverReport request and which have not already been registered. The list has a finite size and it is sorted upon the arrival. The first element is the newest one. Once full, the oldest ones are replaced by the new ones.

reporting_system_list cont'd	The reporting system list is updated: – when a DiscoverReport CI_PDU is received by the server system (whatever its state: non configured or configured): the CIASE adds the reporting system-title at the beginning of the list, and verifies that it does not exist anywhere else in the list, if so it destroys the old one. A system-title can only be present once in the list; – when a Register CI_PDU is received by the server system (whatever its state: non configured or configured): the CIASE checks the reporting-system list. If a system-title is present in the reporting-system-list and in the Register CI-PDU, the CIASE deletes the system-title in the reporting-system-list: this system is no more considered as a reporting system.

5.9 Interface classes for setting up the LLC layer for ISO/IEC 8802-2

5.9.1 General

This subclause specifies the ICs available for setting up the ISO/IEC 8802-2 LLC layer, used in some DLMS/COSEM communication profiles, in the various types of operation.

For definitions related to the ISO/IEEE 8802-2 LLC layer see 1.4.2 of ISO/IEC 8802-2:1998.

5.9.2 ISO/IEC 8802-2 LLC Type 1 setup (class_id = 57, version = 0)

An instance of the “ISO/IEC 8802-2 LLC Type 1 setup” IC holds the parameters necessary to set up the ISO/IEC 8802-2 LLC layer in Type 1 operation.

ISO/IEC 8802-2 LLC Type 1 setup		0...n	class_id = 57, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	max_octets_ui_pdu (static)	long unsigned			128	x + 0x08
Specific methods		m/o				

Attribute description

logical_name Identifies the “ISO/IEC 8802-2 LLC Type 1 setup” object instance. See 6.2.23.

max_octets_ui_pdu Refer to the appropriate MAC protocol specification for any limitation on the maximum number of octets in a UI PDU. No restrictions are imposed by the LLC sublayer. However, in the interest of having a value that all users of Type 1 LLC may depend upon, all MACs shall at least be capable of accommodating UI PDUs with information fields up to and including 128 octets in length.

See ISO/IEC 8802-2:1998, 6.8.1 *Maximum number of octets in a UI PDU*.

5.9.3 ISO/IEC 8802-2 LLC Type 2 setup (class_id = 58, version = 0)

An instance of the “ISO/IEC 8802-2 LLC Type 2 setup” IC holds the parameters necessary to set up the ISO/IEC 8802-2 LLC layer in Type 2 operation.

ISO/IEC 8802-2 LLC Type 2 setup		0...n	class_id = 58, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. transmit_window_size_k	(static)	unsigned	1	127	1	x + 0x08
3. receive_window_size_rw	(static)	unsigned	1	127	1	x + 0x10
4. max_octets_i_pdu_n1	(static)	long unsigned			128	x + 0x18
5. max_number_transmissions_n2	(static)	unsigned				x + 0x20
6. acknowledgement_timer	(static)	long-unsigned				x + 0x28
7. p_bit_timer	(static)	long-unsigned				x + 0x30
8. reject_timer	(static)	long-unsigned				x + 0x38
9. busy_state_timer	(static)	long-unsigned				x + 0x40
Specific methods		m/o				

Attribute description

logical_name Identifies the “ISO/IEC 8802-2 LLC Type 2 setup” object instance. See 6.2.23.

transmit_window_size_k The transmit window size (k) shall be a data link connection parameter that can never exceed 127. It shall denote the maximum number of sequentially numbered I PDUs that the sending LLC may have outstanding (i.e., unacknowledged). The value of k is the maximum number by which the sending LLC send state variable V(S) can exceed the N(R) of the last received I PDU.

See ISO/IEC 8802-2:1998, 7.8.4 *Transmit window size, k*.

receive_window_size_rw The receive window size (RW) shall be a data link connection parameter that can never exceed 127. It shall denote the maximum number of unacknowledged sequentially numbered I PDUs that the local LLC allows the remote LLC to have outstanding. It is transmitted in the information field of XID (see ISO/IEC 8802-2:1998, 5.4.1.1.2) and applies to the XID sender. The XID receiver shall set its transmit window (k) to a value less than or equal to the receive window of the XID sender to avoid overrunning the XID sender.

See ISO/IEC 8802-2:1998, 7.8.6 *Receive window size, RW*.

max_octets_i_pdu_n1 N1 is a data link connection parameter that denotes the maximum number of octets in an I PDU. Refer to the various MAC descriptions to determine the precise value of N1 for a given medium access method. LLC itself places no restrictions on the value of N1. However, in the interest of having a value of N1 that all users of Type 2 LLC may depend upon, all MACs shall at least be capable of accommodating I PDUs with information fields up to and including 128 octets in length.

See ISO/IEC 8802-2:1998, 7.8.3 *Maximum number of octets in an I PDU, N1*.

max_number_transmissions_n2	N2 is a data link connection parameter that indicates the maximum number of times that a PDU is sent following the running out of the acknowledgment timer, the P-bit timer, the reject timer, or the busy-state timer. See ISO/IEC 8802-2:1998, 7.8.2 <i>Maximum number of transmissions, N2</i>.
acknowledgement_timer	The acknowledgment timer is a data link connection parameter that shall define the time interval during which the LLC shall expect to receive an acknowledgment to one or more outstanding I PDUs or an expected response PDU to a sent unnumbered command PDU. The unit is seconds. See ISO/IEC 8802-2:1998, 7.8.1.1 <i>Acknowledgement timer</i>.
p_bit_timer	The P-bit timer is a data link connection parameter that shall define the time interval during which the LLC shall expect to receive a PDU with the F bit set to “1” in response to a sent Type 2 command with the P bit set to “1”. The unit is seconds. See ISO/IEC 8802-2:1998, 7.8.1.2 <i>P-bit timer</i>.
reject_timer	The reject timer is a data link connection parameter that shall define the time interval during which the LLC shall expect to receive a reply to a sent REJ PDU. The unit is seconds. See ISO/IEC 8802-2:1998, 7.8.1.3 <i>Reject timer</i>.
busy_state_timer	The busy-state timer is a data link connection parameter that shall define the timer interval during which the LLC shall wait for an indication of the clearance of a busy condition at the other LLC. The unit is seconds. See ISO/IEC 8802-2:1998, 7.8.1.4, <i>Busy-state timer</i>.

5.9.4 ISO/IEC 8802-2 LLC Type 3 setup (class_id = 59, version = 0)

An instance of the “ISO/IEC 8802-2 LLC Type 3 setup” IC holds the parameters necessary to set up the ISO/IEC 8802-2 LLC layer in Type 3 operation.

ISO/IEC 8802-2 LLC Type 3 setup		0...n				class_id = 59, version = 0			
Attributes		Data type		Min.	Max.	Def.	Short name		
1.	logical_name (static)	octet-string					X		
2.	max_octets_acn_pdu_n3 (static)	long unsigned					x + 0x08		
3.	max_number_transmissions_n4 (static)	unsigned					x + 0x10		
4.	acknowledgement_time_t1 (static)	long unsigned					x + 0x18		
5.	receive_lifetime_var_t2 (static)	long unsigned					x + 0x20		
6.	transmit_lifetime_var_t3 (static)	long unsigned					x + 0x28		
Specific methods		m/o							

Attribute description

logical_name	Identifies the “ISO/IEC 8802-2 LLC Type 3 setup” object instance. See 6.2.23.
max_octets_acn_pdu_n3	N3 is a logical link parameter that denotes the maximum number of octets in an ACn command PDU. Refer to the various MAC descriptions to determine the precise value of N3 for a given medium access method. LLC places no restrictions on the value of N3. See ISO/IEC 8802-2:1998, 8.6.2 <i>Maximum number of octets in an ACn command PDU, N3</i>.

max_number_transmissions_n4	N4 is a logical link parameter that indicates the maximum number of times that an ACn command PDU is sent by LLC trying to accomplish a successful information exchange. Normally, N4 is set large enough to overcome the loss of a PDU due to link error conditions. If the medium access control sublayer has its own retransmission capability, the value of N4 may be set to one so that LLC does not itself requeue a PDU to the medium access control sublayer. See ISO/IEC 8802-2:1998, 8.6.1 <i>Maximum number of transmissions, N4</i>.
acknowledgement_time_t1	The acknowledgment time is a logical link parameter that determines the period of the acknowledgment timers, and as such shall define the time interval during which the LLC shall expect to receive an ACn response PDU from a specific LLC from which the LLC is awaiting a response PDU. The acknowledgment time shall take into account any delay introduced by the MAC sublayer and whether the timer is started at the beginning or at the end of the sending of the ACn command PDU by the LLC. The proper operation of the procedure shall require that the acknowledgment time be greater than the normal time between the sending of an ACn command PDU and the reception of the corresponding ACn response PDU. If the medium access control sublayer performs its own retransmissions and if the logical link parameter N4 is set to one to prevent LLC from re-queuing a PDU, then the acknowledgment time T1 may be set to infinity, making the acknowledgment timers unnecessary. The unit is seconds. Infinity is indicated by all bits set to 1. See ISO/IEC 8802-2:1998, 8.6.4 <i>Acknowledgement time, T1</i>.
receive_lifetime_var_t2	This time value is a logical link parameter that determines the period of all of the receive variable lifetime timers. T2 shall be longer by a margin of safety than the longest possible period during which the first transmission and all retries of a single PDU may occur. The margin of safety shall take into account anything affecting LLCs perception of the arrival time of PDUs, such as LLC response time, timer resolution, and variations in the time required for the medium access control sublayer to pass received PDUs to LLC. If the destruction of the received state variables is not desired, the value of time T2 may be set to infinity. In this case the receive variable lifetime timer needs not be implemented. The unit is seconds. Infinity is indicated by all bits set to 1. See ISO/IEC 8802-2:1998, 8.6.5 <i>Receive lifetime variable, T2</i>.

transmit_lifetime_var_t3 This time value is a logical link parameter that determines the minimum lifetime of the transmit sequence state variables. T3 shall be longer by a margin of safety than

- 1) the logical link variable T2 at stations to which ACn commands are sent; and
- 2) the longest possible lifetime of an ACn command-response pair. The lifetime of an ACn command-response pair shall take into account the sum of processing time, queuing delays, and transmission time for the command and response PDUs at the local and remote stations.

If the destruction of the transmit state variables is not desired, the value of time T3 may be set to infinity. Note, if the receive variable lifetime parameter, T2 is set to infinity at remote stations to which ACn commands are sent, then the T3 parameter shall be set to infinity at the local station.

The unit is seconds. Infinity is indicated by all bits set to 1.

See ISO/IEC 8802-2:1998, 8.6.6 *Transmit lifetime variable, T3*.

5.10 Interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for PRIME networks

5.10.1 Overview

See also Annex D.

COSEM objects for data exchange using narrowband OFDM PLC profile for PRIME networks, if implemented, shall be located in the Management Logical Device of COSEM servers.

Figure 17 shows an example with a COSEM physical device comprising three logical devices.

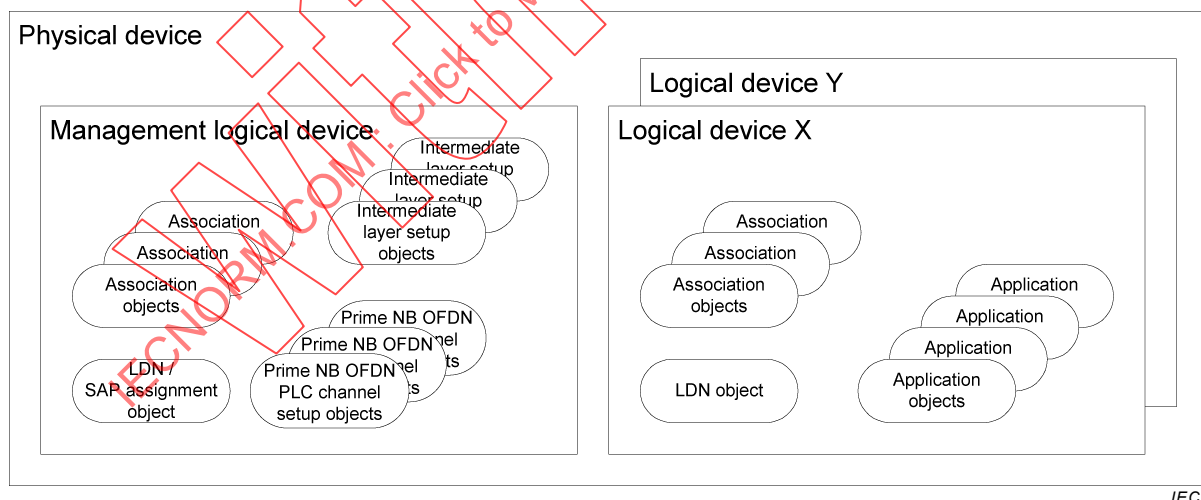


Figure 17 – Object model of DLMS/COSEM servers

Each logical device shall contain a Logical Device Name (LDN) object.

NOTE As in this example there is more than one logical device, the mandatory Management logical device contains a “SAP Assignment” object instead of a Logical Device object.

Each logical device contains one or more “Association” objects, one for each client supported.

The management logical device contains the setup objects of the physical and MAC layers of narrowband OFDM PLC profile for PRIME networks as well as setup objects for the intermediate layer(s). It may contain further application objects.

The other logical devices, in addition to the “Association” and Logical Device Name objects mentioned above, contain further application objects, holding parameters and measurement values.

To set up and manage the 61334-4-32 LLC SCS, one IC is specified:

- “61334-4-32 LLC SCS setup”, see 5.10.3.

To manage the PRIME NB OFDM PLC physical layer (PhL), one IC is specified:

- “PRIME NB OFDM PLC Physical layer counters”, see 5.10.5;

To set up and manage the PLC PRIME OFDM MAC layer, four ICs are specified:

- “PRIME NB OFDM PLC MAC setup”: see 5.10.6;
- “PRIME NB OFDM PLC MAC functional parameters”: see 5.10.7;
- “PRIME NB OFDM PLC MAC counters”: see 5.10.8;
- “PRIME NB OFDM PLC MAC network administration data”: see 5.10.9.

For application identification, one IC is specified:

- “PRIME NB OFDM PLC Application identification” see 5.10.11.

5.10.2 Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes

ITU-T G.9904:2012 defines variables in Table 10-1 and Table 10-2 for PHY PIB attributes, Table 10-3 to Table 10-8 for MAC PIB attributes and Table 10-9 for Applications PIB attributes.

Table 16 shows the mapping of PRIME NB OFDM PLC PIB attributes to attributes of COSEM ICs. Only variables related to the switch and Terminal nodes are mapped. Variables relevant for the base node are not mapped, because the base node acts as a client regarding the distribution network.

Table 16 – Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes

Name	Identifier	Interface class	class_id / attribute
PHY PIB attributes – PHY read-only variable that provide statistical information ¹			
phyStatsCRCIncorrectCount	0x00A0	PRIME NB OFDM PLC Physical layer counters (class_id = 81, version = 0)	81 / Attr. 2
PhyStatsCRCFailCount	0x00A1		81 / Attr. 3
phyStatsTxDropCount	0x00A2		81 / Attr. 4
phyStatsRxDropCount	0x00A3		81 / Attr. 5
phyStatsRxTotalCount	0x00A4		Not modelled
phyStatsBlkAvgEvm	0x00A5		Not modelled
phyEmaSmoothing	0x00A8		Not modelled

Table 16 continued			
Name	Identifier	Interface class	class_id / attribute
PHY read-only parameters, providing information on specific implementation ²			
phyTxQueueLen	0x00B0		Not modelled
phyRxQueueLen	0x00B1		
phyTxProcessingDelay	0x00B2		
phyRxProcessingDelay	0x00B3		
PhyAgcMinGain	0x00B4		
PhyAgcStepValue	0x00B5		
PhyAgcStepNumber	0x00B6		
MAC read-write variables, read-only variables ³			
macMinSwitchSearchTime	0x0010	PRIME NB OFDM PLC MAC setup (class_id = 82, version = 0)	82 / Attr. 2
macMaxPromotionPdu	0x0011		82 / Attr. 3
macMaxPromotionPduTxPeriod	0x0012		82 / Attr. 4
macBeaconsPerFrame	0x0013		82 / Attr. 5
macSCPMaTxAttempts	0x0014		82 / Attr. 6
macCtlReTxTimer	0x0015		82 / Attr. 7
macMaxCtlReTx	0x0018		82 / Attr. 8
macEMASmoothing	0x0019		
macSCPRBO	0x0016	Not modelled	
macSCPChSenseCount	0x0017	Not modelled	
MAC read-only variables that provide functional information ⁴			
macLNID	0x0020	PRIME NB OFDM PLC MAC functional parameters (class_id = 83 version = 0)	83 / Attr. 2
macLSID	0x0021		83 / Attr. 3
macSID	0x0022		83 / Attr. 4
macSNA	0x0023		83 / Attr. 5
macState	0x0024		83 / Attr. 6
macSCPLength	0x0025		83 / Attr. 7
macNodeHierarchyLevel	0x0026		83 / Attr. 8
macBeaconSlotCount	0x0027		83 / Attr. 9
macBeaconRxSlot	0x0028		83 / Attr. 10
macBeaconTxSlot	0x0029		83 / Attr. 11
macBeaconRxFrequency	0x002A		83 / Attr. 12
macBeaconTxFrequency	0x002B		83 / Attr. 13
macCapabilities	0x002C		83 / Attr. 14
MAC read-only variable that provide statistical information ⁵			
macTxDataPktCount	0x0040	PRIME NB OFDM PLC MAC counters (class_id = 84, version = 0)	84 / Attr. 2
macRxDataPktCount	0x0041		84 / Attr. 3
macTxCtrlPktCount	0x0042		84 / Attr. 4
macRxCtrlPktCount	0x0043		84 / Attr. 5
macCSMAFailCount	0x0044		84 / Attr. 6
macCSMAChBusyCount	0x0045		84 / Attr. 7

Table 16 continued			
Name	Identifier	Interface class	class_id / attribute
Read-only lists, made available by MAC layer through management interface ⁶			
macListMcastEntries	0x0052	PRIME NB OFDM PLC MAC network administration data (class_id = 85, version = 0)	85 / Attr. 2
macListSwitchTable	0x0053		85 / Attr. 3
macListDirectTable	0x0055		85 / Attr. 4
macListAvailableSwitches	0x0056		85 / Attr. 5
macListPhyComm	0x0057		85 / Attr. 6
Application PIB attributes ⁷			
AppFwVersion	0x0075	PRIME NB OFDM PLC Application identification (class_id = 86, version = 0)	86 / Attr. 2
AppVendorId	0x0076		86 / Attr. 3
AppProductId	0x0077		86 / Attr. 4
¹ See ITU-T G.9904:2012, Table 10-1.			
² See ITU-T G.9904:2012, Table 10-2.			
³ See ITU-T G.9904:2012, Table 10-3, 10-4.			
⁴ See ITU-T G.9904:2012, Table 10-5.			
⁵ See ITU-T G.9904:2012, Table 10-6.			
⁶ See ITU-T G.9904:2012, Table 10-7.			
⁷ See ITU-T G.9904:2012, Table 10-9.			
NOTE Whereas in COSEM interface class specifications the underscore notation is used, in Recommendation ITU-T G.9904:2012 and in Table 1 – the camel notation is used.			

5.10.3 61334-4-32 LLC SCS setup (class_id = 80, version = 0)

An instance of the “61334-4-32 LLC SCS” (Service Specific Convergence Sublayer, 432 CL) setup IC holds addresses that are provided by the base node during the opening of the convergence layer, as a response to the establish request of the service node. They allow the service node to be part of the network managed by the base node.

61334-4-32 LLC SCS setup		0...n	class_id = 80, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. service_node_address	(dyn.)	long-unsigned				x + 0x08
3. base_node_address	(dyn.)	long-unsigned				x + 0x10
Specific methods		m/o				
1. reset (data)		o				x + 0x20

Attribute description

logical_name	Identifies the “61334-4-32 LLC SCS setup object instance”. See 6.2.24.
service_node_address	Holds the value of the address assigned to the service node during its registration by the base node. After deregistration, the value of this address is NEW, meaning 0xFFE.
base_node_address	Holds the value of the base node address to which the service node is registered. After deregistration this address is 0.

Method description

reset (data) This method is used for deallocating the service node address.
The value of the *service_node_address* becomes NEW and the value of the *base_node_address* becomes 0.

5.10.4 PRIME NB OFDM PLC Physical layer parameters

The physical layer parameters are not modelled.

5.10.5 PRIME NB OFDM PLC Physical layer counters (class_id = 81, version = 0)

An instance of the “PRIME NB OFDM PLC Physical layer counters” IC stores counters related to the physical layers exchanges. The objective of these counters is to provide statistical information for management purposes.

The attributes of instances of this IC shall be read only. They can be reset using the reset method.

PRIME NB OFDM PLC Physical layer counters		0...n	class_id = 81, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	phy_stats_crc_incorrect_count (dyn.)	long-unsigned				x + 0x08
3.	phy_stats_crc_fail_count (dyn.)	long-unsigned				x + 0x10
4.	phy_stats_tx_drop_count (dyn.)	long-unsigned				x + 0x18
5.	phy_stats_rx_drop_count (dyn.)	long-unsigned				x + 0x20
Specific methods		m/o				
1.	reset (data)	o				

Attribute description

logical_name Identifies the “PRIME NB OFDM PLC Physical layer counters” object instance. See 6.2.24.

phy_stats_crc_incorrect_count PIB attribute 0x00A0: Number of bursts received on the physical layer for which the CRC was incorrect.

phy_stats_crc_failed_count PIB attribute 0x00A1: Number of bursts received on the physical layer for which the CRC was correct, but the *Protocol* field of PHY header had invalid value. This count would reflect number of times corrupt data was received and the CRC calculation failed to detect it.

phy_stats_tx_drop_count PIB attribute 0x00A2: Number of times when PHY layer received new data to transmit (PHY_DATA.request) and had to either overwrite on existing data in its transmit queue or drop the data in new request due to full queue.

phy_stats_rx_drop_count PIB attribute 0x00A3: Number of times when the PHY layer received new data on the channel and had to either overwrite on existing data in its receive queue or drop the newly received data due to full queue.

NOTE When a counter reaches the maximum value (0xFFFF), it is automatically rolled-over.

Method description

reset (data) This method is used for resetting all the counters held by an instance of this interface.

5.10.6 PRIME NB OFDM PLC MAC setup (class_id = 82, version = 0)

An instance of the “PRIME NB OFDM PLC MAC setup” IC holds the necessary parameters to set up and manage the PRIME NB OFDM PLC MAC layer.

These attributes influence the functional behaviour of an implementation. These attributes may be defined external to the MAC, typically by the management entity and implementations may allow changes to their values during normal running, i.e. even after the device start-up sequence has been executed.

PRIME NB OFDM PLC MAC setup		0...n	class_id = 82, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. mac_min_switch_search_time	(static)	unsigned	16	32	24	x + 0x08
3. mac_max_promotion_pdu	(static)	unsigned	1	4	2	x + 0x10
4. mac_promotion_pdu_tx_period	(static)	unsigned	2	8	5	x + 0x18
5. mac_beacons_per_frame	(static)	unsigned	1	5	5	x + 0x20
6. mac_scp_max_tx_attempts	(static)	unsigned	2	5	5	x + 0x28
7. mac_ctl_re_tx_timer	(static)	unsigned	2	20	15	x + 0x30
8. mac_max_ctl_re_tx	(static)	unsigned	3	5	3	x + 0x38
Specific methods		n/o				

Attribute description

logical_name Identifies the “PRIME NB OFDM PLC MAC setup” object instance. See 6.2.24.

mac_min_switch_search_time PIB attribute 0x0010: Minimum time for which a service node in *Disconnected* status should scan the channel for beacons before it can broadcast PNPDU. This attribute is not maintained in base nodes.
The unit of this attribute is seconds.

mac_max_promotion_pdu PIB attribute 0x0011: Maximum number of PNPDU's that may be transmitted by a service node in a period of *mac_promotion_pdu_tx_period* seconds. This attribute is not maintained in base nodes.

mac_promotion_pdu_tx_period PIB attribute 0x0012: Time quantum for limiting the number of PNPDU's transmitted from a service node. No more than *mac_max_promotion_pdu* may be transmitted in a period of *mac_promotion_pdu_tx_period*.
The unit of this attribute is seconds.

mac_beacons_per_frame PIB attribute 0x0013: Maximum number of beacon slots that may be provisioned in a frame. This attribute is maintained in base nodes.

mac_scp_max_tx_attempts	PIB attribute 0x0014: Number of times the CSMA algorithm would attempt to transmit requested data when a previous attempt was withheld due to PHY indicating channel busy.
mac_ctl_re_tx_timer	PIB attribute 0x0015: Number of seconds for which a MAC entity waits for acknowledgement of receipt of MAC control packet from its peer entity. On expiry of this time, the MAC entity may retransmit the MAC control packet. The unit of this attribute is seconds.
mac_max_ctl_re_tx	PIB attribute 0x0018: Maximum number of times a MAC entity will try to retransmit an unacknowledged MAC control packet. If the retransmit count reaches this maximum, the MAC entity shall abort further attempts to transmit the MAC control packet. NOTE When a counter reaches the maximum value (0xFFFF), it is automatically rolled-over.

5.10.7 PRIME NB OFDM PLC MAC functional parameters (class_id = 83 version = 0)

The attributes of an instance of the “PRIME NB OFDM PLC MAC functional parameters” IC belong to the functional behaviour of MAC. They provide information on specific aspects.

The attributes of instances of this IC shall be read only.

PRIME NB OFDM PLC MAC functional parameters		0...n	class_id = 83, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. mac_LNID	(static)	long	0	16 383		x + 0x08
3. mac_LSID	(static)	unsigned	0	255		x + 0x10
4. mac_SID	(static)	unsigned	0	255		x + 0x18
5. mac_SNA	(static)	octet-string				x + 0x20
6. mac_state	(static)	enum	0	3		x + 0x28
7. mac_scp_length	(static)	long				x + 0x30
8. mac_node_hierarchy_level	(static)	unsigned	0	63		x + 0x38
9. mac_beacon_slot_count	(static)	unsigned	0	7		x + 0x40
10. mac_beacon_rx_slot	(static)	unsigned	0	7		x + 0x48
11. mac_beacon_tx_slot	(static)	unsigned	0	7		x + 0x50
12. mac_beacon_rx_frequency	(static)	unsigned	0	31		x + 0x58
13. mac_beacon_tx_frequency	(static)	unsigned	0	31		x + 0x60
14. mac_capabilities	(static)	long-unsigned				x + 0x68
Specific methods		m/o				

Attribute description

logical_name	Identifies the “PRIME NB OFDM PLC MAC functional parameters” object instance. See 6.2.24.
mac_LNID	PIB attribute 0x0020: LNID allocated to this node at time of its registration.

mac_LSID	PIB attribute 0x0021: LSID allocated to this node at the time of its promotion. This attribute is not maintained if the node is in a <i>Terminal</i> functional state.
mac_SID	PIB attribute 0x0022: SID of the switch node through which this node is connected to the subnetwork. This attribute is not maintained in a base node.
mac_SNA	PIB attribute 0x0023: Subnetwork address to which this node is registered. The base node returns the SNA it is using.
mac_state	PIB attribute 0x0024: Present functional state of the node. enum: (0) Disconnected, (1) Terminal, (2) Switch, (3) Base
mac_scp_length	PIB attribute 0x0025: The SCP length, in symbols, in present frame.
mac_node_hierarchy_level	PIB attribute 0x0026: Level of this node in subnetwork hierarchy.
mac_beacon_slot_count	PIB attribute 0x0027: Number of beacon slots provisioned in present frame structure.
mac_beacon_rx_slot	PIB attribute 0x0028: Beacon slot in which this device's switch node transmits its beacon. This attribute is not maintained in a base node.
mac_beacon_tx_slot	PIB attribute 0x0029: Beacon slot in which this device transmits its beacon. This attribute is not maintained in service nodes that are in a <i>Terminal</i> functional state.
mac_beacon_rx_frequency	PIB attribute 0x002A: Number of frames between receptions of two successive beacons. A value of 0x0 indicates beacons are received in every frame. This attribute is not maintained in a base node.
mac_beacon_tx_frequency	PIB attribute 0x002B: Number of frames between transmissions of two successive beacons. A value of 0x0 indicates beacons are transmitted in every frame. This attribute is not maintained in service nodes that are in a <i>Terminal</i> functional state.

mac_capabilities	PIB attribute 0x002C: This attribute defines the capabilities of the node. It is a bitmap each bit defining a capability.
	Bit 0: Switch Capable
	Bit 1: Packet Aggregation
	Bit 2: Contention Free Period
	Bit 3: Direct connection
	Bit 4: Multicast
	Bit 5: PHY Robustness Management
	Bit 6: ARQ
	Bit 7: Reserved for future use
	Bit 8: Direct Connection Switching
	Bit 9: Multicast Switching Capability
	Bit 10: PHY Robustness Management Switching Capability
	Bit 11: ARQ Buffering Switching Capability
	Bit 12 to 15: Reserved for future use

5.10.8 PRIME NB OFDM PLC MAC counters (class_id = 84, version = 0)

An instance of the “PRIME NB OFDM PLC MAC counters” IC stores statistical information on the operation of the MAC layer for management purposes. The attributes of instances of this IC shall be read only. They can be reset using the reset method.

PRIME NB OFDM PLC MAC counters		0...n	class_id = 84, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. mac_tx_data_pkt_count	(dyn.)	double-long-unsigned		4 294 967 295		x + 0x08
3. mac_rx_data_pkt_count	(dyn.)	double-long-unsigned		4 294 967 295		x + 0x10
4. mac_tx_ctrl_pkt_count	(dyn.)	double-long-unsigned		4 294 967 295		x + 0x18
5. mac_rx_ctrl_pkt_count	(dyn.)	double-long-unsigned		4 294 967 295		x + 0x20
6. mac_csma_fail_count	(dyn.)	double-long-unsigned		4 294 967 295		x + 0x28
7. mac_csma_ch_busy_count	(dyn.)	double-long-unsigned		4 294 967 295		x + 0x30
Specific methods		m/o				
1. reset (data)		o				x + 0x40

Attribute description

logical_name	Identifies the “PRIME NB OFDM PLC MAC counters” object instance. See 6.2.24.
mac_tx_data_pkt_count	PIB attribute 0x0040: Count of successfully transmitted MSDUs.
mac_rx_data_pkt_count	PIB attribute 0x0041: Count of successfully received MSDUs whose destination address was this node.
mac_tx_ctrl_pkt_count	PIB attribute 0x0042: Count of successfully transmitted MAC control packets.

mac_rx_ctrl_pkt_count	PIB attribute 0x0043: Count of successfully received MAC control packets whose destination was this node.
mac_csma_fail_count	PIB attribute 0x0044: Count of failed CSMA transmit attempts.
mac_csma_ch_busy_count	PIB attribute 0x0045: Count of number of times this node has to back off SCP transmission due to channel busy state. NOTE When a counter reaches the maximum value (0xFFFFFFFF), it is automatically rolled-over.

Method description

reset (data)	This method is used for resetting all the counters held by an instance of this interface class. data ::= integer (0)

5.10.9 PRIME NB OFDM PLC MAC network administration data (class_id = 85, version = 0)

This IC holds the parameters related to the management of the devices connected to the network.

PRIME NB OFDM PLC MAC network administration data	0...n	class_id = 85, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. mac_list_multicast_entries (dyn.)	array				x + 0x08
3. mac_list_switch_table (dyn.)	array				x + 0x10
4. mac_list_direct_table (dyn.)	array				x + 0x18
5. mac_list_available_switches (dyn.)	array				x + 0x20
6. mac_list_phy_comm (dyn.)	array				x + 0x28
Specific methods	m/o				
1. reset (data)	o				x + 0x30

Attribute description

logical_name Identifies the “PRIME NB OFDM PLC MAC network administration data” object instance. See 6.2.24.

mac_list_multicast_entries PIB attribute 0x0052: List of entries in multicast switching table. This list is not maintained in service nodes in a *Terminal* functional state.

mac_list_multicast_entries_type ::= array
mac_list_multicast_entries_element

```
mac_list_multicast_entries_element ::= structure
{
    mcast_entry_LCID:      integer, -- LCID of multicast group
    mcast_entry_members:  long      -- number of child nodes
}
```

The number of child nodes is the number of the members of this group, including the Node itself.

mac_list_switch_table	PIB attribute 0x0053: Switch table. This table is not maintained by service nodes in a <i>Terminal</i> state. mac_switch_table::= array stbl_entry_LSID stbl_entry_LSID::= long -- SID of attached Switch node
mac_list_direct_table	PIB attribute 0x0055: Direct table. mac_direct_table::= array mac_direct_table_element mac_direct_table_element::= structure <pre> { dconn_entry_src_SID: long, dconn_entry_src_LNID: long, dconn_entry_src_LCID: long, dconn_entry_dst_SID: long, dconn_entry_dst_LNID: long, dconn_entry_dst_LCID: long, dconn_entry_DID: octet-string (size 6 bytes) } </pre> Where: – dconn_entry_src_SID is the SID of switch through which the source service node is connected; – dconn_entry_src_LNID is the NID allocated to the source service node; – dconn_entry_src_LCID is the LCID allocated to this connection at the source; – dconn_entry_dst_SID is the SID of the switch through which the destination service node is connected; – dconn_entry_dst_LNID is the NID allocated to the destination service node; – dconn_entry_dst_LCID is the LCID allocated to this connection at the destination; – dconn_entry_DID is the EUI-48 of the direct switch.
mac_list_available_switches	PIB attribute 0x0056: List of switch nodes whose beacons are received. mac_list_available_switches::= array mac_list_available_switches_element mac_list_available_switches_element::= structure <pre> { slist_entry_SNA: octet-string (size 6 bytes), slist_entry_LSID: long, slist_entry_level: integer, slist_entry_rx_level: integer, slist_entry_rx_snr: integer } </pre> Where: – slist_entry_SNA is EUI-48 of the subnetwork; – slist_entry_LSID is SID of this switch; – slist_entry_level is level of this switch in subnetwork hierarchy; – slist_entry_rx_level is the received signal level for this Switch; – slist_entry_rx_snr is the signal to noise ratio for this switch.

mac_list_phy_comm PIB attribute 0x0057: List of PHY communication parameters. It is maintained in every node. For terminal nodes it contains only one entry for the switch the node is connected through. For other nodes it contains also entries for every directly connected child node.

mac_list_phy_comm::= array phy_comm_element

phy_comm_element::= structure

```
{
    phy_Comm_EUI:           octet-string,
    phy_Comm_Tx_Pwr:        integer,
    phy_Comm_Tx_Cod:        integer,
    phy_Comm_Rx_Cod:        integer,
    phy_Comm_Rx_Lvl:        integer,
    phy_Comm_SNR:           integer,
    phy_Comm_Tx_Pwr_Mod:    integer,
    phy_Comm_Tx_Cod_Mod:    integer,
    phy_Comm_Rx_Cod_Mod:    integer
}
```

Where:

- phy_Comm_EUI is the EUI-48 of the other device;
- phy_Comm_Tx_Pwr is the Tx power of GPDU packets sent to the device;
- phy_Comm_Tx_Cod is the Tx coding of GPDU packets sent to the device;
- phy_Comm_Rx_Cod is the Rx coding of GPDU packets received from the device;
- phy_Comm_Rx_Lvl is the Rx power level of GPDU packets received from the device;
- phy_Comm_SNR is the SNR of GPDU packets received from the device;
- phy_Comm_Tx_Pwr_Mod is the number of times the Tx power was modified;
- phy_Comm_Tx_Cod_Mod is the number of times the Tx coding was modified;
- phy_Comm_Rx_Cod_Mod is the number of times the Rx coding was modified.

Method description

reset (data) This method is used for resetting all the entries (to an array of 0 elements) of the attributes 2 to 6 of the instance of this interface class.

data::= integer (0)

5.10.10 PRIME NB OFDM PLC MAC address setup (class_id = 43, version = 0)

An instance of the MAC address setup IC holds the EUI-48 MAC address of the device. The size of this octet string is 6 due to the fact that this address is a EUI-48 and is unique. See also 5.7.4 and 6.2.24.

5.10.11 PRIME NB OFDM PLC Application identification (class_id = 86, version = 0)

An instance of the “PRIME NB OFDM PLC Application identification IC” holds identification information related to administration and maintenance of PRIME NB OFDM PLC devices. They are not communication parameters but allow the device management.

PRIME NB OFDM PLC Application identification		0...n				class_id = 86, version = 0	
Attributes		Data type	Min.	Max.	Def.	Short name	
1.	logical_name (static)	octet-string				x	
2.	firmware_version (static)	octet-string		128		x + 0x08	
3.	vendor_id (static)	long-unsigned				x + 0x10	
4.	product_id (static)	long-unsigned				x + 0x18	
Specific methods		m/o					

Attribute description

logical_name Identifies the device setup object instance. See 6.2.24.

firmware_version PIB attribute 0x0075: Textual description of the firmware version running on the device.

vendor_id PIB attribute 0x0076: Unique vendor identifier assigned by PRIME Alliance.

product_id PIB attribute 0x0077: Vendor assigned unique identifier for specific product.

5.11 Interface classes for setting up and managing the DLMS/COSEM narrowband OFDM PLC profile for G3-PLC networks

5.11.1 Overview

This subclause 5.11 specifies version 1 of interface classes for setting up and managing the MAC and 6LoWPAN Adaptation layers of the DLMS/COSEM G3-PLC profile, based on ITU-T G.9903:2014.

NOTE 1 The use of version 0 of these interface classes based on ITU-T G.9903 Amd. 1:2013 – see subclauses 7.18 to 7.20 – is deprecated.

For this purpose, the elements of the PAN Information Base (PIB) have been mapped to attributes of COSEM ICs.

COSEM objects for data exchange using G3-PLC, if implemented, shall be located in the Management Logical Device of COSEM servers.

To set up and manage the DLMS/COSEM G3-PLC profile layers (including PHY, IEEE 802.15.4:2006 MAC and 6LoWPAN), three ICs are specified:

- “G3-PLC MAC layer counters”, see 5.11.3;
- “G3-PLC MAC setup”, see 5.11.4;
- “G3-PLC 6LoWPAN adaptation layer setup”, see 5.11.5.

An instance of the existing COSEM interface class “MAC address” (class_id = 43, version = 0) is needed to indicate the EUI-48 MAC address of the G3-PLC modem (corresponding to aExtendedAddress constant in IEEE 802.15.4:2006).

IPv6 configuration is provided by an instance of “IPv6 setup” class.

NOTE 2 The PHY layer of ITU-T G.9903:2014 is out of scope of the G3-PLC setup ICs.

5.11.2 Mapping of G3-PLC PIB attributes to COSEM IC attributes

In terms of IEEE 802.15.4:2006, a meter is a Reduced Function Device (RFD) while a concentrator / Neighbourhood Network Access Point (NNAP) is a Full Function Device (FFD) / PAN coordinator. In terms of DLMS/COSEM the meter is the server and the concentrator / NNAP is the client (or an agent for a client).

As COSEM models only the server and not the client, the G3-PLC setup classes concern only the RFD (Reduced Function Device) and not the PAN coordinator.

Table 17 shows the mapping of G3-PLC PIB attributes to attributes of COSEM interface classes.

Table 17 – Mapping of G3-PLC IB attributes to COSEM IC attributes

Name	Identifier	Interface class	class_id / attribute
MAC counters – Read only PIB attributes that provide statistic information ¹			
mac_Tx_data_packet_count	0x0101	G3-PLC MAC layer counters (class_id = 90, version = 1)	90 / Att. 2
mac_Rx_data_packet_count	0x0102		90 / Att. 3
mac_Tx_cmd_packet_count	0x0103		90 / Att. 4
mac_Rx_cmd_packet_count	0x0104		90 / Att. 5
mac_CSMA_fail_count	0x0105		90 / Att. 6
mac_CSMA_no_ACK_count	0x0106		90 / Att. 7
mac_bad_CRC_count	0x0109		90 / Att. 8
mac_Tx_data_broadcast_count	0x0108		90 / Att. 9
mac_Rx_data_broadcast_count	0x0107		90 / Att. 10
MAC setup PIB attributes – Read only and read-write and write only variables ^{1 2}			
mac_short_address	0x0053	G3-PLC MAC setup (class_id = 91, version = 1)	91 / Att. 2
mac_RC_coord	0x010F		91 / Att. 3
mac_PAN_id	0x0050		91 / Att. 4
mac_key_table	0x0071		91 / Att. 5
mac_frame_counter	0x0077		91 / Att. 6
mac_tone_mask	0x0110		91 / Att. 7
mac_TMR_TTL	0x010D		91 / Att. 8
mac_max_frame_retries	0x0059		91 / Att. 9
mac_neighbour_table_entry_TTL	0x010E		91 / Att. 10
mac_neighbour_table	0x010A		91 / Att. 11
mac_high_priority_window_size	0x0100		91 / Att. 12
mac_CSMA_fairness_limit	0x010C		91 / Att. 13
mac_beacon_randomization_window_length	0x0111		91 / Att. 14
mac_A	0x0112		91 / Att. 15
mac_K	0x0113		91 / Att. 16
mac_min_CW_attempts	0x0114		91 / Att. 17
mac_cenelec_legacy_mode	0x0115		91 / Att. 18
mac_FCC_legacy_mode	0x0116		91 / Att. 19
mac_max_BE	0x0047		91 / Att. 20
mac_max_CSMA_backoffs	0x004E		91 / Att. 21
mac_min_BE	0x004F		91 / Att. 22

Table 17 continued			
Name	Identifier	Interface class	class_id / attribute
6LoWPAN adaptation layer IB attributes – Read only and read-write variables ^{3 4}			
adp_max_hops	0x0F	G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 1)	92 / Att. 2
adp_weak_LQI_value	0x1A		92 / Att. 3
adp_security_level	0x00		92 / Att. 4
adp_prefix_table	0x01		92 / Att. 5
adp_routing_configuration	0x09, 0x0A, 0x0D, 0x11-0x19, 0x1B, 0x1F		92 / Att. 6
adp_broadcast_log_table_entry_TTL	0x02		92 / Att. 7
adp_routing_table	0x0C		92 / Att. 8
adp_context_information_table	0x07		92 / Att. 9
adp_blacklist_table	0x1E		92 / Att. 10
adp_broadcast_log_table	0x0B		92 / Att. 11
adp_group_table	0x0E		92 / Att. 12
adp_max_join_wait_time	0x20		92 / Att. 13
adp_path_discovery_time	0x21		92 / Att. 14
adp_active_key_index	0x22		92 / Att. 15
adp_metric_type	0x03		92 / Att. 16
adp_coord_short_address	0x08		92 / Att. 17
adp_disable_default_routing	0xF0		92 / Att. 18
adp_device_type	0x10		92 / Att. 19
¹⁾ See ITU-T G.9903:2014, 9.3.6.2.2 and 9.3.6.2.3.			
²⁾ The following attributes of the G3-PLC MAC sublayer IB attributes have been excluded as there is no need to expose them: <i>macBSN</i> , <i>macDSN</i> , <i>macAckWaitDuration</i> , <i>macFreqNotching</i> , <i>macTimeStampSupported</i> , <i>macPromiscuousMode</i> , <i>macSecurityEnabled</i> .			
³⁾ See ITU-T G.9903:2014, 9.4.1.1.			
⁴⁾ The following attributes of the G3-PLC Adaptation sublayer IB attributes have been excluded as there is no need to expose them: <i>adpSoftVersion</i> , <i>adpSnifferMode</i> .			
NOTE Whereas in ITU-T G.9903:2014 the camel-case notation is used, in COSEM interface class specifications – and in this table – the underscore notation is used.			

5.11.3 G3-PLC MAC layer counters (class_id = 90, version = 1)

An instance of the “G3-PLC MAC layer counters” IC stores counters related to the MAC layer exchanges. The objective of these counters is to provide statistical information for management purposes.

The attributes of instances of this IC shall be read only. They can be reset using the reset method.

G3-PLC MAC layer counters		0...n	class_id = 90, version = 1			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	mac_Tx_data_packet_count (dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x08
3.	mac_Rx_data_packet_count (dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x10
4.	mac_Tx_cmd_packet_count (dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x18
5.	mac_Rx_cmd_packet_count (dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x20
6.	mac_CSMA_fail_count (dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x28
7.	mac_CSMA_no_ACK_count (dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x30
8.	mac_bad_CRC_count (dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x38
9.	mac_Tx_data_broadcast_count (dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x40
10.	mac_Rx_data_broadcast_count (dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x48
Specific methods		m/o				
1.	reset (data)	O				x + 0x50

Attribute description

NOTE When a counter reaches the maximum value (0xFFFFFFFF), it's automatically rolled-over.

logical_name	Identifies the "G3-PLC MAC layer counters" object instance. See 6.2.25.
mac_Tx_data_packet_count	PIB attribute 0x0101: Statistic counter of successfully transmitted data packets (MSDUs).
mac_Rx_data_packet_count	PIB attribute 0x0102: Statistic counter of successfully received data packets (MSDUs).
mac_Tx_cmd_packet_count	PIB attribute 0x0103: Statistic counter of successfully transmitted command packets.
mac_Rx_cmd_packet_count	PIB attribute 0x0104: Statistic counter of successfully received command packets.
mac_CSMA_fail_count	PIB attribute 0x0105: Counts the number of times when CSMA backoffs reach macMaxCSMABackoffs.
mac_CSMA_no_ACK_count	PIB attribute 0x0106: Counts the number of times when an ACK is not received while transmitting a unicast data frame (The loss of ACK is attributed to collisions).
mac_bad_CRC_count	PIB attribute 0x0109: Statistic counter of the number of frames received with bad CRC.
mac_Tx_data_broadcast_count	PIB attribute 0x0108: Statistic counter of the number of broadcast frames sent.
mac_Rx_data_broadcast_count	PIB attribute 0x0107: Statistic counter of successfully received broadcast packets.

Method description

reset (data)	This method forces a reset of the object. By invoking this method, the value of all counters is set to 0. data::= integer (0)

5.11.4 G3-PLC MAC setup (class_id = 91, version = 1)

An instance of the "G3-PLC MAC setup" IC holds the necessary parameters to set up and manage the G3-PLC IEEE 802.15.4:2006 MAC sub-layer.

These attributes influence the functional behaviour of an implementation. Implementations may allow changes to the attributes during normal running, i.e. even after the device start-up sequence has been executed.

G3-PLC MAC setup		0...n	class_id = 91, version = 1			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. mac_short_address	(dyn.)	long-unsigned	0x0000	0xFFFF	0xFFFF	x + 0x08
3. mac_RC_coord	(dyn.)	long-unsigned	0x0000	0xFFFF	0xFFFF	x + 0x10
4. mac_PAN_id	(dyn.)	long-unsigned	0x0000	0xFFFF	0xFFFF	x + 0x18
5. mac_key_table	(dyn.)	array				x + 0x20
6. mac_frame_counter	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x28
7. mac_tone_mask	(static)	bit-string			0x00000000FF FFFFFFFF	x + 0x30
8. mac_TMR_TTL	(static)	unsigned	0	255	2	x + 0x38
9. mac_max_frame_retries	(static)	unsigned	0	10	5	x + 0x40
10. mac_neighbour_table_entry_TTL	(static)	unsigned	0	255	255	x + 0x48
11. mac_neighbour_table	(dyn.)	array				x + 0x50
12. mac_high_priority_window_size	(static)	unsigned	1	7	7	x + 0x58
13. mac_CSMA_fairness_limit	(static)	unsigned	See below	255	25	x + 0x60
14. mac_beacon_randomization_window_length	(static)	unsigned	1	254	12	x + 0x68
15. mac_A	(static)	unsigned	3	20	8	x + 0x70
16. mac_K	(static)	unsigned	1	See below	5	x + 0x78
17. mac_min_CW_attempts	(static)	unsigned	0	255	10	x + 0x80
18. mac_cenelec_legacy_mode	(static)	unsigned	0	255	1	x + 0x88
19. mac_FCC_legacy_mode	(static)	unsigned	0	255	1	x + 0x90
20. mac_max_BE	(static)	unsigned	0	20	8	x + 0x98
21. mac_max_CSMA_backoffs	(static)	unsigned	0	255	50	x + 0xA0
22. mac_min_BE	(static)	unsigned	0	20	3	x + 0xA8
Specific methods		m/o				
1. mac_get_neighbour_table_entry(data)		o				

Attribute description					
logical_name	Identifies the “G3-PLC MAC setup” object instance. See 6.2.25.				
mac_short_address	PIB attribute 0x0053: The 16-bit address the device is using to communicate through the PAN. Its value shall be equal to 0xFFFF when the device does not have a short address. An associated device necessarily has a short address, so that a device cannot be in the state where it is associated but does not have a short address.				
mac_RC_coord	PIB attribute 0x010F: Route cost to coordinator, to be used in the beacon payload as RC_COORD				
mac_PAN_id	PIB attribute 0x0050: The 16-bit identifier of the PAN through which the device is operating. A value equal to 0xFFFF indicates that the device is not associated.				
mac_key_table	PIB attribute 0x0071: This attribute holds GMK keys required for MAC layer ciphering. The attribute can hold up to two 16-bytes keys. The Key Identifier value shall be different for each key. For security reason, the key entries cannot be read, only written. <pre>array mac_GMK mac_GMK ::= structure { key_id: unsigned, key: octet-string }</pre> <table> <tr> <td>key_id</td><td>The Key Identifier used to refer to this key, can take the value 0 or 1.</td></tr> <tr> <td>key</td><td>The AES-128 key used for ciphering the frames exchanged at MAC layer.</td></tr> </table>	key_id	The Key Identifier used to refer to this key, can take the value 0 or 1.	key	The AES-128 key used for ciphering the frames exchanged at MAC layer.
key_id	The Key Identifier used to refer to this key, can take the value 0 or 1.				
key	The AES-128 key used for ciphering the frames exchanged at MAC layer.				
mac_frame_counter	PIB attribute 0x0077: The outgoing frame counter for this device, used when ciphering frames at MAC layer.				
mac_tone_mask	PIB attribute 0x0110: Defines the tone mask to use during symbol formation.				
mac_TMR_TTL	PIB attribute 0x010D: Maximum time to live of tone map parameters entry in the neighbour table in minutes.				
mac_max_frame_retries	PIB attribute 0x0059: Maximum number of retransmissions.				
mac_neighbour_table_entry_TTL	PIB attribute 0x010E: Maximum time to live for an entry in the neighbour table in minutes.				

mac_neighbour_table

PIB attribute 0x010A: See ITU-T G.9903:2014, 9.3.7.2 for CENELEC and FCC bands.

The neighbour table contains information about all the devices within the POS of the device. One element of the table represents one PLC direct neighbour of the device.

array neighbour_table

neighbour_table::= structure

```
{
    short_address:                long-unsigned,
    payload_modulation_scheme:    boolean,
    tone_map:                     bit-string,
    modulation:                   enum,
    tx_gain:                      integer,
    tx_res:                       enum,
    tx_coeff:                     bit-string,
    lqi:                          unsigned,
    phase_differential            integer,
    TMR_valid_time:              unsigned,
    neighbour_valid_time:         unsigned
}
```

NOTE 1 This table is actualized each time any frame is received from a neighbour device, and each time a Tone Map Response is received.

short_address	The MAC Short Address of the node which this entry refers to.

payload_modulation_scheme	Payload Modulation scheme to be used when transmitting to this neighbour. FALSE: Differential, TRUE: Coherent

tone_map	The Tone Map parameter defines which frequency sub-band can be used for communication with the device. A bit set to 1 means that the frequency sub-band can be used, and a bit set to 0 means that frequency sub-band shall not be used.

modulation	The modulation type to use for communicating with the device. enum : (0) Robust Mode, (1) DBPSK, (2) DQPSK, (3) D8PSK, (4) 16-QAM

NOTE 2 The 16-QAM modulation is optional and only applicable for FCC band.

tx_gain	Defines the Tx Gain to use to transmit frames to that device.

tx_res	Defines the Tx Gain resolution corresponding to one gain step. 0: 6 dB, 1: 3 dB

mac_neighbour_table (continued)	tx_coeff	A parameter that specifies transmitter gain for each group of tones represented by one valid bit of the tone map. The receiver measures the frequency-dependent attenuation of the channel and may request the transmitter to compensate for this attenuation by increasing the transmit power on sections of the spectrum that are experiencing attenuation in order to equalize the received signal. Each group of tones is mapped to a 4-bit value for CENELEC-A or a 2-bit value for FCC where a "0" in the most significant bit indicates a positive gain value, hence an increase in the transmitter gain scaled by TXRES is requested for that section and a "1" indicates a negative gain value, hence a decrease in the transmitter gain scaled by TXRES is requested for that section. Implementing this feature is optional and it is intended for frequency selective channels. If this feature is not implemented, the value zero shall be used. <i>Example: in CENELEC bands, with gain values equal to [1, -5, 4, -2, 0, 1], tx_coeff encoding is equal to: '00011101 0100 1010 0000 0001'</i> NOTE 3 One group of tones gathers 6 consecutive tones (or carriers) for CENELEC bands, and 3 consecutive tones for FCC band.
	lqi	Link Quality Indicator of the link to the neighbour (reverse LQI) NOTE 4 LQI value measured during reception of the PPDU from the neighbour. The LQI measurement is a characterization of the strength and/or quality of a received packet.
	phase_differential	Phase difference in multiples of 60 degrees between the mains phase of the local node and the neighbour node. PhaseDifferential can assume six integer values between 0 and 5.
	TMR_valid_time	Remaining time in minutes until which the tone map response parameters in the neighbour table are considered valid. – When the entry is created, this value shall be set to the default value 0. – When it reaches 0, a tone map request may be issued if data is sent to this device. Upon successful reception of a tone map response, this value is set to mac_TMR_TTL.
	neighbour_valid_time	Remaining time in minutes until which this entry in the neighbour table is considered valid. Every time an entry is created or a frame (data or ACK) is received from this neighbour, it is set to mac_neighbour_table_entry_TTL. When it reaches zero, this entry is no longer valid in the table and may be removed.
mac_high_priority_window_size	PIB attribute 0x0100: The high priority contention window size in number of slots.	

mac_CSMA_fairness_limit	PIB attribute 0x010C: Channel access fairness limit. Specifies how many failed back-off attempts, back-off exponent is set to minBE. This attribute can take a value between $2 \times (\text{macMaxBE} - \text{macMinBE})$ and 255.
mac_beacon_randomization_window_length	PIB attribute 0x0111: Duration time in seconds for the beacon randomization.
mac_A	PIB attribute 0x0112: This parameter controls the adaptive CW linear decrease.
mac_K	PIB attribute 0x0113: Rate adaptation factor for channel access fairness limit. This attribute can take a value between 1 and macCSMAFairnessLimit.
mac_min_CW_attempts	PIB attribute 0x0114: Number of consecutive attempts while using minimum CW.
mac_cenelec_legacy_mode	PIB attribute 0x0115: This read only attribute indicates the capability of the node. 0: The following configuration is used (legacy mode): – Elementary interleaving; – Interleaver parameters n_i and n_j are not swapped when $I(i,j) = 0$. 1: The following configuration is used (non legacy mode): – Full Block interleaving; – Interleaver parameters n_i and n_j are swapped when $I(i,j) = 0$.
mac_FCC_legacy_mode	PIB attribute 0x0116: This read only attribute indicates the capability of the node. 0: The following configuration is used (legacy mode): – Differential FCH modulation; – Elementary interleaving; – Interleaver parameters n_i and n_j are not swapped when $I(i,j) = 0$; – Single RS block. 1: The following configuration is used (non legacy mode): – Coherent FCH modulation; – Full Block interleaving; – Interleaver parameters n_i and n_j are swapped when $I(i,j) = 0$; – Two RS blocks.
mac_max_BE	PIB attribute 0x0047: Maximum value of backoff exponent. It should always be greater than macMinBE.
mac_max_CSMA_backoffs	PIB attribute 0x004E: Maximum number of backoff attempts.
mac_min_BE	PIB attribute 0x004F: Minimum value of backoff exponent.

Method description**mac_get_neighbour_table_entry (data)**

This method is used to retrieve the mac neighbour table for one MAC short address. It may be used to perform topology monitoring by the client.

The method invocation parameter contains a mac_short_address.

data::= long-unsigned

The response parameter includes the neighbour table for this mac_short_address.

data::= array neighbour_table

where mac_neighbour_table is as defined in the mac_neighbour_table attribute of the present IC.

5.11.5 G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 1)

An instance of the “G3-PLC 6LoWPAN adaptation layer setup” IC holds the necessary parameters to set up and manage the G3-PLC 6LoWPAN Adaptation layer.

These attributes influence the functional behaviour of an implementation. Implementations may allow changes to their values during normal running, i.e. even after the device start-up sequence has been executed.

G3-PLC 6LoWPAN adaptation layer setup		0...n	class_id = 92, version = 1			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				X
2. adp_max_hops	(static)	unsigned	1	14	8	x + 0x08
3. adp_weak_LQI_value	(static)	unsigned	0	255	52	x + 0x10
4. adp_security_level	(static)	unsigned	0	7	5	x + 0x18
5. adp_prefix_table	(dyn.)	array				x + 0x20
6. adp_routing_configuration	(static)	array				x + 0x28
7. adp_broadcast_log_table_entry_TTL	(static)	long-unsigned	0	65535	2	x + 0x30
8. adp_routing_table	(dyn.)	array				x + 0x38
9. adp_context_information_table	(dyn)	array				x + 0x40
10. adp_blacklist_table	(dyn)	array				x + 0x48
11. adp_broadcast_log_table	(dyn)	array				x + 0x50
12. adp_group_table	(dyn)	array				x + 0x58
13. adp_max_join_wait_time	(static)	long-unsigned	0	1 023	20	x + 0x60
14. adp_path_discovery_time	(static)	unsigned	0	255	40	x + 0x68
15. adp_active_key_index	(static)	unsigned	0	1	0	x + 0x70
16. adp_metric_type	(static)	unsigned	0x00	0x0F	0x0F	x + 0x78
17. adp_coord_short_address	(static)	long-unsigned	0x0000	0x7FFF	0x0000	x + 0x80
18. adp_disable_default_routing	(static)	boolean			FALSE	x + 0x88
19. adp_device_type	(static)	enum	0	2	2	x + 0x90
Specific methods		m/o				

Attribute description																			
logical_name	Identifies the “G3-PLC OFDM 6LoWPAN adaptation layer setup” object instance. See 6.2.25.																		
adp_max_hops	PIB attribute 0x0F: Defines the maximum number of hops to be used by the routing algorithm.																		
adp_weak_LQI_value	PIB attribute 0x1A: The weak link value defines the LQI value below which a link to a neighbour is considered as a weak link. A value of 52 represents an SNR of 3 dB.																		
adp_security_level	PIB attribute 0x00: The minimum security level to be used for incoming and outgoing adaptation frames. Only values 0 (no ciphering) and 5 (ciphering with 32 bits integrity code) are supported.																		
adp_prefix_table	PIB attribute 0x01: Contains the list of prefixes defined on this PAN.																		
adp_routing_configuration	The routing configuration element specifies all parameters linked to the routing mechanism described in ITU-T G.9903:2014. The elements are specified in 9.4.1.2 of that Recommendation. NOTE 1 The Link cost calculation is provided in ITU-T G.9903:2014 , Annex B. <pre> array routing_configuration routing_configuration ::= structure { adp_net_traversal_time: unsigned, adp_routing_table_entry_TTL: long-unsigned, adp_Kr: unsigned, adp_Km: unsigned, adp_Kc: unsigned, adp_Kq: unsigned, adp_Kh: unsigned, adp_Krt: unsigned, adp_RREQ_retries: unsigned, adp_RREQ_RERR_wait: unsigned, adp_blacklist_table_entry_TTL: long-unsigned, adp_unicast_RREQ_gen_enable: boolean, adp_RLC_Enabled: boolean, adp_add_rev_link_cost: unsigned } </pre> <table> <tr> <td>adp_net_traversal_time</td><td>PIB attribute 0x11: Maximum time that a packet is expected to take to reach any node from any node in seconds.</td></tr> <tr> <td></td><td>Range : 0-255</td></tr> <tr> <td></td><td>Default value : 20</td></tr> <tr> <td>adp_routing_table_entry_TTL</td><td>PIB attribute 0x12: Maximum time-to-live of a routing table entry (in minutes).</td></tr> <tr> <td></td><td>Range : 0-65 535</td></tr> <tr> <td></td><td>Default value : 60</td></tr> <tr> <td>adp_Kr</td><td>PIB attribute 0x13: A weight factor for the Robust Mode to calculate link cost.</td></tr> <tr> <td></td><td>Range : 0-31</td></tr> <tr> <td></td><td>Default value : 0</td></tr> </table>	adp_net_traversal_time	PIB attribute 0x11: Maximum time that a packet is expected to take to reach any node from any node in seconds.		Range : 0-255		Default value : 20	adp_routing_table_entry_TTL	PIB attribute 0x12: Maximum time-to-live of a routing table entry (in minutes).		Range : 0-65 535		Default value : 60	adp_Kr	PIB attribute 0x13: A weight factor for the Robust Mode to calculate link cost.		Range : 0-31		Default value : 0
adp_net_traversal_time	PIB attribute 0x11: Maximum time that a packet is expected to take to reach any node from any node in seconds.																		
	Range : 0-255																		
	Default value : 20																		
adp_routing_table_entry_TTL	PIB attribute 0x12: Maximum time-to-live of a routing table entry (in minutes).																		
	Range : 0-65 535																		
	Default value : 60																		
adp_Kr	PIB attribute 0x13: A weight factor for the Robust Mode to calculate link cost.																		
	Range : 0-31																		
	Default value : 0																		

adp_routing_configuration cont'd	adp_Km	PIB attribute 0x14: A weight factor for modulation to calculate link cost. Range : 0-31 Default value : 0
	adp_Kc	PIB attribute 0x15: A weight factor for number of active tones to calculate link cost. Range : 0-31 Default value : 0
	adp_Kq	PIB attribute 0x16: A weight factor for LQI to calculate route cost. Range : 0-50 Default value : 10 for CENELEC A band / 40 for FCC band.
	adp_Kh	PIB attribute 0x17: A weight factor for hop to calculate link cost. Range : 0-31 Default value : 4 for CENELEC A band / 2 for FCC band.
	adp_Krt	PIB attribute 0x1B: A weight factor for the number of active routes in the routing table to calculate link cost. Range : 0-31 Default value : 0
	adp_RREQ_retries	PIB attribute 0x18: The number of RREQ re-transmission in case of RREP reception time out. Range : 0-255 Default value : 0
	adp_RREQ_RERR_wait	PIB attribute 0x19: The number of seconds to wait between two consecutive RREQ – RERR generations. Range : 0-255 Default value : 30
	adp_blacklist_table_entry_TTL	PIB attribute 0x1F: Maximum time-to-live of a blacklisted neighbour entry (in minutes). Range : 0-65 535 Default value : 10
	adp_unicast_RREQ_gen_enable	PIB attribute 0x0D: If TRUE, the RREQ shall be generated with its "unicast RREQ" flag set to '1'. If FALSE, the RREQ shall be generated with its "unicast RREQ" flag set to '0'. Default value : TRUE
	adp_RLC_enabled	PIB attribute 0x09: Enable the sending of RLCREQ frame by the device. Default value : FALSE

adp_routing_configuration cont'd	adp_add_rev_ link_cost	PIB attribute 0x0A: It represents an additional cost to take into account a possible asymmetry in the link. Range : 0-255 Default value : 0
adp_broadcast_log_table_entry_TTL		PIB attribute 0x02: Maximum time to live of an adpBroadcastLogTable entry (in minutes).
adp_routing_table		PIB attribute 0x0C: Contains the routing table. array routing_table routing_table::= structure { destination_address: long-unsigned, next_hop_address: long-unsigned, route_cost: long-unsigned, hop_count: unsigned, weak_link_count: unsigned, valid_time: long-unsigned }
		NOTE 2 This table is actualized each time a route is built or updated (triggered by data traffic) and each time the TTL timer expires.
	destination_address	Address of the destination.
	next_hop_address	Address of the next hop on the route towards the destination.
	route_cost	Cumulative link cost along the route towards the destination.
	hop_count	Number of hops of the selected route to the destination. Range: 0-14 NOTE 3 Practically the maximum allowed value is limited by adp_max_hops.
	weak_link_count	Number of weak links to destination. Range : 0-14 NOTE 4 Practically the maximum allowed value is limited by adp_max_hops.
	valid_time	Remaining time in minutes until when this entry in the routing table is considered valid.
adp_context_information_table		PIB attribute 0x07: Contains the context information associated to each CID extension field. array context_information_table context_information_table::= structure { CID: bit-string, context_length: unsigned, context: octet-string, C: boolean, valid_lifetime: long-unsigned }

adp_context_information_table (continued)	CID	Corresponds to the 4-bit context information used for source and destination addresses (SCI, DCI). Range: 0x00-0x0F
	context_length	Indicates the length of the carried context (up to 128-bit contexts may be carried). Range: 0-128
	context	Corresponds to the carried context used for compression/decompression purposes. Range: 0x0000:0000:0000:0000:0000:0000:0000:0000 – 0xFFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF
	C	Indicates if the context is valid for use in compression. FALSE: Only decompression is allowed. TRUE: Compression and decompression are allowed A context may be used for decompression purposes only. Moreover, recommendations made in RFC 6775 should be followed to take into account the propagation of the context to all nodes of the PAN.
	valid_lifetime	Remaining time in minutes during which the context information table is considered valid. It is updated upon reception of the advertised context. Range: 0-65 535
adp_blacklist_table	PIB attribute 0x1E: Contains the list of the blacklisted neighbours.	
	array blacklisted_neighbour_set	
	blacklisted_neighbour_set::= structure	
	<pre> { blacklisted_neighbour_address: long-unsigned, valid_time: long-unsigned } </pre>	
	blacklisted_neighbour_address	The 16-bit address of the blacklisted neighbour.
	valid_time	Remaining time in minutes until which this entry in the blacklisted neighbour table is considered valid.
adp_broadcast_log_table	PIB attribute 0x0B: Contains the broadcast log table.	
	NOTE 5 This table provides a list of the broadcast packets recently received by this device. array broadcast_log_table broadcast_log_table::= structure <pre> { source_address: long-unsigned, sequence_number: unsigned, valid_time: long-unsigned } </pre>	

adp_broadcast_log_table	source_address	The 16-bit source address of a broadcast packet. This is the address of the broadcast initiator.
cont'd	sequence_number	The sequence number contained in the BC0 header.
	valid_time	Remaining time in minutes until when this entry in the broadcast log table is considered valid.
adp_group_table	PIB attribute 0x0E: Contains the group addresses to which the device belongs. array group_address group_address::= long-unsigned	
	group_address	Group address to which this node has been subscribed.
adp_max_join_wait_time	PIB attribute 0x20: Network join timeout in seconds for LBD.	
adp_path_discovery_time	PIB attribute 0x21: Timeout for path discovery in seconds.	
adp_active_key_index	PIB attribute 0x22: Index of the active GMK to be used for data transmission	
adp_metric_type	PIB attribute 0x03: Metric Type to be used for routing purposes	
adp_coord_short_address	PIB attribute 0x08: Defines the short address of the coordinator.	
adp_disable_default_routing	PIB attribute 0xF0: If TRUE, the default routing (LOADng) is disabled. If FALSE, the default routing (LOADng) is enabled.	
adp_device_type	PIB attribute 0x10: Defines the type of the device connected to the modem: enum: (0) PAN device, (1) PAN coordinator, (2) Not Defined	

5.12 ZigBee® setup classes

5.12.1 Overview

This subclause 5.12 specifies COSEM interface classes required for the external configuration and management of a ZigBee® network to allow interfacing with a multi-part installation that internally uses ZigBee® communications. ZigBee® is a low-power radio communications technology and open standard that is operated by the ZigBee® Alliance, see www.zigbee.org.

ZigBee® is a registered trademark of the ZigBee® Alliance.

NOTE 1 A multi-part installation is one where the meter provides information and/or services to the householder on behalf of the utility. For example, the meter interacts with an in home display, and/or an external load control switch, and/or a smart appliance, to inform the customer of their usage in real time, to control heating devices, and possibly to disconnect peak loads when supply is constrained. While it is possible that the consumer will control the ZigBee® network, in normal operations the utility will control the radio system. This is to ensure that security is maintained for PAN, so that ZigBee® devices such as load switches controlled by the utility operate in a secure manner.

ZigBee® defines a local network of devices linked by radio, with routing and forwarding of messages and with encryption for privacy at network-level.

NOTE 2 Such a local network is known as a PAN – a Personal Area Network. This name is used in the ZigBee® community as ZigBee® is underpinned by the IEEE 802.15.4:2006 standard, which uses the term PAN. This is broadly equivalent to a HAN (Home Area Network – name used in the context of smart metering in the UK) or PAN.

Each PAN has one device designated as ZigBee® coordinator, which has responsibility for creating and managing the network, and which normally acts also as a ZigBee® Trust Center for the management of ZigBee® network, PCLK's and APS Link keys.

There is a process of PAN creation (and corresponding destruction) which is performed by the coordinator; this declares the existence of the network without any devices apart from the coordinator forming part of it. Other ZigBee® devices can join a network created with cooperation of the coordinator, and equally can choose to leave, or can be invited to leave by the coordinator (this is not currently enforceable). Normally devices are members of the network indefinitely; they do not repeatedly join and leave. To create a PAN the coordinator has to receive an external trigger and needs to have setup information including:

- extended PAN ID;
- link keys or install code (for initial communication with new devices);
- radio channel information.

During the process of creating the PAN, the coordinator scans for nearby radio devices, “exchanges” keys, chooses the short addresses, and confirms use of radio channels. Details of the information available from the ZigBee® servers on each device are also exchanged.

NOTE 3 Full details of the joining process are documented in ZigBee® 053474, the ZigBee® specification. More information on ZigBee® technology can be sought at <http://www.zigbee.org/>.

Figure 18 shows an example architecture with a Comms hub on the left, that comprises a DLMS/COSEM server as well as the ZigBee® coordinator. The DLMS/COSEM server has interface objects needed to set up and control the ZigBee® network and may have other COSEM objects to support a metering application. Further, there are two native ZigBee® devices and another DLMS/COSEM server – an electricity meter or another meter type – on the right which is also a “normal” ZigBee® network device.

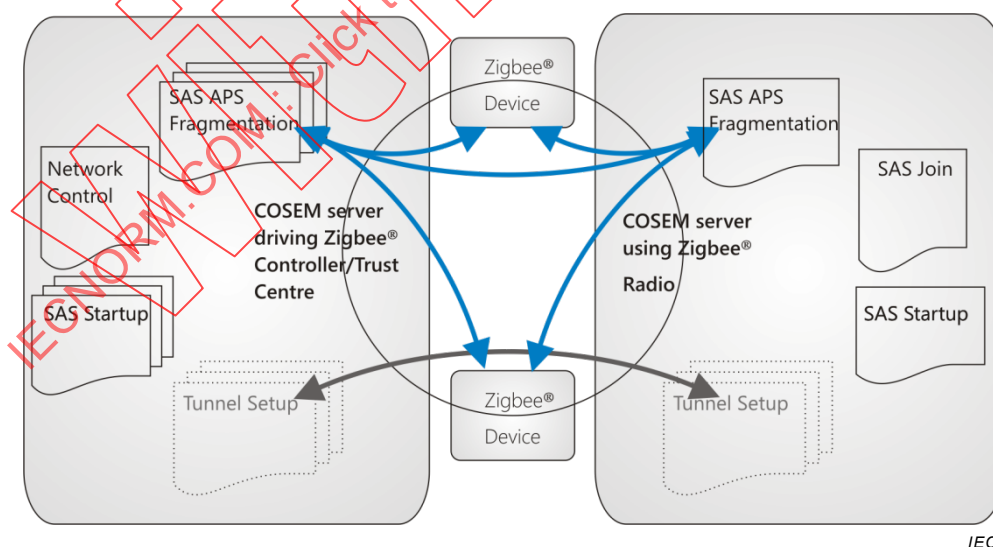


Figure 18 – Example of a ZigBee® network

Any ZigBee® device can be “joined” to the network by remote control.

It is assumed that the Comms hub will also have a further network connection to the WAN, and it is assumed that this is managed by existing DLMS structures – e.g. PSTN, GSM/3G, PLC etc. – but this is out of scope of this standard.

The role of COSEM interface objects in the process of creating / destructing the PAN is to set up ZigBee® parameters and allow a DLMS/COSEM client to trigger actions, typically when commissioning the installation, in a system where WAN communications between a central system and a smart meter installation is by means of DLMS.

Operation of the ZigBee® network is not the responsibility of DLMS/COSEM. The DLMS/COSEM server is merely the vehicle for controlling the ZigBee® network by an external manager (DLMS/COSEM Client).

The use of ZigBee® setup classes in the ZigBee® coordinator and other DLMS/COSEM ZigBee® devices is shown in Table 18.

Table 18 – Use of ZigBee® setup COSEM interface classes

ZigBee® coordinator	Other DLMS/COSEM ZigBee® devices	Reference
ZigBee® SAS startup	ZigBee® SAS startup	5.12.2
–	ZigBee® SAS join	5.12.3
ZigBee® SAS APS fragmentation	ZigBee® SAS APS fragmentation	5.12.4
ZigBee® network control	–	5.12.5
Optionally: ZigBee® tunnel setup	Optionally: ZigBee® tunnel setup	5.12.6

This set of COSEM ICs supports the ZigBee® 2007 and ZigBee® PRO protocol stacks. The ZigBee® IP protocol stack is not supported at this time.

5.12.2 ZigBee® SAS startup (class_id = 101, version = 0)

NOTE 1 In the specification of the ZigBee® COSEM ICs, the length of the octet-strings is indicated for information.

Instances of this IC are used to configure a ZigBee® PRO device with information necessary to create or join the network. The functionality that is driven by this object and the effect on the network depends on whether the object is located in a ZigBee® coordinator or in another ZigBee® device.

ZigBee® SAS startup	0...n	class_id = 101, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. short_address (dyn.)	long-unsigned			0xFFFF	x + 0x08
3. extended_pan_id (dyn.)	octet-string			0	x + 0x10
4. pan_id (dyn.)	long-unsigned			0xFFFF	x + 0x18
5. channel_mask (dyn.)	double-long-unsigned			0	x + 0x20
6. protocol_version (static)	unsigned	0x02		0x02	x + 0x28
7. stack_profile (static)	enum			0x02	x + 0x30
8. start_up_control (dyn.)	unsigned			2	x + 0x38
9. trust_center_address (dyn.)	octet-string			0	x + 0x40
10. link_key (dyn.)	octet-string			0	x + 0x48
11. network_key (dyn.)	octet-string			0	x + 0x50
12. use_insecure_join (static)	boolean			FALSE	x + 0x58
Specific methods	m/o				

Attribute description	
logical_name	Identifies the “ZigBee® SAS startup” object instance. See 6.2.26.
short_address	Defines the 16-bit address by which this device is known locally on the ZigBee® network. Value 0x0000 is given to a coordinator / Trust Center device, and only to these devices. NOTE 2 This short address will be issued to the device by the coordinator when the device joins the network.
extended_pan_ID	Defines the unique address by which the network is known externally. The length of the octet-string is 8 octets.
pan_ID	Defines the 16-bit address by which network is known locally.
channel_mask	Defines (as a bit mask) the set of radio channels which this PAN is permitted to use. Actual usage of channels within this set is determined by the coordinator on the basis of local conditions. The mask is as defined in the ZigBee® specification.
protocol_version	Defines the version of the ZigBee® protocol to be used on the network.
stack_profile	Identifies the capabilities of the ZigBee® stack. enum: (1) ZigBee®, (2) ZigBee® PRO
start_up_control	Indicates the commissioning state of the ZigBee® device: 2: un-commissioned. Indicates that the device will seek to join the network if/when it is established; 0: commissioned. Indicates that the device should consider itself a part of the network indicated by the <i>extended_pan_id</i> attribute. In this case it will not perform any explicit join or rejoin operation. See NOTE 3 below.
trust_center_address	Defines the unique extended address of the Trust Center used for this network. This is often, but not necessarily, the address of the ZigBee® coordinator. The length of the octet-string is 8 octets.
link_key	Defines the key value used to secure point-to-point communications between particular devices within the Smart Energy Profile. The way in which the raw key value is protected is not defined in this specification. This is the APS link key or the PCLK. It is down to the implementation if this attribute is Hashed or in the clear or even if this is used. If this is the Trust Center then this is not usually implemented. This is usually read only, but may need to be writable for testing. The length of the octet-string is 16 octets.

network_key Defines the key value used to secure general communications between devices. The way in which the raw key value is protected is not defined in this specification. The network key value may be set internally by the coordinator.

It is down to the implementation if this attribute is Hashed or in the clear or even if this is used.

If this is the TC then this is not usually implemented.

This is read only.

The length of the octet-string is 16 octets.

use_insecure_join Indicates whether the coordinator is permitted to allow insecure joining as defined by ZigBee® specification.

NOTE 3 The “ZigBee® SAS startup” object is reflecting the state of the ZigBee® HAN to the WAN (or a diagnostic tool connected to a ZigBee® connected DLMS/COSEM Server). For example, if the object is in a Comms hub, then the attribute can be read out to show that the ZigBee® HAN has not been commissioned. The main function of this object is to provide information about a ZigBee® network to a client on the WAN (or via the optical port). The reason that there are multiple instances of the “ZigBee® SAS startup” object in Figure 18 is to express the idea that there may be multiple ZigBee® networks operated from a single Comms Hub.

5.12.3 ZigBee® SAS join (class_id = 102, version = 0)

Instances of this IC configure the behaviour of a ZigBee® PRO device on joining or loss of connection to the network. “ZigBee® SAS join” objects are present in all devices where a DLMS/COSEM server controls the ZigBee® Radio behaviour, but it is not used when a device is acting as coordinator (as the coordinator device creates rather than joins a network). “ZigBee® SAS join” objects can be factory configured, or configured using another communications technique – e.g. an optical port.

ZigBee® SAS join		0...n	class_id = 102, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	scan_attempts (static)	unsigned			3	x + 0x08
3.	time_between_scans (static)	long-unsigned			1	x + 0x10
4.	rejoin_interval	long-unsigned			60	x + 0x18
5.	rejoin_retry_interval (static)	long-unsigned			900	x + 0x20
Specific methods		m/o				

Attribute description

logical_name Identifies the “ZigBee® SAS join” object instance. See 6.2.26.

scan_attempts Defines the number of consecutive scans that a ZigBee® device will perform on each attempt to rejoin a network, in the event of losing contact.

time_between_scans Defines the period in seconds between consecutive scans in a single attempt to rejoin a network.

rejoin_interval Defines the period in seconds that the device should wait after apparently becoming disconnected from a network, before attempting to rejoin.

rejoin_retry_interval Defines the period in seconds for which the device should wait after a failed attempt to rejoin a network before trying again.

Remarks on usage

At boot time or when instructed to join a network, the device should complete up to three (3) (as per the default) scan attempts to find a ZigBee® coordinator or router with which to associate.

If a device has not been commissioned, this means that when the user presses a button or uses another methodology to instruct the device to join a network, it will scan all of the channels up to three times (as per the default) to find a network that allows joining. If it has already been commissioned, it should scan up to three times (as per the default) to find its original PAN to join. (ZigBee® Pro devices should try to find a network using their original extended PAN ID and ZigBee® devices can only try to find a network using their original PAN ID).

Remark on the *rejoin_retry_interval*

Imposes an upper bound on the *rejoin_interval* parameter – this is restarted if device is touched by human user, i.e. by a button press. This parameter is intended to restrict how often a device will scan to find its network in case the network is no longer present and therefore a scan attempt by the device would always fail (i.e., if a device finds it has lost network connectivity, it will try to re-join the network, scanning all channels if necessary). If the scan fails to successfully re-join, the device will wait for 15 min before attempting to re-join again. To be network friendly, it would be recommended to adaptively extend this time period if successive re-joins fail. It would also be recommended the device should try a re-join when triggered (via a control, button, etc.) and fall back to the *rejoin_retry_interval* if attempts to re-join fail again.

5.12.4 ZigBee® SAS APS fragmentation (class_id = 103, version = 0)

Instances of this IC configure the fragmentation feature of ZigBee® PRO transport layer. This fragmentation is not of concern to COSEM; the object merely allows configuration of the fragmentation function by an external manager (DLMS/COSEM client).

Instances of this IC are present in all devices where a DLMS/COSEM server controls the ZigBee® Radio behaviour.

ZigBee® SAS APS fragmentation		class_id = 103, version = 0			
Attributes		Data type		Min.	Max.
1.	logical_name (static)	octet-string			
2.	aps_interframe_delay (static)	long-unsigned			50
3.	aps_max_window_size (static)	long-unsigned			1
Specific methods		m/o			

Attribute description

logical_name	Identifies the “ZigBee® SAS APS fragmentation” object instance. See 6.2.26.
aps_interframe_delay	Defines the delay in milliseconds between sending two blocks of a fragmented transmission.
aps_max_window_size	Defines the maximum number of unacknowledged frames that can be transmitted consecutively.

NOTE These initial values will allow for installation, and setting these values will depend on the specification of the ZigBee® Interface.

5.12.5 ZigBee® network control (class_id = 104, version = 0)

There will be a single instance of the “ZigBee® network control” IC in any device that can act as a ZigBee® coordinator controlled by the DLMS/COSEM client. This class allows interaction between a DLMS/COSEM client (head-end system) and a ZigBee® coordinator at times such as when the installation is commissioned.

ZigBee® network control	0..1	class_id = 104, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. enable_disable_joining	boolean			FALSE	x + 0x08
3. join_timeout (static)	long-unsigned			60	x + 0x10
4. active_devices (dyn.)	array				x + 0x18
Specific methods	m/o				
1. register_device (data)	m				x + 0x20
2. unregister_device (data)	m				x + 0x28
3. unregister_all_devices (data)	o				x + 0x30
4. backup_PAN (data)	o				x + 0x38
5. restore_PAN (data)	o				x + 0x40
6. identify_device (data)	o				x + 0x48
7. remove_mirror (data)	o				x + 0x50
8. update_network_key (data)	o				x + 0x58
9. update_link_key (data)	o				x + 0x60
10. create_PAN (data)	m				x + 0x68
11. remove_PAN (data)	m				x + 0x70

Attribute description

logical_name Identifies the “ZigBee® network control” object instance. See 6.2.26.

enable_disable_joining A flag controlling whether devices are allowed to join the ZigBee® network. The flag is normally FALSE (joining disabled). At certain times – when a new device is expected to join – the flag is set externally to TRUE and then remains set for the duration of the *join_timeout*, or until externally reset if that happens sooner.

join_timeout Defines the time period in seconds during which the coordinator device will permit joining of new devices, following setting of the *enable_disable_joining* flag.

active_devices This attribute shows all of the currently authorised devices within the ZigBee® PAN.

array active_device

active_device::= structure

```
{
    mac_address:                octet-string,
    status:                     bit-string,
    maxRSSI:                    integer,
    averageRSSI:                integer,
    minRSSI:                    integer,
    maxLQI:                     unsigned,
    averageLQI:                 unsigned,
    minLQI:                     unsigned,
    last_communication_date-time: octet-string,
    number_of_hops:              unsigned,
    transmission_failures:       unsigned,
    transmission_successes:      unsigned,
    application_version:         unsigned,
    stack_version:               unsigned
}
```

The length of the mac_address is 8 octets.

The length of the status is 8 bits.

The Max/Average/Min values are taken over the last 24 h period for each device. Period is from 00:00:01 to 00:00:00. They are always historical i.e. they refer to the last day.

status::= bit-string[8]

Bit 0 =	Authorised on PAN
Bit 1 =	Actively reporting on PAN
Bit 2 =	Unauthorised on PAN but has reported
Bit 3 =	Authorised after swap-out
Bit 4 =	SEP Transmitting
Bit 4 =	Reserved
Bit 6 =	Reserved
Bit 7 =	Reserved

Other elements are detailed in the ZigBee® specification.

Method description

register_device (data)

This method is called externally to instruct the coordinator that a device should be added to the list of authorised devices. This method does not actually join the devices to the network, they are just authorised to join at some time in the future.

data::= structure

```
{
    ieee_address:      octet-string,
    key_type:          enum,
    key:               octet-string,
    device_type:       enum,
}
```

Where:

- ieee_address holds the IEEE address of the device. The length of the octet-string is 8 octets;
- key_type determines the type of content of key.

enum:

- (0) Pre-configured Link Key,
- (1) Install code,
- (2)...(255) Reserved,

Install Codes will be subject to agreement across the members who are implementing a ZigBee® network. Therefore the length of install codes, the use of a CRC, and how they are padded when transported in this method needs to be agreed as part of a project specific companion specification.

- key is a pre-configured link key or install code. This will depend on the device being authorised to join the network. Its maximum length is 16 octets,
- device_type is linked to the enumeration shown below so that the coordinator has a method of understanding the possible services that the joining device may require.

NOTE 1 For example, it is likely that an implementation would require Mirror function for Gas, Water and Heat Meters connected by ZigBee®. However, determination of which ZigBee® support is needed for which device will be dependent on the organisation designing the smart metering solution, perhaps a government or a large utility.

register_device
(data)
 (continued)

device-type::= enum

Value	Description
(0)	Electricity Meter
(1)	Gas Meter
(2)	Water Meter
(3)	Thermal Meter
(4)	Pressure Meter
(5)	Heat Meter
(6)	Cooling Meter
(7)	Electric Vehicle charging Meter
(8)	PV Generation Meter
(9)	Wind Turbine Generation Meter
(10)	Water Turbine Generation Meter
(11)	Micro Generation Meter
(12)	Solar Hot Water Generation Meter
(13)	ZigBee® Controlled Load Switch
(14)	ZigBee® Based Boost Button
(128)	IHD
(129)	Range extender
(130)	CAD (Consumer Access Device)
(131)	Thermostat
(132)	Prepayment Terminal
(133)	ZigBee® Controlled Load Switch
(134)	ZigBee® Based Boost Button

unregister_device
(data)

This method is called externally to instruct the coordinator that a device should leave the network.

data::= octet-string

It holds the ieee_address. The length of the octet-string is 8 octets.

unregister_all_devices
(data)

This method is called externally to instruct the coordinator that all devices should leave the network.

data::= integer (0)

NOTE 2 The likely use of this function is to ensure a network is empty of devices prior to destroying it.

backup_PAN
(data)

This method instructs the coordinator to create a back-up of information that would be necessary to re-create the PAN.

NOTE 3 The storage location of the back-up is not currently defined and is an internal function of the DLMS/COSEM server.

data::= integer (0)

Method invocation return parameters are detailed below:

data::= structure

```

{
    date_time:           octet-string,
    extended_PAN_ID:     octet-string,
    devices_to_backup:   array device_to_backup
}

```

backup_PAN (data) cont'd	Where: – date-time refers to the time at which the backup process is due to begin. It is formatted as specified in 4.6.1; – extended_PAN_ID identifies the PAN. The length of the octet-string is 8. <pre>device_to_backup ::= structure { MAC_address: octet-string, hashed_TC_link_key: octet-string }</pre> Where: – MAC_address hold the MAC address. The length of the octet-string is 8; – the length of octet-string holding the hashed_TC_link_key is 16 octets. NOTE 4 The method of hashing the link key is not part of this specification; it is defined in the ZigBee® Smart Energy Specification. MMO is currently used.
restore_PAN (data)	This method instructs the coordinator to restore a PAN using backup information. The storage location of the back-up is not currently defined and is an internal function of the DLMS/COSEM Server. <pre>data ::= structure { extended_PAN_ID: octet-string, devices_to_restore: array device_to_restore }</pre> Where: – extended_PAN_ID identifies the PAN. The length of the octet-string is 8; <pre>device_to_restore ::= structure { MAC_address: octet-string, hashed_TC_link_key: octet-string }</pre> Where: – MAC_address holds the MAC address. The length of the octet-string is 8; – the length of octet-string holding the hashed_TC_link_key is 16 octets. NOTE 5 The method of hashing the link key is not part of this specification; it is defined in the ZigBee® Smart Energy Specification. MMO is currently used.
identify_device (data)	This method is called externally to instruct a device to identify itself to an engineer present on site, for example by sounding a buzzer. <pre>data ::= ieee_address</pre> ieee_address: octet-string ieee_address holds the IEEE address of the device. The length of the octet-string is 8 octets.

remove_mirror (data) This method causes the removal of a ZigBee® mirror that reflects the real device identified by the mac_address parameter.

data::= structure

```
{
    mac_address:      octet-string,
    mirror_control:   bit-string
}
```

Where:

- MAC_address holds the MAC address. The length of the octet-string is 8;
- the mirror_control parameter is provided to support the execution of implementation-specific actions, which should be defined in a project specific companion specification.

EXAMPLE The following example is one of the possible options for the bit-string functionality.

0 =	Force Gas Meter Removal. Where a device removal is forced, the keys values for this device are removed, an APS ack from the device is not required.
1 =	Clear all Mirror Data
2 =	Clear Consumption registers / indexes
3 =	Clear Demand & Max Demand registers
4 =	Clear ZigBee® attributes
5 =	Clear MPAN
6 =	Clear Billing Information
7 =	Clear Logs
8 =	Clear OTA Firmware waiting
9...14 =	Reserved
15 =	Action all

Should the bit be set then the action is carried out.

- The full meaning and actions that the DLMS/COSEM server should undertake are implementation-specific actions, which should be defined in a project specific companion specification.

update_network_key (data) This method requests that the ZigBee® coordinator updates the network key and propagate it to all devices on the PAN. For details of ZigBee® key management, see the ZigBee® specification.

data::= integer (0)

update_link_key (data) This method requests that the ZigBee® coordinator updates the link key and propagate it to the identified device on the PAN. For details of ZigBee® key management, see the ZigBee® specification.

data::= ieee_address

ieee_address: octet-string

ieee_address holds the IEEE address of the device. The length of the octet-string is 8 octets.

create_PAN (data) This method is called externally to instruct a coordinator to create a network using the configuration held in the ZigBee® SAS startup object.

data::= integer (0)

remove_PAN (data) This method is called externally to instruct a coordinator to destroy a network by turning off the ZigBee® radio and removing all of the settings associated with the current PAN.

data::= integer (0)

5.12.6 ZigBee® tunnel setup (class_id = 105, version = 0)

A ZigBee® tunnel is established between two ZigBee® PRO devices to allow DLMS APDUs to be transferred between them. The tunnel in effect extends WAN connectivity to ZigBee® devices not connected to the WAN through a ZigBee® device connected to the same ZigBee® network and connected to the WAN.

The ZigBee® tunnel setup objects would be present on the coordinator and on all other DLMS/COSEM devices that are not connected to the WAN.

Creation of the tunnel is managed on demand and invisibly from the point of view of the DLMS/COSEM client. The target device is implicitly identified by the COSEM addressing information.

NOTE 1 See also the Gateway specification in DLMS UA 1000-2, the “Green Book” Ed. 8.1:2015.

ZigBee® tunnel setup	0...n	class_id = 105, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. maximum_incoming_transfer_size (static)	long-unsigned			0x05DC	x + 0x08
3. maximum_outgoing_transfer_size (static)	long-unsigned			0x05DC	x + 0x10
4. protocol_address (static)	octet-string length			0	x + 0x18
5. close_tunnel_timeout (static)	long-unsigned			0xFFFF	x + 0x20
Specific methods	n/o				

Attribute description

logical_name Identifies the “ZigBee® tunnel setup” object instance. See 6.2.26.

maximum_incoming_transfer_size Defines the maximum size, in octets, of the data packet that can be transferred to the tunnel client in the payload of a single **TransferData** (TransferData is a ZigBee® parameter) command. Behaviour on receipt of a larger packet is not defined in this specification.

maximum_outgoing_transfer_size Defines the maximum size, in octets, of the data packet that can be sent from the tunnel client in the payload of a single **TransferData** command. Behaviour on transmission of a larger packet is not defined in this specification. See NOTE 2 below.

protocol_address The *protocol_address* is implementation-specific, which should be defined in a project specific companion specification.

The length of the octet-string is 6 octets.

close_tunnel_timeout Defines the time, in seconds that the ZigBee® server waits before closing an inactive tunnel on its own (without waiting for the *CloseTunnel* Command from the client within the ZigBee® specification) and freeing its resources

The timer is re-started with each reception of a command.

NOTE 2 The word *outcoming* is unusual, but is adopted for commonality with ZigBee® specifications.

5.13 Maintenance of the interface classes

5.13.1 New versions of interface classes

Any modification of an existing IC affecting the transmission of service requests or responses results in a new version (version::= version+1) and shall be documented accordingly. The following rules shall be followed:

- a) new attributes and methods may be added;
- b) existing attributes and methods may be invalidated BUT the indices of the invalidated attributes and methods shall not be re-used by other attributes and methods;
- c) if these rules cannot be met, then a new IC shall be created;

Any modification of ICs will be recorded by moving the old version of an IC into Clause 7.

5.13.2 New interface classes

The DLMS UA reserves the right to be the exclusive administrator of interface classes.

5.13.3 Removal of interface classes

Besides one association object and the logical device name object no instantiation of an IC is mandatory within a meter. Therefore, even unused ICs will not be removed from the standard. They will be kept to ensure compatibility with possibly existing implementations.

6 Relation to OBIS

6.1 General

This Clause 6 specifies the use of COSEM interface objects to model various data items.

It also specifies the logical names of the objects. The naming system is based on OBIS, the Object Identification System: each logical name is an OBIS code.

OBIS codes are specified in the following subclauses:

- 6.2 specifies the use and the logical names of abstract COSEM objects, i.e. objects not related to an energy type;
- 6.3 specifies the use and logical names for electricity related COSEM objects;
- the detailed OBIS code allocations are specified in IEC 62056-6-1:2015.

Unless otherwise specified the use of value group B shall be:

- if just one object is instantiated, the value in value group B shall be 0;
- if more than one object is instantiated in the same physical device, the value group B shall number the measurement or communication channels as appropriate, from 1...64. This is indicated by the letter “*b*”.

Unless otherwise specified the use of value group E shall be:

- if just one object is instantiated, value in value group E shall be 0;
- if more than one object is instantiated in the same physical device, the value group E shall number the instantiations from zero to the maximum value needed. This is indicated by the letter “*e*”. For the values allocated, see IEC 62056-6-1:2015.

All codes, which are not explicitly listed, but which are outside the manufacturer, utility or consortia specific ranges are reserved for future use.

6.2 Abstract COSEM objects

6.2.1 Use of value group C

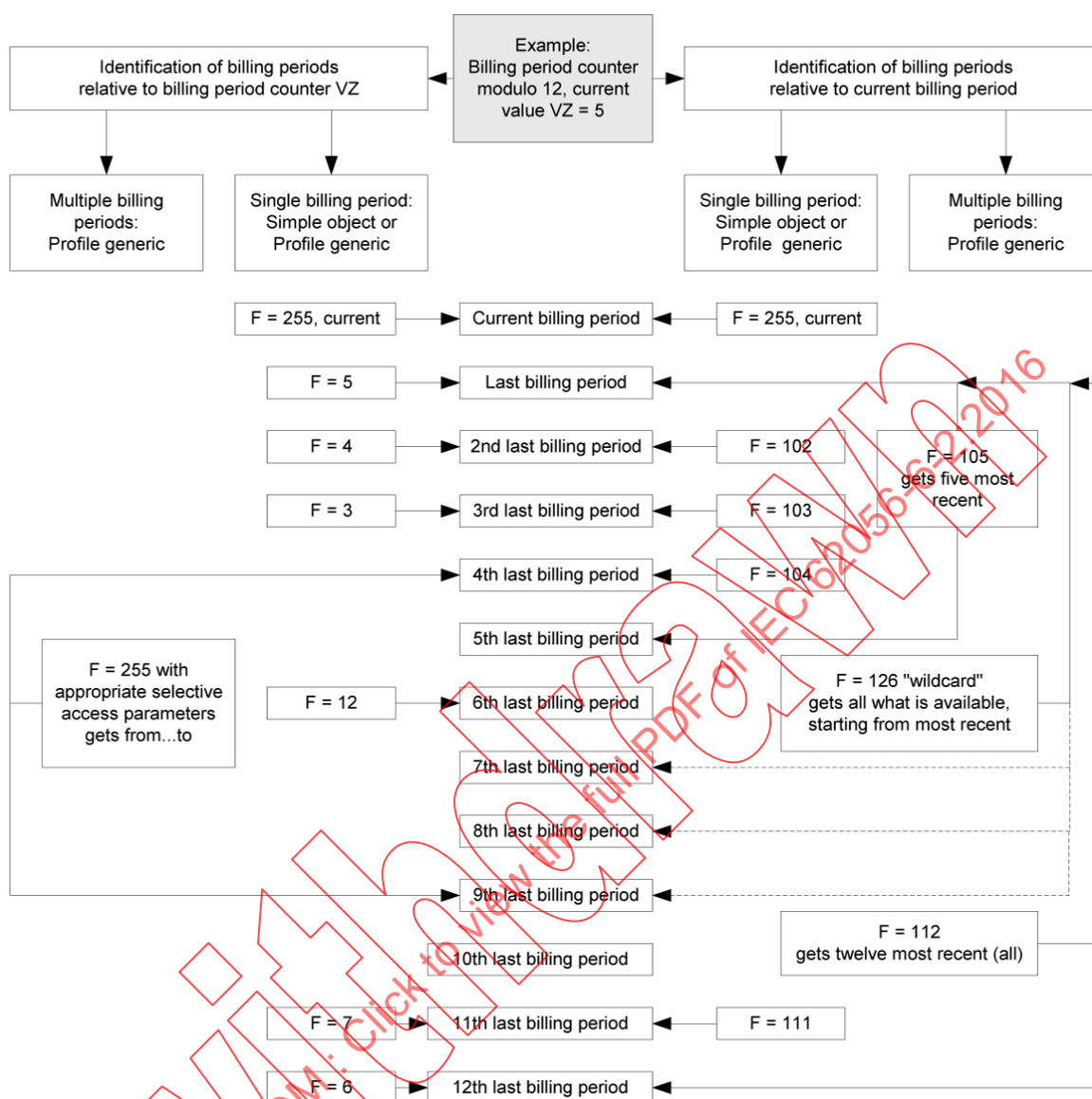
Table 19 shows the use of value group C for abstract objects in the COSEM context. See also IEC 62056-6-1:2015, Table 4.

Table 19 – Use of value group C for abstract objects in the COSEM context

Value group C Abstract objects (A = 0)	
0	General purpose COSEM objects
1	Instances of IC "Clock"
2	Instances of IC "Modem configuration" and related ICs
10	Instances of IC "Script table"
11	Instances of IC "Special days table"
12	Instances of IC "Schedule"
13	Instances of IC "Activity calendar"
14	Instances of IC "Register activation"
15	Instances of IC "Single action schedule"
16	Instances of IC "Register monitor", "Parameter monitor"
17	Instances of IC "Limiter"
20	Instances of IC "IEC local port setup"
21	Standard readout definitions
22	Instances of IC "IEC HDLC setup"
23	Instances of IC "IEC twisted pair (1) setup"
24	COSEM objects related to M-Bus
25	Instances of IC "TCP-UDP setup", "IPv4 setup", "IPv6 setup", "MAC address setup", "PPP setup", "GPRS modem setup", "SMTP setup", "GSM diagnostic", "FTP setup", "Push setup"
26	COSEM objects for data exchange using S-FSK PLC networks
27	COSEM objects for ISO/IEC 8802-2 LLC layer setup
28	COSEM objects for data exchange using narrow-band OFDM PLC for PRIME networks
29	COSEM objects for data exchange using narrow-band OFDM PLC for G3-PLC networks
30	COSEM objects for data exchange using ZigBee®
31	Instances of IC "Wireless Mode Q" (M-Bus)
40	Instances of IC "Association SN/LN"
41	Instances of IC "SAP assignment"
42	COSEM logical device name
43	COSEM objects related to security: Instances of IC "Security setup"
44	Instances of IC "Image transfer"
65	Instances of IC "Utility tables"
128...199	Manufacturer specific COSEM related abstract objects
All other	Reserved

6.2.2 Data of historical billing periods

COSEM provides three mechanisms to represent values of historical billing periods. This is shown in Figure 19 and is described below:



IEC

Figure 19 – Data of historical billing periods – example with module 12, VZ = 5

- a value of a single historical billing period may be represented using the same IC as used for representing the value of the current billing period. With $F = 0 \dots 99$, the billing period is identified by the value of the billing period counter VZ. $F = VZ$ identifies the youngest value, $F = VZ - 1$ identifies the second youngest value etc. $F = 101 \dots 125$ identifies the last, second last ... 25th last billing period. ($F = 255$ identifies the current billing period). Simple objects can only be used to represent values of historical billing periods, if "Profile generic" objects are not implemented;
- a value of a single historical billing period may also be represented by "Profile generic" objects, which are one entry deep, and contain the historical value itself and the time stamp of the storage. With $F = 0 \dots 99$, the billing period is identified by the value of the billing period counter VZ. $F = VZ$ identifies the youngest value, $F = VZ - 1$ identifies the second youngest value etc. $F = 101$ identifies the most recent billing period;
- values of multiple historical billing periods are represented with "Profile generic" objects, with suitable controlling attributes. With $F = 102 \dots 125$ the two last, ... 25 last values can be reached. $F = 126$ identifies an unspecified number of historical values;

6.2.5 Clock objects (class_id = 8)

Instances of the IC “Clock” – see 5.4.1 – control the system clock of the physical device.

Instances of the Interface class “Data” hold the UNIX time with data type *double-long-unsigned*. The epoch time is 1st January 1970.

Clock objects	IC	OBIS code					
		A	B	C	D	E	F
Clock	8, Clock	0	<i>b</i>	1	0	<i>e</i>	255
UNIX clock	1, Data ^a	0	<i>b</i>	1	1	<i>e</i>	255

^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

6.2.6 Modem configuration and related objects

In this group, the following objects are available:

- Instances of the IC “Modem configuration” – see 5.5.4 – define and control the behaviour of the device regarding the communication through a modem;
- Instances of the IC “Auto connect” – see 5.5.6 – define the necessary parameters for the management of sending information from the metering device to one or more destinations and for connection to the network;
- Instances of the IC “Auto answer” – see 5.5.5 – define and control the behaviour of the device regarding the auto answering function using a modem and handling wake-up calls and messages.

Modem configuration and related objects	IC	OBIS code					
		A	B	C	D	E	F
Modem configuration	27, Modem configuration	0	<i>b</i>	2	0	0	255
Auto connect	29, Auto connect	0	<i>b</i>	2	1	0	255
Auto answer	28, Auto answer	0	<i>b</i>	2	2	0	255

6.2.7 Script table objects (class_id = 9)

Instances of the IC “Script table” – see 5.4.2 – control the behaviour of the device.

Several instances are predefined and normally available as hidden scripts only with access to the *execute* () method. The following table contains only the identifiers for the “standard” instances of the listed scripts. Implementation specific instances of these scripts should use values different from zero in value group D.

- *MDI reset / End of billing period* “Script table” objects define the actions to be performed at the end of the billing period, for example the reset of maximum demand indicator registers and archiving data. If there are several billing period schemes available, then there shall be one script present in the array of scripts for each billing period scheme.
- *Tariffication* “Script table” objects define the entry point into tariffication by standardizing utility-wide how to invoke the activation of certain tariff conditions;
- *Disconnect control* “Script table” objects hold the scripts to invoke the methods of “Disconnect control” objects;
- *Image activation* “Script table” objects are used to locally activate an Image transferred to the server, at the date and time held by an Image activation “Single action schedule” object;
- *Push* “Script table” objects hold scripts to activate the push operation. Normally every entry in the array of scripts calls the push method of one “Push setup” object instance;
- *Broadcast* “Script table” objects allow standardising utility wide the entry point into regularly needed functionality.

Script table objects	IC	OBIS code					
		A	B	C	D	E	F
Global meter reset ^a Script table	9, Script table	0	<i>b</i>	10	0	0	255
MDI reset / End of billing period ^a Script table		0	<i>b</i>	10	0	1	255
Tariffication Script table		0	<i>b</i>	10	0	100	255
Activate test mode ^a Script table		0	<i>b</i>	10	0	101	255
Activate normal mode ^a Script table		0	<i>b</i>	10	0	102	255
Set output signals Script table		0	<i>b</i>	10	0	103	255
Switch optical test output ^{b, c} Script table		0	<i>b</i>	10	0	104	255
Power quality measurement management Script table		0	<i>b</i>	10	0	105	255
Disconnect control Script table		0	<i>b</i>	10	0	106	255
Image activation Script table		0	<i>b</i>	10	0	107	255
Push Script table		0	<i>b</i>	10	0	108	255
Broadcast Script table		0	<i>b</i>	10	0	125	255
^a The activation of these scripts is performed by calling the execute() method to the script identifier 1 of the corresponding script object.							
^b The optical test output is switched to measuring quantity Y and the test mode is activated by calling the execute method of the script table object 0.x.10.0.104.255 using Y as parameter, where Y is given by see IEC 62056-6-1:2015, Table 12. The default value of A is 1 (Electricity).							
EXAMPLE In the case of electricity meters, A = 1, default, execute (21) switches the test output to display the active power + of phase 1.							
^c The optical test output is also switched back to its default value when this script is activated.							

6.2.8 Special days table objects (class_id = 11)

Instances of the IC “Special days table” – see 5.4.4 – define and control the behaviour of the device regarding calendar functions on special days for clock control.

Special days table objects	IC	OBIS code					
		A	B	C	D	E	F
Special days table	11, Special days table	0	<i>b</i>	11	0	<i>e</i>	255

6.2.9 Schedule objects (class_id = 10)

Instances of the IC “Schedule” – see 5.4.3 – define and control the behaviour of the device in a sequenced way.

Schedule objects	IC	OBIS code					
		A	B	C	D	E	F
Schedule	10, Schedule	0	<i>b</i>	12	0	<i>e</i>	255

6.2.10 Activity calendar objects (class_id = 20)

Instances of the IC “Activity calendar” – see 5.4.5 – define and control the behaviour of the device in a calendar-based way.

Activity calendar objects	IC	OBIS code					
		A	B	C	D	E	F
Activity calendar	20, Activity calendar	0	<i>b</i>	13	0	<i>e</i>	255

6.2.11 Register activation objects (class_id = 6)

Instances of the IC “Register activation” – see 5.2.5 – are used to handle different tariffication structures.

Register activation objects	IC	OBIS code					
		A	B	C	D	E	F
Register activation	6, Register activation	0	<i>b</i>	14	0	<i>e</i>	255

6.2.12 Single action schedule objects (class_id = 22)

Instances of the IC “Single action schedule” – see 5.4.7 – control the behaviour of the device. Implementation specific instances should use values different from zero in value group D.

Instances of Push “Single action schedule” objects activate scripts in Push “Script table” objects, which invoke the push method of the appropriate “Push setup” objects.

Single action schedule objects	IC	OBIS code					
		A	B	C	D	E	F
End of billing period Single action schedule	22, Single action schedule	0	<i>b</i>	15	0	0	255
Disconnect control Single action schedule		0	<i>b</i>	15	0	1	255
Image activation Single action schedule		0	<i>b</i>	15	0	2	255
Output control Single action schedule		0	<i>b</i>	15	0	3	255
Push Single action schedule		0	<i>b</i>	15	0	4	255

6.2.13 Register monitor objects (class_id = 21)

Instances of the IC “Register monitor” – see 5.4.6 – control the register monitoring function of the device. They define the value to be monitored, the set of thresholds to which the value is compared, and the actions to be performed when a threshold is crossed.

In general, the logical name(s) shown in the table below shall be used. See also 6.3.9 and 6.3.10.

Alarm monitor objects monitor *Alarm register* or *Alarm descriptor* objects.

Register monitor objects	IC	OBIS code					
		A	B	C	D	E	F
Register monitor	21, Register monitor	0	<i>b</i>	16	0	<i>e</i>	255
Alarm monitor		0	<i>b</i>	16	1	0...9	255

6.2.14 Parameter monitor objects (class_id = 65)

Instances of the IC “Parameter monitor” – see 5.4.10 – control the Parameter monitoring function of the device. They define the list of parameters to be monitored and hold the identifier and the value of the last parameter changed, as well as the *capture_time*.

Parameter monitor objects	IC	OBIS code					
		A	B	C	D	E	F
Parameter monitor	65, Parameter monitor	0	<i>b</i>	16	2	<i>e</i>	255

6.2.15 Limiter objects (class_id = 71)

Instances of the IC “Limiter” handle the monitoring of values in normal and emergency conditions. See also 5.4.9.

Limiter objects	IC	OBIS code					
		A	B	C	D	E	F
Limiter	71, Limiter	0	<i>b</i>	17	0	<i>e</i>	255

6.2.16 IEC local port setup objects (class_id = 19)

These objects define and control the behaviour of local ports using the protocol specified in IEC 62056-21:2002. See also 5.5.1.

IEC local port setup objects	IC	OBIS code					
		A	B	C	D	E	F
IEC optical port setup	19, IEC local port setup	0	<i>b</i>	20	0	0	255
IEC electrical port setup		0	<i>b</i>	20	0	1	255

6.2.17 Standard readout profile objects (class_id = 7)

A set of objects is defined to carry the standard readout as it would appear with IEC 62056-21:2002 (modes A to D). See also 5.2.6.

Standard readout objects	IC	OBIS code					
		A	B	C	D	E	F
General local port readout	7, Profile generic	0	<i>b</i>	21	0	0	255
General display readout		0	<i>b</i>	21	0	1	255
Alternate display readout		0	<i>b</i>	21	0	2	255
Service display readout		0	<i>b</i>	21	0	3	255
List of configurable meter data		0	<i>b</i>	21	0	4	255
Additional readout profile 1		0	<i>b</i>	21	0	5	255
.....							
Additional readout profile <i>n</i>		0	<i>b</i>	21	0	N	255

For the parametrization of the standard readout “Data” objects can be used.

Standard readout parametrization objects	IC	OBIS code					
		A	B	C	D	E	F
Standard readout parametrization	1, Data	0	<i>b</i>	21	0	<i>e</i>	255

6.2.18 IEC HDLC setup objects (class_id = 23)

Instances of the IC “IEC HDLC setup” – see 5.5.2 – hold the parameters of the HDLC based data link layer.

IEC HDLC setup objects	IC	OBIS code					
		A	B	C	D	E	F
IEC HDLC setup	23, IEC HDLC setup	0	<i>b</i>	22	0	0	255

6.2.19 IEC twisted pair (1) setup objects (class_id =24)

An instance of the IC “IEC twisted pair (1) set up” IC – see 5.5.3 – stores the parameters necessary to manage a communication profile specified in IEC 62056-3-1:2013.

An instance of the IC “MAC address set up” IC stores the Secondary Station Address ADS.

An instance of the “Data” stores the Fatal Error register.

Instances of the IC “Profile generic” IC instances allow the configuration of IEC 62056-3-1 readout lists.

IEC twisted pair (1) setup and related objects	IC	OBIS code					
		A	B	C	D	E	F
IEC twisted pair (1) setup	24, IEC twisted pair (1) setup	0	b	23	0	0	255
IEC twisted pair (1) MAC address setup	43, MAC address setup	0	b	23	1	0	255
IEC twisted pair (1) Fatal Error register	1, Data	0	b	23	2	0	255
IEC 62056-3-1 Short readout	7, Profile generic	0	b	23	3	0	255
IEC 62056-3-1 Long readout		0	b	23	3	1	255
IEC 62056-3-1 Alternate readout profile 0		0	b	23	3	2	255
IEC 62056-3-1 Additional readout profile 1		0	b	23	3	3	255
IEC 62056-3-1 Additional readout profile 2		0	b	23	3	4	255
IEC 62056-3-1 Additional readout profile 7		0	b	23	3	9	255

6.2.20 Objects related to data exchange over M-Bus

The following objects are available to model and control data exchange using the M-Bus protocol specified in the EN 13757 series:

- instances of the IC “M-Bus slave port setup” define and control the behaviour of M-Bus slave ports of a DLMS/COSEM device. See 5.6.1;
- instances of the IC “M-Bus client” are used to configure DLMS/COSEM devices as M-Bus clients. There is one “M-Bus client” object for each M-Bus slave. Value group B identifies the M-Bus channels. See 5.6.2;
- M-Bus value objects, instances of the IC “Extended register”, hold the values captured from M-Bus slave devices on the relevant channel. The link between the M-Bus client setup objects and the M-Bus value objects is provided by the channel number.
- M-Bus “Profile generic” objects capture M-Bus value objects possibly along with other, not M-Bus specific objects;
- M-Bus “Disconnect control” objects control disconnect devices of M-Bus devices (e.g. gas valves);
- instances of the IC “Wireless mode Q” define and control the behaviour of the device regarding the communication parameters according to mode Q of EN 13757-5:2015. A node having more than one network address, i.e. a multi-homed node, will have multiple objects of these types. See 5.6.3;
- M-Bus control log objects are instances of the IC “Profile generic”. They log the changes of the state of the disconnect devices;
- instances of the IC “M-Bus master port setup” define and control the behaviour of M-Bus master ports of DLMS/COSEM devices, allowing to exchange data with M-Bus slaves. See 5.6.4.

Objects related to data exchange over M-Bus	IC	OBIS code					
		A	B	C	D	E	F
M-Bus slave port setup	25, M-Bus slave port setup	0	b	24	0	0	255
M-Bus client	72, M-Bus client	0	b	24	1	0	255
M-Bus value	4, Extended register	0	b	24	2	e ^a	255
M-Bus profile generic	7, Profile generic	0	b	24	3	e	255
M-Bus disconnect control	70, Disconnect control	0	b	24	4	0	255
M-Bus control log	7, Profile generic	0	b	24	5	0	255
M-Bus master port setup	74, M-Bus master port setup	0	b	24	6	0	255
Wireless Mode Q channel	73, Wireless Mode Q channel	0	b	31	0	0	255
^a “e” is equal to the index of the captured value in accordance to index of <i>capture_definition</i> element in the <i>capture_definition</i> attribute of the M-Bus client object. ^b If there is more than one M-Bus network interface present then there may be one object instantiated for each interface. For example, if a device has two interfaces (one wired M-Bus and one wireless M-Bus) and uses the DLMS/COSEM M-Bus communication profiles on both, there shall be one instance for each interface.							

6.2.21 Objects to set up data exchange over the Internet

In this group, the following objects are available:

- Instances of the IC “TCP-UDP setup” – see 5.7.1 – handle all information related to the setup of the TCP and UDP layer of the Internet based communication profile(s), and point to the IP setup object(s) handling the setup of the IP layer on which the TCP-UDP connection(s) is (are) used;
- Instances of the IC “IPv4 setup” – see 5.7.2 – handle all information related to the setup of the IPv4 layer of the Internet based communication profile(s) and point to the data link layer setup object(s) handling the setup of the data link layer on which the IP connections is (are) used;
- Instances of the IC “IPv6 setup” – see 5.7.3 – handle all information related to the setup of the IPv6 layer of the Internet based communication profile(s) and point to the data link layer setup object(s) handling the setup of the data link layer on which the IP connections is (are) used;
- Instances of the IC “MAC address setup” – see 5.7.4 – handle all information related to the setup of the Ethernet data link layer of the Internet based communication profile(s);
- Instances of the IC “PPP setup” – see 5.7.5 – handle all information related to the setup of the PPP data link layer of the Internet based communication profiles;
- Instances of the IC “GPRS modem setup” – see 5.5.7 – handle all information related to the setup of the GPRS modem;
- Instances of the IC “SMTP setup” – see 5.7.6 – handle all information related to the setup of the SMTP service.

NOTE The following objects have internet related OBIS codes, although they are not strictly related to use over the internet only.

- Instances of the IC “GSM diagnostic” – see 5.5.8 – handle all diagnostic information related to the GSM/GPRS network.
- Instances of the IC “Push setup” – see 5.3.8 – handle all information about the data to be pushed, the push destination and the method the data should be pushed.

Objects to set up data exchange over the Internet	IC	OBIS code					
		A	B	C	D	E	F
TCP-UDP setup	41, TCP-UDP setup	0	b	25	0	0	255
IPv4 setup	42, IPv4 setup	0	b	25	1	0	255
MAC address setup	43, MAC address setup	0	b	25	2	0	255
PPP setup	44, PPP setup	0	b	25	3	0	255
GPRS modem setup	45, GPRS modem setup	0	b	25	4	0	255
SMTP setup	46, SMTP setup	0	b	25	5	0	255
GSM diagnostic	47, GSM diagnostic	0	b	25	6	0	255
IPv6 setup	48, IPv6 setup	0	b	25	7	0	255
Reserved for FTP setup							
Push setup	40, Push setup	0	b	25	9	0	255

6.2.22 Objects for setting up data exchange using S-FSK PLC

In this group, the following objects are available:

- Instances of the IC “S-FSK Phy&MAC setup” – see 5.8.3 – handle all information related to setting up the PLC S-FSK lower layer profile specified in IEC 61334-5-1:2001.
- Instances of the IC “S-FSK Active initiator” – see 5.8.4 – handle all information related to the active initiator in the PLC S-FSK lower layer profile specified in IEC 61334-5-1:2001.
- Instances of the IC “S-FSK MAC synchronization timeouts” – see 5.8.5 – manage all timeouts related to the synchronization process of devices using the PLC S-FSK lower layer profile specified in IEC 61334-5-1:2001.
- Instances of the IC “S-FSK MAC counters” – see 5.8.6 – store counters related to the frame exchange, transmission and repetition phases in the PLC S-FSK lower layer profile specified in IEC 61334-5-1:2001.
- Instances of the IC “IEC 61334-4-32 LLC setup” – see 5.8.7 – handle all information related to the LLC layer specified in IEC 61334-4-32:1996.
- Instances of the IC “S-FSK Reporting system list” – see 5.8.8 – hold information on reporting systems in the PLC S-FSK lower layer profile specified in IEC 61334-5-1:2001.

Objects to set up data exchange using S-FSK PLC	IC	OBIS code					
		A	B	C	D	E	F
S-FSK Phy&MAC setup	50, S-FSK Phy&MAC setup	0	b	26	0	0	255
S-FSK Active initiator	51, S-FSK Active initiator	0	b	26	1	0	255
S-FSK MAC synchronization timeouts	52, S-FSK MAC synchronization	0	b	26	2	0	255
S-FSK MAC counters	53, S-FSK MAC counters	0	b	26	3	0	255
NOTE This is a placeholder for a Monitoring IC to be specified.							
IEC 61334-4-32 LLC setup	55, IEC 61334-4-32 LLC setup	0	b	26	5	0	255
S-FSK Reporting system list	56, S-FSK Reporting system list	0	b	26	6	0	255

6.2.23 Objects for setting up the ISO/IEC 8802-2 LLC layer

In this group, the following objects are available:

- Instances of the IC “ISO/IEC 8802-2 LLC Type 1 setup” – see 5.9.2 – handle all information related to the LLC layer specified in ISO/IEC 8802-2:1998 in Type 1 operation.
- Instances of the IC “ISO/IEC 8802-2 LLC Type 2 setup” – see 5.9.3 – handle all information related to the LLC layer specified in ISO/IEC 8802-2:1998 in Type 2 operation.

- Instances of the IC “ISO/IEC 8802-2 LLC Type 3 setup” – see 5.9.4 – handle all information related to the LLC layer specified in ISO/IEC 8802-2:1998 in Type 3 operation.

Objects to set up the ISO/IEC 8802-2 LLC layer	IC	OBIS code					
		A	B	C	D	E	F
ISO/IEC 8802-2 LLC Type 1 setup	57, ISO/IEC 8802-2 LLC Type 1 setup	0	<i>b</i>	27	0	0	255
ISO/IEC 8802-2 LLC Type 2 setup	58, ISO/IEC 8802-2 LLC Type 2 setup	0	<i>b</i>	27	1	0	255
ISO/IEC 8802-2 LLC Type 3 setup	59, ISO/IEC 8802-2 LLC Type 3 setup	0	<i>b</i>	27	2	0	255

6.2.24 Objects for data exchange using narrowband OFDM PLC for PRIME networks

For setting up and managing data exchange using narrowband OFDM PLC for PRIME networks one instance of each following classes shall be implemented for each interface:

- an instance of the 61334-4-32 LLC SSCS setup – see 5.10.3 – holds the addresses related to the CL_432 layer;
- an instance of the IC “PRIME NB OFDM PLC Physical layer counters” – see 5.10.5 – stores counters related to the physical layers exchanges;
- an instance of the IC “PRIME NB OFDM PLC MAC setup” – see 5.10.6 – holds the necessary parameters to set up the PRIME NB OFDM PLC MAC layer;
- an instance of the IC “PRIME NB OFDM PLC MAC functional parameters” – see 5.10.7 – provides information on specific aspects concerning the functional behaviour of the MAC layer;
- an instance of the IC “PRIME NB OFDM PLC MAC counters” – see 5.10.8 – stores statistical information on the operation of the MAC layer for management purposes;
- an instance of the IC “PRIME NB OFDM PLC MAC network administration data” – see 5.10.9 – holds the parameters related to the management of the devices connected to the network;
- an instance of the IC “MAC address setup” – holds the MAC address of the device. See 5.10.10;
- an instance of the IC “PRIME NB OFDM PLC Application identification” – see 5.10.11 – holds identification information related to administration and maintenance of PRIME NB OFDM PLC devices.

Objects for data exchange using Type 1 OFDM PLC	IC	OBIS code					
		A	B	C	D	E	F
61334-4-32 LLC SSCS setup	80, 61334-4-32 LLC SSCS setup	0	<i>b</i>	28	0	0	255
PRIME NB OFDM PLC Physical layer counters	81, PRIME NB OFDM PLC Physical layer counters	0	<i>b</i>	28	1	0	255
PRIME NB OFDM PLC MAC setup	82, PRIME NB OFDM PLC MAC setup	0	<i>b</i>	28	2	0	255
PRIME NB OFDM PLC MAC functional parameters	83, PRIME NB OFDM PLC MAC functional parameters	0	<i>b</i>	28	3	0	255
PRIME NB OFDM PLC MAC counters	84, PRIME NB OFDM PLC MAC counters	0	<i>b</i>	28	4	0	255
PRIME NB OFDM PLC MAC network administration data	85, PRIME NB OFDM PLC MAC network administration data	0	<i>b</i>	28	5	0	255
PRIME NB OFDM PLC MAC address setup	43, MAC address setup	0	<i>b</i>	28	6	0	255
PRIME NB OFDM PLC Application identification	86, PRIME NB OFDM PLC Application identification	0	<i>b</i>	28	7	0	255

6.2.25 Objects for data exchange using narrow-band OFDM PLC for G3-PLC networks

For setting up and managing data exchange using G3-PLC profile, one instance of each following classes shall be implemented for each interface:

- an instance of the IC “G3-PLC MAC layer counters” – see 5.11.3 – to store counters related to the MAC layer exchanges;
- an instance of the IC “G3-PLC MAC setup” – see 5.11.4 – to hold the necessary parameters to set up the G3-PLC MAC IEEE 802.15.4:2006 layer;
- an instance of the IC “G3-PLC 6LoWPAN adaptation layer setup” – see 5.11.5 – to hold the necessary parameters to set up the Adaptation layer.

Objects for data exchange using G3-PLC	IC	OBIS code					
		A	B	C	D	E	F
G3-PLC MAC layer counters	90, G3-PLC MAC layers counters	0	b	29	0	0	255
G3-PLC MAC setup	91, G3-PLC MAC setup	0	b	29	1	0	255
G3-PLC 6LoWPAN adaptation layer setup	92, G3-PLC 6LoWPAN adaptation layer setup	0	b	29	2	0	255

6.2.26 ZigBee® setup objects

The following objects are available for setting up and managing a ZigBee® network; see also 5.12.

Objects set up and manage ZigBee® networks	IC	OBIS code					
		A	B	C	D	E	F
ZigBee® SAS startup	101, ZigBee® SAS Startup	0	b	30	0	e	255
ZigBee® SAS join	102, ZigBee® SAS join	0	b	30	1	e	255
ZigBee® SAS APS fragmentation	103, ZigBee® SAS APS fragmentation	0	b	30	2	e	255
ZigBee® network control	104, ZigBee® network control	0	b	30	3	e	255
ZigBee® tunnel setup	105, ZigBee® tunnel setup	0	b	30	4	e	255

6.2.27 Association objects (class_id = 12, 15)

A series of Association SN / LN objects – see 5.3.3 and 5.3.4 – are available to model application associations between a DLMS/COSEM client and server.

Association objects	IC	OBIS code					
		A	B	C	D	E	F
Current association	12, Association SN 15, Association LN	0	0	40	0	0	255
Association, instance 1		0	0	40	0	1	255
.....							
Association, instance n		0	0	40	0	n	255

6.2.28 SAP assignment object (class_id = 17)

An instance of the IC “SAP assignment” – see 5.3.5 – holds information about the addresses (Service Access Points, SAPs) of logical devices within a physical device.

SAP Assignment object	IC	OBIS code					
		A	B	C	D	E	F
SAP assignment of current physical device	17, SAP assignment	0	0	41	0	0	255

6.2.29 COSEM logical device name object

Each COSEM logical device shall be identified by its Logical Device Name, unique worldwide. See 4.8.2. It is held by the *value* attribute of a “Data” object, with data type *octet-string* or *visible-string*. For short name referencing, the base_name of the object is fixed. See 4.3.

COSEM logical device name object	IC	OBIS code					
		A	B	C	D	E	F
COSEM logical device name	1, Data ^a	0	0	42	0	0	255
^a If the IC “Data” is not available, “Register” (with scaler = 0, unit = 255) may be used.							

6.2.30 Information security related objects

Instances of the IC “Security setup” – see 5.3.7 – are used to set up the message security features. For each Association object, there is one Security setup object managing security within that AA. See 7.4 and 7.5. Value group E numbers the instances.

Security setup objects	IC	OBIS code					
		A	B	C	D	E	F
Security setup	64, Security setup	0	0	43	0	e	255

Frame counter objects hold the invocation counter element of the initialization vector. They are instances of the IC “Data”. The value in value group B identifies the communication channel.

NOTE The same client may use different communication channels e.g. a remote port and a local port. The invocation counter on the different channels may be different.

The value in value group E shall be the same as in the logical name of the corresponding “Security setup” object.

Frame counter objects	IC	OBIS code					
		A	B	C	D	E	F
Frame counter	1, Data ^a	0	b	43	1	e	255
^a If the IC “Data” is not available, “Register” (with scaler = 0, unit = 255) may be used.							

6.2.31 Image transfer objects (class_id = 18)

Instances of the IC “Image transfer” – see 5.3.6 – control the Image transfer process.

Image transfer related objects	IC	OBIS code					
		A	B	C	D	E	F
Image transfer	18, Image transfer	0	0	44	0	e	255

6.2.32 Utility table objects (class_id = 26)

Instances of the IC “Utility tables” – see 5.2.7 – allow representing ANSI utility tables. The Utility table IDs are mapped to OBIS codes as follows:

- value group A: use value of 0 to specify abstract object;
- value group B: instance of table set;
- value group C: use value 65 – signifies utility tables specific definitions;
- value group D: table group selector;
- value group E: table number within group;
- value group F: use value 0xFF for data of current billing period.

Utility table objects	IC	OBIS code					
		A	B	C	D	E	F
Standard tables 0-127	26, Utility tables	0	<i>b</i>	65	0	<i>e</i>	255
Standard tables 128-255		0	<i>b</i>	65	1	<i>e</i>	255
...							
Standard tables 1920-2047		0	<i>b</i>	65	15	<i>e</i>	255
Manufacturer tables 0-127		0	<i>b</i>	65	16	<i>e</i>	255
Manufacturer tables 128-255		0	<i>b</i>	65	17	<i>e</i>	255
...							
Manufacturer tables 1920-2047		0	<i>b</i>	65	31	<i>e</i>	255
Std pending tables 0-127		0	<i>b</i>	65	32	<i>e</i>	255
Std pending tables 128-255		0	<i>b</i>	65	33	<i>e</i>	255
...							
Std pending tables 1920-2047		0	<i>b</i>	65	47	<i>e</i>	255
Mfg pending tables 0-127		0	<i>b</i>	65	48	<i>e</i>	255
Mfg pending tables 128-255		0	<i>b</i>	65	49	<i>e</i>	255
...							
Mfg pending tables 1920-2047		0	<i>b</i>	65	63	<i>e</i>	255

6.2.33 Device ID objects

A series of objects are used to hold ID numbers of the device. These ID numbers can be defined by the manufacturer (e.g. manufacturing number) or by the user.

They are held by the *value* attribute of "Data" objects, with data type *double-long-unsigned*, *octet-string*, *visible-string*, *utf8-string*, *unsigned*, *long-unsigned*. If more than one of those is used, it is allowed to combine them into a "Profile generic" object. In this case, the captured objects are *value* attributes of the device ID "Data" objects, the capture period is 1 to have just actual values, the sort method is FIFO and the profile entries are limited to 1. Alternatively, a "Register table" object – see 5.2.8 – can be used. See also see IEC 62056-6-1:2015, Table 7.

Device ID objects	IC	OBIS code					
		A	B	C	D	E	F
Device ID 1... 10 object (manufacturing number)	1, Data ^a	0	<i>b</i>	96	1	0...9	255
Device ID-s object	7, Profile generic	0	<i>b</i>	96	1	255	255
Device ID-s object	61, Register table	0	<i>b</i>	96	1	255	255

^a If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

6.2.34 Metering point ID objects

One object is available to store a media type independent metering point ID. It is held by the *value* attribute of a "Data" object, with data type *double-long-unsigned*, *octet-string*, *visible-string*, *utf8-string*, *unsigned*, *long-unsigned*.

Metering point ID objects	IC	OBIS code					
		A	B	C	D	E	F
Metering point ID	1, Data ^a	0	<i>b</i>	96	1	10	255

^a If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

6.2.35 Parameter changes and calibration objects

A set of simple COSEM objects describes the history of the configuration of the device. All values are modelled by instances of the IC "Data".

Parameter changes objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2015, Table 7.	1, Data ^a	0	<i>b</i>	96	2	<i>e</i>	255
^a If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.							

6.2.36 I/O control signal objects

A series of objects are available to define and control the status of I/O lines of the physical metering equipment.

The status is held by the *value* attribute of a "Data" object, with data type *octet-string* or *bit-string*. Alternatively, the status is held by a "Status mapping" object, see 5.2.9, which holds both the status word and the mapping of its bits to the reference table. If there are several I/O control status objects used, it is allowed to combine them into an instance of the IC "Profile generic" or "Register table", using the OBIS code of the global state of I/O control signals object. See also see IEC 62056-6-1:2015, Table 7.

I/O control signal objects	IC	OBIS code					
		A	B	C	D	E	F
I/O control signal objects, contents manufacturer specific	1, Data ^a	0	<i>b</i>	96	3	0...4	255
I/O control signal objects, contents mapped to a reference table	63, Status mapping	0	<i>b</i>	96	3	0...4	255
I/O control signal objects, global	7, Profile generic or 61, Register table	0	<i>b</i>	96	3	0	255
^a If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.							

6.2.37 Disconnect control objects (class_id = 70)

Instances of the IC "Disconnect control" – see 5.4.8 – manage internal or external disconnect units (e.g. electricity breaker, gas valve) in order to connect or disconnect – partly or entirely – the premises of the consumer to / from the supply. See also 6.2.20.

Disconnect control objects	IC	OBIS code					
		A	B	C	D	E	F
Disconnect control	70, Disconnect control	0	<i>b</i>	96	3	10	255

6.2.38 Status of internal control signals objects

A series of objects are available to hold the status of internal control signals.

The status carries binary information from a bitmap, and it shall be held by the *value* attribute of a "Data" object, with data type *bit-string*, *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-unsigned* or *octet-string*. Alternatively, the status is held by a "Status mapping" object, see 5.2.9, which holds both the status word and the mapping of its bits to the reference table. If there are several status of internal control signals objects used, it is allowed to combine them into an instance of the IC "Profile generic" or "Register table", using the OBIS code of the global "Internal control signals" object. See also see IEC 62056-6-1:2015, Table 7.

Internal control signals objects	IC	OBIS code					
		A	B	C	D	E	F
Internal control signals, contents manufacturer specific	1, Data ^a	0	<i>b</i>	96	4	0...4	255
Internal control signals, contents mapped to a reference table	63, Status mapping	0	<i>b</i>	96	4	0...4	255
Internal control signals, global	7, Profile generic or 61, Register table	0	<i>b</i>	96	4	0	255

^a If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

6.2.39 Internal operating status objects

A series of objects are available to hold internal operating statuses.

The status carries binary information from a bitmap, and it shall be held by the value attribute of a "Data" object, with data type *bit-string*, *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-unsigned* or *octet-string*. Alternatively, the status is held by a "Status mapping" object, see 5.2.9, which holds both the status word and the mapping of its bits to the reference table. If there are several status of internal control signals objects used, it is allowed to combine them into an instance of the IC "Profile generic" or "Register table", using the OBIS code of the global "Internal operating status" object. See also see IEC 62056-6-1:2015, Table 7.

Internal operating status objects	IC	OBIS code					
		A	B	C	D	E	F
Internal operating status objects, contents manufacturer specific	1, Data ^a	0	<i>b</i>	96	5	0...4	255
Internal operating status objects, contents mapped to a reference table	63, Status mapping	0	<i>b</i>	96	5	0...4	255
Internal operating status objects, global	7, Profile generic or 61, Register table	0	<i>b</i>	96	5	0	255

^a If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

Internal operating status objects can also be related to an energy type. See 6.3.7.

6.2.40 Battery entries objects

A series of objects are available for holding information relative to the battery of the device. These objects are instances of IC "Data", "Register" or "Extended register" as appropriate.

Battery entries objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2015, Table 7.	1, Data, 3, Register or 4, Extended register	0	<i>b</i>	96	6	0...6	255

6.2.41 Power failure monitoring objects

A series of objects are available for power failure monitoring:

- For simple power failure monitoring, it is possible to count the number of power failure events affecting all three phases, one of the three phases, any of the phases, and the auxiliary supply;
- For advanced power failure monitoring, it is possible to define a time threshold to make a distinction between short and long power failure events. It is possible to count the number of such long power failure events separately from the short ones, as well as to store their time of occurrence and duration (time from power down to power up) in all three phases, in one of the three phases and in any of the phases;

- The number of power failure events objects are represented by instances of the IC “Data”, “Register” or “Extended register” with data types *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*;
- The power failure duration, time and time threshold data are represented by instances of the IC “Data”, “Register” or “Extended register” with appropriate data types;
- If power failure duration objects are represented by instances of the IC “Data”, then the default scaler shall be 0, and the default unit shall be the second.

Power failure monitoring objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2015, Table 7.	1, Data, 3, Register or 4, Extended register	0	<i>b</i>	96	7	0... 24	255

These objects may be collected in a “Power failure event log” object. See IEC 62056-6-1:2015, Table 22.

6.2.42 Operating time objects

A series of objects are available for holding the cumulated operating time and the various tariff registers of the device. These objects are instances of the IC “Data”, “Register” or “Extended register”. The data type shall be *unsigned*, *long-unsigned* or *double-long-unsigned* with appropriate scaler and unit. If the IC “Data” is used, the unit shall be the second by default.

Operating time objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2015, Table 7.	1, Data, 3, Register or 4, Extended register	0	<i>b</i>	96	8	0... 63	255

6.2.43 Environment related parameters objects

A series of objects are available to store environmental related parameters. They are held by the *value* attribute of instances of the IC “Register” or “Extended register”, with appropriate data types.

Environment related parameters objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2015, Table 7.	3, Register or 4, Extended register	0	<i>b</i>	96	9	0...2	255

6.2.44 Status register objects

A series of objects are available to hold statuses that can be captured in load profiles. See also see IEC 62056-6-1:2015, Table 7.

Status register objects	IC	OBIS code					
		A	B	C	D	E	F
Status register, contents manufacturer specific	1, Data ^a	0	<i>b</i>	96	10	1... 10	255
Status register, contents mapped to reference table	63, Status mapping	0	<i>b</i>	96	10	1... 10	255
^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.							

The status register is held by the value attribute of a “Data” object, with data type *bit-string*, *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-unsigned* or *octet-string*. It carries binary information from a bitmap. Its contents is not specified.

Alternatively, the status register may be held by the *status_word* attribute of a “Status mapping” object, see 5.2.9. The *mapping_table* attribute holds mapping information between the bits of the status word and entries of a reference table.

6.2.45 Event code objects

In the meter or in its environment, various events may be generated. A series of objects are available to hold an identifier of a most recent event (event code). Different instances of event code objects may be captured in different instances of event logs; see 6.2.54.

NOTE The definition of event identifiers is out of the Scope of this standard.

Event code objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2015, Table 7.	1, Data, 3, Register, or 4, Extended register	0	<i>b</i>	96	11	0...99	255

Events may also set flags in error registers and alarm registers. See also 6.2.52.

6.2.46 Communication port log parameter objects

A series of objects are available to hold various communication log parameters. They are represented by instances of IC “Data”, “Register” or “Extended register”.

Communication port log parameter objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2015, Table 7.	1, Data, 3, Register, or 4, Extended register	0	<i>b</i>	96	12	0...6	255

6.2.47 Consumer message objects

A series of objects are available to store information sent to the energy end-user. The information may appear on the display of the meter and / or on a consumer information port.

Consumer message objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2015, Table 7.	1, Data, 3, Register, or 4, Extended register	0	<i>b</i>	96	13	0, 1	255

6.2.48 Currently active tariff objects

A series of objects are available to hold the identifier of the currently active tariff. They carry the same information as the *active_mask* attribute of the corresponding “Register activation” object.

Currently active tariff objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2015, Table 7.	1, Data, 3, Register, or 4, Extended register	0	<i>b</i>	96	14	0...15	255

6.2.49 Event counter objects

A series of objects are available to count events. The number of the events is held by the value attribute.

Event counter objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2015, Table 7.	1, Data, 3, Register, or 4, Extended register	0	<i>b</i>	96	15	0... 99	255

6.2.50 Meter tamper event related objects

A series of objects are available to register characteristics of various meter tamper events. These objects are instances of the IC “Data”, “Register” or “Extended register”. The data type shall be *unsigned*, *long-unsigned* or *double-long-unsigned* with appropriate scaler and unit. For time stamps, the data type shall be *double-long-unsigned* (in the case of UNIX time), *octet-string* or *date-time* formatted as specified in 4.6.1.

- Meter open events are related to cases when the meter case is open;
- Terminal cover open events are related to cases when a terminal cover is removed (open);
- Tilt events are related to cases when the meter is not in its normal operation position;
- Strong DC magnetic field events are related to cases when the presence of a strong DC magnetic field is detected;
- Metrology tamper events are related to cases when an anomaly in the operation of the metrology is detected due to a perceived tamper;
- Communication tamper events are related to cases when an anomaly in the operation of the communication interfaces is detected due to a perceived tamper.

The method of detecting the various tampers is out of the Scope of this standard.

Meter tamper event related objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2015, Table 7.	1, Data, 3, Register, or 4, Extended register	0	<i>b</i>	96	20	<i>e</i>	255

6.2.51 Error register objects

A series of objects are used to communicate error indications of the device. The different error registers are held by the *value* attribute of “Data” objects, with data type or *bit-string*, *octet-string*, *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*.

The individual bits of the error register may be set and cleared by a pre-defined selection of events – see 6.2.45. Depending on the type of the error, some errors may clear themselves when the reason setting the error flag disappears.

If more than one of those objects is used, it is allowed to combine them into one instance of the IC "Profile generic". In this case, the captured objects are the *value* attributes "Data" objects, the capture period is 1 to have just actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively, an instance of the IC "Register table" can be used.

Error register objects can also be related to an energy type and to a channel. See see IEC 62056-6-1:2015, 6.2 and 7.5.2.

Error register objects	IC	OBIS code					
		A	B	C	D	E	F
Error register 1...10 object	1, Data ^a	0	b	97	97	0...9	255
Error profile object	7, Profile generic	0	b	97	97	255	255
Error table object	61, Register table	0	b	97	97	255	255

^a If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

6.2.52 Alarm register, Alarm filter and Alarm descriptor objects

A number of objects are available to hold alarm registers. The different alarm registers are held by the value attribute of “Data” objects, with data type *bit-string*, *octet-string*, *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*. When selected events occur, they set the corresponding flag and the device may raise an alarm. Depending on the type of alarm, some alarms may clear themselves when the reason setting the alarm flag disappears.

If more than one of those objects is used, it is also allowed to combine them into one instance of the IC “Profile generic”. In this case, the captured objects are the *value* attributes of “Data” objects, the capture period is 1 to have just actual values, the sort method is FIFO, and the profile entries are limited to 1. Alternatively, an instance of the IC “Register table” can be used.

Alarm filter objects are available to define if an event is to be handled as an alarm when it appears. The different alarm filters are held by the value attribute of “Data” objects, with data type *bit-string*, *octet-string*, *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*. The bit mask has the same structure as the corresponding alarm register object. If a bit in the alarm filter is set, then the corresponding alarm is enabled, otherwise it is disabled. *Alarm filter* objects act on *Alarm register* and *Alarm descriptor* objects the same way.

Alarm descriptor objects are available to persistently hold the occurrence of alarms. The different alarm descriptors are of the same type as the corresponding *Alarm register*. When a selected event occurs, the corresponding flag is set in the *Alarm register* as well as in the *Alarm descriptor* objects. An alarm descriptor flag remains set even if the corresponding alarm condition has disappeared. Alarm descriptor flags do not reset themselves; they can be reset by writing the value attribute only.

NOTE The alarm conditions, the structure of the *Alarm register* / *Alarm filter* / *Alarm descriptor* objects are subject to a project specific companion specification.

Alarm register, Alarm filter and Alarm descriptor objects	IC	OBIS code					
		A	B	C	D	E	F
Alarm register objects 1...10	1, Data ^a	0	b	97	98	0...9	255
Alarm register profile object	7, Profile generic	0	b	97	98	255	255
Alarm register table object	61, Register table	0	b	97	98	255	255
Alarm filter objects 1...10	1, Data ^a	0	b	97	98	10...19	255
Alarm descriptor objects 1...10	1, Data ^a	0	b	97	98	20...29	255

^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

6.2.53 General list objects

Instances of the IC “Profile generic” are used to model lists of any kind of data, for example measurement values, constants, statuses, events. They are modelled by “Profile generic” objects. One standard object per billing period scheme is defined.

List objects may be also related to an energy type and to a channel.

General list objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2015, 6.3 and 7.5.3	7, Profile generic	0	b	98	d	e	255 ^a

^a F = 255 means a wildcard here. See IEC 62056-6-1:2015, Clause A.3.

6.2.54 Event log objects

Instances of the IC "Profile generic" are used to store Event logs. Event logs may be also media related. In this case, the value of value group A shall be the relevant media identifier. See also see IEC 62056-6-1:2015, 6.5 and 7.5.4.

Event log objects	IC	OBIS code					
		A	B	C	D	E	F
Event log	7, Profile generic	a	b	99	98	e	255 ^a
^a F = 255 means a wildcard here. See IEC 62056-6-1:2015, Clause A.3.							
NOTE 1 Event logs may capture for example the time of occurrence of the event, the event code and other relevant data.							
NOTE 2 Project specific companion specifications may specify a more precise meaning of the instances of the different event logs, i.e. the data captured and the number of events captured.							

6.2.55 Inactive objects

Inactive objects are objects, which are present in the meter, but which do not have an assigned functionality. Inactive instances of any IC may be present. See also See IEC 62056-6-1:2015, 5.3.2.

Inactive objects	IC	OBIS code					
		A	B	C	D	E	F
Inactive objects	Any	0	b	127	0	e	255

6.3 Electricity related COSEM objects

6.3.1 Value group D definitions

The different ways of processing measurement values as defined by value group D – see See IEC 62056-6-1:2015, 7.2.1 – are modelled as shown in Table 20.

Table 20 – Representation of various values by appropriate ICs

Type of value	Represented by
cumulative values	Instances of IC "Register" or "Extended register".
maximum and minimum values	Instances of IC "Profile generic" with sorting method <i>maximum</i> or <i>minimum</i> , depth according to implementation and captured objects according to implementation. A single maximum value or minimum value can alternatively be represented by an instance of the IC "Register" or "Extended register".
current and last average values	Instances of IC "Demand register". The logical name is the OBIS code of the current average value (D = 4, 14 or 24). For display purposes: Instances of IC "Register" or "Demand register". The logical name is the OBIS code of current average (D = 4, 14 or 24) or last average (D = 5, 15 or 25) as appropriate.
instantaneous values	Instances of IC "Register".
time integral values	Instances of IC "Register" or "Extended register".
occurrence counters	Instances of IC "Data" or "Register".
contracted values	Instances of IC "Register" or "Extended register".

6.3.2 Electricity ID numbers

The different electricity ID numbers are held by instances of the IC "Data", with data type *unsigned*, *long-unsigned*, *double-long-unsigned*, *octet-string* or *visible-string*. If more than one of those is used, it is allowed to combine them into a "Profile generic" object. In this case, the captured objects are *value* attributes of electricity ID "Data" objects, the capture period is 1 to

have just actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively, a “Register table” object can be used. See also IEC 62056-6-1:2015, Table 19.

Electricity ID objects	IC	OBIS code					
		A	B	C	D	E	F
Electricity ID 1...10 object	1, Data ^a	1	<i>b</i>	0	0	0...9	255
Electricity ID-s object	7, Profile generic	1	<i>b</i>	0	0	255	255
Electricity ID-s object	61, Register table	1	<i>b</i>	0	0	255	255

^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

6.3.3 Billing period values / reset counter entries

These values are represented by instances of the IC “Data”.

For billing period / reset counters and for number of available billing periods the data type shall be *unsigned*, *long-unsigned* or *double-long-unsigned*. For time stamps of billing periods, the data type shall be *double-long-unsigned* (in the case of UNIX time), *octet-string* or *date-time* formatted as specified in 4.6.1.

These objects may be related to channels.

When the values of historical periods are represented by “Profile generic” objects, the time stamp of the billing period objects shall be part of the captured objects.

Billing period values / reset counter entries	IC	OBIS code					
		A	B	C	D	E	F
For item names and OBIS codes see IEC 62056-6-1:2015, Table 19.	1, Data ^a	1	<i>b</i>	0	1	<i>e</i>	255

^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

6.3.4 Other electricity related general purpose objects

Program entries shall be represented by instances of the IC “Data” with data type *unsigned*, *long-unsigned*, *octet-string* or *visible-string*. For “Meter connection diagram ID” objects data type *enumerated* can be used as well. Program entries can also be related to a channel.

Output pulse constant, reading factor, CT/VT ratio, nominal value, input pulse constant, transformer and line loss coefficient values shall be represented by instances of the IC “Data”, “Register” or “Extended register”. For the *value* attribute, only simple data types are allowed.

Measurement period, recording interval and billing period duration values shall be represented by instances of IC “Data”, “Register” or “Extended register” with the data type of the *value* attribute *unsigned*, *long-unsigned* or *double-long-unsigned*. The default unit is the second.

Time entry values shall be represented by instances of IC “Data”, “Register” or “Extended register” with the data type of the *value* attribute *octet-string*, formatted as *date-time* in 4.6.1. The data types *unsigned*, *integer*, *long-unsigned* or *double-long-unsigned* can also be used where appropriate.

The *Clock synchronization method* shall be represented by an instance of an IC “Data” with data type *enum*.

Synchronization method

- enum:
- (0) no synchronization,
 - (1) adjust to quarter,
 - (2) adjust to measuring period,
 - (3) adjust to minute,
 - (4) reserved,
 - (5) adjust to preset time,
 - (6) shift time

For the detailed OBIS codes, see IEC 62056-6-1:2015, Table 19.

Electricity related general purpose objects	IC	OBIS code					
		A	B	C	D	E	F
Program entries	1, Data ^a	1	b	0	2	e	255
Output pulse values or constants		1	b	0	3	e	255
Reading factor and CT/VT ratio		1	b	0	4	e	255
Nominal values		1	b	0	6	e	255
Input pulse values or constants		1	b	0	7	e	255
Measurement period- / recording interval- / billing period duration		1	b	0	8	e	255
Time entries		1	b	0	9	e	255
Transformer and line loss coefficients		1	b	0	10	e	255
^a If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.							

6.3.5 Measurement algorithm

These values are represented by instances of the IC "Data", with data type *enum*.

Measurement algorithm objects	IC	OBIS code					
		A	B	C	D	E	F
Measuring algorithm for active power	1, Data ^a	1	<i>b</i>	0	11	1	255
Measurement algorithm for active energy		1	<i>b</i>	0	11	2	255
Measurement algorithm for reactive power		1	<i>b</i>	0	11	3	255
Measurement algorithm for reactive energy		1	<i>b</i>	0	11	4	255
Measurement algorithm for apparent power		1	<i>b</i>	0	11	5	255
Measurement algorithm for apparent energy		1	<i>b</i>	0	11	6	255
Measurement algorithm for power factor calculation		1	<i>b</i>	0	11	7	255
^a In cases where the IC “Data” is not available, “Register” (with scaler = 0, unit = 255) may be used.							

The enumerated values are specified in Table 21 below:

Table 21 – Measuring algorithms – enumerated values

Measuring algorithm for active power and energy	
(0)	not specified
(1)	only the fundamentals of voltage and current are used
(2)	all harmonics of voltage and current are used
(3)	only the DC part of voltage and current is used
(4)	all harmonics and the DC part of voltage and current are used
Measuring algorithm for reactive power and energy	
(0)	not specified
(1)	(sum of) reactive power of each phase, calculated from the fundamental of the per phase voltage and the per phase current
(2)	polyphase reactive power calculated from polyphase apparent power and polyphase active power
(3)	(sum of) reactive power calculated from per phase apparent power and per phase active power
Measurement algorithm for apparent power and energy	
(0)	not specified
(1)	$S = U \times I$, with voltage: only fundamental, and current: only fundamental
(2)	$S = U \times I$, with voltage: only fundamental, and current: all harmonics
(3)	$S = U \times I$, with voltage: only fundamental, and current: all harmonics and DC part
(4)	$S = U \times I$, with voltage: all harmonics, and current: only fundamental
(5)	$S = U \times I$, with voltage: all harmonics, and current: all harmonics
(6)	$S = U \times I$, with voltage: all harmonics, and current: all harmonics and DC part
(7)	$S = U \times I$, with voltage: all harmonics and DC part, and current: only fundamental
(8)	$S = U \times I$, with voltage: all harmonics and DC part, and current: all harmonics
(9)	$S = U \times I$, with voltage: all harmonics and DC part, and current: all harmonics and DC part
(10)	$S = \sqrt{P^2 + Q^2}$, with P : only fundamental in U and I , and Q : only fundamental in U and I , where P and Q are polyphase quantities
(11)	$S = \sqrt{P^2 + Q^2}$, with P : all harmonics in U and I , and Q : only fundamental in U and I where P and Q are polyphase quantities
(12)	$S = \sqrt{P^2 + Q^2}$, with P : all harmonics and DC part in U and I , and Q : only fundamental in U and I where P and Q are polyphase quantities
(13)	$S = \sum \sqrt{P^2 + Q^2}$, with P : only fundamental in U and I , and Q : only fundamental in U and I where P and Q are single phase quantities
(14)	$S = \sum \sqrt{P^2 + Q^2}$, with P : all harmonics in U and I , and Q : only fundamental in U and I where P and Q are single phase quantities
(15)	$S = \sum \sqrt{P^2 + Q^2}$, with P : all harmonics and DC part in U and I , and Q : only fundamental in U and I where P and Q are single-phase quantities
Measurement algorithm for power factor calculation	
(0)	not specified
(1)	displacement power factor: the displacement between fundamental voltage and current vectors, which can be calculated directly from fundamental active power and apparent power, or another appropriate algorithm,
(2)	true power factor, the power factor produced by the voltage and current, including their harmonics . It may be calculated from apparent power and active power, including the harmonics.

6.3.6 Metering point ID (electricity related)

A series of objects are available to hold electricity related metering point IDs. They are held by the *value* attribute of “Data” objects, with data type *unsigned*, *long-unsigned*, *double-long unsigned*, *octet-string* or *visible-string*. If more than one of those is used, it is allowed to combine them into one instance of the IC “Profile generic”. In this case, the captured objects are the *value* attributes of the electricity related metering point ID “Data” objects, the capture period is 1 to have just actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively, an instance of the IC “Register table” can be used. For detailed OBIS codes, see IEC 62056-6-1:2015, Table 19.

Metering point ID objects	IC	OBIS code					
		A	B	C	D	E	F
Metering point ID 1...10 (electricity related)	1, Data ^a	1	<i>b</i>	96	1	0...9	255
Metering point ID-s object	7, Profile generic	1	<i>b</i>	96	1	255	255
Metering point ID-s object	64, Register table	1	<i>b</i>	96	1	255	255

^a If the IC “Data” is not available, “Register” (with scaler = 0, unit = 255) may be used.

6.3.7 Electricity related status objects

A number of electricity related objects are available to hold information about the internal operating status, the starting of the meter and the status of voltage and current circuits.

The status is held by the value attribute of a “Data” object, with data type *bit-string*, *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-unsigned* or *octet-string*.

Alternatively, the status is held by a “Status mapping” object, which holds both the status word and the mapping of its bits to the reference table.

If there are several electricity related internal operating status objects used, it is allowed to combine them into an instance of the IC “Profile generic” or “Register table”, using the OBIS code of the global internal operating status. For detailed OBIS codes, see IEC 62056-6-1:2015, Table 19.

Electricity related status objects	IC	OBIS code					
		A	B	C	D	E	F
Internal operating status signals, electricity related, contents manufacturer specific	1, Data ^a	1	<i>b</i>	96	5	0...5	255
Internal operating status signals, electricity related, contents mapped to reference table	63, Status mapping	1	<i>b</i>	96	5	0...5	255
Electricity related status data, contents manufacturer specific	1, Data ^a	1	<i>b</i>	96	10	0...3	255
Electricity related status data, contents mapped to reference table	63, Status mapping	1	<i>b</i>	96	10	0...3	255

^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

6.3.8 List objects – Electricity (class_id = 7)

These COSEM objects are used to model lists of any kind of data, for example measurement values, constants, statuses, events. They are modelled by “Profile generic” objects.

One standard object per billing period scheme is defined. See also IEC 62056-6-1:2015, 7.5.3.

List objects – Electricity	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2015, Table 21.	7, Profile generic	1	<i>b</i>	98	<i>d</i>	<i>e</i>	255 ^a
^a F = 255 means a wildcard here. See IEC 62056-6-1:2015, Clause A.3.							

6.3.9 Threshold values

A number of objects are available for representing thresholds for instantaneous quantities. The thresholds may be “under limit”, “over limit”, “missing” and “time thresholds”. Time thresholds are used to detect “under limit”, “over limit” and “missing” conditions.

Objects are also available to represent the number of occurrences when these thresholds are exceeded, the duration of such events and the magnitude of the quantity during such events.

These values are represented by instances of IC “Data”, “Register” or “Extended register”.

All these quantities may be related to tariffs.

As defined in IEC 62056-6-1:2015, 7.4.2, value group F may be used to identify multiple thresholds.

For OBIS codes, see Table 22 below and IEC 62056-6-1:2015, Table 13.

Table 22 – Threshold objects, electricity

Threshold objects	IC	OBIS code					
		A	B	C	D	E	F
Threshold objects for instantaneous values	1, Data, 3, Register, 4, Extended register	1	<i>b</i>	1...10, 13, 14, 16...20, 21...30, 33, 34, 36...40, 41...50, 53, 54, 56...60, 61...70, 73, 74, 76...80, 82, 84...89	31...34, 35...38, 39...42, 43...45	0...63	0...99. 255
Threshold objects for harmonics of voltage, current and active power		1	<i>b</i>	11, 12, 15, 31, 32, 35, 51, 52, 55, 71, 72, 75, 90...92		0...120, 124... 127	

For monitoring the supply voltage, a more sophisticated functionality is also available, allowing to count the number of occurrences classified by the duration of the event and the depth of the voltage dip. For OBIS codes, see IEC 62056-6-1:2015, Table 18.

6.3.10 Register monitor objects (class_id = 21)

Further to 6.2.13, the following definitions apply, see Table 23:

- For monitoring instantaneous values, the logical name of the “Register monitor” object may be the OBIS identifier of the threshold;

- For monitoring current average and last average values, the logical name of the “Register monitor” object may be the OBIS identifier of the demand value monitored.

Table 23 – Register monitor objects, electricity

Register monitor objects	IC	OBIS code					
		A	B	C	D	E	F
Register monitor object, instantaneous values	21, Register monitor	1	<i>b</i>	1-80, 82, 84... 92	31, 35, 39	<i>e</i>	255
Register monitor object, current average and last average values		1	<i>b</i>	1-80, 82, 84... 92	4, 5, 14, 15, 24, 25	<i>e</i>	255

The use of value group E shall be to identify the tariff rate. If the total quantity (without tariffication) is monitored, then its value shall be 0.

6.4 Coding of OBIS identifications

To identify different instances of the same IC, their logical name shall be different. In COSEM, the logical_name is taken from the OBIS definition (see 6.2, 6.3 and IEC 62056-6-1:2015).

OBIS codes are used within the COSEM environment as an *octet-string* [6]. Each octet contains the unsigned value of the corresponding OBIS value group, coded without tags.

If a data item is identified by less than six value groups, all unused value groups shall be filled with 255.

Octet 1 contains the binary coded value of A (A = 0, 1, 2 ...9) in the four rightmost bits. The four leftmost bits contain the information on the identification system. The four leftmost bits set to zero indicate that the OBIS identification system (version 1) is used as *logical_name*.

Identification system used	Four leftmost bits of octet 1 (MSB left)
OBIS, see IEC 62056-6-1:2015	0 0 0 0
Reserved for future use	0 0 0 1
	...
	1 1 1 1

Within all value groups, the usage of a certain selection is fully defined; others are reserved for future use. If in the value groups B to F a value belonging to the manufacturer specific range (see IEC 62056-6-1:2015, 4.2) is used, then the whole OBIS code shall be considered as manufacturer specific, and the value of the other groups does not necessarily carry a meaning defined neither by Clause 5 of this standard nor by IEC 62056-6-1:2015.

7 Previous versions of interface classes

7.1 General

This Clause 7 lists those IC specifications which were included in previous editions of this standard. The previous IC versions differ from the current versions by at least one attribute and/or method and by the version number.

For new implementations in metering devices, only the current versions should be used.

Communication drivers at the client side should also support previous versions.

7.2 Profile generic (class_id = 7, version = 0)

The version listed here was valid in edition 1 and it is replaced as of edition 2 and 3.

Profile generic		0...n	class_id = 7, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	buffer (dyn.)	array			x	x + 0x08
3.	capture_objects (static)	array				x + 0x10
4.	capture_period (static)	double-long-unsigned			x	x + 0x18
5.	sort_method (static)	enum			x	x + 0x20
6.	sort_object (static)	ObjectDefinition			x	x + 0x28
7.	entries_in_use (dyn.)	double-long-unsigned	0		0	x + 0x30
8.	profile_entries (static)	double-long-unsigned	1		1	x + 0x38
Specific methods		m/o				
1.	reset ()					x + 0x58
2.	capture ()					x + 0x60
3.	get_buffer_by_range ()					x + 0x68
4.	get_buffer_by_index ()					x + 0x70

Attribute description

buffer The buffer attribute contains a sequence of entries. Each entry contains values of the captured objects (as returned by calling *read (current_value)*). The sequence is ordered according to the specified sort method. The buffer gets filled by subsequent calls to *capture ()*.

array entry

entry::= structure

Instance Specific

Def. The buffer is empty at installation.

Remark: Reading the entire buffer delivers only those entries which are “in use”

capture_objects	Specifies the list of capture objects (registers, clocks and profiles) that are assigned to this profile object. Upon a call of the <i>capture ()</i> service, the specified attributes of these objects are copied into the buffer of the profile. array ObjectDefinition ObjectDefinition ::= structure <pre> { logical_name: octet-string, class_id: long-unsigned, attribute_index: unsigned } </pre> where attribute_index is a pointer to the attribute within the object. attribute_index = 1 refers to the first attribute (i.e. <i>logical_name</i>), attribute_index = 2 to the 2nd, etc.)
capture_period	>= 1: Automatic capturing assumed. Specifies the capturing period in seconds 0: no automatic capturing, capturing is triggered externally or capture events occur asynchronously
sort_method	If the profile is unsorted, it works as a „first in first out“ buffer (it is hence sorted by capturing, and not necessarily by the time maintained in the clock object). If the buffer is full, the next call to <i>capture ()</i> will push out the first (oldest) entry of the buffer to make space for the new entry. If the profile is sorted, a call to <i>capture ()</i> will store the new entry at the appropriate position in the buffer, moving all following entries and probably losing the least interesting entry. If the new entry would enter the buffer after the last entry and if the buffer is already full, the new entry will not be retained at all. enum: 1: fifo, 2: lifo (last in first out), 3: largest, 4: smallest, 5: nearest_to_zero, 6: farthest_from_zero Def. fifo
sort_object	If the profile is sorted, this attribute specifies the register or clock that the ordering is based upon. ObjectDefinition see above Def. no object to sort by (only possible with <i>sort_method</i> = fifo or lifo)
entries_in_use	Counts the number of entries stored in the buffer. After a call of <i>reset ()</i> the buffer does not contain any entries, and this value is zero. Upon each subsequent call of <i>capture ()</i>, this value will be incremented up to the maximum number of entries that will get stored (see <i>profile_entries</i>). double-long-unsigned 0...profile_entries Def. 0
profile_entries	Specifies how many entries should be retained in the buffer. double-long-unsigned 1... (limited by physical size) Def. 1

Method description

reset (data)	Clears the buffer. The buffer has no valid entries afterwards; <i>entries_in_use</i> is zero after this call. This call does not trigger any additional operations of the capture objects, specifically, it does not reset any captured buffers or registers. data = integer (0)																																		
capture (data)	Copies the values of the objects to capture into the buffer by calling <i>read (<object_attribute>)</i> of each capture object. Depending on the <i>sort_method</i> and the actual state of the buffer this produces a new entry or a replacement for the less significant entry. As long as not all entries are already used, the <i>entries_in_use</i> attribute will be incremented. This call does not trigger any additional operations within the capture objects such as <i>capture ()</i> or <i>reset ()</i> . Note that only some attributes of the captured objects might be stored, not the complete object. data = integer (0)																																		
write (...)	Any write access to one of the attributes describing the static structure of the buffer will automatically call a <i>reset ()</i> and this call will propagate to all other profiles capturing this profile. If writing to <i>profile_entries</i> is attempted with a value too large for the buffer, it will be set to the maximum possible value (restricted by physical size, typically laid down in the firmware).																																		
get_buffer_by_range (data)	<table><tr><td colspan="4">Reads all entries between the given limits.</td></tr><tr><td>restricting_object</td><td>in</td><td>ObjectDefinition</td><td>Defines the register or clock restricting the range of entries to be retrieved.</td></tr><tr><td>from_value</td><td>in</td><td>instance_specific</td><td>Oldest or smallest entry to retrieve.</td></tr><tr><td>to_value</td><td>in</td><td>instance_specific</td><td>Newest or largest entry to retrieve.</td></tr><tr><td>selected_values</td><td>in</td><td colspan="2">List of columns to retrieve: array ObjectDefinition If the array is empty (has no entries), all captured data is returned. Otherwise, only the columns specified in the array are returned. The type <i>ObjectDefinition</i> is specified above (<i>Capture_Objects</i>)</td></tr><tr><td>entries</td><td>out</td><td colspan="2">See <i>entries_in_use</i> above.</td></tr><tr><td>array (of <i>instance_specific_value</i>)</td><td>out</td><td colspan="2">Sorted sequence of entries containing the requested values of the capture objects.</td></tr><tr><td>data::= structure selected_values}</td><td colspan="3">{restricting_object; from_value; to_value;</td></tr></table> In the response “data” is an array of instancespecific value.			Reads all entries between the given limits.				restricting_object	in	ObjectDefinition	Defines the register or clock restricting the range of entries to be retrieved.	from_value	in	instance_specific	Oldest or smallest entry to retrieve.	to_value	in	instance_specific	Newest or largest entry to retrieve.	selected_values	in	List of columns to retrieve: array ObjectDefinition If the array is empty (has no entries), all captured data is returned. Otherwise, only the columns specified in the array are returned. The type <i>ObjectDefinition</i> is specified above (<i>Capture_Objects</i>)		entries	out	See <i>entries_in_use</i> above.		array (of <i>instance_specific_value</i>)	out	Sorted sequence of entries containing the requested values of the capture objects.		data::= structure selected_values}	{restricting_object; from_value; to_value;		
Reads all entries between the given limits.																																			
restricting_object	in	ObjectDefinition	Defines the register or clock restricting the range of entries to be retrieved.																																
from_value	in	instance_specific	Oldest or smallest entry to retrieve.																																
to_value	in	instance_specific	Newest or largest entry to retrieve.																																
selected_values	in	List of columns to retrieve: array ObjectDefinition If the array is empty (has no entries), all captured data is returned. Otherwise, only the columns specified in the array are returned. The type <i>ObjectDefinition</i> is specified above (<i>Capture_Objects</i>)																																	
entries	out	See <i>entries_in_use</i> above.																																	
array (of <i>instance_specific_value</i>)	out	Sorted sequence of entries containing the requested values of the capture objects.																																	
data::= structure selected_values}	{restricting_object; from_value; to_value;																																		

get_buffer_by_index (data)	Read all entries between the given limits.		
	from_index	in	double-long-unsigned First entry to retrieve.
	to_index	in	double-long-unsigned Last entry to retrieve.
	from_selected_value	in	long-unsigned index of first value to retrieve
	to_selected_value	in	long-unsigned index of last value to retrieve.
	entries	out	See <i>entries_in_use</i> above.
	array of <i>instance_specific_value</i>	out	Sorted sequence of entries containing the requested values of the capture objects.
	data::= structure {from_index; to_index; selected_values}.		
In the response “data” is an array of instance_specific_value.			

7.3 Association SN (class_id = 12, version = 0)

The version listed here was valid in Edition 1 and it is replaced as of Edition 2 and 3.

Device Association View		0..1 ^a	class_id = 12, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	object_list (static)	objlist_type				x + 0x08
Specific methods		m/o				
1.	getlist_by_classid	o				x + 0x20
2.	getobj_by_logicalname	o				x + 0x28
3.	read_by_logicalname()	o				x + 0x30
4.	get_attributes&services()	o				x + 0x38
5.	change_LLS_secret()	o				x + 0x40
6.	change_HLS_secret ()	o				x + 0x48
7.	get_HLS_challenge ()	o				x + 0x50
8.	reply_to_HLS_challenge()	o				x + 0x58
^a Per client server association.						

Attribute description

logical_name Identifies the type of the client-server association. See 6.2.27.

object_list	Contains the list of all objects with their short_name (DLMS ObjectName of the first attribute), class_id, version⁵ and <i>logical_name</i>. <pre> Objlist_type ::= array objlist_element objlist_element ::= structure { short_name: long, class_id: long-unsigned, version = unsigned, logical_name: octet-string } </pre>
Method description	
getlist_by_classid (data)	Delivers the subset of the object_list for a specific class_id. <pre> data ::= class_id: long-unsigned </pre> For the response: data ::= objlist_type
getobj_by_logicalname (data)	Delivers the entry of the object_list for a specific <i>logical_name</i> and class_id. <pre> data ::= structure { class_id: long-unsigned, logical_name: octet-string } </pre> For the response: data ::= objlist_element
read_by_logicalname (data)	Reads attributes for specific objects. The objects are specified by their class_id and their logical_name. <pre> data ::= array AttributeIdentification AttributeIdentification ::= structure { class_id: long-unsigned, logical_name: octet-string, attribute_index: unsigned } </pre> where attribute_index is a pointer (i.e. offset) to the attribute within the object. attribute_index = 0 delivers all attributes⁶, attribute_index = 1 delivers the first attribute (i.e. <i>logical_name</i>), etc. For the response: data is according to the type of the attribute.

⁵ Within an client-server association, there are never two objects with the same class_id and logical_name differing only in the versions.

⁶ If at least one attribute has no read access right under the current association, then a *read_by_logicalname* () to attribute index = 0 reveals the error message "scope-of-access-violated" (see IEC 61334-4-41:1996, p. 221).

get_attributes&services (data)	Delivers information about the access rights to the attributes and services within the actual association. The objects are specified by their <code>class_id</code> and their <code>logical_name</code>. array ObjectIdentification ObjectIdentification ::= structure { class_id: long-unsigned, logical_name: octet-string } For the response data ::= array AccessDescription AccessDescription ::= structure { read_attributes: bit-string, write_attributes: bit-string, services: bit-string } The position in the bit-string identifies the attribute/service (first position ↔ first attribute, first position ↔ first service), and the value of the bit specifies whether the attribute/service is available (bit set) or not available (bit clear).
change_LLS_secret (data)	Changes the LLS secret (for example password). data ::= octet-string new LLS secret
change_HLS_secret (data)	Changes the HLS secret (for example encryption key). data ::= octet-string⁷ new HLS secret
get_HLS_challenge (data)	Asks the server for the client “challenge” (for example random number). data ::= octet-string client challenge
reply_to_HLS_challenge (data)	Delivers the “secretly” processed “challenge” back to the server. data ::= octet-string client’s response to the challenge If the authentication is accepted, then the response is successful [0]. If the authentication is not accepted, then the response is set to data-access-error [1].

7.4 Association SN (`class_id` = 12, `version` = 1)

This IC allows modelling of AAs between a server and a client, using the application context *short name referencing*. A COSEM logical device may have one instance of this IC for each association the device is able to support.

The **short_name** of the current “Association SN” object itself is fixed within the COSEM context. It is given in 4.3 as 0xFA00.

⁷ The structure of the “new secret” depends on the security mechanism implemented. The “new secret” may contain additional checkbits and it may be encrypted.

Association SN	0...n	class_id = 12, version = 1			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. object_list (static)	objlist_type				x + 0x08
Specific methods	m/o				
1. reserved from previous versions	o				
2. reserved from previous versions	o				
3. read_by_logicalname (data)	o				x + 0x30
4. get_attributes&methods (data)	o				x + 0x38
5. change_LLS_secret (data)	o				x + 0x40
6. change_HLS_secret (data)	o				x + 0x48
7. reserved from previous versions					
8. reply_to_HLS_authentication (data)	o				x + 0x58

Attribute description

logical_name Identifies the “Association SN” object instance. See 6.2.27.

object_list Contains the list of all objects with their base_names (short_name), class_id, version and logical_name. The base_name is the DLMS objectName of the first attribute (logical_name).

objlist_type ::= array objlist_element

objlist_element ::= structure
 {
 base_name: long,
 class_id: long-unsigned,
 version: unsigned,
 logical_name: octet-string
 }

selective access (see 4.4) to the attribute *object_list* may be available (optional). The selective access parameters are as defined below.

Parameters for selective access to the *object_list* attribute

Access selector value	Parameter	Comment
1	class_id: long-unsigned	Delivers the subset of the object_list for a specific class_id. For the response: data ::= objlist_type
2	structure { class_id: long-unsigned, logical_name: octet-string }	Delivers the entry of the object_list for a specific class_id and logical_name. For the response: data ::= objlist_element

Method description

read_by_
logicalname

Reads attributes for selected objects. The objects are specified by their *class_id* and their *logical_name*. With this method, the parameterized access feature can also be used.

data::= array attribute_identification

attribute_identification::= structure

```
{
    class_id:      long-unsigned,
    logical_name:  octet-string,
    attribute_index: integer
}
```

where attribute_index is a pointer (i.e. offset) to the attribute within the object.

attribute_index 0 delivers all attributes ^a, attribute_index 1 delivers the first attribute (i.e. *logical_name*), etc.).

For the response: data is according to the type of the attribute.

get_attributes
&methods
(data)

Delivers information about the access rights to the attributes and methods within the actual association. The objects are specified by their *class_id* and their *logical_name*. With this method, the parameterized access feature can also be used.

data::= array object_identification

object_identification::= structure

```
{
    class_id:      long-unsigned,
    logical_name:  octet-string
}
```

For the response

data:= array access_description

access_description::= structure

```
{
    read_attributes:  bit-string,
    write_attributes: bit-string,
    methods:         bit-string
}
```

The position in the bit-string identifies the attribute/method (first position ↔ first attribute, first position ↔ first method) and the value of the bit specifies whether the attribute/method is available (bit set) or not available (bit clear).

Depending on the implementation, some attributes or methods of some objects may not be needed. In this case, such attributes or methods may not be accessible (neither read access, nor write access to attributes, nor access to methods).

change_LLS_
secret (data)

Changes the LLS secret (for example password).

data::= octet-string new LLS secret

change_HLS_secret (data)	Changes the HLS secret (for example encryption key). data::= octet-string ^b new HLS secret
reply_to_HLS_authentication (data)	The remote invocation of this method delivers the client's "secretly" processed "challenge StoC" (f(StoC)) back to the server as the <i>data</i> service parameter of the invoked Write.request service. data::= octet-string client's response to the challenge data::= octet-string server's response to the challenge If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back.

^a If at least one attribute has no read access right under the current association, then a *read_by_logicalname* () to attribute index 0 reveals the error message "scope of access violation" (see IEC 61334-4-41:1996, p. 221).

^b The structure of the "new secret" depends on the security mechanism implemented. The "new secret" may contain additional check bits and it may be encrypted.

7.5 Association SN (class_id = 12, version = 2)

COSEM logical devices able to establish AAs within a COSEM context using SN referencing, model the AAs using instances of the "Association SN" IC. A COSEM logical device may have one instance of this IC for each AA the device is able to support.

The **short_name** of the "Association SN" object itself is fixed within the COSEM context. See 4.3.

Association SN		0...n	class_id = 12, version = 2			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	object_list (static)	objlist_type				x + 0x08
3.	access_rights_list (static)	access_rights_type				x + 0x10
4.	security_setup_reference (static)	octet-string				x + 0x18
Specific methods		m/o				
1.	reserved from previous versions	o				
2.	reserved from previous versions	o				
3.	read_by_logicalname (data)	o				x + 0x30
4.	reserved from previous versions	o				
5.	change_secret (data)	o				x + 0x40
6.	reserved from previous versions	o				
7.	reserved from previous versions					
8.	reply_to_HLS_authentication (data)	o				x + 0x58

Attribute description	
logical_name	Identifies the “Association SN” object instance. See 6.2.27.
object_list	Contains the list of all objects with their <i>base_name</i> (<i>short_name</i>), <i>class_id</i>, <i>version</i> and <i>logical_name</i>. The <i>base_name</i> is the DLMS objectName of the first attribute (<i>logical_name</i>). <pre> objlist_type ::= array objlist_element objlist_element ::= structure { base_name: long, class_id: long-unsigned, version: unsigned, logical_name: octet-string } </pre> <i>selective access</i> (see 4.4) to the attribute <i>object_list</i> may be available. The access selector values and their parameters are as defined below.
access_rights_list	Contains the access rights to attributes and methods. The link between the <i>object_list</i> and the <i>access_rights_list</i> is the <i>base_name</i>, present in both the <i>objlist_element</i> structure and the <i>access_right_element</i> structure. Therefore, the <i>base_names</i> on the two lists shall be the same. The number – and preferably, the order – of the elements in the array of <i>objlist_element</i> and the array of <i>access_right_element</i> shall also be the same. <pre> access_rights_type ::= array access_rights_element access_rights_element ::= structure { base_name: long, attribute_access: attribute_access_descriptor, method_access: method_access_descriptor } attribute_access_descriptor ::= array attribute_access_item attribute_access_item ::= structure { attribute_id: integer, access_mode: enum: (0) no_access, (1) read_only, (2) write_only, (3) read_and_write, (4) authenticated_read_only, (5) authenticated_write_only, (6) authenticated_read_and_write access_selectors: CHOICE { null-data [0], array integer [1] } } method_access_descriptor ::= array method_access_item </pre>

access_rights_list cont'd	method_access_item::= structure <pre> { method_id: integer, access_mode: enum: (0) no_access, (1) access, (2) authenticated_access } </pre> <i>selective access</i> (see 4.4) to the attribute <i>access_rights_list</i> may be available (optional). The access selector values and their parameters are as defined below.
security_setup_reference	References the “Security setup” object by its <i>logical_name</i> . The referenced object manages security for a given “Association SN” object instance.

Parameters for selective access to the *object_list* and *access_rights_list* attribute

Access selector value	Parameter	Available with attribute	Comment
1	class_id: long-unsigned	2	Delivers the subset of the <i>object_list</i> for a specific <i>class_id</i> . For the response: data::= objlist_type
2	<pre> structure { class_id: long-unsigned, logical_name: octet-string } </pre>	2	Delivers the entry of the <i>object_list</i> for a specific <i>class_id</i> and <i>logical_name</i> . For the response: data::= objlist_element
3	base_name: long	2, 3	In the case of attribute 2, delivers the entry of the <i>object_list</i> for a specific <i>base_name</i> . For the response: data::= objlist_element In the case of attribute 3, delivers the entry of the <i>access_rights_list</i> for a specific <i>base_name</i> . For the response: data::=access_rights_element

Method description

read_by_logicalname (data) Reads attributes for selected objects. The objects are specified by their *class_id* and their *logical_name*. With this method, the parameterized access feature can also be used.

data::= array attribute_identification

attribute_identification::= structure

```

{
    class_id:          long-unsigned,
    logical_name:      octet-string,
    attribute_index:    integer
}

```

where *attribute_index* is a pointer (i.e. offset) to the attribute within the object.

attribute_index 0 delivers all attributes; *attribute_index* 1 delivers the first attribute (i.e. *logical_name*), etc.).

For the response: data is according to the type of the attribute.

NOTE 1 If at least one attribute has no read access right under the current association, then a *read_by_logicalname* () to attribute index 0 reveals the error message “scope-of-access-violated”, see IEC 62056-5-3:2016, Clause 8.

change_secret (data) Changes the LLS or HLS secret (for example password).

data::= octet-string new secret

NOTE 2 The structure of the “new secret” depends on the security mechanism implemented. The “new secret” may contain additional check bits and it may be encrypted.

NOTE 3 In the case of HLS with GMAC, the (HLS_) secret is held by the “Security setup” object referenced in attribute.

reply_to_HLS_authentication (data) The remote invocation of this method delivers to the server the result of the secret processing by the client of the server’s challenge to the client, f(StoC), as the data service parameter of the Read.request primitive invoked with parameterised access.

data::= octet-string client’s response to the challenge

If the authentication is accepted, then the response (Read.confirm primitive) contains Result == OK and the result of the secret processing by the server of the client’s challenge to the server, f(CtoS) in the data service parameter of the Read.response service.

data::= octet-string server’s response to the challenge

If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back.

7.6 Association LN (class_id = 15, version = 0)

This IC allows modelling of AAs between a server and a client, using the application context *logical name referencing*. Each AA is modelled by one Association LN object.

Association LN		0...MaxNborAss	class_id = 15, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	object_list (static)	object_list_type				x + 0x08
3.	associated_partners_id	associated_partners_type				x + 0x10
4.	application_context_name	context_name_type				x + 0x18
5.	xDLMS_context_info	xDLMS-context-type				x + 0x20
6.	authentication_mechanism_name	mechanism_name_type				x + 0x28
7.	LLS_secret	octet-string				x + 0x30
8.	association_status	enum				x + 0x38
Specific methods		m/o				
1.	reply_to_HLS_authentication (data)	o				x + 0x60
2.	change_HLS_secret (data)	o				x + 0x68
3.	add_object (data)	o				x + 0x70
4.	remove_object (data)	o				x + 0x78

Attribute description

logical_name	Identifies the “Association LN” object instance. See 6.2.27.
object_list	Contains the list of visible COSEM objects with their class_id, version, logical name and the access rights to their attributes and methods within the given application association. object_list_type ::= array object_list_element object_list_element ::= structure <pre> { class_id: long-unsigned, version: unsigned, logical_name: octet-string, access_rights: access_right } </pre> access_right ::= structure <pre> { attribute_access: attribute_access_descriptor, method_access: method_access_descriptor } </pre> attribute_access_descriptor ::= array attribute_access_item attribute_access_item ::= structure <pre> { attribute_id: integer, access_mode: enum; (0) no_access, (1) read_only, (2) write_only, (3) read_and_write access_selectors: CHOICE { null-data [0], array integer [1] } } </pre> method_access_descriptor ::= array method_access_item method_access_item ::= structure <pre> { method_id: integer, access_mode: boolean } </pre> Where: – the attribute_access_descriptor and the method_access_descriptor always contain all implemented attributes or methods; – access_selectors contain a list of the supported selector values; <i>selective access</i> (see 4.4) to the attribute <i>object_list</i> may be available (optional). The selective access parameters are as defined below.

associated_partners_id	Contains the identifiers of the COSEM client and server (logical device) APs within the physical devices hosting these processes, which belong to the AA modelled by the “Association LN” object. <pre>associated_partners_type ::= structure { client_SAP: integer, server_SAP: long-unsigned } The range for the client_SAP is 0...0x7F. The range for the server_SAP is 0x000...0x3FFF.</pre>
application_context_name	In the COSEM environment, it is intended that an application context pre-exists and is referenced by its name during the establishment of an AA. This attribute contains the name of the application context for that AA. <pre>context_name_type ::= CHOICE { context_name_structure [2], octet-string [9] } The application context name is specified as OBJECT IDENTIFIER in IEC 62056-5-3:2016, 7.2.2.2</pre> When the context_name_type is encoded as a structure, it includes the arc labels of the OBJECT IDENTIFIER. <pre>context_name_structure ::= structure { joint_iso_ctt_element: unsigned, country_element: unsigned, country_name_element: long-unsigned, identified_organization_element: unsigned, DLMS_UA_element: unsigned, application_context_element: unsigned, context_id_element: unsigned } Example 1: In the case of context_id(1) the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 01 11 01 (all values are hexadecimal). When the context_name_type is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See IEC 62056-5-3:2016, Clause C.4. Example 2: In the case of context_id(1) the A-XDR encoding is: 09 07 60 85 74 05 08 01 01 (all values are hexadecimal). </pre>
xDLMS_context_info	Contains all the necessary information on the xDLMS context for the given AA. <pre>xDLMS-context-type ::= structure { conformance: bit-string, max_receive_pdu_size: long-unsigned, max_send_pdu_size: long-unsigned, dlms_version_number: unsigned, quality_of_service: integer, cyphering_info: octet-string }</pre>

xDLMS_ context_info cont'd	Where: – the conformance element contains the xDLMS conformance block supported by the server; – the max_receive_pdu_size element contains the maximum length for an xDLMS APDU, expressed in bytes that the client may send. This is the same as the server-max-receive-pdu-size parameter of the xDLMS initiateResponse APDU (see IEC 62056-5-3:2016, Clause 8); – the max_send_pdu_size, in an active association contains the maximum length for an xDLMS APDU, expressed in bytes that the server may send. This is the same as the client-max-receive-pdu-size parameter of the xDLMS initiateRequest APDU (see IEC 62056-5-3:2016, Clause 8); – the dlms_version_number element contains the DLMS version number supported by the server; – the quality_of_service element is not used; – the cyphering_info, in an active association, contains the dedicated key parameter of the xDLMS initiateRequest APDU (see IEC 62056-5-3:2016, Clause 8).
authentication_ mechanism_ name	Contains the name of the authentication mechanism for the AA. mechanism_name_type ::= CHOICE <pre> { mechanism_name_structure [2] octet-string [9] } </pre> The authentication mechanism name is specified as an OBJECT IDENTIFIER in IEC 62056-5-3:2016, 7.2.2.3. When the mechanism_name_type is encoded as a structure, it includes the arc labels of the OBJECT IDENTIFIER. mechanism_name_structure ::= structure <pre> { joint_iso_ctt_element: unsigned, country_element: unsigned, country_name_element: long-unsigned, identified_organization_element: unsigned, DLMS_UA_element: unsigned, authentication_mechanism_name_element: unsigned, mechanism_id_element: unsigned } </pre> EXAMPLE 3: In the case of mechanism_id(1) the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 02 11 01 (all values are hexadecimal): When the mechanism_name_type is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See IEC 62056-5-3:2016, Clause C.4. EXAMPLE 4: In the case of mechanism_id(1) the A-XDR encoding is: 09 07 60 85 74 05 08 02 01 (all values are hexadecimal). No mechanism_name is required when no authentication is used.
LLS_secret	Contains the authentication value for the LLS authentication process.

association_status	Indicates the current status of the association, which is modelled by the object. enum: (0) non-associated, (1) association-pending, (2) associated

Parameters for selective access to the *object_list* attribute

- if no selective access is requested, (no *Access_Selection_Parameters* parameter is present in the *GET.request (.indication)* service primitive for the *object_list* attribute) the corresponding *.response (.confirmation)* service shall contain all *object_list_element* of the *object_list* attribute;
- when selective access is requested to the *object_list* attribute (the *Access_Selection_Parameters* parameter is present), the response shall contain a 'filtered' list of *object_list_element*, as follows:

Access selector	Access parameter	Comment
1	NULL	All information excluding the <i>access_rights</i> shall be included in the response.
2	<i>class_list</i>	Access by class. In this case, only those <i>object_list_element</i> of the <i>object_list</i> shall be included in the response, which have a <i>class_id</i> equal to one of the <i>class_id</i> -s of the class-list. No <i>access_right</i> information is included. <i>class_list</i> ::= array <i>class_id</i> <i>class_id</i> ::= long-unsigned
3	<i>object_id_list</i>	Access by object. The full information record of object instances on the <i>object_id_list</i> shall be returned. <i>object_id_list</i> ::= array <i>object_id</i> <i>object_id</i> ::= structure { <i>class_id</i> : long-unsigned, <i>logical_name</i> : octet-string }
4	<i>object_id</i>	The full information record of the required COSEM object instance shall be returned. <i>object_id</i> : See above.

Method description

reply_to_HLS_authentication (data)

The remote invocation of this method delivers the client's "secretly" processed "challenge StoC" (f(StoC)) back to the server as the *data* service parameter of the *ACTION.request* primitive invoked.

data ::= octet-string client's response to the challenge

If the authentication is accepted, then the response (*ACTION.confirm* primitive) contains *Result* == OK and the server's "secretly" processed "challenge CtoS" (f(CtoS)) back to the client in the *data* service parameter of the response service.

data ::= octet-string server's response to the challenge

If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back.

change_HLS_secret (data)	Changes the HLS secret (for example encryption key). data::= octet-string ^a new HLS secret
add_object (data)	Adds the referenced object to the object_list. data::= object_list_element (see above)
remove_object (data)	Removes the referenced object from the object_list. data::= object_list_element (see above)

^a The structure of the “new secret” depends on the security mechanism implemented. The “new secret” may contain additional check bits and it may be encrypted.

7.7 Association LN (class_id = 15, version = 1)

COSEM logical devices able to establish AAs within a COSEM context using LN referencing, model the AAs through instances of the “Association LN” IC. A COSEM logical device has one instance of this IC for each AA the device is able to support.

Association LN		0...MaxNbOfAss.	class_id = 15, version = 1			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	object_list (static)	object_list_type				x + 0x08
3.	associated_partners_id	associated_partners_type				x + 0x10
4.	application_context_name	context_name_type				x + 0x18
5.	xDLMS_context_info	xDLMS_context_type				x + 0x20
6.	authentication_mechanism_name	mechanism_name_type				x + 0x28
7.	secret	octet-string				x + 0x30
8.	association_status	enum				x + 0x38
9.	security_setup_reference (static)	octet-string				x + 0x40
Specific methods		m/o				
1.	reply_to_HLS_authentication (data)	o				x + 0x60
2.	change_HLS_secret (data)	o				x + 0x68
3.	add_object (data)	o				x + 0x70
4.	remove_object (data)	o				x + 0x78

Attribute description

logical_name	Identifies the “Association LN” object instance. See 6.2.27.
object_list	Contains the list of visible COSEM objects with their class_id, version, logical name and the access rights to their attributes and methods within the given application association. object_list_type::= array object_list_element

object_list	object_list_element ::= structure
cont'd	<pre> { class_id: long-unsigned, version: unsigned, logical_name: octet-string, access_rights: access_right } access_right ::= structure { attribute_access: attribute_access_descriptor, method_access: method_access_descriptor } attribute_access_descriptor ::= array attribute_access_item attribute_access_item ::= structure { attribute_id: integer, access_mode: enum: (0) no_access, (1) read_only, (2) write_only, (3) read_and_write, (4) authenticated_read_only, (5) authenticated_write_only, (6) authenticated_read_and_write access_selectors: CHOICE { null-data [0], array integer [1] } } method_access_descriptor ::= array method_access_item method_access_item ::= structure { method_id: integer, access_mode: enum: (0) no_access, (1) access, (2) authenticated_access } </pre>
	Where:
	- the attribute_access_descriptor and the method_access_descriptor elements always contain all implemented attributes or methods; - access_selectors contain a list of the supported selector values.
	<i>selective access</i> (see 4.4) to the attribute <i>object_list</i> may be available (optional). The selective access parameters are as defined below.

associated_partners_id	Contains the identifiers of the COSEM client and server (logical device) APs within the physical devices hosting these APs, which belong to the AA modelled by the “Association LN” object. <pre>associated_partners_type ::= structure { client_SAP: integer, server_SAP: long-unsigned }</pre> The range for the client_SAP is 0...0x7F. The range for the server_SAP is 0x0000...0x3FFF. The SAPs shall be in the range allowed by the data type and the media.
application_context_name	In the COSEM environment, it is intended that an application context pre-exists and is referenced by its name during the establishment of an AA. This attribute contains the name of the application context for that AA. <pre>context_name_type ::= CHOICE { context_name_structure [2], octet-string [9] }</pre> The application context name is specified as OBJECT IDENTIFIER in IEC 62056-5-3:2016, 7.2.2.2. When the context_name_type is encoded as a structure, it includes the arc labels of the OBJECT IDENTIFIER. <pre>context_name_structure ::= structure { joint_iso_ctt_element: unsigned, country_element: unsigned, country_name_element: long-unsigned, identified_organization_element: unsigned, DLMS_UA_element: unsigned, application_context_element: unsigned, context_id_element: unsigned }</pre> EXAMPLE 1: In the case of context_id(1) the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 01 11 01 (all values are hexadecimal). When the context_name_type is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See IEC 62056-5-3:2016, Clause C.4. EXAMPLE 2: In the case of context_id(1) the A-XDR encoding is: 09 07 60 85 74 05 08 01 01 (all values are hexadecimal).
xDLMS_context_info	Contains all the necessary information on the xDLMS context for the given AA. <pre>xDLMS_context_type ::= structure { conformance: bit-string, max_receive_pdu_size: long-unsigned, max_send_pdu_size: long-unsigned, dlms_version_number: unsigned, quality_of_service: integer, cyphering_info: octet-string }</pre>

xDLMS_ context_info cont'd	Where: – the conformance element contains the xDLMS conformance block supported by the server; – the max_receive_pdu_size element contains the maximum length for an xDLMS APDU, expressed in bytes that the client may send. This is the same as the server-max-receive-pdu-size parameter of the xDLMS initiateResponse APDU; – the max_send_pdu_size element, in an active AA contains the maximum length for an xDLMS APDU, expressed in bytes that the server may send. This is the same as the client-max-receive-pdu-size parameter of the xDLMS initiateRequest APDU; – the dlms_version_number element contains the DLMS version number supported by the server; – the quality_of_service element is not used; – the cyphering_info element, in an active association, contains the dedicated key parameter of the xDLMS initiateRequest APDU. See IEC 62056-5-3:2016, Clause 8.
authentication_ mechanism_ name	Contains the name of the authentication mechanism for the AA. mechanism_name_type ::= CHOICE <pre> { mechanism_name_structure [2], octet-string [9] } </pre> The authentication mechanism name is specified as an OBJECT IDENTIFIER in IEC 62056-5-3:2016, 7.2.2.3. When the mechanism_name_type is encoded as a structure, it includes the arc labels of the OBJECT IDENTIFIER. mechanism_name_structure ::= structure <pre> { joint_iso_ctt_element: unsigned, country_element: unsigned, country_name_element: long-unsigned, identified_organization_element: unsigned, DLMS_UA_element: unsigned, authentication_mechanism_name_element: unsigned, mechanism_id_element: unsigned } </pre> Example 3: In the case of mechanism_id(1) the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 02 11 01 (all values are hexadecimal): When the mechanism_name_type is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See IEC 62056-5-3:2016, Clause C.4. EXAMPLE 4 In the case of mechanism_id(1) the A-XDR encoding is: 09 07 60 85 74 05 08 02 01 (all values are hexadecimal). No mechanism_name is required when no authentication is used.
secret	Contains the secret for the LLS or HLS authentication process. NOTE In the case of HLS with GMAC, the (HLS_)secret is held by the Security setup object referenced in attribute 9.

association_status	Indicates the current status of the association, which is modelled by the object. enum: (0) non-associated, (1) association-pending, (2) associated
security_setup_reference	References the Security setup object by its logical name. The referenced object manages security for a given Association LN object instance.

A SET operation on an attribute of an association LN object becomes effective when this association object is used to establish a new association.

Parameters for selective access to the object_list attribute

- If no selective access is requested, (no Access_Selection_Parameters parameter is present in the GET.request (.indication) service primitive for the object_list attribute) the corresponding .response (.confirmation) service shall contain all object_list_elements of the object_list attribute.
- When selective access is requested to the object_list attribute (the Access_Selection_Parameter parameter is present), the response shall contain a 'filtered' list of object_list_elements, as follows:

Access selector	Access parameter	Comment
1	NULL	All information excluding the access_rights shall be included in the response.
2	class_list	Access by class_id. In this case, only those object_list_elements of the object_list shall be included in the response, which have a class_id equal to one of the class_id-s of the class_list. No access_right information is included. class_list ::= array class_id class_id: long-unsigned
3	object_id_list	Access by object. The full information record of object instances on the object_id_list shall be returned. object_id_list ::= array object_id object_id ::= structure { class_id: long-unsigned, logical_name: octet-string }
4	object_id	The full information record of the required COSEM object instance shall be returned. object_id ::= structure See above.

Method description**reply_to_HLS_authentication (data)**

The remote invocation of this method delivers to the server the result of the secret processing by the client of the server's challenge to the client, *f(StoC)*, as the *data* service parameter of the ACTION.request primitive invoked.

data::= octet-string client's response to the challenge

If the authentication is accepted, then the response (ACTION.confirm primitive) contains Result == OK and the result of the secret processing by the server of the client's challenge to the server, *f(CtoS)* in the *data* service parameter of the response service.

data::= octet-string server's response to the challenge

If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back.

change_HLS_secret (data)

Changes the HLS secret (for example encryption key).

data::= octet-string new HLS secret

The structure of the "new secret" depends on the security mechanism implemented. The "new secret" may contain additional check bits and it may be encrypted.

add_object (data)

Adds the referenced object to the *object_list*.

data::= object_list_element (see above)

remove_object (data)

Removes the referenced object from the *object_list*.

data::= object_list_element (see above)

7.8 IEC local port setup (class_id = 19, version = 0)

Instances of this IC define the operational parameters for communication using IEC 62056-21:2002. Several ports can be configured.

IEC local port setup		0...n	class_id = 19, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. default_mode	(static)	enum				x + 0x08
3. default_baud	(static)	enum				x + 0x10
4. prop_baud	(static)	enum				x + 0x18
5. response_time	(static)	enum				x + 0x20
6. device_addr	(static)	octet-string				x + 0x28
7. pass_p1	(static)	octet-string				x + 0x30
8. pass_p2	(static)	octet-string				x + 0x38
9. pass_w5	(static)	octet-string				x + 0x40
Specific methods		m/o				

Attribute description	
logical_name	Identifies the “IEC local port setup” object instance. See 6.2.16.
default_mode	Defines the protocol used by the meter on the port. enum: (0) protocol according to IEC 62056-21:2002 (modes A...E) (1) protocol according to IEC 62056-46:2007, Clause 8. Using this enumeration value all other attributes of this IC are not applicable.
default_baud	Defines the baud rate for the opening sequence enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud
prop_baud	Defines the baud rate to be proposed by the meter enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud
response_time	Defines the minimum time between the reception of a request (end of request telegram) and the transmission of the response (begin of response telegram). enum: (0) 20 ms, (1) 200 ms
device_addr	Device address according to IEC 62056-21:2002.
pass_p1	Password 1 according to IEC 62056-21:2002.
pass_p2	Password 2 according to IEC 62056-21:2002.
pass_w5	Password W5 reserved for national applications.

7.9 IEC HDLC setup, (class_id = 23, version = 0)

An instance of the “IEC HDLC setup” contains all data necessary to set up a communication channel according to IEC 62056-46:2007. Several communication channels can be configured.

IEC HDLC setup		0...n	class_id = 23, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	comm_speed (static)	enum	0	9	5	x + 0x08
3.	window_size_transmit (static)	unsigned	1	7	1	x + 0x10
4.	window_size_receive (static)	unsigned	1	7	1	x + 0x18
5.	max_info_field_length_transmit (static)	unsigned	32	128	128	x + 0x20
6.	max_info_field_length_receive (static)	unsigned	32	128	128	x + 0x28
7.	inter_octet_time_out (static)	long-unsigned	20	1000	25	x + 0x30
8.	inactivity_time_out (static)	long-unsigned	0		120	x + 0x38
9.	device_address (static)	long-unsigned	0x0010	0x3FFD		x + 0x40
Specific methods		m/o				

Attribute description

logical_name Identifies the “IEC HDLC setup” object instance. See 6.2.18.

comm_speed The communication speed supported by the corresponding port:

enum:

- (0) 300 baud,
- (1) 600 baud,
- (2) 1 200 baud,
- (3) 2 400 baud,
- (4) 4 800 baud,
- (5) 9 600 baud,
- (6) 19 200 baud,
- (7) 38 400 baud,
- (8) 57 600 baud,
- (9) 115 200 baud

This communication speed can be overridden if the HDLC mode of a device is entered through a special mode of another protocol.

window_size_transmit The maximum number of frames that a device or system can transmit before it needs to receive an acknowledgement from a corresponding station. During logon, other values can be negotiated.

window_size_receive The maximum number of frames that a device or system can receive before it needs to transmit an acknowledgement to the corresponding station. During logon, other values can be negotiated.

max_info_length_transmit The maximum information field length that a device can transmit. During logon, a smaller value can be negotiated.

max_info_length_receive The maximum information field length that a device can receive. During logon, a smaller value can be negotiated.

inter_octet_time_out Defines the time, expressed in milliseconds, over which, when no character is received from the primary station, the device will treat the already received data as a complete frame.

inactivity_time_out Defines the time, expressed in seconds over which, when no frame is received from the primary station, the device will process a disconnection.

When this value is set to 0, this means that the *inactivity_time_out* is not operational.

device_address Contains the physical device address of a device:

In the case of single byte addressing:

0x00 NO_STATION Address,
0x01...0x0F Reserved for future use,
0x10...0x7D Usable address space,
0x7E 'CALLING' device address,
0x7F Broadcast address

In the case of double byte addressing:

0x0000 NO_STATION address,
0x0001...0x000F Reserved for future use,
0x0010...0x3FFD Usable address space,
0x3FFE 'CALLING' physical device address,
0x3FFF Broadcast address

7.10 IEC twisted pair (1) setup (class_id = 24, version = 0)

NOTE The use of version 0 of the IEC twisted pair (1) setup IC is deprecated.

This IC allows modelling and configuring communication channels according to IEC 62056-31:1999. Several communication channels can be configured.

IEC twisted pair (1) setup	0...n	class_id = 24, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. secondary_address (static)	octet-string				x + 0x08
3. primary_address_list (static)	primary_address_list_type				x + 0x10
4. tabi_list (static)	tabi_list_type				x + 0x18
5. fatal_error (dyn.)	enum				x + 0x20
Specific methods	m/o				

Attribute description

logical_name Identifies the "IEC twisted pair setup" object instance. See 6.2.19.

secondary_address *Secondary_address* memorizes the ADS of the secondary station that corresponds to the real equipment.
octet-string (SIZE(6))

primary_address_list *Primary_address_list* memorizes the list of ADP or primary station physical addresses for which each logical device of the real equipment (the secondary station) has been programmed.

primary_address_list_type ::= array *primary_address_element*

primary_address_element ::= octet-string

The length of the octet-string is one octet.

tabi_list *tabi_list* represents the list of the TAB(i) for which the real equipment (the secondary station) has been programmed in case of forgotten station call.

tabi_list_type ::= array *tabi_element*

tabi_element ::= integer

fatal_error *fatal_error* represents the last occurrence of one of the fatal errors of the protocols described in IEC 62056-31:1999.

The initial default value of this variable is 0x00. Then, each fatal error is spotted.

enum:

- (0) No-error,
- (1) t-EP-1F,
- (2) t-EP-2F,
- (3) t-EL-4F,
- (4) t-EL-5F,
- (5) eT-1F,
- (6) eT-2F,
- (7) e-EP-3F,
- (8) e-EP-4F,
- (9) e-EP-5F,
- (10) e-EL-2F

7.11 PSTN modem configuration (class_id = 27, version = 0)

NOTE The name of this IC was changed to "Modem configuration" in Edition 6.0.

An instance of the "PSTN modem configuration" IC stores data related to the initialization of modems, which are used for data transfer from/to a device. Several modems can be configured.

PSTN modem configuration		0...n	class_id = 27, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. comm_speed	(static)	enum	0	9	5	x + 0x08
3. initialization_string	(static)	array				x + 0x10
4. modem_profile	(static)	array				x + 0x18
Specific methods		m/o				

Attribute description

logical_name	Identifies the "PSTN modem configuration" object instance. See 6.2.6.
comm_speed	The communication speed between the device and the modem, not necessarily the communication speed on the WAN.

enum:

- (0) 300 baud,
- (1) 600 baud,
- (2) 1 200 baud,
- (3) 2 400 baud,
- (4) 4 800 baud,
- (5) 9 600 baud,
- (6) 19 200 baud,
- (7) 38 400 baud,
- (8) 57 600 baud,
- (9) 115 200 baud

initialization_string This data contains all the necessary initialization commands to be sent to the modem in order to configure it properly. This may include the configuration of special modem features.

array initialization_string_element

initialization_string_element ::= structure
 {
 request: octet-string,
 response: octet-string
 }

If the array contains more than one initialization string element, they are subsequently sent to the modem after receiving an answer matching the defined response.

REMARK It is assumed that the modem is pre-configured so that it accepts the initialization_string. If no initialization is needed, the initialization string is empty.

modem_profile This data defines the mapping from Hayes standard commands/responses to modem specific strings.

array modem_profile_element

modem_profile_element: octet-string

The *modem_profile* array shall contain the corresponding strings for the modem used in following order:

- Element 0: OK,
- Element 1: CONNECT,
- Element 2: RING,
- Element 3: NO CARRIER,
- Element 4: ERROR,
- Element 5: CONNECT 1 200,
- Element 6: NO DIAL TONE,
- Element 7: BUSY,
- Element 8: NO ANSWER,
- Element 9: CONNECT 600,
- Element 10: CONNECT 2 400,
- Element 11: CONNECT 4 800,
- Element 12: CONNECT 9 600,
- Element 13: CONNECT 14 400,
- Element 14: CONNECT 28 800,
- Element 15: CONNECT 36 600,
- Element 16: CONNECT 56 000

7.12 Auto answer (class_id = 28, version = 0)

This IC allows modelling how the device manages the “Auto answer” function of the modem, i.e. answering of incoming calls. Several modems can be configured.

Auto answer		0...n	class_id = 28, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	mode (static)	enum				x + 0x08
3.	listening_window (static)	array				x + 0x10
4.	status (dyn.)	enum				x + 0x18
5.	number_of_calls (static)	unsigned				x + 0x20
6.	number_of_rings (static)	nr_rings_type				x + 0x28
Specific methods		m/o				

Attribute description

logical_name Identifies the “Auto answer” object instance. See 6.2.6.

mode Defines the working mode of the line when the device is auto answer.

enum:

(0) line dedicated to the device,

(1) shared line management with a limited number of calls allowed. Once the number of calls is reached, the window status becomes inactive until the next start date, whatever the result of the call,

(2) shared line management with a limited number of successful calls allowed. Once the number of successful communications is reached, the window status becomes inactive until the next start date,

(3) currently no modem connected,

(200...255) manufacturer specific modes

listening_window Contains the start and end instant when the window becomes active (for the start instant), and inactive (for the end instant). The start_date defines implicitly the period.

EXAMPLE When the day of month is not specified (equal to 0xFF) this means that we have a daily share line management. Daily, monthly ...window management can be defined.

array window_element

window_element ::= structure

```
{
    start_time: octet-string,
    end_time: octet-string
}
```

start_time and end_time are formatted as specified in 4.6.1 for *date-time*.

status	Here is defined the status of the window. enum: (0) Inactive: the device will manage no new incoming call. This status is automatically reset to Active when the next listening window starts, (1) Active: the device can answer to the next incoming call, (2) Locked: This value can be set automatically by the device or by a specific client when this client has completed its reading session and wants to give the line back to the customer before the end of the window duration. This status is automatically reset to Active when the next listening window starts.
number_of_calls	This number is the reference used in modes 1 and 2. When set to 0, this means there is no limit.
number_of_rings	Defines the number of rings before the meter connects the modem. Two cases are distinguished: number of rings within the window defined by attribute <i>listening_window</i> and number of rings outside the <i>listening_window</i>. <pre>nr_rings_type ::= structure { nr_rings_in_window: unsigned, (0: no connect in window) nr_rings_out_of_window: unsigned (0: no connect out of window) }</pre>

7.13 PSTN auto dial (class_id = 29, version = 0)

NOTE The name of this IC was changed to "Auto connect" in IEC 62056-6-2:2006.

An instance of the "PSTN auto dial" IC stores data related to the management data transfer between the device and the modem to perform auto dialling. Several modems can be configured. The logical_names shall be as defined in 6.2.6.

PSTN auto dial		0...n	class_id = 29, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. mode	(static)	enum				x + 0x08
3. repetitions	(static)	unsigned				x + 0x10
4. repetition_delay	(static)	long-unsigned				x + 0x18
5. calling_window	(static)	array				x + 0x20
6. phone_list	(static)	array				x + 0x28
Specific methods		m/o				

Attribute description	
logical_name	Identifies the “PSTN auto dial” object instance. See 6.2.6.
mode	Defines if the device can perform auto-dialling. enum: (0) no auto dialling, (1) auto dialling allowed anytime, (2) auto dialling allowed within the validity time of the calling window, (3) “regular” auto dialling allowed within the validity time of the calling window; “alarm” initiated auto dialling allowed anytime, (200...255) manufacturer specific modes
repetitions	The maximum number of trials in the case of unsuccessful dialling attempts.
repetition_delay	The time delay, expressed in seconds until an unsuccessful dial attempt can be repeated. repetition_delay 0 means delay is not specified
calling_window	Contains the start and end date/time stamp when the window becomes active (for the start instant), or inactive (for the end instant). The start_date defines implicitly the period. EXAMPLE When day of month is not specified (equal to 0xFF) this means that we have a daily share line management. Daily, monthly ...window management can be defined. array window_element window_element ::= structure { start_time: octet-string, end_time: octet-string } start_time and end_time are formatted as specified in 4.6.1 for <i>date-time</i> .
phone_list	Contains phone numbers, the device modem has to call under certain conditions. The link between entries in the array and the conditions are not contained in here. array phone_number phone_number ::= octet-string

7.14 Auto connect (class_id = 29, version = 1)

This IC allows modelling the management of data transfer from the device to one or several destinations.

The messages to be sent, the conditions on which they shall be sent and the relation between the various modes, the calling windows and destinations are not defined here.

Depending on the mode, one or more instances of this IC may be necessary to perform the function of sending out messages.

Auto connect		0...n	class_id = 29, version = 1			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	mode (static)	enum				x + 0x08
3.	repetitions (static)	unsigned				x + 0x10
4.	repetition_delay (static)	long-unsigned				x + 0x18
5.	calling_window (static)	array				x + 0x20
6.	destination_list (static)	array				x + 0x28
Specific methods		m/o				

Attribute description

logical_name Identifies the “Auto connect” object instance. See 6.2.6.

mode Defines the mode controlling the auto dial functionality concerning the timing, the message type to be sent and the infrastructure to be used.
enum:
(0) no auto dialling,
(1) auto dialling allowed anytime,
(2) auto dialling allowed within the validity time of the calling window,
(3) “regular” auto dialling allowed within the validity time of the calling window; “alarm” initiated auto dialling allowed anytime,
(4) SMS sending via Public Land Mobile Network (PLMN),
(5) SMS sending via PSTN,
(6) email sending.
(200..255) manufacturer specific modes

repetitions The maximum number of trials in the case of unsuccessful dialling attempts.

repetition_delay The time delay, expressed in seconds until an unsuccessful dial attempt can be repeated.
repetition_delay 0 means delay is not specified

calling_window Contains the start and end date/time stamp when the window becomes active (for the start instant), or inactive (for the end instant). The start_date defines implicitly the period.

EXAMPLE When day of month is not specified (equal to 0xFF) this means that we have a daily share line management. Daily, monthly ...window management can be defined.

array window_element

window_element ::= structure

```
{
    start_time: octet-string,
    end_time: octet-string
}
```

start_time and end_time are formatted as specified in 4.6.1 for *date-time*.

destination_list Contains the list of destinations (for example phone numbers, email addresses or their combinations) where the message(s) have to be sent under certain conditions. The conditions and their link to the elements of the array are not defined here.

array destination

destination::= octet-string

7.15 S-FSK Phy&MAC setup (class_id = 50, version = 0)

NOTE 1 The use of this version is deprecated.

An instance of the “S-FSK Phy&MAC setup” IC stores the data necessary to set up and manage the physical and the MAC layer of the PLC S-FSK lower layer profile.

S-FSK Phy&MAC setup		0...n				class_id = 50, version = 0			
Attributes		Data type		Min.	Max.	Def.	Short name		
1.	logical_name (static)	octet-string					x		
2.	initiator_electrical_phase (static)	enum		0	3		x + 0x08		
3.	delta_electrical_phase (dyn.)	enum		0	6		x + 0x10		
4.	max_receiving_gain (static)	unsigned					x + 0x18		
5.	max_transmitting_gain (static)	unsigned					x + 0x20		
6.	search_initiator_gain (static)	unsigned					x + 0x28		
7.	frequencies (static)	frequencies_type					x + 0x30		
8.	mac_address (dyn.)	long-unsigned				FFE	x + 0x38		
9.	mac_group_addresses (static)	array					x + 0x40		
10.	repeater (static)	enum				1	x + 0x48		
11.	repeater_status (dyn.)	boolean					x + 0x50		
12.	min_delta_credit (dyn.)	unsigned					x + 0x58		
13.	initiator_mac_address (dyn.)	long-unsigned					x + 0x60		
14.	synchronization_locked (dyn.)	boolean					x + 0x68		
Specific methods		m/o							

Attribute description

logical_name Identifies the “S-FSK Phy&MAC setup” object instance. See 6.2.22.

initiator_electrical_phase Holds the MIB variable initiator-electrical-phase (variable 18) specified in IEC 61334-4-512:2001, 5.8.

It is written by the client system to indicate the phase to which it is connected.

enum:

- (0) Not defined (default),
- (1) Phase 1,
- (2) Phase 2,
- (3) Phase 3.

NOTE 2 This enumeration is different from that of IEC 61334-4-512:2001.

delta_ electrical_phase	Holds the MIB variable delta-electrical-phase (variable 1) specified in IEC 61334-4-512:2001, 5.2 and IEC 61334-5-1:2001, 3.5.5.3. It indicates the phase difference between the client's connecting phase and the server's connecting phase. The following values are predefined: enum: (0) Not defined: the server is temporarily not able to determine the phase difference, (1) The server system is connected to the same phase as the client system. The phase difference between the server's connecting phase and the client's connecting phase is equal to: (2) 60 degrees, (3) 120 degrees, (4) 180 degrees, (5) -120 degrees, (6) -60 degrees.
max_ receiving_gain	Holds the MIB variable max-receiving-gain (variable 2) specified in IEC 61334-4-512:2001, 5.2 and in IEC 61334-5-1:2001, 3.5.5.3. Corresponds to the maximum allowed gain bound to be used by the server system in the receiving mode. The default unit is dB. NOTE 3 In IEC 61334-4-512:2001, no units are specified. The possible values of the gain may depend on the hardware. Therefore, after writing a value to this attribute, the value should be read back to know the actual value.
max_ transmitting_gain	Holds the value of the max-transmitting-gain. Corresponds to the maximum attenuation bound to be used by the server system in the transmitting mode. The default unit is dB. The possible values of the gain may depend on the hardware. Therefore, after writing a value to this attribute, the value should be read back to know the actual value.
search_ initiator_gain	This attribute is used in the intelligent search initiator process. If the value of the <i>max_receiving_gain</i> is below the value of this attribute, a fast synchronization process is possible.
frequencies	Contains frequencies required for S-FSK modulation. <pre>frequencies_type ::= structure { mark_frequency: double-long-unsigned, space_frequency: double-long-unsigned }</pre> The default unit is Hz.
mac_address	Holds the MIB variable mac-address (variable 3) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. NOTE 4 MAC addresses are expressed on 12 bits. Contains the value of the address of the physical attachment (MAC address) associated to the local system. In the unconfigured state, the MAC address is "NEW-address".

mac_address cont'd	This attribute is locally written by the CIASE when the system is registered (with a Register service). The value is used in each outgoing or incoming frame. The default value is "NEW-address". This attribute is set to NEW: – by the MAC sub-layer, once the time-out-not-addressed delay is exceeded; – when a client system "resets" the server system. See the "S-FSK Active initiator" IC in 5.8.4. When this attribute is set to NEW: – the system loses its synchronization (function of the MAC-sublayer); – the <i>mac_group_address</i> attribute is reset (array of 0 elements); – the system automatically releases all AAs which can be released. NOTE 5 The second item is not present in IEC 61334-4-512:2001. The predefined MAC addresses are shown in Table 15.
mac_group_addresses	Holds the MIB variable <i>mac-group-address</i> (variable 4) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. Contains a set of MAC group addresses used for broadcast purposes. array mac-address mac-address::= long-unsigned The ALL-configured-address, ALL-physical-address and NO-BODY addresses are not included in this list. These ones are internal predefined values. This attribute shall be written by the initiator using DLMS services to declare specific MAC group addresses on a server system. This attribute is locally read by the MAC sublayer when checking the destination address field of a MAC frame not recognized as an individual address or as one of the three predefined values (ALL-configured-address, ALL-physical-address and NO-BODY).
repeater	Holds the MIB variable <i>repeater</i> (variable 5) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. It specifies whether the server system effectively repeats all frames or not. enum: (0) never repeater, (1) always repeater, (2) dynamic repeater If the repeater variable is equal to 0, the server system should never repeat the frames. If it is set to 1, the server system is a repeater: it has to repeat all frames received without error and with a current credit greater than zero. If it is set to 2, then the repeater status can be dynamically changed by the server itself. NOTE 6 The value 2 value is not specified in IEC 61334-4-512:2001. This attribute is internally read by the MAC sub-layer each time a frame is received. The default value is 2, and the server system is a repeater (<i>repeater_status</i> = TRUE).

repeater_status	Holds the current repeater status of the device. boolean: FALSE = no repeater, TRUE = repeater
min_delta_credit	Holds the MIB variable min-delta-credit (variable 9) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. NOTE 7 Only the three least significant bits are used. The Delta Credit (DC) is the subtraction of the Initial Credit (IC) and Current Credit (CC) fields of a correct received MAC frame. The delta-credit minimum value of a correct received MAC frame, directed to a server system, is held by this variable. The default value is set to the maximal initial credit (see IEC 61334-5-1:2001, 4.2.3.1 for further explanations on the credit and the value of MAX_INITIAL_CREDIT). A client system can reinitialise this variable by setting its value to the maximal initial credit.
initiator_mac_address	Holds the MIB variable initiator-mac-address specified in IEC 61334-5-1:2001, 4.3.7.6. Its value is either the MAC address of the active-initiator or the NO-BODY address, depending on the value of the synchronization_locked attribute (see below). See also IEC 61334-5-1:2001, 3.5.3, 4.1.6.3 and 4.1.7.2. If the value NO-BODY is written then the server mac_address (see the <i>mac_address</i> attribute) has to be set to NEW.
synchronization_locked	Holds the MIB variable synchronization-locked (variable 10) specified in IEC 61334-4-512:2001, 5.3. Controls the synchronization locked / unlocked state. See IEC 61334-5-1:2001 for more details. If the value of this attribute is equal to TRUE, the system is in the synchronization-locked state. In this state, the initiator-mac-address is always equal to the MAC address field of the active-initiator MIB object. See attribute 2 of the S-FSK Active initiator IC in 5.8.4. If the value of this attribute is equal to FALSE, the system is in the synchronization-unlocked state. In this state, the <i>initiator_mac_address</i> attribute is always set to the NO-BODY value: a value change in the MAC address field of the active-initiator MIB object does not affect the content of the <i>initiator_mac_address</i> attribute which remains at the NO-BODY value. The default value of this variable shall be specified in the implementation specifications. NOTE 8 In the synchronization-unlocked state, the server synchronizes on any valid frame. In the synchronization locked state, the server only synchronizes on frames issued or directed to the client system the MAC address of which is equal to the value of the <i>initiator_mac_address</i> attribute.

7.16 S-FSK IEC 61334-4-32 LLC setup (class_id = 55, version = 0)

An instance of the “S-FSK IEC 61334-4-32 LLC setup” IC holds parameters necessary to set up and manage the LLC layer as specified in IEC 61334-4-32:1996.

S-FSK IEC 61334-4-32 LLC setup		0...n	class_id = 55, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	max_frame_length (static)	unsigned	26	242	134	x + 0x08
3.	reply_status_list (dyn.)	array				x + 0x10
Specific methods		m/o				

Attribute description

logical_name	Identifies the “S-FSK IEC 61334-4-32 LLC setup” object instance. See 6.2.22.
max_frame_length	Holds the length of the LLC frame in bytes. See IEC 61334-4-32:1996, 5.1.4. In the case of the S-FSK profile, as specified in IEC 61334-5-1:2001 4.2.2, the maximum value is 242, but lower values may be chosen due to performance considerations.
reply_status_list	Holds the MIB variable <i>reply_status_list</i> (variable 11) specified in IEC 61334-4-512:2001, 5.4. Lists the L-SAPs that have a not empty RDR (Reply Data on Request) buffer, which has not already been read. The length of a waiting L-SDU is specified in number of sub frames (different from zero). The variable is locally generated by the LLC sub layer. array reply_status reply_status::= structure <pre> { L-SAP-selector: unsigned, length-of-waiting-L-SDU: unsigned } </pre> length-of-waiting-LSDU in the case of the S-FSK profile is in number of sub-frames, valid values are 1 to 7.

7.17 M-Bus client (class_id = 72, version = 0)

Instances of this IC allow setting up M-Bus slave devices and to exchange data with them. Each M-Bus client object controls one M-Bus slave device. For details on the M-Bus dedicated application layer, see EN 13757-3:2004.

The M-Bus client device may have one or more physical M-Bus interfaces, which can be configured using instances of the M-Bus master port setup IC, see 5.6.4.

An M-Bus slave device is identified with its Primary Address, Identification Number, Manufacturer ID, etc., as defined in EN 13757-3:2004 Clause 5, *Variable Data respond*. These parameters are carried by the respective attributes of the M-Bus client IC, see 7.14.

Values to be captured from an M-Bus slave device are identified by the *capture_definition* attribute, containing a list of data identifiers (DIB, VIB) for the M-Bus slave device. The data are captured periodically or on an appropriate trigger. Each data element is stored in an M-Bus value object, of IC “Extended register”. M-Bus value objects may be captured in M-Bus Profile generic objects, eventually along with other, not M-Bus specific objects.

Using the methods of M-Bus client objects, M-Bus slave devices can be installed and de-installed. It is also possible to send data to M-Bus slave devices and to perform operations like resetting alarms, setting the clock, transferring an encryption key, etc.

NOTE 1 In Edition 10, the mapping of attributes to short name has been corrected, without a change in the version.

M-Bus client		0...n	class_id = 72, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	mbus_port_reference (static)	octet-string				x + 0x08
3.	capture_definition (static)	array				x + 0x10
4.	capture_period (static)	double-long-unsigned				x + 0x18
5.	primary_address (dyn.)	unsigned				x + 0x20
6.	identification_number (dyn.)	double-long-unsigned				x + 0x28
7.	manufacturer_id (dyn.)	long-unsigned				x + 0x30
8.	version (dyn.)	unsigned				x + 0x38
9.	device_type (dyn.)	unsigned				x + 0x40
10.	access_number (dyn.)	unsigned				x + 0x48
11.	status (dyn.)	unsigned				x + 0x50
12.	alarm (dyn.)	unsigned				x + 0x58
Specific methods		m/o				
1.	slave_install (data)	o				x + 0x60
2.	slave_deinstall (data)	o				x + 0x68
3.	capture (data)	o				x + 0x70
4.	reset_alarm (data)	o				x + 0x78
5.	synchronize_clock (data)	o				x + 0x80
6.	data_send (data)	o				x + 0x88
7.	set_encryption_key (data)	o				x + 0x90
8.	transfer_key (data)	o				x + 0x98

Attribute description

logical_name Identifies the “M-Bus client” object instance. See 6.2.20.

mbus_port_reference Provides reference to an “M-Bus master port setup” object, used to configure an M-Bus port, each interface allowing to exchange data with one or more M-Bus slave devices.

capture_definition Provides the capture_definition for M-Bus slave devices.

array capture_definition_element

capture_definition_element ::= structure

```
{
    data_information_block:      octet-string,
    value_information_block:     octet-string
}
```

NOTE 2 The elements data_information_block and value_information_block correspond to Data Information Block (DIB) and Value Information Block (VIB) described in 6.2 and Clause 7 respectively.

capture_period >= 1: Automatic capturing assumed. Specifies the capture period in seconds.

0: No automatic capturing: capturing is triggered externally or capture events occur asynchronously.

primary_address	Carries the primary address of the M-Bus slave device, in the range 0...250. If the slave device is already configured and thus, its primary address is different from 0, then this value shall be written to the <i>primary_address</i> attribute. From this moment, the data exchange with the M-Bus slave device is possible. Otherwise, the <i>slave_install</i> method shall be used; see below. NOTE 3 The <i>primary_address</i> attribute cannot be used to store a desired primary address for an unconfigured slave device. If the primary address attribute is set, this means that the M-Bus client can immediately operate with this primary address, which is not the case with an unconfigured slave device.
identification_number	Carries the Identification Number element of the data header as specified in EN 13757-3:2004, 5.4. It is either a fixed fabrication number or a number changeable by the customer, coded with 8 BCD packed digits (4 Byte), and which thus runs from 00 000 000 to 99 999 999. It can be preset at fabrication time with a unique number, but could be changeable afterwards, especially if in addition a unique and not changeable fabrication number (DIF = 0x0C, VIF = 0x78, see 7.2 is provided).
manufacturer_id	Carries the Manufacturer Identification element of the data header as specified in EN 13757-3:2004, 5.5. It is coded unsigned binary with 2 bytes. This <i>manufacturer_id</i> is calculated from the ASCII code of the IEC 62056-21 manufacturer ID (three uppercase letters), using the formula specified in EN 13757-3:2004, 5.5.
version	Carries the Version element of the data header as specified in EN 13757-3:2004, 5.6. It specifies the generation or version of the meter and depends on the manufacturer. It can be used to make sure, that within each version number the identification # is unique.
device_type	Carries the Device type identification element of the data header as specified in EN 13757-3:2004, 5.7, Table 3.
access_number	Carries the Access Number element of the data header as specified in EN 13757-3:2004, 5.8. It has unsigned binary coding, and it is incremented (modulo 256) by one before or after each RSP-UD from the slave. Since it can also be used to enable private end users to detect an unwanted over-frequently readout of their consumption meters, it should not be resettable by any bus communication.
status	Carries the Status byte element of the data header as specified in EN 13757-3:2004, 5.9, Tables 4 and 5.
alarm	Carries the Alarm state specified in EN 13757-3:2004, Annex D. It is coded with data type D (Boolean, in this case 8 bit). Set bits signal alarm bits or alarm codes. The meaning of these bits is manufacturer specific.

Method description

slave_install (data) Installs a slave device, which is yet unconfigured (its primary address is 0).

This method can be successfully invoked only if the value of the *primary_address* attribute is 0. The following actions are performed:

- the M-Bus address 0 is checked for presence of a new device.
- if no uninstalled M-Bus slave is found, the method invocation fails;
- if the *slave_install* method is invoked with no parameter, then the primary address is assigned automatically. This is done by checking the *primary_address* attribute of all M-Bus client objects in the DLMS/COSEM device and then selecting the first unused number. The *primary_address* attribute is set to this address and it is then transferred to the M-Bus slave device;
- if the *slave_install* method is invoked with a primary address as a parameter, then the *primary_address* attribute is set to this value and it is then transferred to the M-Bus slave device.

data::= unsigned (no data, or a valid primary address)

NOTE 4 Unconfigured slave devices are configured with primary address as specified in EN 13757-3:2004, Annex E.5.

slave_deinstall (data) De-installs the slave device. The main purpose of this service is to uninstall the M-Bus slave device and to prepare the master for the installation of a new device.

The following actions are performed:

- the M-Bus address is set to 0 in the M-Bus slave device;
- the encryption key transferred previously to the M-Bus slave device is destroyed; the default key is not affected.
- the attribute *primary_address* is also set to 0.

NOTE 5 A new M-Bus slave can be installed only, once the value of the *primary_address* attribute is 0.

data::= integer (0)

capture Capture values – as specified by the *capture_definition* attribute – from the M-Bus slave device.

data::= integer (0)

reset_alarm Reset alarm state of the M-Bus slave device.

data::= integer (0)

synchronize_clock Synchronize the clock of the M-Bus slave device with that of the M-Bus client device.

data::= integer (0)

data_send	Send data to the M-Bus slave device. <pre> data ::= array data_definition_element data_definition_element ::= structure { data_information_block: octet-string, value_information_block: octet-string data: CHOICE { -- simple data types null-data [0], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], utf8-string [12], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], float32 [23], float64 [24] } }</pre>
set_encryption_key	Sets encryption key to be used with the M-Bus slave device. Changing the encryption key requires two steps: First, the key is sent to the M-Bus slave, encrypted with the master key, using the <i>transfer_key</i> method. Second, the key is set in the M-Bus master using the <i>set_encryption_key</i> method. <pre>data ::= octet-string (encryption_key)</pre> After the installation of the M-Bus slave, the M-Bus client holds an empty encryption key. With this, encryption of M-Bus telegrams is disabled. Encryption can be disabled by invoking the <i>set_encryption_key</i> method with null- data as a parameter.
transfer_key	Transfers an encryption key to the M-Bus slave. <pre>data ::= octet-string (encrypted_key)</pre> Each M-Bus slave device shall be delivered with a default encryption key. Before encrypted M-Bus telegrams can be used, an operational encryption key has to be sent to the M-Bus slave, by invoking the <i>transfer_key</i> method. The method invocation parameter is the operational encryption key encrypted with the default key of the M-Bus slave device. The M-Bus telegram sent is not encrypted. After the execution, the encryption is enabled and all further telegrams are encrypted. A new encryption key can be set in the M-Bus client by invoking the <i>set_encryption_key</i> method with the new encryption key as a parameter. With further invocations of the <i>transfer_key</i> method, new encryption keys can be sent to the M-Bus slave. The method invocation parameter is the new encryption key encrypted with the default key. The M-Bus telegram is encrypted. When an M-Bus slave is de-installed, the encryption key is destroyed, but the default key is not affected. Encryption remains disabled until a new encryption is transferred.

7.18 G3 NB OFDM PLC MAC layer counters (class_id = 90, version = 0)

An instance of the “G3 NB OFDM PLC MAC layer counters” IC stores counters related to the MAC layer exchanges. The objective of these counters is to provide statistical information for management purposes.

The attributes of instances of this IC shall be read only. They can be reset using the reset method.

G3 NB OFDM PLC MAC layer counters		0...n	class_id = 90, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. mac_Tx_data_packet_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x08
3. mac_Rx_data_packet_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x10
4. mac_Tx_cmd_packet_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x18
5. mac_Rx_cmd_packet_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x20
6. mac_CSMA_fail_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x28
7. mac_CSMA_no_ACK_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x30
8. mac_bad_CRC_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x38
9. mac_broadcast_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x40
10. mac_multicast_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x48
Specific methods		m/o				
1. reset (data)		o				
						x + 0x50

Attribute description

logical_name	Identifies the “G3 NB OFDM PLC MAC layer counters” object instance. See 6.2.25.
mac_Tx_data_packet_count	PIB attribute 0x02000101: Statistic counter of successfully transmitted data packets (MSDUs).
mac_Rx_data_packet_count	PIB attribute 0x02000102: Statistic counter of successfully received data packets (MSDUs).
mac_Tx_cmd_packet_count	PIB attribute 0x02000201: Statistic counter of successfully transmitted command packets.
mac_Rx_cmd_packet_count	PIB attribute 0x02000202: Statistic counter of successfully received command packets.
mac_CSMA_fail_count	PIB attribute 0x02000103: Counts the number of times when CSMA backoffs exceed macMaxCSMABackoffs.
mac_CSMA_no_ACK_count	PIB attribute 0x02000104: Counts the number of times when an ACK is not received while transmitting an unicast data frame (The loss of ACK is attributed to collisions).
mac_bad_CRC_count	PIB attribute 0x02000108: Statistic counter of the number of frames received with bad CRC.
mac_broadcast_count	PIB attribute 0x02000106: Statistic counter of the number of broadcast frames sent.

mac_multicast_count PIB attribute 0x02000107: Statistic counter of the number of multicast frames sent.

NOTE When a counter reaches the maximum value (0xFFFFFFFF), it's automatically rolled-over.

Method description

reset (data) This method forces a reset of the object. By invoking this method, the value of all counters is set to 0.

data::= integer (0)

7.19 G3 NB OFDM PLC MAC setup (class_id = 91, version = 0)

An instance of the “G3 NB OFDM PLC MAC setup” IC holds the necessary parameters to set up and manage the G3 NB OFDM PLC IEEE 802.15.4:2006 MAC sub-layer.

These attributes influence the functional behaviour of an implementation. Implementations may allow changes to the attributes during normal running, i.e. even after the device start-up sequence has been executed.

G3 NB OFDM PLC MAC setup		0...n				class_id = 91, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name			
1.	logical_name (static)	octet-string				X			
2.	mac_short_address (dyn.)	long-unsigned	0x0000	0xFFFF	0xFFFF	x + 0x08			
3.	mac_coord_short_address (dyn.)	long-unsigned	0x0000	0xFFFF	0x0000	x + 0x10			
4.	mac_PAN_id (dyn.)	long-unsigned	0x0000	0xFFFF	0xFFFF	x + 0x18			
5.	mac_max_orphan_timer (static)	double-long-unsigned	0	4 294 967 295	0	x + 0x20			
6.	mac_security_enabled (static)	boolean			TRUE	x + 0x28			
7.	mac_freq_notching (static)	boolean			FALSE	x + 0x30			
8.	mac_TMR_TTL (static)	double-long-unsigned	0	262 143	120	x + 0x38			
9.	mac_max_frame_retries (static)	unsigned	0	10	5	x + 0x40			
10.	mac_neighbour_table_entry_TTL (static)	double-long-unsigned	0	262 143	15 300	x + 0x48			
11.	mac_neighbour_table (dyn.)	array							
Specific methods		m/o							
1.	mac_get_neighbour_table_entry (data)	o					x + 0x60		

Attribute description	
logical_name	Identifies the “G3 NB OFDM PLC MAC setup” object instance. See 6.2.25.
mac_short_address	PIB attribute 0x01000112: Device short address. The 16-bit address the device is using to communicate on the PAN. Its value shall be equal to 0xFFFF when the device does not have a short address. An associated device necessarily has a short address, so that a device cannot be in the state where it is associated but does not have a short address.
mac_coord_short_address	PIB attribute 0x01000107: Coordinator short address. NOTE 1 The short address assigned to the coordinator through which the device is associated.
mac_PAN_id	PIB attribute 0x0100010F: PAN ID. NOTE 2 The 16-bit identifier of the PAN on which the device is operating. A value equal to 0xFFFF indicates that the device is not associated.
mac_max_orphan_timer	PIB attribute 0x02000109: The maximum number of seconds without communication with a particular device after which it is declared as an orphan.
mac_security_enabled	PIB attribute 0x01000111: Security enabled. boolean: TRUE: security enabled, FALSE: security disabled
mac_freq_notching	PIB attribute 0x0000006D: S-FSK 63 and 74 kHz frequency notching. boolean: TRUE: notching enabled, FALSE: notching disabled Default value is FALSE (disabled).
mac_TMR_TTL	PIB attribute 0x02000113: Maximum valid time of tone map parameters in the neighbour table in seconds.
mac_max_frame_retries	PIB attribute 0x0100010D: Maximum number of retransmissions.
mac_neighbour_table_entry_TTL	PIB attribute 0x02000114: Maximum valid time for an entry in the neighbour table in seconds.

mac_neighbour_table PIB attribute 0x0000006B: See ITU-T G.9903 Amd. 1:2013, 9.3.7.2 for CENELEC and FCC bands.

The neighbour table contains information about all the devices within the POS of the device. One element of the table represents one PLC direct neighbour of the device.

array mac_neighbour_table_element

mac_neighbour_table_element::= structure

```
{
    mac_short_address:          long-unsigned,
    payload_modulation_scheme:  boolean,
    tone_map:                   bit-string,
    modulation:                  enum,
    tx_gain:                     integer,
    tx_res:                      boolean,
    tx_coeff:                    array,
    lqi:                         unsigned,
    TMR_valid_time:              double-long-unsigned,
    neighbour_valid_time:        double-long-unsigned
}
```

NOTE 3 This table is actualized each time any frame is received from a neighbour device, and each time a Tone Map Response is received.

short_address	The MAC Short Address of the node which this entry refers to.

payload_modulation_scheme	0: Differential, 1: Coherent

tone_map	The Tone Map parameter defines which frequency sub-band can be used for communication with the device. A bit set to 1 means that the frequency sub-band can be used, and a bit set to 0 means that frequency sub-band shall not be used.

modulation	The modulation type to use for communicating with the device. enum: (0) Robust Mode (1) DBPSK (2) DQPSK (3) D8PSK (4) 16-QAM

NOTE 4 The 16-QAM modulation is optional and only applicable for FCC band.

tx_gain	Defines the Tx Gain to use to transmit frames to that device.

tx_res	Defines the Tx Gain resolution corresponding to one gain step. 0: 6 dB 1: 3 dB

**mac_neighbour_
table**

(continued)

tx_coeff	A parameter that specifies transmitter gain for each group of tones represented by one valid bit of the tone map. The receiver measures the frequency-dependent attenuation of the channel and may request the transmitter to compensate for this attenuation by increasing the transmit power on sections of the spectrum that are experiencing attenuation in order to equalize the received signal. Each group of tones is mapped to a 4-bit value for CENELEC-A or a 2-bit value for FCC where a "0" in the most significant bit indicates a positive gain value, hence an increase in the transmitter gain scaled by TXRES is requested for that section and a "1" indicates a negative gain value, hence a decrease in the transmitter gain scaled by TXRES is requested for that section. Implementing this feature is optional and it is intended for frequency selective channels. If this feature is not implemented, the value zero shall be used. NOTE 5 One group of tones gathers 6 consecutive tones (or carriers) for CENELEC bands, and 3 consecutive tones for FCC band.
lqi	Link quality indicator NOTE 6 LQI value measured during reception of the PPDU from the neighbour. The LQI measurement is a characterization of the strength and/or quality of a received packet.
TMR_valid_time	Remaining time in seconds until which the tone map response parameters in the neighbour table are considered valid. – When the entry is created, this value shall be set to the default value 0. – When it reaches 0, a tone map request may be issued if data is sent to this device. Upon successful reception of a tone map response, this value is set to macTMRRTTL.
neighbour_valid_time	Remaining time in seconds until which this entry in the neighbour table is considered valid. Every time an entry is created or a frame (data or ACK) is received from this neighbour, it is set to macNeighbourTableEntryTTL. When it reaches zero, this entry is no longer valid in the table and may be removed.

Method description

mac_get_neighbour_table_entry (data)

This method is used to retrieve the mac neighbour table for one MAC short address. It may be used to perform topology monitoring by the client.

The method invocation parameter contains a mac_short_address.

data::= long-unsigned

The response parameter includes the neighbour table for this mac_short_address.

data::= array mac_neighbour_table_element

where mac_neighbour_table-element is as defined in the *mac_neighbour_table* attribute of the present IC.

7.20 G3 NB OFDM PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 0)

An instance of the “G3 NB OFDM PLC 6LoWPAN adaptation layer setup” IC holds the necessary parameters to set up and manage the G3 NB OFDM PLC 6LoWPAN Adaptation layer.

These attributes influence the functional behaviour of an implementation. Implementations may allow changes to their values during normal running, i.e. even after the device start-up sequence has been executed.

G3 NB OFDM PLC 6LoWPAN adaptation layer setup		0...n	class_id = 92, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	adp_max_hops (static)	unsigned	1	14	8	x + 0x10
3.	adp_weak_LQI_value (static)	unsigned	0	255	3 for CENELEC A 5 for FCC	x + 0x18
4.	adp_tone_mask (static)	bit-string			All bits set to 1	x + 0x20
5.	adp_discovery_attempts_wait_time (static)	long-unsigned	1	3 600	60	x + 0x28
6.	adp_routing_configuration (static)	array				x + 0x30
7.	adp_broadcast_log_table_entry_TL (static)	long-unsigned	0	3 600	90	x + 0x38
8.	adp_routing_table (dyn.)	array				x + 0x40
9.	adp_context_information_table (dyn)	array				x + 0x48
10.	adp_blacklist_table (dyn)	array				x + 0x50
11.	adp_broadcast_log_table (dyn)	array				x + 0x58
12.	adp_group_table (dyn)	array				x + 0x60
13.	adp_max_join_wait_time (static)	long-unsigned	0	1 023	20	x + 0x68
14.	adp_path_discovery_time (static)	long-unsigned	0	6553 5	5 000	x + 0x70
15.	adp_use_new_GMK_time (static)	long-unsigned	0	6553 5	3 600	x + 0x78
16.	adp_exp_prec_GMK_time (static)	long-unsigned	0	3 600	3 600	x + 0x80
Specific methods		m/o				

Attribute description

logical_name Identifies the “G3 NB OFDM 6LoWPAN adaptation layer setup” object instance. See 6.2.25

adp_max_hops PIB attribute 0x10: Defines the maximum number of hops to be used by the routing algorithm.

adp_weak_LQI_value PIB attribute 0x1B: The weak link value defines the threshold below which a direct neighbour is not taken into account during the commissioning procedure (compared to the LQI measured).

adp_tone_mask PIB attribute 0x0F: Defines the Tone Mask to use during symbol formation. The length of the bit-string is 70 bits.

adp_discovery_attempts_wait_time PIB attribute 0x06: Allows programming the maximum wait time between invocations of two consecutive network discovery primitives (in seconds).

adp_routing_configuration

The routing configuration element specifies all parameters linked to the routing mechanism described in ITU-T G.9903 Amd. 1:2013. The elements are specified in 9.4.1.2 of that Recommendation.

NOTE 1 The Link cost calculation is provided in ITU-T G.9903 Amd. 1:2013 Annex B.

array routing_configuration

routing_configuration ::= structure

```
{
    adp_net_traversal_time:      long-unsigned,
    adp_routing_table_entry_TTL: double-long-unsigned,
    adp_Kr:                     unsigned,
    adp_Km:                     unsigned,
    adp_Kc:                     unsigned,
    adp_Kq:                     unsigned,
    adp_Kh:                     unsigned,
    adp_Krt:                    unsigned,
    adp_RREQ_retries:           unsigned,
    adp_RREQ_RERR_wait:        long-unsigned,
    adp_blacklist_table_entry_TTL: long-unsigned
}
```

adp_net_traversal_time	PIB attribute 0x12: The Max duration between RREQ and the correspondent RREP (in seconds). Range: 0-65 535 Default value: 20
adp_routing_table_entry_TTL	PIB attribute 0x13: The time to live of a route request table entry (in seconds). Range: 0-262 143 Default value: 90
adp_Kr	PIB attribute 0x14: A weight factor for ROBO to calculate link cost. Range: 0-31 Default value: 0
adp_Km	PIB attribute 0x15: A weight factor for modulation to calculate link cost. Range: 0-31 Default value: 0
adp_Kc	PIB attribute 0x16: A weight factor for number of active tones to calculate link cost. Range: 0-31 Default value: 0
adp_Kq	PIB attribute 0x17: A weight factor for LQI to calculate route cost. Range: 0-31 Default value: 10 for CENELEC A band / 40 for FCC band.
adp_Kh	PIB attribute 0x18: A weight factor for hop to calculate link cost. Range: 0-31 Default value: 4 for CENELEC A band / 2 for FCC band.
adp_Krt	PIB attribute 0x1C: A weight factor for the number of active routes in the routing table to calculate link cost. Range: 0-31 Default value: 0

adp_routing_configuration (continued)	adp_RREQ_retries	PIB attribute 0x19: The number of RREQ re-transmission in case of RREP reception time out. Range: 0-255 Default value: 0
	adp_RREQ_RERR_wait	PIB attribute 0x1A: The number of seconds to wait between two consecutive RREQ – RERR generations. Range: 0-65 535 Default value: 30
	adp_blacklist_table_entry_TTL	PIB attribute 0x26: Time to live of a blacklisted neighbour set entry in minutes. Range: 0-65 535 Default value: 10
adp_broadcast_log_table_entry_TTL	PIB attribute 0x02: Defines the time while an entry in the adpBroadcastLogTable remains active in the table (in seconds).	
adp_routing_table	PIB attribute 0x0C: The routing table contains information about the different routes in which the device is implicated. array routing_table routing_table::= structure <pre> { destination_address: long-unsigned, next_hop_address: long-unsigned, route_cost: long-unsigned, hop_count: unsigned, weak_link_count: unsigned, status: enum, valid_time: unsigned } </pre> NOTE 2 This table is actualized each time a route is built or updated (triggered by data traffic) and each time the TTL timer expires.	
	destination_address	Address of the destination.
	next_hop_address	Address of the next hop on the route towards the destination.
	route_cost	Cumulative link cost along the route towards the destination.
	hop_count	Number of hops of the selected route to the destination. Range: 0-15 NOTE 3 Practically the maximum allowed value is limited by adp_max_hops.
	weak_link_count	Number of weak links to destination. Range : 0-15
	NOTE 4 Practically the maximum allowed value is limited by adp_max_hops.	
	status	Status of the routing table entry. enum : (0) Invalid route, (1) Valid route, (2) Route under construction
	valid_time	Remaining time in seconds until which this entry in the routing table is considered valid.

adp_context_information_table	PIB attribute 0x07: Contains the context information associated to each CID extension field. <pre> array context_information_table context_information_table ::= structure { CID: bit-string, context_length: unsigned, context: octet-string, C: boolean, valid_lifetime: long-unsigned } </pre>				
CID	Corresponds to the 4-bit context information used for source and destination addresses (SCI, DCI). Range: 0x00-0x0F				
context_length	Indicates the length of the carried context (up to 128-bit contexts may be carried). Range: 0-128				
context	Corresponds to the carried context used for compression/decompression purposes. Range: 0x0000:0000:0000:0000:0000:0000:0000:0000 – 0xFFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF				
C	Indicates if the context is valid for use in compression 0: Only decompression is allowed, 1: Compression and decompression are allowed A context may be used for decompression purposes only. Moreover, recommendations made in RFC 6775 should be followed to take into account the propagation of the context to all nodes of the PAN.				
valid_lifetime	Remaining time in minutes during which the context information table is considered valid. It is updated upon reception of the advertised context. Range: 0-65 535				
adp_blacklist_table	PIB attribute 0x25: Contains the list of the blacklisted neighbours. <pre> array blacklisted_neighbour_set blacklisted_neighbour_set ::= structure { blacklisted_neighbour_address: long-unsigned, valid_time: long-unsigned } </pre> <table> <tr> <td data-bbox="459 1691 614 1780">blacklisted_neighbour_address</td><td data-bbox="662 1691 1388 1724">The 16-bit address of the blacklisted neighbour.</td></tr> <tr> <td data-bbox="459 1803 590 1836">valid_time</td><td data-bbox="662 1803 1388 1859">Remaining time in minutes until which this entry in the blacklisted neighbour table is considered valid.</td></tr> </table>	blacklisted_neighbour_address	The 16-bit address of the blacklisted neighbour.	valid_time	Remaining time in minutes until which this entry in the blacklisted neighbour table is considered valid.
blacklisted_neighbour_address	The 16-bit address of the blacklisted neighbour.				
valid_time	Remaining time in minutes until which this entry in the blacklisted neighbour table is considered valid.				

adp_broadcast_log_table	PIB attribute 0x0B: Contains the broadcast log table. NOTE 5 This table provides a list of the broadcast packets recently received by this device. <pre> array broadcast_log_table broadcast_log_table ::= structure { source_address: long-unsigned, sequence_number: unsigned, time_to_live: long-unsigned } </pre> <table> <tr> <td>source_address</td><td>The 16-bit source address of a broadcast packet. This is the address of the broadcast initiator.</td></tr> <tr> <td>sequence_number</td><td>The sequence number contained in the BC0 header.</td></tr> <tr> <td>time_to_live</td><td>The remaining time to live of this entry in the broadcast log table, in seconds.</td></tr> </table>	source_address	The 16-bit source address of a broadcast packet. This is the address of the broadcast initiator.	sequence_number	The sequence number contained in the BC0 header.	time_to_live	The remaining time to live of this entry in the broadcast log table, in seconds.
source_address	The 16-bit source address of a broadcast packet. This is the address of the broadcast initiator.						
sequence_number	The sequence number contained in the BC0 header.						
time_to_live	The remaining time to live of this entry in the broadcast log table, in seconds.						
adp_group_table	PIB attribute 0x0E: Contains the group addresses to which the device belongs. <pre> array group_table group_table ::= structure { group_address: long-unsigned } </pre> <table> <tr> <td>group_address</td><td>Group address to which this node has been subscribed.</td></tr> </table>	group_address	Group address to which this node has been subscribed.				
group_address	Group address to which this node has been subscribed.						
adp_max_join_wait_time	PIB attribute 0x21: Network join timeout in seconds for LBD.						
adp_path_discovery_time	PIB attribute 0x22: Timeout for path discovery in msec.						
adp_use_new_GMK_time	PIB attribute 0x23: The wait time in seconds for a device to use new GMK after rekeying.						
adp_exp_prec_GMK_time	PIB attribute 0x24: The time in seconds to keep PrecGMK after switching to a new GMK.						

Annex A (informative)

Additional information on Auto answer and Auto connect ICs

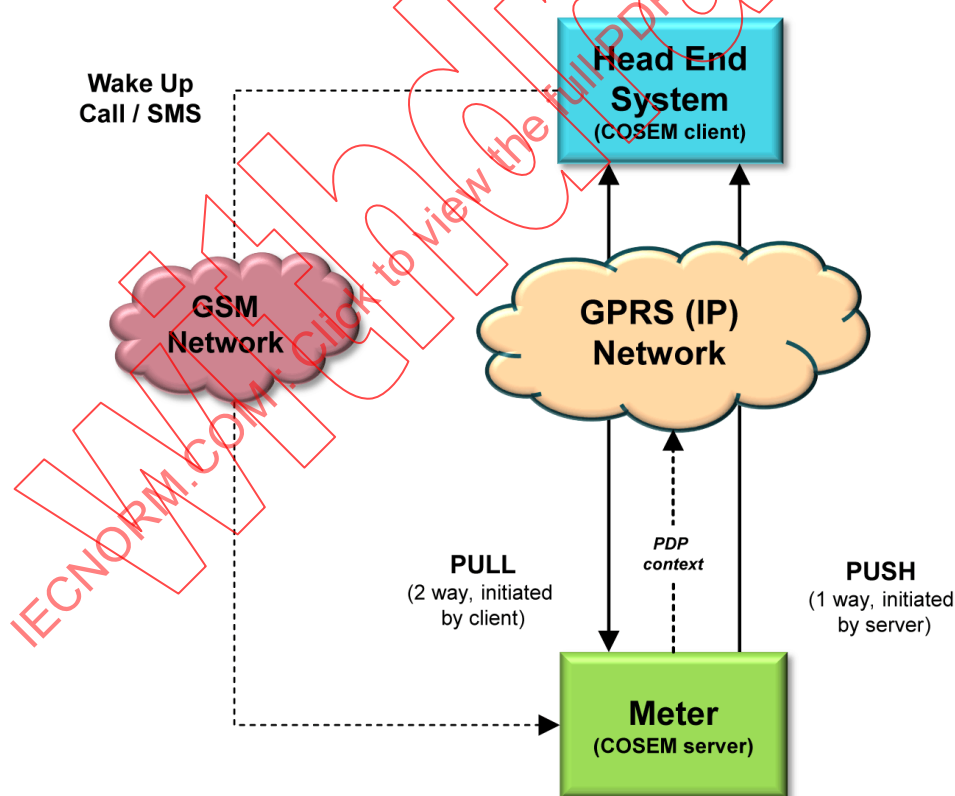
NOTE This information is related to the “Auto answer” (class_id = 28, version = 2, see 5.5.5) and “Auto connect” (class_id = 29, version = 2, see 5.5.6) interface classes.

Since the capabilities (e.g. connection time, number of parallel connections) of communication networks (e.g. GPRS) are limited, devices e.g. meters are not permanently connected to the communication network.

Devices may connect to the network in regular intervals or on special events either to send unsolicited data or just to become accessible.

If a DLMS/COSEM client e.g. a Head End System needs to access a server e.g. a meter that is not connected to the communication network a wake-up request can be sent. This may be a wake-up call or a wake-up message, e.g. an SMS message. After successfully processing the wake-up request the device connects to network.

Figure A.1 below shows an example for a GSM/GPRS communication network. Note that the dashed lines represent the network services, the solid lines refer to possible application layer services.



IEC

Figure A.1 – Network connectivity example for a GSM/GPRS network

The basic network connectivity in the case of a mobile network (GPRS or equivalent service) is modelled by the “Auto connect” IC. Depending on the mode the connection can be 'always on', 'always on in a time window' or 'only on after a wake-up'. If the device is connected to the network it has the PDP context attached and it is accessible from by the HES via its IP address. If necessary the current IP address of the server (meter) can be sent to the client (HES) using the DataNotification xDLMS service.

The wake-up process is modelled using an instance of the “Auto answer” IC which provides additional security (check calling number) compared to today's solution.

Also note that the “Auto answer” class is fully decoupled from the xDLMS application layer services. The main reason is to have a clear separation between the communication layers as well as to avoid creating an unsecured backdoor to execute application layer services with almost no protection. The execution of xDLMS services via SMS should be handled by sending ciphered xDLMS APDUs in a pre-established AA.

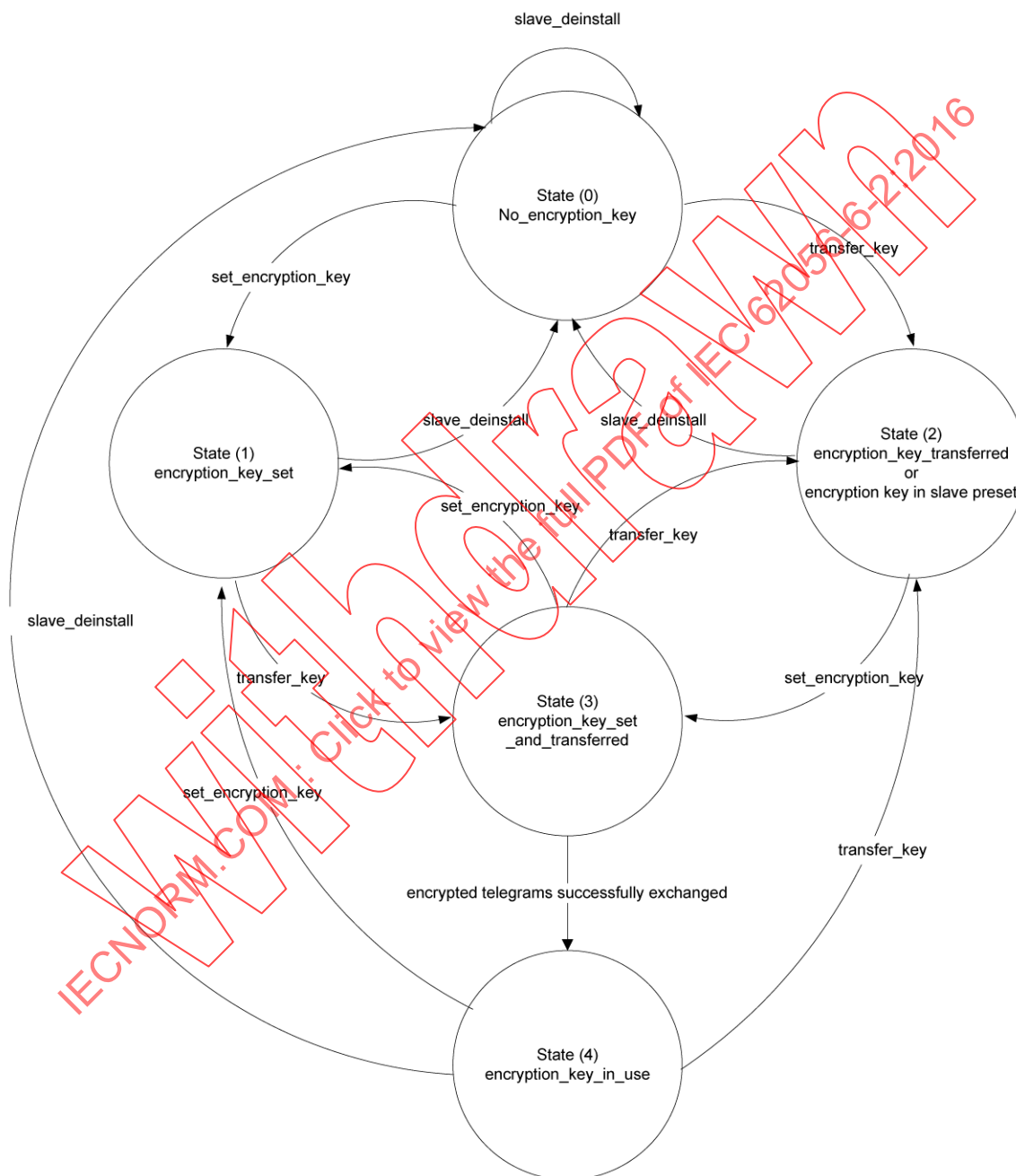
IECNORM.COM: Click to view the full PDF of IEC 62056-6-2:2016

Withdwn

Annex B (informative)

Additional information to M-Bus client (class_id = 72, version 1)

State transitions of the *encryption_key_status* attribute for different use cases are shown in Figure B.1.



IEC

Figure B.1 – Encryption key status diagram

State transitions of the *encryption_key_status* attribute for different use cases are given below. At the time of installation of the slave four cases are possible:

- a) Encryption key is preset in the slave and cannot be changed, see Table B.1;

- b) Encryption key is preset in the slave and new key is set after installation, see Table B.2;
- c) Encryption key is not preset in the slave, but can be set, see Table B.3 and Table B.4;
- d) The slave is used without encryption. In this case, the *encryption_key_status* stays in state (0).

Table B.1 – Encryption key is preset in the slave and cannot be changed

Step	State	Condition for state transition (event or successfully invoked method)
1	(2)	set_encryption_key
2	(3)	encrypted telegrams successfully exchanged
3	(4)	–

Table B.2 – Encryption key is preset in the slave and new key is set after installation

Step	State	Condition for state transition (event or successfully invoked method)
1	(2)	set_encryption_key
2	(3)	transfer_key
3	(2)	set_encryption_key
4	(3)	encrypted telegrams successfully exchanged
5	(4)	–

Table B.3 – Encryption key is not preset in the slave, but can be set, case a)

Step	State	Condition for state transition (event or successfully invoked method)
1a	(0)	set_encryption_key
2a	(1)	transfer_key
3a	(3)	encrypted telegrams successfully exchanged
4a	(4)	–

Table B.4 – Encryption key is not preset in the slave, but can be set, case b)

Step	State	Condition for state transition (event or successfully invoked method)
1b	(0)	transfer_key
2b	(2)	set_encryption_key
3b	(3)	encrypted telegrams successfully exchanged
4b	(4)	–

Annex C (informative)

Additional information on IPv6 setup class (class_id = 48, version = 0)

C.1 General

In most regards, IPv6 is a conservative extension of IPv4. Most transport and application-layer protocols need little or no change to operate over IPv6; exceptions are application protocols that embed internet-layer addresses, such as FTP or NTPv3.

IPv6 specifies a new packet format, designed to minimize packet-header processing. Since the headers of IPv4 packets and IPv6 packets are significantly different, the two protocols are not interoperable.

C.2 IPv6 addressing

The most important feature of IPv6 is a much larger address space than that of IPv4: addresses in IPv6 are 128 bits long, compared to 32-bit addresses in IPv4. Furthermore, compared to IPv4, IPv6 supports multi-addressing on one physical interface (global, unique or link local IPv6 addresses).

IPv6 addresses are typically composed of two logical parts: a 64-bit (sub-)network prefix used for routing, and a 64-bit host part used to identify a host within the network.

The formats allowed for an IPv6 address are shown in Figure C.1 (see <http://www.iana.org/assignments/ipv6-address-space/>). Note that to facilitate the IPv6 address writing, a specific notation defined in RFC 4291 has been specified by IETF (e.g. FF00::/8).

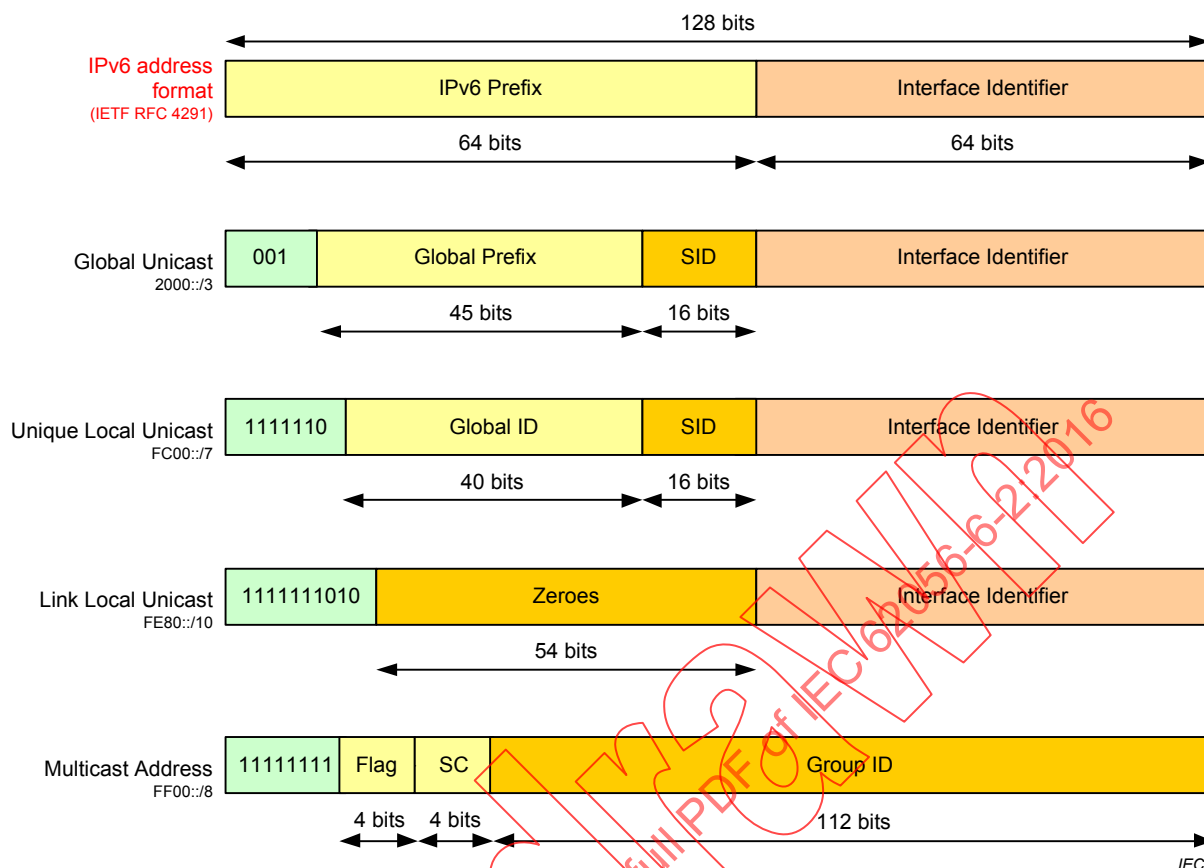


Figure C.1 – IPv6 address formats

Where:

- *Global Unicast* is a routable address in the whole internet network and is composed as follows:
 - Global prefix assigned by IANA (see <http://www.iana.org/assignments/ipv6-unicast-address-assignments/>);
 - Subnet ID (SID) allocated by the network administrator; and
 - Interface Identifier either generated from the interface's MAC address (using modified EUI-64 format), or obtained from a DHCPv6 server, or assigned manually;
- *Unique Local Unicast* is an address only applicable to local network. This type of address is not routable outside the local network. The Global ID and the Subnet ID (SID) are allocated by the network administrator;
- *Link Local Unicast* is a unicast address allowed for a link local (without router). This type of address is not routable outside a local link;
- *Multicast* is an address assigned to different devices of the network. Following the scope (SC) of the address, the multicast group may be either Interface-local, Link-local, Admin-local, Site-local, Organization-local or global. For more information about Flag and SC (scope) parameters, see RFC 4291, 2.7.

It is important to note that there is no broadcast address defined in IPv6.

For more information concerning IPv6 addressing, see RFC 4291.

C.3 IPv6 header format

As defined in RFC 2460, Clause 3, the header of one IPv6 packet is composed as shown in Figure C.2:

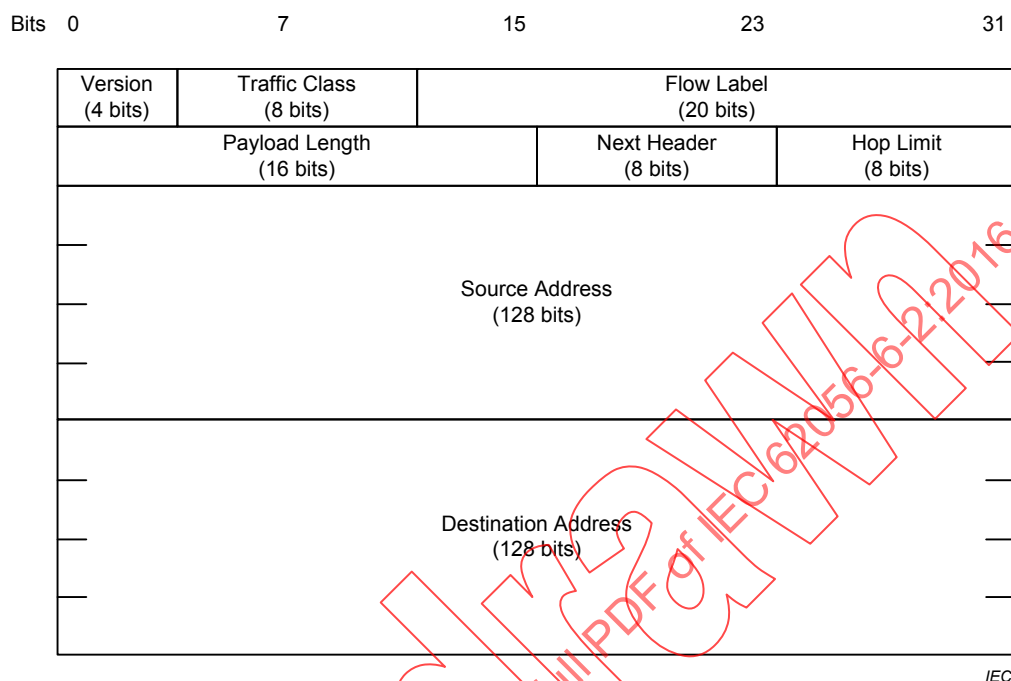


Figure C.2 – IPv6 header format

Where:

- *Version* specifies the version of the protocol. For IPv6, the value is fixed and equals to 6;
- *Traffic class* is used by originating nodes and/or forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets (see RFC 2474, Clause 3). Note that the traffic class value is only a notion of prioritization used to distribute the IPv6 frame and does not secure the transmission (the philosophy of the IP network is to do its best).

Figure C.3 shows the content of the traffic class parameter:

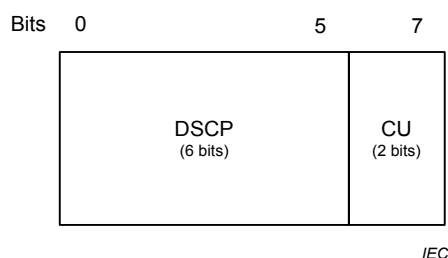


Figure C.3 – Traffic class parameter format

Where:

- DSCP – Differentiated services code point contains the prioritization of the IPv6 packet in the network (see RFC 2474 for details);
- CU – Currently unused;

- *Flow label* may be used by a source to label sequences of packets for which it requests special handling by the IPv6 routers, such as non-default quality of service or "real-time" service. Currently, no full definition of this field is given by IETF and may be seen as reserved for future use;
- *Payload length* indicates the size of the upper layer payload carried by the IPv6 frame;
- *Next header* identifies the type of header immediately following the IPv6 header of the current frame. It may indicate an upper applicative header (ICMP, UDP, TCP...) or extensions;
- *Hop limit* defines the time of life of the IPv6 packet in terms of number of hop. The usage is similar to the one defined for IPv4 (decremented by 1 by each node that forwards the packet, the packet is discarded if Hop Limit is decremented to zero);
- *Source and destination addresses* indicate the originator of the IPv6 packet and the intended recipient.

Table C.1 summarizes the IPv6 headers managed / not managed by the IPv6 setup class.

Table C.1 – IPv6 header vs. IPv6 IC

Parameters	IPv6 setup IC
Version	Not managed
Traffic class	Managed
Flow label	Not managed
Payload length	Not managed
Next header	Not managed
Hop limit	Not managed
Source and destination addresses	Not managed

C.4 IPv6 header extensions

C.4.1 Overview

Contrary to IPv4, IPv6 provides an extensible header by adding optional elements one by one. Currently, the following list of optional headers is defined by RFC 2460, see Table C.2.

Table C.2 summarizes the options managed / not managed by the IPv6 setup objects.

Table C.2 – Optional IPv6 header extensions vs. IPv6 IC

Extensions	IPv6 setup IC	See clause
Hop-by-Hop options	Not managed	C.4.2
Destination options	Not managed	C.4.3
Routing options	Not managed	C.4.4
Fragment options	Not managed	C.4.5
Security options (Authentication and Encapsulating Security Payload)	Not managed	C.4.6
NOTE The options are listed in the order as they appear in the packet.		

C.4.2 Hop-by-Hop options

The Hop-by-Hop options field shall be examined by all devices on the path. Four elements currently defined may compose this header (see RFC 2460, 4.3):

- A padding form Pad1 which introduces one byte of padding (see RFC 2460, 4.2);
- A padding form PadN which introduces more than 2 bytes of padding (see RFC 2460, 4.2);
- A Jumbogram field to indicate the length of the IPv6 datagram in case of jumbo payload (higher than the maximum size of length field of IPv6 header) (see RFC 2460 and RFC 2675); and
- A router alert: The option indicates that the contents of the datagram may be interesting to the router (see RFC 2711).

These optional elements are directly managed by the IPv6 protocol stack. Therefore, they are not managed by the COSEM IPv6 setup class.

C.4.3 Destination options

The Destination options field shall be examined only by the target device of the IP datagram. Four elements are currently defined by IETF (see RFC 2460, 4.6):

- A padding form Pad1 which introduces one byte of padding (see RFC 2460, 4.2);
- A padding form PadN which introduces more than 2 bytes of padding (see RFC 2460, 4.2);
- Mobility parameters (linked to new IP wireless networks – UMTS, LTE...) (see RFC 3775); and
- Tunnelling options which permit to communicate between two IPv6 networks separated by an IPv4 network (6To4 mechanism) (see RFC 2473).

NOTE The Mobility parameters and the Tunnelling options have been defined separately from RFC 2460 and are not mentioned in this standard. See appropriate RFCs for more details.

These optional elements are directly managed by the IPv6 protocol stack. These optional fields are not managed by the COSEM IPv6 setup class.

C.4.4 Routing options

Routing options element is used to specify some routing parameters (see RFC 2460, 4.4). Currently, only one type is defined by IETF (type 2), which is used in case of mobility IPv6.

This notion is out of scope of the COSEM IPv6 setup class.

For information, due to an important security issue (see RFC 5095), the type 0 previously defined has been depreciated and is forbidden. Furthermore, the type 1 has been temporarily defined for experimental Nimrod and is obsolete since 1996.

C.4.5 Fragment options

Fragment options field is used in the case of fragmentation of the applicative payload if its size is higher than the MTU of the network (see RFC 2460, 4.5).

This element is directly managed by the IPv6 protocol stack. These optional fields are not managed by the COSEM IPv6 setup class.

C.4.6 Security options

Contrary to IPv4, IPv6 protocol layer includes natively the IPSec protocol. These extensions are fully defined in the RFC 4302 and RFC 4303.

The security options are composed of two extensible headers:

- Authentication Header (AH): Contains information used to verify the authenticity of most parts of the packet (see RFC 4302, Clause 2);
- Encapsulating Security Payload (ESP): Carries encrypted data for secure communication (see RFC 4303, Clause 2).

Due to the complexity of the IPsec protocol, the configuration of IPsec is out of scope of this standard and shall be treated separately.

IECNORM.COM: Click to view the full PDF of IEC 62056-6-2:2016

Annex D (informative)

Overview of the narrow-band OFDM PLC technology for PRIME networks

For the specification of the PRIME narrow-band OFDM PLC setup classes, see 5.10.

NOTE This technology is supported by the PRIME Alliance, <http://www.prime-alliance.org>.

ITU-T G.9904:2012 specifies a physical layer, a medium access control layer and convergence layers for cost-effective narrowband (<200 kbps) data transmission over electrical power lines, intended for use in smart metering and smart grid applications. It is based on Orthogonal Frequency Division Multiplexing (OFDM).

The specification currently describes the following:

- a low-cost PHY capable of achieving rates of encoded 128 kbps;
- a Master-Slave MAC optimised for the power line environment;
- a convergence layer for the LLC layer specified in IEC 61334-4-32;
- a convergence layer for IPv4;
- a convergence layer for IPv6;

CLC/TS 52056-8-4:2015 specifies the DLMS/COSEM narrowband OFDM PLC profile for PRIME networks.

Annex E (informative)

Overview of the narrow-band OFDM PLC technology for G3-PLC networks

For the specification of the G3 narrow-band OFDM PLC setup classes, see 5.11.

NOTE This specification is supported by the G3-PLC Alliance, <http://www.G3-PLC.com>.

ITU-T G.9903:2014 specifies the physical, MAC and 6LoWPAN Adaptation layers of the G3-PLC technology while ITU-T G.9901:2014 deals with frequency bandplan allocation and associated transmission level limitations.

Power line communication has been used for many decades, but a variety of new services and applications require more reliability and higher data rates. However, the power line channel is very hostile. Channel characteristics and parameters vary with frequency, location, time and the type of equipment connected to it. The lower frequency regions from 10 kHz to 200 kHz are especially susceptible to interference. Furthermore, the power line is a very frequency selective channel. Besides background noise, it is subject to impulsive noise often occurring at 50 Hz/60 Hz and group delays up to several hundred microseconds.

G3-PLC uses advanced modulation and channel coding techniques, which enables efficient use of the limited bandwidth of the CENELEC bands and facilitates communication over the power line channel. This combination enables a very robust communication in the presence of narrow-band interference, impulsive noise, and frequency selective attenuation. The specification addresses the following main objectives:

- provide robust communication on extremely harsh power line channels;
- provide a minimum of 20 kbps effective data rate in the normal mode of operation;
- ability to notch selected frequencies, to allow the cohabitation with other Narrow-band PLC communication technologies (e.g. IEC 61334-5-1:2001 S-FSK) or to be compliant with specific regulatory requirements;
- dynamic tone adaptation capability to select frequencies on the channel that do not have major interference, thereby ensuring a robust communication;
- access control, authentication, confidentiality and integrity to ensure high level of security.

To this end, the G3-PLC protocol stack aggregates several layers and sub-layers that form the G3-PLC profile:

- a robust high-performance PHY layer based on OFDM and adapted to the narrow-band PLC environment;
- a MAC layer of the IEEE 802.15.4:2006 type (extended), well suited to low data rates;
- IPv6, the new generation of IP (Internet Protocol), which widely opens the range of potential applications and services; and
- to allow good IPv6 and MAC interoperability, an Adaptation sublayer taken from the Internet world (IETF) and called 6LoWPAN (RFC 4944 extended and RFC 6282). The adaptation sub layer also embeds the LOADng routing algorithm to allow multi-hop mesh connectivity.

For more information about the extensions of IEEE 802.15.4:2006, RFC 4944 and RFC 6282 (AKA 6LoWPAN) standards, and LOADng, see ITU-T G.9903:2014.

CLC/TS 52056-8-5:2015 specifies the DLMS/COSEM narrowband OFDM PLC profile for G3-PLC networks.

Annex F (informative)

Significant technical changes with respect to IEC 62056-6-2:2013

This second edition of IEC 62056-6-2 includes the following significant technical changes with respect to the first edition of IEC 62056-6-2:2013:

- an Introduction explaining the relationship to the DLMS UA Blue Book Editions has been added;
- Clause 3, Terms, definitions and abbreviations: new definitions and abbreviations related to the new interface classes added;
- 4.6.1, Date and time formats: precisions to the elements deviation and clock_status added;
- 5.1, Overview: Updated with the new interface classes. A Table listing all interface classes has been added;
- 5.2.2, Table 4 – Enumerated values for physical units: new units added;
- 5.2.9, Status mapping (class_id = 63, version = 0): specification of the mapping_table attribute amended (editorial);
- 5.3.1: An Overview clause of interface classes for access control and management has been added;
- 5.3.2 Client user identification mechanism added;
- 5.3.3, Association SN (class_id = 12, version = 3): New version supporting the client user identification mechanism added;
- 5.3.4, Association LN (class_id = 15, version = 2): New version supporting the client user identification mechanism added. Specification of the *application_context_name* and *authentication_mechanism_name* attributes amended (editorial);
- 5.3.6, Image transfer: Precisions to the text added;
- 5.3.8, Push interface classes and objects: New interface class supporting push operation added;
- 5.4.1, Clock (class_id = 8, version = 0): Precisions to the text added;
- 5.4.10 Parameter monitor (class_id = 65, version = 0): New interface class added;
- 5.5.3, IEC twisted pair (1) setup (class_id = 24, version = 1): New version added;
- 5.5.5, Auto answer (class_id = 28, version = 2): New version added;
- 5.5.6, Auto connect (class_id = 29, version = 2): New version added;
- 5.5.8, GSM diagnostic (class_id = 47, version = 0): New interface class added;
- 5.6.2, M-Bus client (class_id = 72, version = 1): New version added;
- 5.7.2, IPv4 setup (class_id = 42, version = 0): Precisions to the text added;
- 5.7.3, IPv6 setup (class_id = 48, version = 0): New interface class added;
- 5.7.5, PPP setup (class_id = 44, version = 0): Precisions to the text added;
- 5.10, Interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for PRIME networks added;
- 5.11, Interface classes for setting up and managing the DLMS/COSEM narrowband OFDM PLC profile for G3-PLC networks added;
- 5.12 ZigBee® setup classes added;
- 6.2, Abstract COSEM objects: new OBIS codes and data types added;
- 6.2.4, Other abstract general purpose OBIS codes added;

- 6.2.5, Clock objects (class_id = 8): OBIS code for UNIX clock added;
- 6.2.7, Script table objects (class_id = 9): OBIS code for Push script table added;
- 6.2.12, Single action schedule objects (class_id = 22): OBIS codes for Output control Single action schedule and Push Single action schedule added;
- 6.2.13, Register monitor objects (class_id = 21): OBIS code for Alarm monitor added;
- 6.2.14, Parameter monitor objects (class_id = 65): OBIS code added;
- 6.2.19, IEC twisted pair (1) setup objects (class_id = 24): OBIS codes for readout profiles added;
- 6.2.21, Objects to set up data exchange over the Internet: OBIS codes for GSM diagnostic, IPv6 setup, Push setup added;
- 6.2.24, Objects for data exchange using narrowband OFDM PLC for PRIME networks: OBIS codes added;
- 6.2.25, Objects for data exchange using narrow-band OFDM PLC for G3-PLC networks: OBIS codes added;
- 6.2.26, ZigBee® setup objects: OBIS codes added;
- 6.2.50, Meter tamper event related objects: OBIS codes added;
- 6.2.52 Alarm register, Alarm filter and Alarm descriptor objects: OBIS codes for Alarm descriptor objects added;
- 6.3, Electricity related COSEM objects: Precisions to the text and new data types added.
- Clause 7, Previous versions of interface classes: Earlier versions of interface classes have been brought here;
- Annex A (informative), Additional information on Auto answer and Auto connect ICs added;
- Annex B (informative), Additional information to M-Bus client (class_id = 72, version 1) added;
- Annex C (informative), Additional information on IPv6 setup class (class_id = 48, version = 0) added;
- Annex D (informative), Overview of the narrow-band OFDM PLC technology for PRIME networks added;
- Annex E (informative), Overview of the narrow-band OFDM PLC technology for G3-PLC networks added.

Bibliography

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. 12.1: 2015, COSEM interface classes and OBIS identification system

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 Amendments 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

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 62056-4-7:2015, Electricity metering data exchange – The DLMS/COSEM suite – Part 4-7: DLMS/COSEM transport layer for IP 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 62056-9-7:2013, Electricity metering data exchange – The DLMS/COSEM suite – Part 9-7: Communication profile for TCP-UDP/IP networks

ISO/IEC 10646:2014, Information technology – Universal Coded Character Set (UCS)

CLC/TS 52056-8-4:2015, ELECTRICITY METERING DATA EXCHANGE – THE DLMS/COSEM SUITE – Part 8-4: The narrowband OFDM PLC profile for PRIME networks

CLC/TS 52056-8-5:2015, ELECTRICITY METERING DATA EXCHANGE – THE DLMS/COSEM SUITE – Part 8-5: The narrowband OFDM PLC profile for G3-PLC networks

EN 13757-1:2014, Communication system for meters – Part 1: Data exchange

ITU Recommendation X.217:1995, Information technology – Open Systems Interconnection – Service definition for the association control service element

ITU Recommendation X.227:1995, Information technology – Open Systems Interconnection Connection-oriented protocol for the association control service element: Protocol specification

IETF STD 5, *Internet Protocol*, 1981. (Also IETF RFC 0791, RFC 0792, RFC 0919, RFC 0922, RFC 0950, RFC 1112)

ZigBee® 053474, ZigBee® Specification. The specification can be downloaded free of charge from <http://www.zigbee.org/Standards/ZigBeeSmartEnergy/Specification.aspx>

The following RFCs are available online from the Internet Engineering Task Force (IETF): <http://www.ietf.org/rfc/std-index.txt>, <http://www.ietf.org/rfc/>

RFC 793, *Transmission Control Protocol (Also IETF STD 0007)*, 1981, Updated by: RFC 3168

RFC 940, *Toward an Internet Standard Scheme for Subnetting*, 1985

RFC 950, *Internet Standard Subnetting Procedure*, 1985

RFC 2460, *Internet Protocol, Version 6 (IPv6)*, 1998

RFC 2473, *Generic Packet Tunneling in IPv6*, 1998

RFC 2675, *IPv6 Jumbograms*, 1999

RFC 2711, *IPv6 Router Alert Option*, 1999

RFC 3748, *Extensible Authentication Protocol (EAP)*, 2004

RFC 3775, *Mobility Support in IPv6*, 2004

RFC 4291, *IP Version 6 Addressing Architecture*, 2006

RFC 4302, *IP Authentication Header*, 2005

RFC 4303, *IP Encapsulating Security Payload (ESP)*, 2005

RFC 4944, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, 2007

RFC 5095, *Deprecation of Type 0 Routing Headers in IPv6*, 2007

RFC 6282, *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*, 2011

RFC 6775, *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*, 2012

Index

- 61334-4-32 LLC SCS setup 169
- Abstract object 219
- ac_max_ctl_re_tx 172
- Access rights 37
- Access selector 240
- access_number 129
- access_right 246
- access_rights_list 63, 243
- acknowledgement_time_t1 165
- acknowledgement_timer 164
- actions 99, 105
- activate_passive_ 98
- Active initiator 17, 156
- Active power 230
- active_devices 199
- active_initiator 154
- active_mask 51
- Activity calendar 94, 95, 207, 211
- add_mask 51
- add_object 71, 250, 255
- add_register 51
- addr_state 127, 132
- address_config_mode 138
- address_mask 132
- adjacent_cells 126
- adjust_to_measuring_period 89
- adjust_to_minute 89
- adjust_to_preset_time 89
- adjust_to_quarter 89
- adjustment_method 109
- adp_broadcast_log_table 192
- adp_broadcast_log_table_entry_TTL 190
- adp_max_hops 188
- adp_max_join_wait_time 192
- adp_path_discovery_time 192
- adp_routing_table 190
- adp_weak_LQI_value 188
- Advanced power failure monitoring 222
- alarm 271
- Alarm descriptor 82, 226, 298
- Alarm filter 226, 298
- Alarm monitor 82, 212
- Alarm register 82, 226, 298
- Algorithm 229, 230
- All configured 153
- All Physical 153
- always repeater 151, 267
- APN 123
- Apparent power 230
- Application context 36
- application_context_name 68, 247, 252
- aps_interframe_delay 197
- aps_max_window_size 197
- associated_partners_id 68, 247, 252
- Association 207, 218
- Association LN ... 28, 61, 66, 245, 250, 297
- Association object 166
- Association SN27, 28, 61, 62, 237, 239, 242, 297
- Association view 36, 37
- association_status 70, 249, 254
- Attribute 26, 28, 29, 72
- Authentication 239
- Authentication mechanism 36
- authentication_mechanism_name 69, 248, 253
- Auto answer 118, 261, 297
- Auto connect 121, 263, 297
- Auto dial 262, 263
- avail_baud 127
- backup_HAN 201, 202
- Base node 18, 167
- base_name 27, 63, 240, 243
- base_node_address 169
- Battery 222
- Beacon slot 18
- Billing period 207, 211, 228
- Billing period counter 209
- Billing period, end of 212
- Block demand 47, 49
- Broadcast 210
- Broadcast address 113, 114, 156, 258
- broadcast_frames_ 159
- buffer 53, 57, 234
- bus_address 127
- busy_state_timer 164
- CAD 20
- Calendar 95
- calendar_name 96
- CALLING device address, ... 113
- Calling line identification 120
- calling_window 123, 263, 264
- capture 56, 60, 236, 272
- Capture period 220, 225, 227, 231
- capture_definition 128, 270
- capture_objects 54, 235
- capture_period 54, 235, 270
- capture_time 47, 49
- Captured object 227, 231
- Cardinality 28
- cell_info 125
- Challenge 242, 249
- change_HLS_secret 71, 239, 242, 250, 255
- change_LLS_secret 239, 241
- change_secret 65, 245
- changed_parameter 106
- CHAP protocol 145
- CHAP_algorithm 145
- Christmas 95
- CIASE 151, 267
- class_id 28, 67, 246, 250
- Client user identification 61
- client_SAP 68, 247, 252
- client_system_title 80

Clock.....	38, 87, 207, 210, 235, 297, 298
<i>Clock synchronization method</i>	228
clock_base	89
close_tunnel_timeout.....	204
comm_speed.....	113, 115, 117, 133, 257, 259
Communication port log parameter.....	224
Communication tamper event.....	225
communication_window	85
Configuration	129, 221
Configuration data	28
Conformance block	69, 248, 253
connect_logical_device	72
Consumer message	224
Contracted value	227
control_mode	103
control_state.....	103
Convergence layer, PRIME NB OFDM PLC	295
COSEM logical device name	36, 37
COSEM objects, Abstract.....	206
COSEM objects, Electricity	227
COSEM physical device	146, 166
CRC_NOK_frames	160
CRC_OK_frames	160
create_HAN	203
cs_attachment	125
CT/VT ratio.....	229
Cumulative value	227
Current and last average values	227
Current association.....	37
current_average_value	47, 49
current_user	64, 70
Currently active tariff.....	224
Data.....	38, 42, 59, 98, 220, 223, 225, 228, 229, 232
Data access error	239
Data format	29, 31
Data security, access.....	37
Data security, transport.....	37
Data type.....	29, 30
data_index	54, 84
data_send	273
Date and time format	31, 297
day_profile_table	97
Daylight saving	87, 94
daylight_savings_deviation	89
daylight_savings_begin.....	89
daylight_savings_end	89
Default encryption key	273
default_baud.....	112, 126, 256
default_mode.....	111, 256
delete	93, 95
delete_mask	51
delta_electrical_phase	150, 266
Demand register	38, 47, 50, 51, 227
Demotion.....	20
destination_list	123, 265
Desynchronization	156
desynchronization-listing.....	159
Device ID	220
Device start-up	171
device_addr.....	112, 256
device_address	113, 132, 258
device_serial_number	116
device_type	129, 271
DHCP	137
Disconnect control	101, 210, 212, 221
Disconnected state	19
DiscoverReport request	161
DL_reference.....	135, 138
DLMS version	69, 248, 253
Domain Name Server.....	137
dynamic repeater	151, 267
Electrical port	213
Electricity breaker.....	221
Emergency activation time	104
Emergency duration	104
Emergency profile	104
Emergency profile, Limiter	105
Emergency threshold	104
emergency_profile	105
emergency_profile_active	105
emergency_profile_group.....	105
enable/disable	93
enable_disable	198
Encryption key.....	71, 130, 242, 250, 255, 269, 273
encryption_key_status	130
Entries, schedule	92
Entries, special days.....	95
entries_in_use	55, 235
Environmental related parameters.....	223
equipment_id.....	116
Error register	213, 225
Event code	224
Event counter	224
Event log	227
execute	91
executed_script	100
execution_time	100
Extended register.....	46, 50, 51, 59, 98, 108, 214, 222, 223, 227, 228, 232
extended_pan_ID.....	195
Fatal error register.....	116
fatal_error.....	259
Firmware	209
Floating point format	33
Forgotten system	156
Frame synchronization.....	156
frequencies.....	151, 266
Frequency notching, G3-PLC	296
Full Function Device	179
Fundamental.....	230
Fundamental component.....	230
G3-PLC	296
G3-PLC 6LoWPAN adaptation layer	218
G3-PLC 6LoWPAN adaptation layer setup	180, 187
G3-PLC MAC layer counters	179, 180, 218
G3-PLC MAC setup	179, 181, 218
Gas valve	221
gateway_IP_address.....	137
get_attributes&methods	241
get_attributes&services.....	239

get_buffer_by_index	237	Instances	28
get_buffer_by_range	236	Instantaneous value	227, 232
get_HLS_challenge	239	Instantiation	26, 28, 42, 43, 98, 205
getlist_by_classid	238	Intelligent search initiator	150, 155, 266
getobj_by_logicalname	238	inter_octet_time_out	113, 257
Global Positioning System	89	Interface class	26, 38, 40
global_key_transfer	81	Interface object	26
GPRS modem setup	123, 207, 215	Internal control signals	221, 222
GSM diagnostic	124, 207, 215	Internal operating status	221, 222, 231
Harmonics	230, 232	Internet	215
Hayes standard	117, 260	Invocation counter	219
HDLC mode	113, 257	IP_address	135
HDLC setup	207, 213	IP_options	136
HDLC setup class	112, 256	IP_reference	134
HLS secret	71, 242, 250, 255	IPCP_options	143
I/O control signal	221	IPv4 setup	134, 207, 215, 297
ID numbers, Electricity	227	IPv6 addressing	289
Identification number	269	IPv6 header	291
Identification system	233	IPv6 setup	207, 215
identification_number	129, 271	ISO/IEC 8802-2 LLC layer setup	207
identify_device	202	ISO/IEC 8802-2 LLC Type 1 setup	162, 216
IEC 61334-4-32 LLC setup	216	ISO/IEC 8802-2 LLC Type 2 setup	163, 216
IEC local port setup	207	ISO/IEC 8802-2 LLC Type 3 setup	164, 217
IEC twisted pair (1) Fatal Error register	214	last_average_value	47, 49
IEC twisted pair (1) MAC address setup	214	LCP_options	142
IEC twisted pair (1) setup	114, 115, 213, 214, 258, 297, 298	Limiter	103, 207, 212
IEEE address	20	Link key	20
IHD	20	link_key	195
Image	16, 79	List objects – Electricity	231
Image activation	210, 212	List objects – General	226
Image transfer	61, 72, 73, 207, 219	list_of	120
Image, activation	79	listening_window	119, 261
Image, completeness	76	LLC layer	160, 268
Image, initiate transfer	76	LLC sub-layer	146
image_activate	75	LLS secret	65, 241, 245
image_block_size	73	LLS_secret	248
image_block_transfer	75	Local port setup	111, 213, 255
image_first_not_transferred_block_number	74	local_disconnect	102
image_transfer_enabled	74	local_reconnect	102
image_transfer_initiate	75	Logical device	28, 35, 36, 37, 62, 66, 72, 239, 242, 250
image_transfer_status	74	Logical Device Name	27, 146, 166, 207, 219
image_transferred_blocks_status	74	Logical name	26, 27, 28
image_verify	75	Logical name referencing	66, 250
ImageBlock	17	logical_name	26, 42, 73
ImageBlockNumber	17	login_password	146
ImageBlocks, transfer	76	MAC address	20, 116, 154
ImageBlockSize	17, 76	MAC address setup	207, 215, 217
ImageSize	17	MAC sub-layer	146
In Home Display	20	mac_address	140, 151, 267
Inactive objects	227	mac_coord_short_address	276
inactivity_time_out	113, 134, 258	mac_freq_notching	276
Information security	37	mac_get_neighbour_table_entry	279
Information, measured	26	mac_group_addresses	151, 267
Information, static	26	mac_max_frame_retries	276
initialization_string	117, 260	mac_max_orphan_timer	276
Initiator	17, 153	mac_neighbour_table	277
initiator_electrical_phase	150, 265	mac_neighbour_table_	276
initiator_mac_address	152, 268	mac_PAN_id	276
insert	93, 95	mac_security_enabled	276
Install code	20	mac_short_address	276

mac_TMR_TTL	276	nb_of_sim_conn	134
Maintenance, interface classes	205	neighbor_discovery_setup	139
Management Logical Device 36, 37, 166		Network Control Protocol	143
Mandatory	28, 29	network_key	196
manual_disconnect	102	Never repeater	151, 267
manual_reconnect	102	NEW	153
Manufacturer ID	269	New system	18
Manufacturer specific	28	New system title	18
manufacturer_id	116, 271	next_period	50
mask_list	51	NO-BODY	153
max_frame_length	161, 269	Node	19
max_info_length_transmit	113	Non configured state	156
max_info_length_receive	113, 257	Normal mode	211
max_info_length_transmit	257	Normal threshold	104
max_number_transmissions_n2	164	Notation	27
max_number_transmissions_n4	165	number_of_calls	119, 262
max_octets_acn_pdu_n3	164	number_of_periods	47, 49
max_octets_i_pdu_n1	163	number_of_retries	86
max_octets_ui_pdu	162	number_of_rings	120, 262
max_receiving_gain	150, 266	OBIS	26, 28, 233
max_transmitting_gain	150, 266	Object Identification System	206
Maximum	235	object_list 37, 63, 68, 70, 71, 238, 240, 243, 246, 249, 251, 254	
Maximum and minimum values	227	Occurrence counter	227
Maximum Segment Size	134	Operating time	223
maximum_incoming_transfer_size	204	Operator	124
maximum_outcoming_transfer_size	204	Optical port	213
M-Bus	207	Optical test output	211
M-Bus client .. 214, 269, 274, 275, 279, 287		Optional	28, 29
M-Bus control log object	214	Orthogonal Frequency Division Multiplexing	295
M-Bus disconnect control object	214	Output control	212
M-Bus master port setup	133, 214	Output pulse constant	228
M-Bus port setup	269	Output signals	211
M-Bus profile generic object	214, 269	output_state	103
M-Bus slave	214, 269	output_type	109
M-Bus slave port setup	126, 214	p_bit_timer	164
M-Bus value object	269	Packet switched network	122
mbus_port	128	PAN coordinator	179
mbus_port_reference	270	pan_ID	195
Measurement period	228	Parameter changes	221
Medium access control layer	295	Parameter monitor 106, 207, 212, 297, 298	
Meter open event	225	Parameter monitor log	106
Meter reset	211	parameter_list	107
Meter tamper event	225, 298	pass_p1	112, 256
Metering point ID	220, 231	pass_p2	112, 256
Method	26, 29	pass_w5	112, 256
metrological	108	Passive calendar	98
Metrology tamper event	225	Password	65, 239, 241, 245, 256
MIB variables	146	Period	47
min_delta_credit	268	Personal Area Network	193
min_over_threshold_duration	104	phone_list	263
min_under_threshold_duration	104	PHY_reference	141
mode	115, 119, 122, 261, 263, 264	Physical device	35, 72, 218
Modem	117, 210, 260, 262	Physical layer, PRIME NB OFDM PLC ..	295
Modem configuration	117, 207, 210	PIN_code	123
modem_profile	117, 260	PLC OFDM Type 1 Application identification	217
monitored_value	99, 104, 195, 198	PLC OFDM Type 1 MAC counters	217
MSS	134	PLC OFDM Type 1 MAC functional parameters	217
multicast_IP_address	135		
NB OFDM PLC for G3-PLC networks	207		
NB OFDM PLC for PRIME networks	207		
NB OFDM PLC profile for PRIME networks	166		

PLC OFDM Type 1 MAC network administration data	217	Register26, 42, 46, 50, 51, 59, 98, 223, 227, 228, 232	
PLC OFDM Type 1 MAC setup	217	Register activation	50, 207, 212
PLC OFDM Type 1 physical layer counters	217	Register monitor98, 108, 207, 212, 232, 233, 298	
Power factor	230	Register table	58
Power factor, displacement	230	<i>register_assignment</i>	50, 51
Power factor, true	230	register_device	200
Power failure	94, 134, 222	Registered system	18
Power failure duration	223	Registration	19
Pperiod	49	reject_timer	164
PPP setup	141, 207, 215, 297	rejoin_interval	196
PPP_authentication	145	rejoin_retry_interval	196
Preferred readout	56	remote_disconnect	102, 103
preset_adjusting_time	90	remote_reconnect	102, 103
Primary address	269	remove_HAN	204
primary_address	129, 271	remove_mirror	203
primary_address_list	115, 259	remove_object	250
primary_DNS_address	137, 139	remove_object (data)	71, 255
PRIME NB OFDM PLC Applications identification	178	repeater	151, 267
PRIME NB OFDM PLC MAC counters ..	174	repeater_status	152, 268
PRIME NB OFDM PLC MAC functional parameters	172	Repetition phase	157, 160
PRIME NB OFDM PLC MAC setup	171	repetition_delay	86, 122, 263, 264
PRIME NB OFDM PLC Physical layer counters	170	repetitions	122, 263, 264
Process value	28, 42	repetitions_counter	160
processed_value	110	reply_status_list	269
processed_value_actions	110	reply_to_HLS	66, 71, 245, 249, 255
processed_value_thresholds	110	reply_to_HLS_authentication	242
Profile	234	reply_to_HLS_challenge	239
Profile entries	220, 225, 228, 231	Reporting system	18
Profile generic38, 52, 208, 209, 220, 225, 227, 231		Reporting system list	161
profile_entries	55, 235	Reserved base_names	27
Program entries	228	reset	43, 47, 50, 56, 60, 110, 236
Promotion	19	reset_alarm	272
prop_baud	112, 256	reset_NEW_not	154
Proprietary	26	response_time	112
protocol_address	204	restore_HAN	202
protocol_version	195	Robust Header Compression	144
ps_status	125	SAP	72
PSTN modem configuration	259	SAP assignment27, 36, 37, 61, 72, 207, 218	
Public client	37	SAP_assignment_list	72
<i>Publish/subscribe</i>	81	scaler_unit	43, 60, 109
Push	81, 210, 212, 297	scan_attempts	196
Push setup	83, 207, 215	Schedule	91, 94, 95, 207, 211
Push window	83	Script	90, 92, 93, 100
quality_of_service	124	Script table27, 90, 96, 98, 100, 207, 210, 298	
random delay	83	Script, null	90
randomisation_start_interval	86	scripts	90
raw_value	109	sealing_method	109
raw_value_actions	109	search_initiator	155
raw_value_thresholds	109	season_profile	96
Reactive power	230	secondary_address	258
read_by_logicalname	65, 238, 241, 244	secondary_DNS_address	137, 139
Reading factor	229	Secret	65, 70, 245, 250, 253
Readout	56, 207	Secret, new	242
receive_lifetime_var_t2	165	Security mechanisms	37
receive_window_size_rw	163	Security setup	61, 207, 219
Reduced Function Device	179	security_activate	81
		security_policy	80
		security_setup_reference ..	64, 70, 244, 254

security_suite	80	synchronization-locked	158
Selective access	29, 53, 55, 57, 63, 64, 68, 70, 84, 240, 243, 244, 246, 249, 251, 254	Synchronization-locked state	152, 268
Selective access parameters	29	Synchronization-unlocked state	152, 268
send_	85	synchronize_clock	130, 272
sender_address	146	System Title	154
Sensor manager	107, 108	tabi_list	115, 259
Sensor, pressure	110	table_cell_definition	59
Serial number, sensor	108	table_cell_values	58
serial_number	108	table_ID	57
Server model	35	Tamper	225, 298
server_address	146	Tariffication	50, 211, 212
server_port	146	TCP-UDP setup	133, 207, 215
server_SAP	68, 247, 252	TCP-UDP_port	134
server_system_title	80	Terminal cover open event	225
Service node	19	Terminal node	167
Service Specific Convergence Sublayer	169	Terminal state	19
service_node_address	169	Test mode	211
set_encryption_key	273	threshold	98
S-FSK Active initiator	216	Threshold value	104, 232
S-FSK MAC counters	216	Threshold, missing	232
S-FSK MAC synchronization timeouts	216	Threshold, over limit	232
S-FSK Phy&MAC setup	216	Threshold, under limit	232
S-FSK Physical layer	146	threshold_active	104
S-FSK PLC setup	207	threshold_emergency	104
S-FSK Reporting system list	216	threshold_normal	104
Shared line	119, 261	Tilt event	225
shift_time	90	time	88
Short name	27, 29	Time changes	94
short_address	195	Time entries	228
Single action schedule	100, 207, 212, 298	Time integral	227
slave_deinstall	130, 272	Time setting backward	94
slave_install	130, 272	Time setting forward	94
Sliding demand	47, 49	Time synchronization	94
Smart Energy Profile	195	time_between_scans	196
SMTP setup	145, 215	time_out_frame_not_	156
Sort method	220, 225, 228, 231	time_zone	89
sort_method	54, 235	Timeslot	18
sort_object	54, 235	transfer_key	273
Special days	92	Transmission phase	157, 160
Special days table	94, 96, 207, 211	transmissions_counter	160
stack_profile	195	transmit_lifetime_var_t3	166
Standard readout profile	213	transmit_window_size_k	163
start_time_current	49	trust_center_address, ZigBee®	195
start_up_control	195	Twisted pair	114, 258, 297
status	46, 49, 89, 119, 271	Twisted pair setup	207
Status mapping	60, 297	Unconfigured	272
Status register	223	Unconfigured state	154
Status value	42	unicast_IPv6_addresses	138
status_word	60	Unit	44
Strong DC magnetic field event	225	UNIX clock	210
subnet_mask	137	unregister_all_devices	201
Subnetwork	19	unregister_device	201
Sub-slot	18	update_link_key	203
Switch node	167	update_network_key	203
Switch state	19	use_DHCP_flag	137
SYNCHRO_FOUND	157	use_insecure_join	196
Synchronization	94, 151, 267	User group specific	28
Synchronization process	157	user_list	64, 70
synchronization_confirmation_timeout	156	user_name	146
synchronization_locked	152, 268	UTC	87, 88, 89
		Utility tables	57, 207, 219
		Value	26, 42, 43

Value group C	207	ZigBee®20, 85, 192, 196, 207, 218, 297, 298	
Value group D	227	ZigBee® 053474	300
Value group F	232	ZigBee® 2007	194
Value, default	29	ZigBee® client	20
Value, maximum	29	ZigBee® cluster	21
Value, minimum	29	ZigBee® Communications hub	193
Van Jacobson compression	144	ZigBee® coordinator	21, 193
Version.....28, 67, 205, 233, 246, 250, 271		ZigBee® mirror	21, 203
Version, previous	234, 298	ZigBee® network control	198, 218
Wake-up	285	ZigBee® PAN	199
Wake-up request	121	ZigBee® PAN creation	193
week_profile_table	97	ZigBee® Pro	21, 194
window_size_receive	113, 257	ZigBee® router	21
window_size_transmit	113, 257	ZigBee® SAS APS fragmentation	197, 218
Wireless Mode Q	207, 214	ZigBee® SAS join	196, 218
Wireless Mode Q channel	132	ZigBee® SAS startup	194, 218
write	236	ZigBee® server	21, 193
xDLMS context	36	ZigBee® Smart Energy Specification	202
xDLMS_context_info	69, 248, 253	ZigBee® Trust Center	21, 193, 195
year_of_manufacture	116	ZigBee® tunnel setup	204, 218

Withd
IECNORM.COM: Click to view the full PDF of IEC 62056-6-2:2016

SOMMAIRE

AVANT-PROPOS.....	315
INTRODUCTION.....	318
1 Domaine d'application.....	320
2 Références normatives	320
3 Termes, définitions et abréviations.....	323
3.1 Termes et définitions liés au processus de transfert d'Image (voir 5.3.6)	323
3.2 Termes et définitions liés aux classes S-FSK PLC setup (voir 5.8)	324
3.3 Termes et définitions liés aux classes d'interface PRIME NB OFDM PLC setup (voir 5.10).....	325
3.4 Termes et définitions liés à ZigBee® (voir 5.12)	326
3.5 Abréviations.....	328
4 Principes de base.....	331
4.1 Généralités	331
4.2 Méthodes de référencement	334
4.3 Base_name réservés pour des objets COSEM spéciaux.....	334
4.4 Notation pour la description de classes.....	334
4.5 Types de données communs.....	337
4.6 Formats de données.....	338
4.6.1 Formats des date et heure	338
4.6.2 Formats de nombres en virgule flottante	341
4.7 Modèle de serveur COSEM	343
4.8 Dispositif logique COSEM	344
4.8.1 Généralités	344
4.8.2 Nom de dispositif logique COSEM (LDN)	344
4.8.3 "Vue association" du dispositif logique.....	345
4.8.4 Contenu obligatoire d'un dispositif logique COSEM.....	345
4.8.5 Dispositif logique de gestion.....	345
4.9 Sécurité des informations	346
5 Classes d'interfaces de la COSEM	346
5.1 Vue d'ensemble	346
5.2 Classes d'interfaces pour paramètres et données de mesure	351
5.2.1 Data (class_id = 1, version = 0).....	351
5.2.2 Register (class_id = 3, version = 0)	352
5.2.3 Extended register (class_id = 4, version = 0)	356
5.2.4 Demand register (class_id = 5, version = 0)	357
5.2.5 Register activation (class_id = 6, version = 0).....	361
5.2.6 Profile generic (class_id = 7, version = 1)	363
5.2.7 Utility tables (class_id = 26, version = 0).....	369
5.2.8 Register table (class_id = 61, version = 0)	370
5.2.9 Status mapping (class_id = 63, version = 0).....	373
5.3 Classes d'interfaces pour contrôle d'accès et gestion.....	374
5.3.1 Vue d'ensemble	374
5.3.2 Identification de l'utilisateur du client	375
5.3.3 Association SN (class_id = 12, version = 3)	375
5.3.4 Association LN (class_id = 15, version = 2)	379
5.3.5 SAP assignment (class_id = 17, version = 0)	387

5.3.6	Image transfer (Transfert d'Image)	387
5.3.7	Security setup (class_id = 64, version = 0).....	398
5.3.8	Classes d'interfaces et objets poussés	399
5.4	Classes d'interfaces pour commande à limite temporelle et événementielle	408
5.4.1	Clock (class_id = 8, version = 0).....	408
5.4.2	Script table (class_id = 9, version = 0)	411
5.4.3	Schedule (class_id = 10, version = 0)	412
5.4.4	Special days table (class_id = 11, version = 0)	416
5.4.5	Activity calendar (class_id = 20, version = 0)	417
5.4.6	Register monitor (class_id = 21, version = 0)	420
5.4.7	Single action schedule (class_id = 22, version = 0)	422
5.4.8	Disconnect control (class_id = 70, version = 0)	422
5.4.9	Limiter (class_id = 71, version = 0).....	425
5.4.10	Parameter monitor (class_id = 65, version = 0)	428
5.4.11	Classe d'interfaces Sensor manager.....	430
5.5	Classes d'interfaces pour l'établissement de l'échange de données via des ports locaux et des modems	434
5.5.1	IEC local port setup (class_id = 19, version = 1)	434
5.5.2	IEC HDLC setup (class_id = 23, version = 1)	435
5.5.3	IEC twisted pair (1) setup (class_id = 24, version = 1).....	437
5.5.4	Modem configuration (class_id = 27, version = 1)	440
5.5.5	Auto answer (class_id = 28, version = 2)	442
5.5.6	Auto connect (class_id = 29, version = 2)	447
5.5.7	GPRS modem setup (class_id = 45, version = 0)	449
5.5.8	GSM diagnostic (class_id = 47, version = 0)	450
5.6	Classes d'interfaces pour l'établissement d'échange de données via le M-Bus.....	452
5.6.1	M-Bus slave port setup (class_id = 25, version = 0)	452
5.6.2	M-Bus client (class_id = 72, version = 1)	453
5.6.3	Wireless Mode Q channel (class_id = 73, version = 1)	458
5.6.4	M-Bus master port setup (class_id = 74, version = 0)	459
5.7	Classes d'interfaces pour l'établissement d'échange de données sur Internet	459
5.7.1	TCP-UDP setup (class_id = 41, version = 0)	459
5.7.2	IPv4 setup (class_id = 42, version = 0)	461
5.7.3	IPv6 setup (class_id = 48, version = 0)	464
5.7.4	MAC address setup (class_id = 43, version = 0)	468
5.7.5	PPP setup (class_id = 44, version = 0)	468
5.7.6	SMTP setup (class_id = 46, version = 0).....	472
5.8	Classes d'interfaces pour l'établissement d'échange de données en utilisant le PLC à modulation S-FSK	473
5.8.1	Généralités	473
5.8.2	Vue d'ensemble	474
5.8.3	S-FSK Phy&MAC set-up (class_id = 50, version = 1)	476
5.8.4	S-FSK Active initiator (class_id = 51, version = 0).....	481
5.8.5	S-FSK MAC synchronization timeouts (class_id = 52, version = 0)	482
5.8.6	S-FSK MAC counters (class_id = 53, version = 0)	485
5.8.7	IEC 61334-4-32 LLC setup (class_id = 55, version = 1).....	488
5.8.8	S-FSK Reporting system list (class_id = 56, version = 0).....	489
5.9	Classes d'interfaces pour l'établissement de la couche LLC pour l'ISO/IEC 8802-2	490

5.9.1	Généralités	490
5.9.2	ISO/IEC 8802-2 LLC Type 1 setup (class_id = 57, version = 0)	490
5.9.3	ISO/IEC 8802-2 LLC Type 2 setup (class_id = 58, version = 0)	491
5.9.4	ISO/IEC 8802-2 LLC Type 3 setup (class_id = 59, version = 0)	493
5.10	Classes d'interfaces pour l'établissement et la gestion du profil PLC OFDM à bande étroite DLMS/COSEM pour les réseaux PRIME	495
5.10.1	Vue d'ensemble	495
5.10.2	Mise en correspondance des attributs PRIME NB OFDM PLC PIB avec les attributs d'IC COSEM.....	497
5.10.3	61334-4-32 LLC SSCS setup (class_id = 80, version = 0)	499
5.10.4	Paramètres de couche physique PRIME NB OFDM PLC	499
5.10.5	PRIME NB OFDM PLC Physical layer counters (class_id = 81, version = 0)	499
5.10.6	PRIME NB OFDM PLC MAC setup (class_id = 82, version = 0)	500
5.10.7	PRIME NB OFDM PLC MAC functional parameters (class_id = 83, version = 0)	502
5.10.8	PRIME NB OFDM PLC MAC counters (class_id = 84, version = 0)	503
5.10.9	PRIME NB OFDM PLC MAC network administration data (class_id = 85, version = 0)	504
5.10.10	PRIME NB OFDM PLC MAC address setup (class_id = 43, version = 0)	507
5.10.11	PRIME NB OFDM PLC Application identification (class_id = 86, version = 0)	507
5.11	Classes d'interfaces pour l'établissement et la gestion du profil PLC OFDM à bande étroite DLMS/COSEM pour les réseaux G3-PLC	508
5.11.1	Vue d'ensemble	508
5.11.2	Mise en correspondance des attributs G3-PLC PIB avec les attributs d'IC COSEM	509
5.11.3	G3-PLC MAC layer counters (class_id = 90, version = 1)	510
5.11.4	G3-PLC MAC setup (class_id = 91, version = 1)	512
5.11.5	G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 1)	517
5.12	Classes d'établissement ZigBee®.....	524
5.12.1	Vue d'ensemble	524
5.12.2	ZigBee® SAS startup (class_id = 101, version = 0)	526
5.12.3	ZigBee® SAS join (class_id = 102, version = 0)	528
5.12.4	ZigBee® SAS APS fragmentation (class_id = 103, version = 0)	529
5.12.5	ZigBee® network control (class_id = 104, version = 0)	530
5.12.6	ZigBee® tunnel setup (class_id = 105, version = 0).....	537
5.13	Maintenance des classes d'interfaces	538
5.13.1	Nouvelles versions de classes d'interfaces	538
5.13.2	Nouvelles classes d'interfaces.....	538
5.13.3	Retrait de classes d'interfaces.....	538
6	Relation à l'OBIS	538
6.1	Généralités	538
6.2	Objets COSEM abstraits.....	539
6.2.1	Utilisation du groupe de valeurs C	539
6.2.2	Données relatives aux périodes de facturation historiques	539
6.2.3	Valeurs des périodes de facturation/réinitialisation d'entrées de compteur	541
6.2.4	Autres codes OBIS d'usage général abstraits	542
6.2.5	Objets Clock (class_id = 8).....	542
6.2.6	Configuration de modem et objets connexes	543

6.2.7	Objets Script table (class_id = 9)	543
6.2.8	Objets Special days table (class_id = 11)	544
6.2.9	Objets Schedule (class_id = 10)	544
6.2.10	Objets Activity calendar (class_id = 20)	545
6.2.11	Objets Register activation (class_id = 6)	545
6.2.12	Objets Single action schedule (class_id = 22)	545
6.2.13	Objets Register monitor (class_id = 21)	545
6.2.14	Objets Parameter monitor (class_id = 65)	546
6.2.15	Objets Limiter (class_id = 71)	546
6.2.16	Objets IEC local port setup (class_id = 19)	546
6.2.17	Objets Standard readout profile (class_id = 7)	546
6.2.18	Objets IEC HDLC setup (class_id = 23)	547
6.2.19	Objets IEC twisted pair (1) setup (class_id = 24)	547
6.2.20	Objets liés à l'échange de données sur M-Bus	548
6.2.21	Objets pour établir l'échange de données sur internet	549
6.2.22	Objets pour l'établissement d'échange de données en utilisant PLC S-FSK	550
6.2.23	Objets pour l'établissement de la couche LLC de l'ISO/IEC 8802-2	551
6.2.24	Objets pour l'échange de données en utilisant l'OFDM PLC à bande étroite pour les réseaux PRIME	551
6.2.25	Objets pour l'échange de données en utilisant l'OFDM PLC à bande étroite pour les réseaux G3-PLC	552
6.2.26	Objets d'établissement ZigBee®	553
6.2.27	Objets Association (class_id = 12, 15)	553
6.2.28	Objet SAP assignment (class_id = 17)	553
6.2.29	Objet COSEM logical device name	554
6.2.30	Objets relatifs à la sécurité des informations	554
6.2.31	Objets Image transfer (class_id = 18)	554
6.2.32	Objets Utility table (class_id = 26)	554
6.2.33	Objets Device ID	555
6.2.34	Objets Metering point ID	556
6.2.35	Objets "Parameter changes" et "calibration"	556
6.2.36	Objets I/O control signal	556
6.2.37	Objets Disconnect control (class_id = 70)	556
6.2.38	Objets Status of internal control signals	557
6.2.39	Objets Internal operating status	557
6.2.40	Objets Battery entries	558
6.2.41	Objets Power failure monitoring	558
6.2.42	Objets Operating time	558
6.2.43	Objets Environment related parameters	559
6.2.44	Objets Status register	559
6.2.45	Objets Event code	559
6.2.46	Objets Communication port log parameter	560
6.2.47	Objets Consumer message	560
6.2.48	Objets Currently active tariff	560
6.2.49	Objets Event counter	560
6.2.50	Objets liés à "Meter tamper event"	560
6.2.51	Objets Error register	561
6.2.52	Objets "Alarm register", "Alarm filter" et "Alarm descriptor"	562
6.2.53	Objets General list	562

6.2.54	Objets Event log	563
6.2.55	Objets Inactive	563
6.3	Objets COSEM liés à l'électricité	563
6.3.1	Définition du groupe de valeurs D	563
6.3.2	Numéros d'ID d'électricité	564
6.3.3	Valeurs des périodes de facturation/réinitialisation d'entrées de compteur	564
6.3.4	Autres objets d'usage général liés à l'électricité	565
6.3.5	Algorithme de mesure	566
6.3.6	ID de point de comptage (lié à l'électricité)	568
6.3.7	Objets états liés à l'électricité (Electricity related status)	568
6.3.8	Objets "List" – Électricité (class_id = 7)	569
6.3.9	Valeurs de seuil	569
6.3.10	Objets Register monitor (class_id = 21)	570
6.4	Codage des identifications OBIS	570
7	Précédentes versions des classes d'interfaces	571
7.1	Généralités	571
7.2	Profile generic (class_id = 7, version = 0)	571
7.3	Association SN (class_id = 12, version = 0)	575
7.4	Association SN (class_id = 12, version = 1)	577
7.5	Association SN (class_id = 12, version = 2)	580
7.6	Association LN (class_id = 15, version = 0)	583
7.7	Association LN (class_id = 15, version = 1)	589
7.8	IEC local port setup (class_id = 19, version = 0)	595
7.9	IEC HDLC setup (class_id = 23, version = 0)	596
7.10	IEC twisted pair (1) setup (class_id = 24, version = 0)	598
7.11	PSTN modem configuration (class_id = 27, version = 0)	599
7.12	Auto answer (class_id = 28, version = 0)	601
7.13	PSTN auto dial (class_id = 29, version = 0)	603
7.14	Auto connect (class_id = 29, version = 1)	604
7.15	S-FSK Phy&MAC setup (class_id = 50, version = 0)	605
7.16	S-FSK IEC 61334-4-32 LLC setup (class_id = 55, version = 0)	610
7.17	M-Bus client (class_id = 72, version = 0)	610
7.18	G3 NB OFDM PLC MAC layer counters (class_id = 90, version = 0)	616
7.19	G3 NB OFDM PLC MAC setup (class_id = 91, version = 0)	617
7.20	G3 NB OFDM PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 0)	622
Annex A (informative) Informations supplémentaires relatives aux IC "Auto answer" et "Auto connect"		627
Annex B (informative) Informations supplémentaires relatives à M-Bus client (class_id = 72, version 1)		630
Annex C (informative) Informations supplémentaires relatives à la classe IPv6 setup (class_id = 48, version = 0)		633
C.1	Généralités	633
C.2	Adressage IPv6	633
C.3	Format d'en-tête IPv6 header	635
C.4	Extensions d'en-tête IPv6	637
C.4.1	Vue d'ensemble	637
C.4.2	Options Hop-by-Hop	637

C.4.3	Options de destination	637
C.4.4	Options Routage	638
C.4.5	Options Fragment	638
C.4.6	Options Sécurité	638
Annex D (informative) Vue d'ensemble de la technologie OFDM PLC à bande étroite pour les réseaux PRIME		639
Annex E (informative) Vue d'ensemble de la technologie OFDM PLC à bande étroite pour les réseaux G3-PLC		640
Annex F (informative) Modifications techniques majeures par rapport à l'IEC 62056-6-2:2013		642
Bibliographie		644
Index		646
Figure 1	– Signification des définitions concernant l'Image	323
Figure 2	– Une classe d'interfaces et ses instances	333
Figure 3	– Modèle de serveur COSEM	343
Figure 4	– Dispositif de comptage combiné	344
Figure 5	– Vue d'ensemble des classes d'interfaces – Partie 1	347
Figure 6	– Vue d'ensemble des classes d'interfaces – Partie 2	348
Figure 7	– Attributs de temps pour mesurer une puissance glissante	358
Figure 8	– Attributs dans le cas de la puissance en bloc	358
Figure 9	– Attributs dans le cas de la puissance glissante (nombre de périodes = 3)	359
Figure 10	– Organigramme du processus de transfert d'image	397
Figure 11	– Modèle COSEM d'opération Push	401
Figure 12	– Fenêtres de poussée et délais	402
Figure 13	– Concept de temps généralisé	408
Figure 14	– Diagramme d'états de l'IC "Disconnect control"	423
Figure 15	– Définition des seuils supérieur et inférieur	433
Figure 16	– Modèle d'objet des serveurs DLMS/COSEM	474
Figure 17	– Modèle d'objet des serveurs DLMS/COSEM	496
Figure 18	– Exemple de réseau ZigBee®	525
Figure 19	– Données de périodes de facturation historiques – Exemple avec module 12 VZ = 5	541
Figure A.1	– Exemple de connectivité d'un réseau GSM/GPRS	628
Figure B.1	– Diagramme d'états de clé de chiffrement	631
Figure C.1	– Formats d'adresse IPv6	634
Figure C.2	– Format d'en-tête IPv6	635
Figure C.3	– Format du paramètre de la classe Traffic	636
Tableau 1	– Base_name réservés pour le référencement par SN	334
Tableau 2	– Types de données communs	337
Tableau 3	– Liste des classes d'interfaces par class_id	349
Tableau 4	– Valeurs énumérées pour les unités physiques	354
Tableau 5	– Exemples pour scaler_unit	356
Tableau 6	– Codage des paramètres d'accès sélectif avec data_index	407

Tableau 7 – Schedule (programme).....	413
Tableau 8 – Special days table (Tableau de jours spéciaux)	413
Tableau 9 – IC "Disconnect control" – états et transitions d'état	424
Tableau 10 – Présentation explicite des tableaux de valeurs seuils.....	433
Tableau 11 – Présentation explicite d'action_sets.....	434
Tableau 12 – Éléments de l'adresse ADS	440
Tableau 13 – Registre des erreurs fatales	440
Tableau 14 – Mise en correspondance des variables MIB de l'IEC 61334-4-512:2001 avec les attributs/méthodes des IC de la COSEM	475
Tableau 15 – Adresses MAC dans le profil S-FSK	481
Tableau 16 – Mise en correspondance des attributs PRIME NB OFDM PLC PIB avec les attributs d'IC COSEM	497
Tableau 17 – Mise en correspondance des attributs G3-PLC IB avec les attributs d'IC COSEM.....	509
Tableau 18 – Utilisation des classes d'interfaces ZigBee® setup COSEM	526
Tableau 19 – Utilisation du groupe de valeurs C pour des objets abstraits dans le contexte de COSEM	539
Tableau 20 – Représentation de diverses valeurs par les IC appropriées	564
Tableau 21 – Algorithmes de mesure – valeurs énumérées.....	567
Tableau 22 – Objets "Threshold" (seuils), électricité	570
Tableau 23 – Objets Register monitor, électricité.....	570
Tableau B.1 – La clé de chiffrement est prédéfinie dans l'esclave et ne peut pas être modifiée	631
Tableau B.2 – La clé de chiffrement est prédéfinie dans l'esclave, et une nouvelle clé est définie après l'installation	631
Tableau B.3 – La clé de chiffrement n'est pas prédéfinie dans l'esclave, mais peut être définie, cas a).....	632
Tableau B.4 – La clé de chiffrement n'est pas prédéfinie dans l'esclave, mais peut être définie, cas b).....	632
Tableau C.1 – En-têtes IPv6 par rapport à l'IC IPv6	636
Tableau C.2 – Extensions d'en-tête IPv6 facultatives par rapport à l'IC IPv6.....	637

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

**ÉCHANGE DES DONNÉES DE COMPTAGE DE L'ÉLECTRICITÉ –
LA SUITE DLMS/COSEM –****Partie 6-2: Classes d'interfaces COSEM****AVANT-PROPOS**

- 1) La Commission Electrotechnique 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. A 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 de la présente Publication 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 et de ne pas avoir signalé leur existence.

La Commission Electrotechnique Internationale (IEC) attire l'attention sur le fait qu'il est déclaré que la conformité avec les dispositions du présent document peut impliquer l'utilisation d'un brevet intéressant la procédure de transfert d'image.

L'IEC ne prend pas position quant à la preuve, à la validité et à la portée de ces droits de propriété.

Le détenteur de ces droits de propriété a donné l'assurance à l'IEC qu'il consent à négocier des licences avec des demandeurs du monde entier, soit sans frais soit à des termes et conditions raisonnables et non discriminatoires. À ce propos, la déclaration du détenteur des droits de propriété est enregistrée à l'IEC. Des informations peuvent être demandées à: Itron, Inc., Liberty Lake, Washington, USA.

L'attention est d'autre part attirée sur le fait que certains des éléments du présent document peuvent faire l'objet de droits de propriété autres que ceux qui ont été mentionnés ci-dessus. L'IEC ne saurait être tenue pour responsable de l'identification de ces droits de propriété en tout ou partie.

L'IEC (<http://patents.iec.ch>) maintiennent des bases de données, consultables en ligne, des droits de propriété pertinents à leurs normes. Les utilisateurs sont encouragés à consulter ces bases de données pour obtenir l'information la plus récente concernant les droits de propriété.

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 62056-6-2.

L'IEC ne prend pas position quant à la preuve, à la validité et à la portée de ce service de maintenance.

Le fournisseur du service de maintenance a donné l'assurance à l'IEC qu'il consent à négocier des services avec des demandeurs du monde entier, à des termes et 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:

DLMS¹ User Association
Zug/Suisse
www.dlms.com

La Norme internationale IEC 62056-6-2 a été établie par le comité d'études 13 de l'IEC: Comptage et pilotage de l'énergie électrique.

Cette deuxième édition annule et remplace la première édition de l'IEC 62056-6-2, parue en 2013. Cette édition constitue une révision technique.

Les modifications techniques majeures par rapport à l'IEC 62056-6-2:2013 sont énumérées à l'Annex F (informative).

Le texte de cette norme est issu des documents suivants:

FDIS	Rapport de vote
13/1651A/FDIS	13/1659/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette norme.

Cette publication a été rédigée selon les Directives ISO/IEC, Partie 2.

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*, peut être consultée sur le site web de l'IEC.

¹ Spécification de message de langage de dispositif.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "<http://webstore.iec.ch>" dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

IMPORTANT – Le logo "*colour inside*" qui se trouve sur la page de couverture de cette publication 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.

INTRODUCTION

Cette deuxième édition de l'IEC 62056-6-2 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 D.

Cette édition est conforme à l'Édition 11.0 du Livre Bleu de la DLMS UA. Les nouvelles fonctionnalités principales sont le mécanisme d'identification de l'utilisateur client et les classes d'interface "Push setup", "Parameter monitor", "GSM diagnostic", "IPv6 setup", "Prime NB OFDM PLC setup", "G3-PLC setup" et "ZigBee[®] setup"².

En 2014, le DLMS UA a publié l'Édition 12.0 du Livre Bleu, qui prévoit de nouvelles caractéristiques en matière de fonctionnalité, d'efficacité et de sécurité tout en maintenant une rétrocompatibilité totale.

En outre, l'intention de la DLMS UA est de mettre ces derniers développements en conformité avec la normalisation internationale. Par conséquent, le groupe de travail 14 du comité d'études 13 de l'IEC a lancé un projet visant à rendre conformes ces nouveaux éléments à la série de normes IEC 62056 en vue de présenter l'Édition 3.0 de la norme.

Modélisation d'objet et identification de données

Motivé par les besoins du secteur d'activité des acteurs du marché de l'énergie – généralement dans un environnement compétitif libéralisé – et par le souhait de gérer efficacement les ressources naturelles et d'impliquer le consommateur, le compteur est devenu une partie intégrante d'un système de mesure, de commande et de facturation. Il ne s'agit plus d'un simple dispositif d'enregistrement des données, mais s'appuie de façon critique sur les capacités de communication. La facilité d'intégration, l'interopérabilité et la sécurité des données du système sont des exigences importantes.

Le COSEM (Companion Specification for Energy Metering) relève ces défis en considérant le compteur comme une partie d'un système de mesure et de commande complexe. Le compteur doit pouvoir acheminer les résultats de mesure des points de comptage vers les processus commerciaux qui les utilise. Il doit également pouvoir donner des informations au consommateur et gérer la consommation et éventuellement la production locale.

Pour ce faire, le COSEM utilise des techniques de modélisation d'objet visant à modéliser toutes les fonctions du compteur, sans formuler d'hypothèses quant aux fonctions à prendre en charge, à la manière dont ces fonctions sont mises en œuvre et à la manière dont les données sont transportées. La spécification formelle des classes d'interface COSEM constitue une partie importante du COSEM.

Pour traiter et gérer les informations, il est indispensable d'identifier de manière unique tous les éléments de données indépendamment du fabricant. La définition de l'OBIS (Object Identification System – Système d'identification d'objets) est une autre partie essentielle du COSEM. Elle repose sur la DIN 43863-3:1997, *Electricity meters – Part 3: Tariff metering device as additional equipment for electricity meters – EDIS – Energy Data Identification System*. Le jeu de codes OBIS a été considérablement étendu au fil des années de manière à répondre à ces besoins.

Les modèles COSEM représentent le compteur comme un serveur – voir 4.7 – utilisé par les applications de client qui récupèrent des données du compteur, fournissent des informations de commande au compteur et déclenchent des actions au sein du compteur via l'accès

² ZigBee[®] est une appellation commerciale appartenant à ZigBee. Cette information est donnée à l'intention des utilisateurs du présent document et ne signifie nullement que l'IEC approuve ou recommande l'emploi exclusif du produit ainsi désigné.

contrôlé aux objets COSEM. Les clients agissent comme des agents pour les tierces parties, c'est-à-dire les processus commerciaux des acteurs du marché de l'énergie.

Les classes d'interface COSEM normalisées forment une bibliothèque extensible. Les fabricants utilisent les éléments de cette bibliothèque pour concevoir leurs produits qui satisfont à un grand nombre d'exigences.

Le serveur offre des moyens d'extraction des fonctions prises en charge, c'est-à-dire les objets COSEM instanciés. Les objets peuvent être organisés en dispositifs logiques et en associations d'applications et donnent des droits d'accès particuliers aux différents clients.

Le concept de la bibliothèque normalisée de classes d'interfaces fournit aux différents utilisateurs et fabricants un maximum de diversité tout en assurant l'interopérabilité.

IECNORM.COM: Click to view the full PDF of IEC 62056-6-2:2016

ÉCHANGE DES DONNÉES DE COMPTAGE DE L'ÉLECTRICITÉ – LA SUITE DLMS/COSEM –

Partie 6-2: Classes d'interfaces COSEM

1 Domaine d'application

La présente partie de l'IEC 62056 spécifie un modèle d'un compteur tel qu'il est vu à travers son/ses interface(s) de communication. Des blocs génériques de base sont définis à l'aide de méthodes orientées objet, sous la forme de classes d'interfaces pour modéliser les compteurs à partir d'une fonctionnalité simple jusqu'à une fonctionnalité très complexe.

Les Annexes A à F (informatives) donnent des informations supplémentaires relatives à certaines classes d'interface.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. 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 61334-4-32:1996, *Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs – Partie 4: Protocoles de communication de données – Section 32: Couche liaison de données – Contrôle de liaison logique (LLC)*

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 61334-4-511:2000, *Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs – Partie 4-511: Protocoles de communication de données – Administration de systèmes – Protocole CIASE*

IEC 61334-4-512:2001, *Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs – Partie 4-512: Protocoles de communication de données – Administration du système à l'aide du profil 61334-5-1 – MIB (Base d'Informations d'Administration)*

IEC 61334-5-1:2001, *Automatisation de la distribution à l'aide de systèmes de communication à courants porteurs – Partie 5-1: Profils des couches basses – Profil S-FSK (modulation par saut de fréquences étalées)*

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 62056-21:2002, *Équipements de mesure de l'énergie électrique – Échange des données pour la lecture des compteurs, le contrôle des tarifs et de la charge – Partie 21: Échange des données directes en local*

IEC 62056-31:1999, *Équipements de mesure de l'énergie électrique – Échange des données pour la lecture des compteurs, le contrôle des tarifs et de la charge – Partie 31: Utilisation des réseaux locaux sur paire torsadée avec signal de porteuse*

NOTE Cette Édition est référencée dans la classe d'interface "IEC twisted pair (1) setup" (class_id: 24, version: 0).

IEC 62056-3-1:2013, *Échange des données de comptage de l'électricité – La suite DLMS/COSEM – Partie 3-1: Utilisation des réseaux locaux sur paire torsadée avec signal de porteuse*

NOTE Cette Édition est référencée dans la classe d'interface "IEC twisted pair (1) setup" (class_id: 24, version: 1).

IEC 62056-46:2007, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 46: Data link layer using HDLC protocol* (disponible en anglais seulement)

IEC 62056-5-3:2016, *Échange des données de comptage de l'électricité – La suite DLMS/COSEM – Partie 5-3: Couche application DLMS/COSEM*

IEC 62056-6-1:2015, *Échange des données de comptage de l'électricité – La suite DLMS/COSEM – Partie 6-1: Système d'identification d'objets (OBIS)*

ISO/IEC 8802-2:1998, *Technologies de l'information – Télécommunications et échange d'information entre systèmes – Réseaux locaux et métropolitains – Exigences spécifiques – Partie 2: Contrôle de liaison logique*

ISO/IEC/IEEE 60559:2011, *Information technology – Microprocessor Systems – Floating-Point arithmetic* (disponible en anglais seulement)

EN 13757-2:2004, *Systèmes de communication et de télérelevé de compteurs – Partie 2: Couches physiques et couche de liaison*

EN 13757-3:2004, *Systèmes de communication et de télérelevé de compteurs – Partie 3: Couches d'application spéciale*

NOTE Cette Norme est référencée dans la classe d'interface "M-Bus client setup" version 0.

EN 13757-3:2013, *Systèmes de communication et de télérelevé de compteurs – Partie 3: Couches d'application spéciale*

NOTE Cette Norme est référencée dans la classe d'interface M-Bus client setup version 1.

EN 13757-5:2015, *Systèmes de communication et de télérelevé de compteurs – Partie 5: Transmission sans fil*

IEEE 802.15.4:2006, *Standard for 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)*

NOTE Cette Norme est également disponible sous ISO/IEC/IEEE 8802-15-4:2010.

UIT-T G.9901:2014, *SÉRIES G: SYSTÈMES ET SUPPORTS DE TRANSMISSION, SYSTÈMES ET RÉSEAUX NUMÉRIQUES – Réseaux d'accès – Réseaux intérieurs – Émetteurs-récepteurs de courants porteurs en ligne à multiplexage par répartition orthogonale de la fréquence à bande étroite – Spécification de la densité spectrale de puissance*

ITU-T G.9903 Amd. 1:2013, *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS – Access networks – In premises networks – Narrow-band*

orthogonal frequency division multiplexing power line communication transceivers for G3-PLC networks

ITU-T G.9903:2014, *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS – Access networks – In premises networks – Narrow-band orthogonal frequency division multiplexing power line communication transceivers for G3-PLC networks*

ITU-T G.9904:2012, *SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS – Access networks – In premises networks – Narrow-band orthogonal frequency division multiplexing power line communication transceivers for PRIME networks*

ETSI GSM 05.08, *Digital cellular telecommunications system (Phase 2+); Radio subsystem link control*

ANSI C12.19:1997, *IEEE 1377:1997, Utility industry end device data tables*

Les RFC suivants sont disponibles en ligne sur le site Web de l'Internet Engineering Task Force (IETF): <http://www.ietf.org/rfc/std-index.txt>, <http://www.ietf.org/rfc/>

IETF STD 51, *The Point-to-Point Protocol (PPP)*, 1994. (également RFC 1661, RFC 1662)

RFC 791, *Internet Protocol* (également: IETF STD 0005), 1981. RFC 1332, *The PPP Internet Protocol Control Protocol (IPCP)*, 1992, mise à jour par: RFC 3241. Obsolètes: RFC 1172

RFC 1570, *PPP LCP Extensions*, 1994

IETF STD 51 / RFC 1661, *The Point-to-Point Protocol (PPP)* (également: IETF STD 0051), 1994, mise à jour par: RFC 2153, obsolètes: RFC 1548

IETF STD 51 / RFC 1662, *PPP in HDLC-like Framing*, (également: IETF STD 0051), 1994, obsolètes: RFC 1549

RFC 1994, *PPP Challenge Handshake Authentication Protocol (CHAP)*, 1996. Obsolètes: RFC 1334

RFC 2433, *PPP CHAP Extension*, 1998

RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, 1998

RFC 2507, *IP Header Compression*, 1999

RFC 2759, *Microsoft PPP CHAP Extensions, Version 2*, 2000

RFC 3241, *Robust Header Compression (ROHC) over PPP*, 2002. Mises à jour: RFC1332

RFC 3513, *Internet Protocol Version 6 (IPv6) Addressing Architecture*, 2003

RFC 3544, *IP Header Compression over PPP*, 2003

RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*, 2007

Point-to-Point (PPP) Protocol Field Assignments. Base de données en ligne. Disponible à l'adresse:

<http://www.iana.org/assignments/ppp-numbers/ppp-numbers.xhtml>

3 Termes, définitions et abréviations

Pour les besoins du présent document, les termes, définitions et abréviations suivants s'appliquent.

3.1 Termes et définitions liés au processus de transfert d'Image (voir 5.3.6)

3.1.1

Image

données binaires de la taille spécifiée

Note 1 à l'article: Une Image peut être perçue comme un conteneur. Elle peut être composée d'un ou de plusieurs éléments (*image_to_activate*) qui sont transférés, vérifiés et activés ensemble.

3.1.2

ImageSize

taille de l'ensemble de l'Image à transférer

Note 1 à l'article: ImageSize est exprimé en octets.

3.1.3

ImageBlock

partie de l'Image de la taille ImageBlockSize

Note 1 à l'article: L'Image est transférée dans ImageBlocks. Chaque bloc est identifié par son ImageBlockNumber.

3.1.4

ImageBlockSize

taille de l'ImageBlock en octets

3.1.5

ImageBlockNumber

identifiant d'un ImageBlock. Les ImageBlocks sont numérotés séquentiellement, à partir de 0.

La signification des définitions ci-dessus est représentée à la Figure 1.

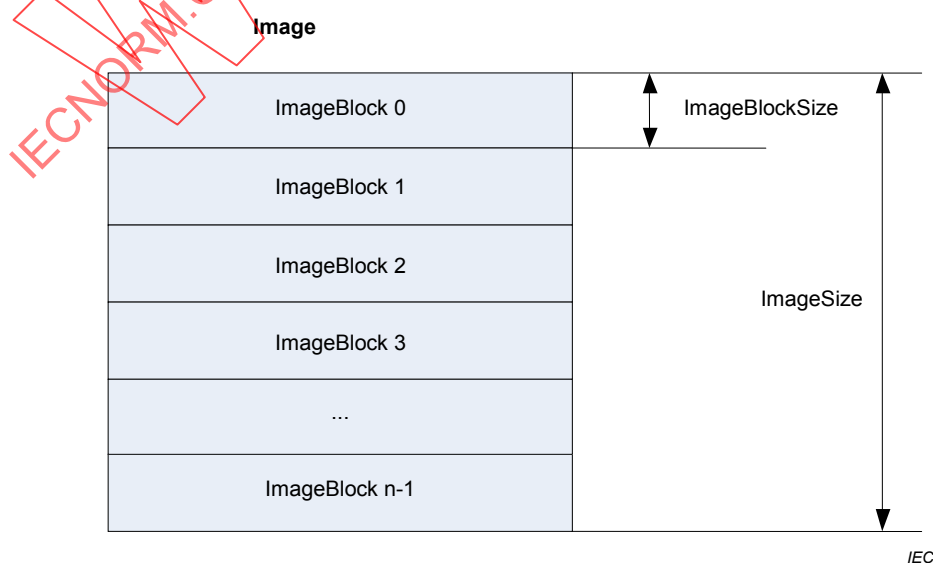


Figure 1 – Signification des définitions concernant l'Image

3.2 Termes et définitions liés aux classes S-FSK PLC setup (voir 5.8)

3.2.1

initiateur

élément utilisateur d'une entité d'application de gestion de système (System Management Application Entity, SMAE) cliente

Note 1 à l'article: L'initiateur utilise l'ASE CIASE et xDLMS et est identifié par son titre système.

[SOURCE: IEC 61334-4-511:2000, 3.8.1]

3.2.2

initiateur actif

initiateur qui émet ou vient d'émettre une demande Register CIASE lorsque le serveur est en état non configuré

[SOURCE: IEC 61334-4-511:2000, 3.9.1]

3.2.3

nouveau système

système serveur qui est dans l'état non configuré: son adresse MAC correspond à l'adresse NEW

[SOURCE: IEC 61334-4-511:2000, 3.9.3]

3.2.4

titre de nouveau système

titre système d'un nouveau système

Note 1 à l'article: Il s'agit du titre système dans le cas d'un système qui est dans le nouvel état.

[SOURCE: IEC 61334-4-511:2000, 3.9.4]

3.2.5

système enregistré

système serveur ayant une adresse MAC individuelle valide (donc différente de "NEW Address" voir l'IEC 61334-5-1:2001: Contrôle d'accès au support)

[SOURCE: IEC 61334-4-511:2000, 3.9.5]

3.2.6

système de compte-rendu

système serveur qui émet un DiscoverReport

[SOURCE: IEC 61334-4-511:2000, 3.9.6, modifié pour corriger une erreur dans l'IEC 61334-4-511:2000]

3.2.7

sous-intervalle

temps nécessaire pour émettre deux octets par la couche physique

Note 1 à l'article: Les intervalles de temps sont divisés en sous-intervalles dans le mode RepeaterCall de la couche physique.

3.2.8

intervalle de temps

temps nécessaire pour émettre une trame physique

Note 1 à l'article: Comme spécifié au 3.3.1 de l'IEC 61334-5-1, une trame physique comprend un préambule de 2 octets, un délimiteur de sous-trame de début de 2 octets, une PSDU de 38 octets et une pause de 3 octets.

3.3 Termes et définitions liés aux classes d'interface PRIME NB OFDM PLC setup (voir 5.10)

Définitions liées à la couche physique

3.3.1

nœud de base

nœud principal qui commande et gère les ressources d'un sous-réseau

[SOURCE: ITU-T G.9904:2012, 3.2.1]

3.3.2

emplacement d'étiquette

emplacement de l'étiquette PDU dans la trame

[SOURCE: ITU-T G.9904:2012, 3.2.2]

3.3.3

nœud

élément d'un sous-réseau, qui est en mesure d'émettre et de recevoir à partir d'autres éléments du sous-réseau

[SOURCE: ITU-T G.9904:2012, 3.2.9]

3.3.4

enregistrement

processus par lequel un nœud de service est accepté comme membre du sous-réseau, auquel un LNID est attribué

[SOURCE: ITU-T G.9904:2012, 3.2.12]

3.3.5

nœud de service

nœud d'un sous-réseau qui n'est pas un nœud de base

[SOURCE: ITU-T G.9904:2012, 3.2.13]

3.3.6

sous-réseau

ensemble d'éléments qui peuvent communiquer en se conformant à la présente spécification et partager un seul nœud de base

[SOURCE: ITU-T G.9904:2012, 3.2.15]

Définitions liées à la couche MAC

3.3.7

état déconnecté <d'un nœud de service>

état fonctionnel initial de tous les nœuds de service. Lorsqu'il est déconnecté, un nœud de service ne peut pas communiquer des données ni commuter les données d'autres nœuds. Sa fonction principale consiste à rechercher un sous-réseau et de tenter de s'y enregistrer

[SOURCE: ITU-T G.9904:2012, 8.1]

3.3.8

état final <d'un nœud de service>

lorsqu'il est dans cet état fonctionnel, un nœud de service peut établir des connexions et communiquer des données, mais il ne peut pas commuter des données d'autres nœuds

[SOURCE: ITU-T G.9904:2012, 8.1]

3.3.9

état de commutation <d'un nœud de service>

lorsqu'il est dans cet état fonctionnel, un nœud de service peut réaliser toutes les fonctions finales. De plus, il peut transférer des données entre d'autres nœuds du même sous-réseau. Il s'agit d'un point de branchement sur la structure de l'arborescence

[SOURCE: ITU-T G.9904:2012, 8.1]

3.3.10

promotion

processus par lequel un nœud de service est qualifié pour commuter (répéter, transférer) le trafic de données provenant d'autres nœuds et faire office de point de branchement sur la structure de l'arborescence du sous-réseau. Une promotion réussie représente la transition entre l'état final et l'état de commutation. Lorsqu'un nœud de service est à l'état déconnecté (Disconnected), il ne peut pas passer directement à l'état de commutation

[SOURCE: ITU-T G.9904:2012, 8.1]

3.3.11

rétrogradation

processus par lequel un nœud de service cesse d'être un point de branchement sur la structure de l'arborescence de sous-réseau. Une rétrogradation réussie représente la transition entre l'état de commutation et l'état final

[SOURCE: ITU-T G.9904:2012, 8.1]

3.4 Termes et définitions liés à ZigBee® (voir 5.12)

NOTE Les termes suivis d'un astérisque (*) sont issus de la spécification ZigBee®.

3.4.1

CAD

(dispositif d'accès grand public) dispositif de passerelle ZigBee® qui agit comme un IHD à l'intérieur du réseau ZigBee®, mais qui est dotée d'une connexion supplémentaire à un autre réseau (c'est-à-dire WiFi)

Note 1 à l'article: L'abréviation «CAD» est dérivée du terme anglais développé correspondant «consumer access device».

3.4.2

IHD

(dispositif d'affichage à domicile) dispositif doté d'un écran pour l'affichage des informations énergétiques au consommateur

Note 1 à l'article: L'abréviation «IHD» est dérivée du terme anglais développé correspondant «in home display».

3.4.3

code d'installation

clé de liaison préconfigurée (PCLK) hachée (via MMO) fournie à un tiers de confiance par l'intermédiaire des communications hors bande

Note 1 à l'article: Un nouveau dispositif qui souhaite intégrer le réseau nécessite d'envoyer ce code d'installation au tiers de confiance, permettant à ce dernier d'exécuter le processus d'intégration, en utilisant ce code d'installation comme partie intégrante des informations de sécurité.

3.4.4

clé de liaison*

clé partagée exclusivement entre deux, et seulement deux, entités de couche d'application homologues au sein d'un réseau PAN

3.4.5

adresse MAC/adresse IEEE

termes utilisés comme synonymes pour représenter le code EUI-64 attribué à ZigBee® Radio

3.4.6

ZigBee®

spécification d'une suite de protocoles de communication de haut niveau utilisée pour créer des réseaux personnels conçus à partir de petites radios basse puissance numériques. ZigBee® repose sur une norme IEEE 802.15. Malgré leur faible puissance, les dispositifs ZigBee® transmettent souvent des données sur de plus longues distances, passant par des dispositifs intermédiaires pour atteindre les plus éloignés, créant ainsi un réseau en mailles

3.4.7

client ZigBee®

s'apparente au rôle du client DLMS/COSEM

Note 1 à l'article: Pour mieux comprendre l'interaction entre le client et le serveur, il convient de lire la spécification ZigBee® PRO.

3.4.8

coordonateur ZigBee® *

coordonateur de réseau personnel IEEE 802.15.4-2003 qui est le principal contrôleur d'un réseau IEEE 802.15.4-2003 chargé de la formation du réseau. Le coordonnateur de réseau personnel doit être un dispositif pleine fonction (FFD – Full Function Device)

3.4.9

cluster ZigBee®

ensemble de types de message liés à une certaine fonction de dispositif (par exemple, comptage, commande de ballast)

3.4.10

miroir ZigBee®

dispositif qui renvoie par écho les données publiées par un dispositif ZigBee® alimenté par pile, permettant aux autres acteurs du réseau d'obtenir les données lorsque le dispositif alimenté par pile est indisponible pour des raisons d'économie d'énergie

3.4.11

ZigBee® PRO

autre nom du protocole ZigBee® 2007, qui est à présent la nouvelle édition de pile, contient deux profils de pile: le profil de pile 1 (tout simplement appelé ZigBee®), pour les utilisations domestiques et commerciales légères, et le profil de pile 2 (appelé ZigBee® PRO). ZigBee® PRO offre plus de fonctionnalités, comme la multidiffusion, le routage à origines multiples et destination unique et la haute sécurité avec SKKE (Symmetric-Key Key Exchange), ZigBee® (profil de pile 1) offrant une plus petite empreinte dans la mémoire RAM et Flash. Les deux protocoles offrent un réseau de mailles et fonctionnent avec tous les profils d'application ZigBee®

3.4.12

routeur ZigBee® *

FFD IEEE 802.15.4-2003 participant à un réseau ZigBee®, qui n'est pas le coordonnateur ZigBee®, mais qui agit comme un coordonnateur IEEE 802.15.4-2003 dans son espace de fonctionnement personnel, capable d'acheminer des messages entre des dispositifs et des associations de prise en charge

3.4.13

serveur ZigBee®

s'apparente au rôle du serveur DLMS/COSEM

Note 1 à l'article: Pour mieux comprendre l'interaction entre le client et le serveur, il convient de lire la spécification ZigBee® PRO.

3.4.14

tiers de confiance ZigBee® *

dispositif digne de confiance au sein d'un réseau ZigBee® pour distribuer les clés pour les besoins de la gestion de configuration du réseau et de l'application de bout en bout

3.5 Abréviations

Abréviation	Explication
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Network
AA	Association d'applications
AARE	A-Associate Response (Réponse A-Associate) – une APDU de l'ACSE
AARQ	A-Associate Request (Demande A-Associate) – une APDU de l'ACSE
ACSE	Association Control Service Element (Élément de service de contrôle d'association)
ADP	Adresse de la station primaire
ADS	Adresse de la station secondaire
CAG	Commande automatique de gain
AL	Application Layer (Couche application)
AP	Application Process (Processus d'application)
APDU	Application Protocol Data Unit (Unité de données de protocole d'application)
APS	Application Support Sublayer (Sous-couche de support d'application) (terme ZigBee®)
ASE	Application Service Element (Élément de service application)
A-XDR	Adapted Extended Data Representation (Représentation de données étendues adaptée) (IEC 61334-6)
base_name	Attribut short_name correspondant au premier attribut ("logical_name") d'un objet COSEM
BCD	Binary Coded Decimal (décimal codé binaire)
TEB	Taux d'erreurs sur les bits
CBCP	CallBack Control Protocol (PPP) (protocole de contrôle de rappel)
CC	Current Credit (crédit courant) (profil S-FSK PLC)
CENELEC	Comité Européen de Normalisation Électrotechnique
CHAP	Challenge Handshake Authentication Protocol (Protocole d'authentification par challenge)
CIASE	Configuration Initiation Application Service Element (élément de service d'application d'initiation de configuration) (profil S-FSK PLC)
class_id	Code d'identification de classe d'interface
CLI	Calling Line Identity (identité de la ligne appelante)
COSEM	Companion Specification for Energy Metering (Spécification d'accompagnement pour le comptage de l'énergie)
Objet COSEM	Instance d'une classe d'interface COSEM
CRC	Contrôle de redondance cyclique
CSD	Circuit Switched Data (données à commutation de circuit)
CSMA	Carrier Sense Multiple Access (accès multiple à détection de porteuse)
CtoS	Client to Server challenge (Défi du client au serveur)
CU	Currently Unused (actuellement non utilisé)
DC	Delta Credit (profil S-FSK PLC)
DHCP	Dynamic Host Control Protocol (protocole de commande d'hôte dynamique)
DIB	Data Information Block (Bloc d'informations de données) (M-Bus)
DIF	Data Information Field (Champ d'informations de données) (M-Bus)
DL	Data Link (Liaison de données)

Abréviation	Explication
DLMS	Device Language Message Specification (Spécification de message de langage de dispositif)
DLMS UA	DLMS User Association (Association des utilisateurs de la DLMS)
DNS	Domain Name System (Serveur de noms de domaine)
DSCP	Differentiated Services Code Point (Code d'accès aux services différenciés)
DSSID	Direct Switch ID (ID de commutation directe)
EAP	Extensible Authentication Protocol (Protocole d'authentification extensible)
EDGE	Enhanced Data rates for GSM Evolution (GSM à débit amélioré)
EUI-48	48-bit Extended Unique Identifier (Identifiant unique étendu 48 bits)
EUI-64	64-bit Extended Unique Identifier (Identifiant unique étendu 48 bits)
FCC	Federal Communications Commission
FFD	Full-Function Device (dispositif pleinement fonctionnel)
FIFO	First-In-First-Out (Premier entré, premier sorti)
FTP	File Transfer Protocol (Protocole de transfert de fichiers)
GCM	Galois/Counter Mode (Mode Galois/Compteur) – un algorithme de chiffrement authentifié avec données associées
GMT	Greenwich Mean Time (Temps moyen de Greenwich) Remplacé par Temps universel coordonné (TUC).
GPRS	General Packet Radio Service (service général de radiocommunication par paquets)
GPS	Global Positioning System (Système de positionnement global)
GSM	Global System for Mobile Communications (système global de communications mobiles)
HAN	Home Area Network (Réseau local domestique)
HART	Highway Addressable Remote Transducer see http://www.hartcomm.org/ (en rapport avec la classe d'interfaces Gestionnaire de capteur)
HDLC	High-level Data Link Control (Commande de liaison de données à haut niveau)
HES	Head End System (Système tête de réseau)
HHT	Hand Held Terminal (Terminal portable)
HLS	High Level Security Authentication (Authentification de niveau de sécurité élevé)
HSDPA	High-Speed Downlink Packet Access (Accès par paquets en liaison descendante haut débit)
IANA	Internet Assigned Numbers Authority
BA	Information Base (Base d'informations)
IC	Interface Class (Classe d'interface) (COSEM)
IC	Initial credit (profil S-FSK PLC)
IEC	International Electrotechnical Commission (Commission Électrotechnique Internationale)
IEEE	Institute of Electrical and Electronics Engineers (Institut des ingénieurs électriciens et électroniciens)
IETF	Internet Engineering Task Force (Groupe de travail d'ingénierie Internet)
IPCP	Internet Protocol Control Protocol (Protocole de contrôle du Protocole Internet)
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
ISO	International Organization for Standardization (Organisation internationale de normalisation)
FAI	Fournisseur d'accès Internet
IT	Information Technology (Technologie de l'information)
UIT-T	Union Internationale des Télécommunications – Télécommunication
KEK	Key Encryption Key (clé de chiffrement de clé)
LA	Local Area (zone locale)
LAC	Local Area Code (code de zone locale)
LAN	Local Area Network (réseau local)
LCID	Local Connection Identifier (Identifiant de connexion local)
LCP	Link Control Protocol (Protocole de commande de liaison)
LDN	Logical Device Name (Nom de dispositif logique)
LLC	Logical Link Control (Commande de liaison logique) (sous-couche)

Abréviation	Explication
LLS	Low Level Security (Sécurité à faible niveau)
LN	Logical Name (Nom logique)
LNID	Local Node Identifier (Identifiant de nœud local)
LOADng	6LoWPAN Ad Hoc On-Demand Distance Vector Routing Next Generation (LOADng)
LQI	Link Quality Indicator (Indicateur de qualité de liaison) (terme ZigBee®)
LSB	Least Significant Bit (Bit de poids faible)
LSID	Local Switch Identifier (Identifiant de commutation local)
LTE	Long Term Evolution (Évolution à long terme) (Communication sans fil)
m	Mandatory (obligatoire)
MAC	Medium access control (Contrôle d'accès au support)
M-Bus	Meter Bus
MD5	Message Digest Algorithm 5 (Algorithme de condensation de message 5)
MIB	Management Information Base (Base d'informations de gestion) (profil S-FSK PLC)
MID	Measuring Instruments Directive (Directives sur les instruments de mesure 2004/22/CE du Parlement européen et du Conseil)
MMO	Hachage Matyas-Meyer-Oseas (terme ZigBee®)
MPAN	(terme britannique) Meter Point Access Number – référence de l'emplacement du compteur électrique sur le réseau de distribution d'électricité.
MPDU	MAC Protocol Data Unit (unité de données de protocole MAC)
MSB	Most Significant Bit (bit de poids fort)
MSDU	MAC Service Data Unit (Unité de données de service MAC)
MT	Mobile Termination (Terminaison mobile)
NB	Narrow-band (Bande étroite)
ND	Neighbour Discovery (Découverte de voisins)
NTP	Protocole NTP (Network Time Protocol)
o	Optional (facultatif)
OBIS	OBject Identification System (Système d'identification d'objet)
OFDM	Orthogonal Frequency Division Multiplexing (multiplexage par répartition orthogonale de la fréquence)
OTA	Over the Air (par liaison radio) – Fait référence à la mise à jour de firmware à l'aide de ZigBee®
PAN	Personal Area Network (réseau local personnel) (terme utilisé en relation avec G3-PLC ¹) et ZigBee®
Pad	Remplissage
PAP	Password Authentication Protocol (Protocole d'authentification de mot de passe)
PCLK	Pre-Configured Link Key (clé de liaison préconfigurée) (terme ZigBee®)
PDU	Protocol Data Unit (Unité de données de protocole)
PhL, PHY	Couche physique
PIB	PLC Information Base (Base d'informations PLC)
PIN	Personal Identity Number (Numéro d'identification personnel)
PLC	Power Line Carrier (Courant porteur sur ligne d'énergie)
PLMN	Public Land Mobile Network (Réseau mobile terrestre public)
PNPDU	Promotion Needed PDU
POS	Personal Operating Space (Espace de fonctionnement personnel) (ZigBee®)
PPDU	Physical Protocol Data Unit (Unité de données de protocole de Couche Physique)
PPP	Point-to-Point Protocol (Protocole point à point)
RTPC	Réseau téléphonique public commuté
REJ PDU	Reject Protocol Data Unit (Unité de données de protocole de rejet)
RFC	Request for Comments (demande de commentaires). Document publié par l'Internet Engineering Task Force
RFD	Reduced Function Device (Dispositif à fonction réduite)
ROHC	Robust Header Compression (Compression robuste des en-têtes)
RREP	Route Reply (Réponse de chemin)

Abréviation	Explication
RREQ	Route Request (Demande de chemin)
RRER	Route Error (Erreur de chemin)
RSSI	Receive Signal Strength Indication (indication de l'intensité du signal reçu) (terme ZigBee®)
SAP	Service Access Point (point d'accès au service)
SAS	Startup Attribute Set (ensemble d'attributs de démarrage) (terme ZigBee®)
SCP	Shared Contention Period (Période de conflit partagé)
SE	Smart Energy (Réseau intelligent)
SEP	Smart Energy Profile (Profil de réseau intelligent) (terme ZigBee®)
S-FSK	Spread – Frequency Shift Keying (modulation par saut de fréquences étalées)
SGERG88	Méthode de calcul de la compressibilité (comptage de gaz)
SHA	Algorithme de hachage sécurisé
SID	Switch identifier (Identifiant de commutation)
SMS	Short Message Service (Service de messages courts)
SMTP	Simple Mail Transfer Protocol (Protocole simple de transfert de courrier)
SN	Short Name (Nom court)
SNA	SubNetwork Address (Adresse de sous-réseau)
SSCS	Service Specific Convergence Layer (Couche de convergence spécifique au service)
StoC	Server to Client Challenge (Défi du serveur au client)
TAB	Dans le cas des profils EURIDIS sans DLMS et sans DLMS/COSEM: code de données. Dans le cas des profils utilisant DLMS ou DLMS/COSEM: valeur à laquelle l'appareil est sensibilisé au Discover
TABi	Liste de champ TAB
TCC	Transmission Control Code (Code de contrôle de transmission) (IPv4)
TCP	Transmission Control Protocol (protocole de contrôle de transmission)
TFTP	Trivial File Transfer Protocol
TOU	Time Of Use (Temps d'utilisation)
TTL	Durée de vie
UDP	User Datagram Protocol (protocole datagramme d'utilisateur)
UMTS	Universal Mobile Telecommunications System (système universel de télécommunications mobiles)
UNC	Unconfigured (Non configuré (profil S-FSK PLC)
TUC	Temps universel coordonné
VIB	Value Information Block (Bloc d'informations sur les valeurs) (M-Bus)
VIF	Value Information Field (Champ d'informations sur les valeurs) (M-Bus)
VZ	Compteur de périodes de facturation (de l'allemand <i>VorwertZähler</i> , voir DIN 43863-3)
réactivation	déclenche le compteur à connecter au réseau de communication mis à la disposition du client (HES, par exemple)
WAN	Wide Area Network (réseau étendu)
wM-Bus	Wireless M-Bus (M-Bus sans fil)
ZTC	ZigBee® Trust Center
¹⁾ Dans le cas de la technologie G3-PLC, PAN peut être défini comme étant un réseau local PLC (PLC Area Network).	

4 Principes de base

4.1 Généralités

Le présent Article 4 décrit les principes de base sur lesquels sont construites les classes d'interfaces COSEM (les IC). Il donne aussi une courte vue d'ensemble sur la manière d'utiliser les objets d'interfaces – instanciations des IC – à des fins de communication. Les systèmes de collecte de données et les équipements de comptage issus de différents

vendeurs, conformes à ces spécifications, peuvent échanger des données d'une manière interopérable.

Pour les besoins de spécification, la présente norme utilise la technique de modélisation d'objets.

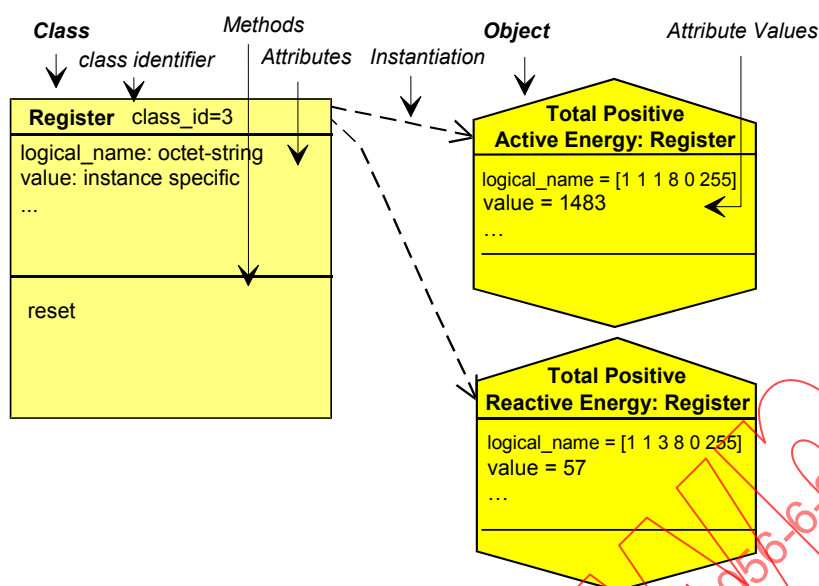
Un objet est un ensemble d'attributs et de méthodes. Les attributs représentent les caractéristiques d'un objet. La valeur d'un attribut peut affecter le comportement d'un objet. Le premier attribut d'un objet est le *logical_name* (c'est-à-dire nom logique). Il fait partie de l'identification de l'objet. Un objet peut proposer un certain nombre de méthodes pour examiner ou modifier les valeurs des attributs.

Les objets qui partagent des caractéristiques communes sont généralisés en une classe d'interface, identifiée par un *class_id*. Au sein d'une IC spécifique, les caractéristiques communes (attributs et méthodes) sont décrites une seule fois pour tous les objets. Les instanciations des IC sont appelées objets d'interfaces COSEM.

Les fabricants peuvent ajouter des méthodes et attributs propriétaires à n'importe quel objet; voir 4.2.

La Figure 2 représente ces termes par un exemple:

Withdrawing the full PDF of IEC 62056-6-2:2016
IECNORM.COM: Click to view the full PDF of IEC 62056-6-2:2016



Anglais	Français
Class	Classe
class identifier	identifiant de classe
Methods	Méthodes
Attributes	Attributs
Instantiation	Instanciation
Object	Objet
Attribute values	Valeurs d'attribut
Total positive active energy	Énergie active positive totale
Total positive reactive energy	Énergie réactive positive totale
Instance specific	Spécifique à l'instance

Figure 2 – Une classe d'interfaces et ses instances

L'IC "Register" est formé en combinant des caractéristiques nécessaires pour modéliser le comportement d'un registre générique (contenant des informations mesurées ou statiques) tel que vu du client (système de collecte de données, terminal portable). Le contenu du registre est identifié par l'attribut *logical_name*. Le *logical_name* contient un identifiant OBIS (Voir l'IEC 62056-6-1:). Le contenu (dynamique) réel du registre est contenu dans son attribut *value* (c'est-à-dire valeur).

Le fait de définir un compteur spécifique signifie définir plusieurs objets spécifiques. Dans l'exemple de la Figure 2, le compteur contient deux registres; c'est-à-dire que deux instances spécifiques de l'IC "Register" sont instanciées. Par le truchement de l'instanciation, un objet COSEM devient un "registre d'énergie active, positive, totale" alors que l'autre devient un "registre de puissance réactive, positive, totale".

NOTE Les objets d'interfaces COSEM (instances des IC COSEM) représentent le comportement du compteur tel que vu de "l'extérieur". Par conséquent, la modification de la valeur d'un attribut (la réinitialisation de l'attribut *value* d'un registre, par exemple) est toujours déclenchée de l'extérieur. Les modifications des attributs déclenchées de l'intérieur (la mise à jour de l'attribut *value* d'un registre, par exemple) ne sont pas décrites dans le présent modèle.

4.2 Méthodes de référencement

Les attributs et les méthodes des objets COSEM peuvent être référencés de deux manières différentes:

À l'aide de noms logiques (référencement par LN): Dans ce cas, les attributs et les méthodes sont référencés via l'identifiant de l'instance d'objet COSEM à laquelle ils appartiennent.

La référence pour un attribut est: class_id, valeur de l'attribut *logical_name*, attribute_index.

La référence pour une méthode est: class_id, valeur de l'attribut *logical_name*, method_index, où:

- attribute_index est utilisé comme l'identifiant de l'attribut requis. Les indices d'attribut sont spécifiés dans la définition de chaque IC. Ce sont des nombres positifs commençant à 1. Des attributs propriétaires peuvent être ajoutés: ceux-ci doivent être identifiés par des nombres négatifs;
- method_index est utilisé comme l'identifiant de la méthode requise. Les indices de méthode sont spécifiés dans la définition de chaque IC. Ce sont des nombres positifs commençant à 1. Des méthodes propriétaires peuvent être ajoutées: celles-ci doivent être identifiées par des nombres négatifs.

À l'aide de noms courts (référencement par SN): Cette sorte de référencement est destinée à être utilisée dans des dispositifs simples. Dans ce cas, chaque attribut et chaque méthode d'un objet COSEM sont identifiés par un nombre entier de 13 bits. La syntaxe pour le nom court est la même que celle du nom d'une variable nommée selon la DLMS. Voir l'IEC 61334-4-41:1996 et IEC 62056-5-3:., Article 8.

4.3 Base_name réservés pour des objets COSEM spéciaux

Afin de faciliter l'accès à des dispositifs à l'aide du référencement par SN, certains short_name sont réservés comme des base_name pour des objets COSEM spéciaux. La plage pour les base_name réservés est comprise entre 0xFA00 et 0xFFFF8. Les base_name spécifiques suivants sont définis (voir le Tableau 1).

Tableau 1 – Base_name réservés pour le référencement par SN

Base_name (objectName)	Objet COSEM
0xFA00	SN d'association
0xFB00	Script table (instanciation: "Tableau de scripts" Broadcast)
0xFC00	SAP assignment (Affectation de SAP)
0xFD00	Objet "Data" (Donnée) ou "Register" (Registre) contenant le "Nom d'appareil logique COSEM" dans l'attribut "value"

4.4 Notation pour la description de classes

Le présent paragraphe décrit la notation utilisée pour définir les IC.

Un texte court décrit la fonctionnalité et l'application de l'IC. Un tableau donne une vue d'ensemble de l'IC, y compris le nom de classe, les attributs et les méthodes. Chaque attribut et chaque méthode doivent être décrits dans le détail. Le modèle est montré ci-dessous.

Nom de la classe	Cardinalité	class_id, version			
Attribut(s)	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. ... (...)	...				x + 0x...
3. ... (...)	...				x + 0x...
Méthodes spécifiques (si requises)	m/o				
1.	...				x + 0x...
2.	...				x + 0x...
3.	...				x + 0x...

Nom de la classe	Décrit la classe d'interfaces ("Register", "Clock", "Profile generic...", par exemple). NOTE 1 Les noms de classes d'interface sont indiqués entre guillemets.
Cardinalité	Spécifie le nombre d'instances de l'IC au sein d'un dispositif logique (Voir 4.8). <i>value</i> L'IC doit être instanciée exactement " <i>value</i> " fois. <i>min...max.</i> L'IC doit être instanciée au moins " <i>min.</i> " fois et au plus " <i>max.</i> " fois. Si <i>min.</i> est zéro (0), l'IC est facultative, autrement (<i>min.</i> > 0) " <i>min.</i> " instanciations de l'IC sont obligatoires.
class_id	Code d'identification de l'IC (plage de 0 à 65 535). Le class_id de chaque objet est récupéré avec le nom logique par lecture de l'attribut <i>object_list</i> d'un objet "LN d'Association"/"SN d'Association". – Les class_id compris entre 0 et 8 191 sont réservés pour être spécifiés par la DLMS UA. – Les class_id compris entre 8 192 et 32 767 sont réservés pour des IC spécifiques à un fabricant. – Les class_id compris entre 32 768 et 65 535 sont réservés pour des IC spécifiques à un groupe d'utilisateurs. La DLMS UA réserve le droit d'attribuer les plages aux fabricants individuels ou aux groupes d'utilisateurs.
Version	Code d'identification de la version de l'IC. La version de chaque objet est récupérée avec le class_id et le nom logique par lecture de l'attribut <i>object_list</i> d'un objet "LN d'Association"/"SN d'Association". Au sein d'un même dispositif logique, toutes les instances d'une certaine IC doivent être de la même version.
Attribut(s)	Spécifie les attributs qui appartiennent à l'IC. (<i>dyn.</i>) Classifie un attribut qui porte une valeur traitée, qui est mis à jour par le compteur lui-même. (<i>static</i>) Classifie un attribut, qui n'est pas mis à jour par le compteur lui-même (par exemple: données de configuration). NOTE 2 Certains attributs peuvent être soit statiques, soit dynamiques en fonction de l'application. Dans ces cas, cette propriété n'est pas indiquée. NOTE 3 Les noms d'attribut utilisent un trait de soulignement. Lorsqu'ils sont mentionnés dans le texte, ils apparaissent en <i>italique</i> . Exemple: <i>logical_name</i>

logical_name	chaîne d'octets	Il s'agit toujours du premier attribut d'une IC. Il identifie l'instanciation (objet COSEM) de cette IC. La valeur du <i>logical_name</i> est conforme à l'OBIS. Voir l'Article 6 et l'IEC 62056-6-1.
Type de données		Définit le type de données d'un attribut; voir 4.5.
Min.		Spécifie si l'attribut a une valeur minimale. x L'attribut a une valeur minimale. <empty> L'attribut n'a pas de valeur minimale.
Max.		Définit si l'attribut a une valeur maximale. x L'attribut a une valeur maximale. <empty> L'attribut n'a pas de valeur maximale.
Def.		Spécifie si l'attribut a une valeur par défaut. Il s'agit de la valeur de l'attribut après réinitialisation. x L'attribut a une valeur par défaut. <empty> La valeur par défaut n'est pas définie par la spécification de l'IC.
Nom court		Lorsque le référencement par nom court (SN) est utilisé, chaque attribut et chaque méthode des instances d'un objet doivent être mis en correspondance avec des noms courts. Le base_name x de chaque instance d'objet est la variable nommée selon la DLMS à laquelle l'attribut nom logique est mis en correspondance. Il est sélectionné dans la phase de mise en œuvre. La définition de l'IC spécifie les décalages pour les autres attributs et pour les méthodes.
Méthode(s) spécifique(s)		Fournit une liste de méthodes spécifiques qui appartiennent à l'objet. Nom de méthode () La méthode doit être décrite dans la sous-section "Description de méthode". NOTE 4 Les noms de méthode utilisent un trait de soulignement. Lorsqu'ils sont mentionnés dans le texte, ils apparaissent en <i>italique</i> . Exemple: <i>add_object</i> .
m/o		Définit si la méthode est obligatoire ou facultative. m (<i>mandatory – obligatoire</i>) La méthode est obligatoire. o (<i>optional – facultatif</i>) La méthode est facultative.

Description d'attribut

Décrit chaque attribut avec son type de données (si le type de données n'est pas simple), son format de données et ses propriétés (valeurs minimum, maximum et valeurs par défaut).

Description de la méthode

Décrit chaque méthode et le comportement invoqué de l'objet ou des objets COSEM instanciés.

NOTE Les services permettant d'accéder aux attributs ou aux méthodes par le protocole sont spécifiés dans l'IEC 62056-5-3, 6.6 à 6.16.

Accès sélectif

Les services liés à l'attribut xDLMS font en général référence à l'ensemble de l'attribut. Cependant, pour certains attributs, un accès sélectif à seulement une partie de l'attribut peut

être assuré. La partie de l'attribut est identifiée par des paramètres d'accès sélectif spécifiques. Ceux-ci sont définis comme partie intégrante de la spécification de l'attribut.

4.5 Types de données communs

Le Tableau 2 contient la liste des types de données utilisables pour les attributs des objets COSEM.

Tableau 2 – Types de données communs

Description de type	Étiquette ^a	Définition	Plage de valeurs
-- types de données simples			
null-data	[0]		
boolean	[3]	booléen	TRUE ou FALSE
bit-string	[4]	Séquence ordonnée de valeurs booléennes	
double-long	[5]	Integer32	-2 147 483 648... 2 147 483 647
double-long-unsigned	[6]	Unsigned32	0...4 294 967 295
	[7]	Étiquette du type "floating-point" dans l'IEC 61334-4-41:1996, non utilisable dans la DLMS/COSEM. Voir les étiquettes [23] et [24]	
octet-string	[9]	Séquence ordonnée d'octets (multiple de huit bits)	
visible-string	[10]	Séquence ordonnée de caractères ASCII	
	[11]	Étiquette du type "time" dans l'IEC 61334-4-41:1996, non utilisable dans la DLMS/COSEM. Voir étiquette [27]	
utf8-string	[12]	Séquence ordonnée de caractères codés en UTF-8	
bcd	[13]	décimal codé binaire	
integer	[15]	Integer8	-128...127
long	[16]	Integer16	-32 768...32 767
unsigned	[17]	Unsigned8	0...255
long-unsigned	[18]	Unsigned16	0...65 535
long64	[20]	Integer64	- 2 ⁶³ ...2 ⁶³ -1
long64-unsigned	[21]	Unsigned64	0...2 ⁶⁴ -1
enum	[22]	Les éléments du type enumeration (énumération) sont définis dans la section " <i>Description d'attribut</i> " ou " <i>Description de la méthode</i> " d'une spécification IC COSEM.	0...255
float32	[23]	OCTET STRING (SIZE(4))	Pour la mise en forme, voir 4.6.2.
float64	[24]	OCTET STRING (SIZE(8))	
date-time	[25]	OCTET STRING SIZE(12))	Pour la mise en forme, voir 4.6.1.
date	[26]	OCTET STRING (SIZE(5))	
time	[27]	OCTET STRING (SIZE(4))	
-- types de données complexes			
array	[1]	Les éléments du type array (tableau) sont définis dans la section " <i>Attribut</i> " ou " <i>Description de la méthode</i> " d'une spécification IC COSEM.	
structure	[2]	Les éléments du type structure sont définis dans la section " <i>Attribut</i> " ou " <i>Description de la méthode</i> " d'une spécification IC COSEM.	
compact array	[19]	Offre une alternative, codage compact des données complexes.	

Description de type	Étiquette ^a	Définition	Plage de valeurs
-- CHOICE		Pour certains attributs des objets d'interface COSEM, le type de données peut être choisi au moment de l'instanciation, dans la phase de mise en œuvre du serveur COSEM. Le serveur doit toujours renvoyer le type de données et la valeur de chaque attribut, ce qui, avec le nom logique, assure une interprétation non ambiguë. La liste des types de données possibles est définie dans la section "Description d'attribut" dans une spécification IC COSEM.	
^a Les étiquettes sont définies dans l'IEC 62056-5-3; Article 8.			

4.6 Formats de données

4.6.1 Formats des date et heure

Les informations de date et d'heure peuvent être représentées à l'aide du type de données *octet-string*.

NOTE 1 Dans ce cas, le codage inclut l'étiquette du type de données *octet-string*, la longueur de la chaîne d'octets et les éléments de *date*, *heure* et/ou *date-heure*, selon le cas.

Les informations de date et d'heure peuvent également être représentées à l'aide des types de données *date*, *time* et *date-time*.

NOTE 2 Dans ces cas, le codage inclut uniquement l'étiquette des types de données *date*, *time* ou *date-time*, selon le cas, et les éléments de date, d'heure ou date-heure.

NOTE 3 Les spécifications (SIZE ()) sont applicables uniquement lorsque la date, l'heure ou la date-heure sont représentées par les types de données *date*, *time* ou *date-time*.

date	OCTET STRING (SIZE(5)) <pre>{ year highbyte, year lowbyte, month, day of month, day of week }</pre> où: year: interprété comme long-unsigned plage 0...big 0xFFFF = non spécifié year highbyte et year lowbyte représentent les deux octets du long-unsigned month, interprété comme unsigned plage 1...12, 0xFD, 0xFE, 0xFF 1 est janvier 0xFD = fin_changement_d'heure_legale 0xFE = debut_changement_d'heure_legale 0xFF = non spécifié dayOfMonth: interprété comme unsigned plage 1...31, 0xFD, 0xFE, 0xFF 0xFD = avant-dernier jour du mois 0xFE = dernier jour du mois 0xE0 à 0xFC = réservés 0xFF = non spécifié dayOfWeek: interprété comme unsigned plage 1...7, 0xFF 1 est lundi 0xFF = non spécifié Pour les dates répétitives, les parties non utilisées doivent être mises à "non spécifié"
	Les éléments dayOfMonth et dayOfWeek doivent être interprétés ensemble. – Si le dernier dayOfMonth est spécifié (0xFE) et dayOfWeek est un caractère générique, cela spécifie le dernier jour calendaire du mois; – Si le dernier dayOfMonth est spécifié (0xFE) et un dayOfWeek explicite est spécifié (par exemple 7, dimanche), il s'agit de la dernière occurrence du weekday (jour de la semaine) spécifié dans le month (mois), c'est-à-dire le dernier dimanche. – Si year (l'année) n'est pas spécifié (0xFFFF), et dayOfMonth et dayOfWeek sont tous deux explicitement spécifiés, il doit être interprété comme le dayOfWeek du dayOfMonth ou suivant celui-ci. – Si year (l'année) et month (le mois) sont spécifiés et dayOfMonth et dayOfWeek sont tous deux explicitement spécifiés, mais les valeurs ne sont pas compatibles, cela doit être considéré comme étant une erreur.

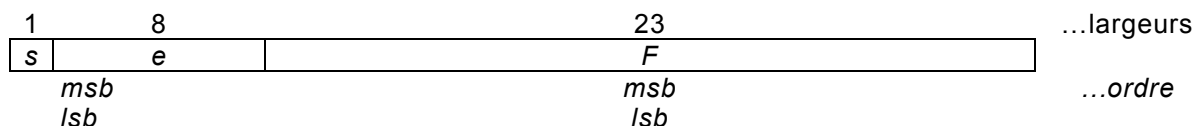
Exemples	
1) year = 0xFFFF, month = 0xFF, dayOfMonth = 0xFE, dayOfWeek = 0xFF: le dernier jour du mois dans chaque année et chaque mois; 2) year = 0xFFFF, month = 0xFF, dayOfMonth = 0xFE, dayOfWeek = 0x07: le dernier dimanche de chaque année et chaque mois; 3) year = 0xFFFF, month = 0x03, dayOfMonth = 0xFE, dayOfWeek = 0x07: le dernier dimanche de mars de chaque année; 4) year = 0xFFFF, month = 0x03, dayOfMonth = 0x01, dayOfWeek = 0x07: le premier dimanche de mars de chaque année; 5) year = 0xFFFF, month = 0x03, dayOfMonth = 0x16, dayOfWeek = 0x05: le quatrième vendredi de mars de chaque année; 6) year = 0xFFFF, month = 0x0A, dayOfMonth = 0x16, dayOfWeek = 0x07: le quatrième dimanche d'octobre de chaque année; 7) year = 0x07DE, month = 0x08, dayOfMonth = 0x13, (2014.08.13, Wednesday) dayOfWeek = 0x02 (Tuesday): erreur, car le dayOfMonth et le dayOfWeek dans l'année et le mois donnés ne concordent pas.	
time	OCTET STRING (SIZE(4)) <pre>{ hour, minute, second, hundredths }</pre> où: hour: interprété comme unsigned plage 0...23, 0xFF, minute: interprété comme unsigned plage 0...59, 0xFF, second: interprété comme unsigned plage 0...59, 0xFF, hundredths: interprété comme unsigned plage 0...99, 0xFF Pour hour (c'est-à-dire heure), minute, second (c'est-à-dire seconde) et hundredths (c'est-à-dire centièmes): 0xFF = non spécifié. Pour les heures répétitives, les parties non utilisées doivent être mises à "non spécifié".

<i>date-time</i>	OCTET STRING (SIZE(12)) <pre>{ year highbyte, year lowbyte, month, day of month, day of week, hour, minute, second, hundredths of second, deviation highbyte, deviation lowbyte, clock status }</pre> Les éléments de <i>date</i> et <i>time</i> sont codés comme indiqué ci-dessus. Certains peuvent être mis à "non spécifié" comme indiqué ci-dessus. De plus: deviation: interprété comme long plage -720...+720 en minutes de l'heure locale par rapport à l'heure au TUC 0x8000 = non spécifié Deviation highbyte et deviation lowbyte représentent les 2 octets du long clock_status interprété comme unsigned. Les bits sont définis comme suit: bit 0 (LSB): valeur non valide ^a, bit 1: valeur douteuse ^b, bit 2: base d'horloge différente ^c, bit 3: statut d'horloge non valide ^d, bit 4: réservé, bit 5: réservé, bit 6: réservé, bit 7 (MSB): heure d'été/hiver active ^e 0xFF = non spécifié
^a L'heure ne pouvait pas être récupérée après un incident. Les conditions détaillées sont spécifiques au fabricant (par exemple après une coupure de l'alimentation de l'horloge). Pour un statut valide, le bit 0 ne doit pas être défini si le bit 1 l'est. ^b L'heure pouvait être récupérée après un incident, mais la valeur ne peut pas être garantie. Les conditions détaillées sont spécifiques au fabricant. Pour un statut valide, le bit 1 ne doit pas être défini si le bit 0 l'est. ^c Le bit est positionné si les informations de temps élémentaires pour l'horloge à l'instant effectif proviennent d'une source de temps différente de la source spécifiée dans clock_base. ^d Ce bit indique qu'au moins un bit du statut de l'horloge est non valide. Certains bits peuvent être corrects. La signification exacte doit être expliquée dans la documentation du fabricant. ^e Fanion mis à vrai: l'heure transmise contient l'écart été/hiver (heure d'été). Fanion mis à faux: l'heure transmise ne contient pas l'écart été/hiver (heure normale).	

4.6.2 Formats de nombres en virgule flottante

Les formats de nombres en virgule flottante sont définis dans l'ISO/IEC/IEEE 60559:2011.

Le format "single" (simple) est:



où:

s est le bit de signe;

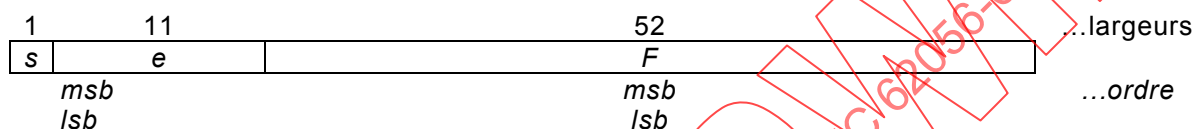
e est l'exposant; il a une largeur de 8 bits et le biais d'exposant est +127;

f est la fraction, il est de 23 bits.

Avec ces valeurs, la valeur est (si $0 < e < 255$):

$$v = (-1)^s \cdot 2^{e-127} \cdot (1.f)$$

Le format "double" (double) est:



où:

s est le bit de signe;

e est l'exposant; il a une largeur de 11 bits et le biais d'exposant est +1 023;

f est la fraction, il est de 52 bits.

Avec ces valeurs, la valeur est (si $0 < e < 2 047$):

$$v = (-1)^s \cdot 2^{e-1023} \cdot (1.f)$$

Pour plus de détails, voir l'ISO/IEC/IEEE 60559:2011.

Les nombres en virgule flottante doivent être représentés sous la forme d'une chaîne d'octets de longueur fixe, contenant les 4 octets (float32) du nombre en virgule flottante de format single ou les 8 octets (float64) du nombre en virgule flottante de format double tels que spécifiés ci-dessus, l'octet de poids fort en premier.

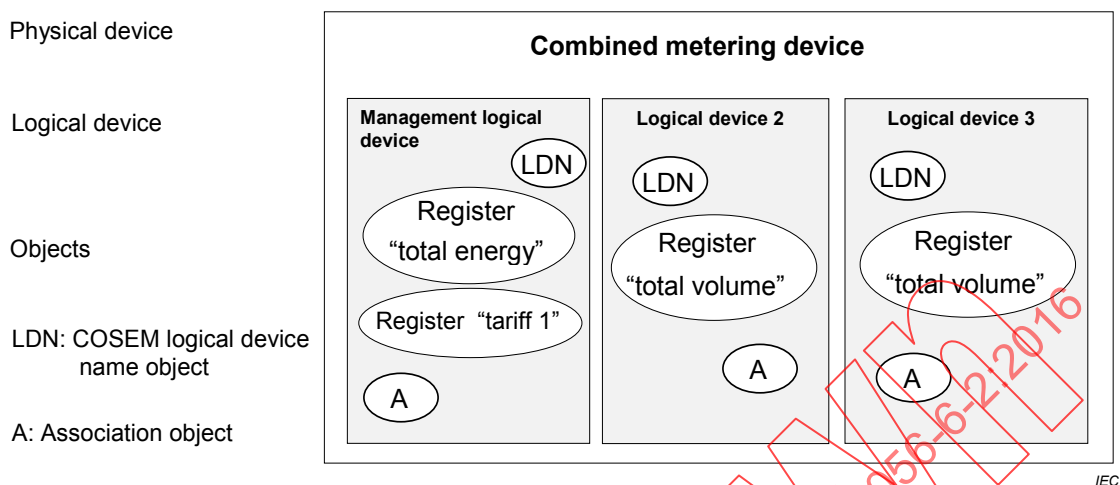
EXEMPLE 1 La valeur décimale "1" représentée au format virgule flottante single est:

Bit 31 Bit de signe 0 0: + 1: -	Bits 30 à 23: Champ exposant: 01111111 La valeur décimale du champ exposant et l'exposant: 127-127 = 0	Bits 22-0 Significande 1,000000000000000000000000 Valeur décimale du significande: 1,0000000

NOTE Le significande est le nombre binaire 1 suivi de la séparation fractionnaire suivie des bits binaires de la fraction.

Le codage, y compris l'étiquette du type de données est (toutes les valeurs sont hexadécimales): 17 3F 80 00 00.

L'exemple de la Figure 4 montre comment un dispositif de comptage combiné peut être structuré à l'aide du modèle de serveur COSEM.



Anglais	Français
Physical device	Dispositif physique
Logical device	Dispositif logique
Combined metering device	Dispositif de comptage combiné
Management logical device	Dispositif logique de gestion
Logical device	Dispositif logique
Objects	Objets
LDN: COSEM logical device name object	LDN: objet nom de dispositif logique COSEM
A: Association object	A: Objet Association
Register «total energy»	Registre «énergie totale»
Register «total volume»	Registre «volume total»

Figure 4 – Dispositif de comptage combiné

4.8 Dispositif logique COSEM

4.8.1 Généralités

Le dispositif logique COSEM contient un ensemble d'objets COSEM. Chaque dispositif physique doit contenir un "Dispositif logique de gestion".

L'adressage des dispositifs logiques COSEM doit être assuré par le plan d'adressage des couches inférieures de la pile protocolaire utilisée.

4.8.2 Nom de dispositif logique COSEM (LDN)

Le dispositif logique COSEM peut être identifié par son LDN COSEM unique. Ce nom peut être récupéré d'une instance de la classe d'interface "d'affectation de SAP" (voir 5.3.5) ou d'un objet COSEM "Nom de dispositif logique COSEM" (voir 6.2.29).

Le LDN est défini comme étant une chaîne d'octets de 16 octets au maximum. Les trois premiers octets doivent porter l'identifiant du fabricant³. Le fabricant doit assurer que le LDN,

³ Administré par la DLMS User Association, en coopération avec la FLAG Association.

commençant par les trois octets identifiant le fabricant et suivi de 13 octets au maximum, est unique.

4.8.3 "Vue association" du dispositif logique

Afin d'accéder aux objets COSEM dans le serveur, une association d'applications (AA) doit d'abord être établie avec un client. Les AA identifient les partenaires et caractérisent le contexte dans lequel les applications associées communiqueront. Les principales parties de ce contexte sont:

- le contexte d'application;
- le mécanisme d'authentification;
- le contexte xDLMS.

Les AA sont modélisées par des objets COSEM particuliers.

- les instances de la classe d'interface "SN d'association" (voir 5.3.3) sont utilisées avec le référencement par nom court;
- les instances de la classe d'interface "LN d'association" (voir 6.3.4) sont utilisées avec le référencement par nom logique;

En fonction de l'AA établie entre le client et le serveur, des droits d'accès différents peuvent être accordés par le serveur. Les droits d'accès concernent un ensemble d'objets COSEM – les objets visibles – qui peuvent être accessibles ('vus') au sein d'une AA donnée. En outre, l'accès aux attributs et méthodes de ces objets COSEM peut aussi être limité au sein de l'AA (par exemple, un certain type de client ne peut lire qu'un attribut particulier d'un objet COSEM, mais ne peut pas l'écrire). Les droits d'accès peuvent également stipuler la protection cryptographique exigée.

La liste des objets COSEM visibles (la "vue association") peut être obtenue par le client par lecture de l'attribut "*object_list*" de l'objet d'association approprié.

4.8.4 Contenu obligatoire d'un dispositif logique COSEM

Les objets suivants doivent être présents dans chaque dispositif logique COSEM. Ils doivent être accessibles pour GET/Read dans toutes les AA avec ce dispositif logique:

- objet Nom de dispositif logique COSEM;
- objet "Association" (LN ou SN) actuel.

Si l'objet "Affectation de SAP" est présent, l'objet Nom de dispositif logique COSEM n'a pas besoin d'être présent.

4.8.5 Dispositif logique de gestion

Comme spécifié en 4.8.1, le dispositif logique de gestion est un élément obligatoire de tout dispositif physique. Il a une adresse réservée. Il doit prendre en charge une AA avec un client public avec l'authentification de sécurité du niveau le plus bas (pas de sécurité). Son rôle est de prendre en charge la révélation de la structure interne du dispositif physique et de prendre en charge la notification d'événements dans le serveur.

En plus de l'objet "Association" modélisant l'AA avec le client public, le dispositif logique de gestion doit contenir un objet "Affectation de SAP", donnant son SAP et le SAP de tous les autres dispositifs logiques au sein du dispositif physique. L'objet Affectation de SAP doit être lisible au moins par le client public.

S'il n'y a qu'un seul dispositif logique au sein du dispositif physique, l'objet "Affectation de SAP" peut être omis.

4.9 Sécurité des informations

DLMS/COSEM fournit plusieurs fonctions de sécurité des informations pour l'accès et le transport des données:

- la sécurité d'accès des données contrôle l'accès aux données détenues par un serveur DLMS/COSEM;
- la sécurité de transport des données permet à la partie émettrice d'appliquer une protection chiffrée aux APDU de xDLMS envoyées. Cela exige des APDU chiffrées. La partie réceptrice peut retirer ou vérifier cette protection

Pour une description de ces mécanismes de sécurité, voir l'IEC 62056-5-3; Article 5.

La sécurité des informations est assurée au niveau de la couche d'application DLMS/COSEM et au niveau de l'objet COSEM. Elle est prise en charge/gérée par les objets suivants:

- "SN d'association", voir 5.3.3;
- "LN d'association", voir 5.3.4; et
- "Établissement de la sécurité", voir 5.3.7.

5 Classes d'interfaces de la COSEM

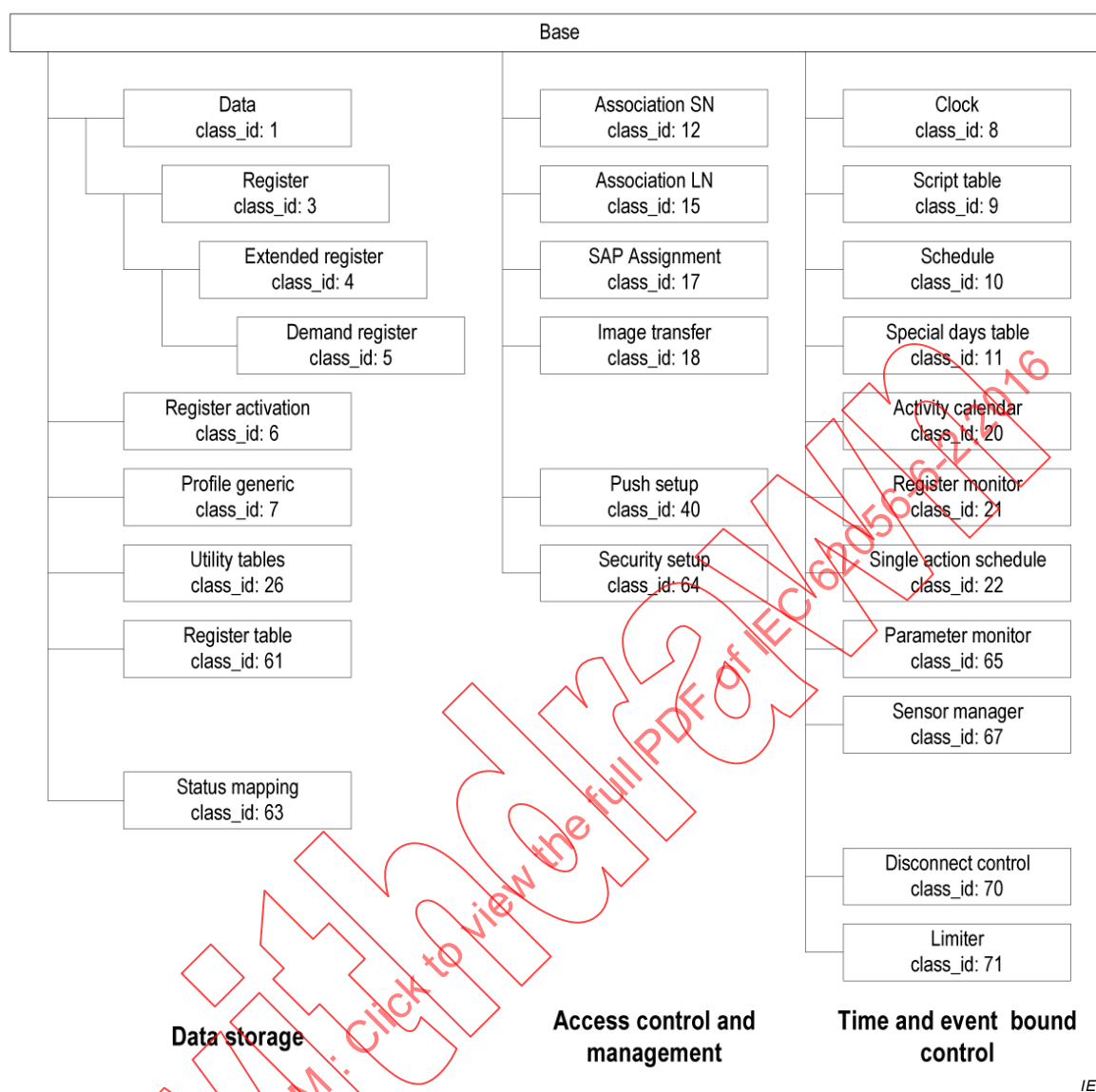
5.1 Vue d'ensemble

Les IC définies à l'heure actuelle et les relations entre elles sont présentées à la Figure 5 et à la Figure 6.

NOTE 1 La "base" d'IC proprement dite n'est pas spécifiée de façon explicite. Elle contient un seul attribut *logical_name*.

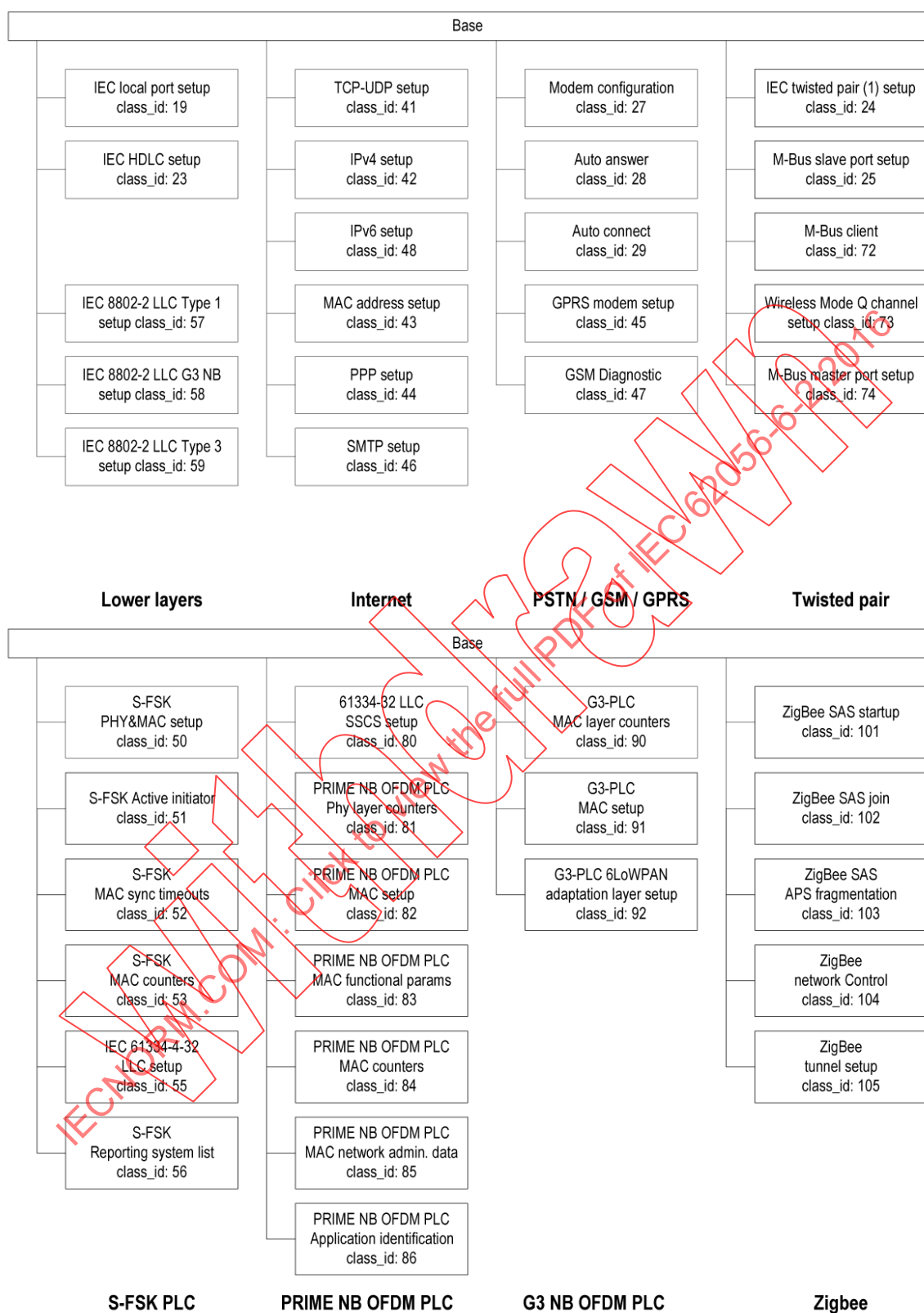
NOTE 2 Dans la description des IC "Demand register" (registre de puissance), "Clock" (horloge) et "Profile generic" (profil générique), les 2^{ème} attributs sont étiquetés différemment de celui du 2^{ème} attribut de l'IC "Data" (données) IC, à savoir *current_average_value*, *time* et *buffer* contre *value*. Le but est de mettre l'accent sur la nature spécifique de *value*.

NOTE 3 Sur ces Figures, les classes d'interfaces sont présentées dans chaque groupe en augmentant *class_id*. Dans les articles spécifiant les différents groupes de classes d'interfaces, les nouvelles classes d'interfaces sont placées à la fin de l'article correspondant.



Anglais	Français
Data storage	Stockage de données
Access control and management	Contrôle d'accès et gestion
Time and event bound control	Commande à limite temporelle et événementielle

Figure 5 – Vue d'ensemble des classes d'interfaces – Partie 1



IEC

Anglais	Français
Lower Layers	Couches inférieures
Twisted pair	Paire torsadée

Figure 6 – Vue d'ensemble des classes d'interfaces – Partie 2

Le Tableau 3 répertorie les classes d'interfaces par class_id.

Tableau 3 – Liste des classes d'interfaces par class_id

Nom de la classe d'interface	class_id	version(s)	Article
Data (Données)	1	0	5.2.1
Register (Registre)	3	0	5.2.2
Extended register (registre étendu)	4	0	5.2.3
Demand register (Registre de puissance)	5	0	5.2.4
Register activation (activation de registre)	6	0	5.2.5
Profile generic (Profil générique)	7	1 0	5.2.6 7.2
Clock (Horloge)	8	0	5.4.1
Script table (tableau de scripts)	9	0	5.4.2
Schedule (programme)	10	0	5.4.3
Special days table (tableau de jours spéciaux)	11	0	5.4.4
SN d'association	12	3	5.3.3
		2	7.5
		1	7.4
		0	7.3
LN d'association	15	2	5.3.4
		1	7.7
		0	7.6
SAP assignment (Affectation de SAP)	17	0	5.3.5
Image transfer (Transfert d'Image)	18	0	5.3.6
IEC local port setup (Établissement de port local IEC)	19	1 0	5.5.1 7.8
Activity calendar (calendrier d'activités)	20	0	5.4.5
Register monitor (Moniteur de registre)	21	0	5.4.6
Single action schedule (programme d'action unique)	22	0	5.4.7
IEC HDLC setup (Établissement de HDLC IEC)	23	1 0	5.5.2 7.9
IEC twisted pair (1) setup (établissement de paire torsadée (1) IEC)	24	1 0	5.5.3 7.10
M-BUS slave port setup (établissement de port de M-Bus esclave)	25	0	5.6.1
Utility tables (Tables de fournisseurs de service d'énergie)	26	0	5.2.7
Modem configuration (configuration de modem)	27	1	5.5.4
PSTN modem configuration (configuration de modems RTPC)		0	7.11
Auto answer (réponse automatique)	28	2	5.5.5
		1	–
		0	7.12
Auto connect (connexion automatique)	29	2	5.5.6
		1	7.14
		0	7.13
PSTN auto dial (Composition automatique de numéro RTPC)			
Push setup (établissement de Push)	40	0	5.3.8.2
TCP-UDP setup (établissement de TCP-UDP)	41	0	5.7.1

Nom de la classe d'interface	class_id	version(s)	Article
IPv4 setup (établissement de IPv4)	42	0	5.7.2
MAC address setup (Ethernet setup) (établissement d'adresse MAC (configuration d'Ethernet))	43	0	5.7.4, 5.10.10
PPP setup (établissement de PPP)	44	0	5.7.5
GPRS modem setup (établissement de modem GPRS)	45	0	5.5.7
SMTP setup (établissement de SMTP)	46	0	5.7.6
GSM diagnostic (diagnostic GSM)	47	0	5.5.8
IPv4 setup (établissement de IPv4)	48	0	5.7.3
S-FSK Phy&MAC setup (établissement de Phy et MAC S-FSK)	50	1 0	5.8.3 7.15
S-FSK Active initiator (Initiateur actif S-FSK)	51	0	5.8.4
S-FSK MAC synchronization timeouts (temps d'expiration de synchronisation MAC S-FSK)	52	0	5.8.5
S-FSK MAC counters (compteurs S-FSK MAC)	53	0	5.8.6
IEC 61334-4-32 LLC setup (établissement de LLC de l'IEC 61334-4-32)	55	1	5.8.7
S-FSK IEC 61334-4-32 LLC setup (établissement de LLC S-FSK de l'IEC 61334-4-32)	55	0	7.16
S-FSK Reporting system list (liste de systèmes de comptes-rendus S-FSK)	56	0	5.8.8
ISO/IEC 8802-2 LLC Type 1 setup (établissement de LLC de Type 1 selon l'ISO/IEC 8802-2)	57	0	5.9.2
ISO/IEC 8802-2 LLC Type 2 setup (établissement de LLC de Type 2 selon l'ISO/IEC 8802-2)	58	0	5.9.3
ISO/IEC 8802-2 LLC Type 3 setup (établissement de LLC de Type 3 selon l'ISO/IEC 8802-2)	59	0	5.9.4
Tableau de registre	61	0	5.2.8
Status mapping (Mise en correspondance d'état)	63	0	5.2.9
Security setup (établissement de la sécurité)	64	0	5.3.7
Parameter monitor (moniteur de paramètre)	65	0	5.4.10
Sensor manager (Gestionnaire de capteur)	67	0	5.4.11
Disconnect control (commande de déconnexion)	70	0	5.4.8
Limiter (Limiteur)	71	0	5.4.9
M-Bus client (client M-Bus)	72	1 0	5.6.2 7.17
Wireless Mode Q channel (canal en mode Q sans fil)	73	0	5.6.3
M-Bus master port setup (établissement de port de M-Bus maître)	74	0	5.6.4
61334-4-32 LLC SSCS setup (établissement de LLC SSCS 61334-4-32)	80	0	5.10.3
PRIME NB OFDM PLC Physical layer counters (Compteurs de couche physique PRIME NB OFDM PLC)	81	0	5.10.5
PRIME NB OFDM PLC MAC setup (établissement PRIME NB OFDM PLC MAC)	82	0	5.10.6
PRIME NB OFDM PLC MAC functional parameters (paramètres fonctionnels PRIME NB OFDM PLC MAC)	83	0	5.10.7
PRIME NB OFDM PLC MAC counters (compteurs PRIME NB OFDM PLC MAC)	84	0	5.10.8
PRIME NB OFDM PLC MAC network administration data (données d'administration de réseau PRIME NB OFDM PLC MAC)	85	0	5.10.9
PRIME NB OFDM PLC Application identification (Identification d'application PRIME NB OFDM PLC)	86	0	5.10.11

Nom de la classe d'interface	class_id	version(s)	Article
G3-PLC MAC layer counters (compteurs de couche G3-PLC MAC)	90	1	5.11.3
G3 NB OFDM PLC MAC layer counters (compteurs de couche G3 NB OFDM PLC MAC)		0	7.18
G3-PLC MAC setup (établissement de G3-PLC MAC)	91	1	5.11.4
G3 NB OFDM PLC MAC setup (établissement de G3 NB OFDM PLC MAC)		0	7.19
G3-PLC 6LoWPAN adaptation layer setup (configuration de la couche d'adaptation G3-PLC 6LoWPAN)	92	1	5.11.5
G3 NB OFDM PLC 6LoWPAN adaptation layer setup (configuration de la couche d'adaptation G3 NB OFDM PLC 6LoWPAN)		0	7.20
ZigBee® SAS startup (démarrage ZigBee® SAS)	101	0	5.12.2
ZigBee® SAS join (liaison ZigBee® SAS)	102	0	5.12.3
ZigBee® SAS APS fragmentation (fragmentation APS ZigBee® SAS)	103	0	5.12.4
ZigBee® network control (contrôle de réseau ZigBee®)	104	0	5.12.5
ZigBee® tunnel setup (établissement du tunnel ZigBee®)	105	0	5.12.6

5.2 Classes d'interfaces pour paramètres et données de mesure

5.2.1 Data (class_id = 1, version = 0)

Cette IC permet de modéliser diverses données, telles que données de configuration et paramètres. Les données sont identifiées par l'attribut *logical_name*.

Data (Données)	0...n	class_id = 1, version = 0			
Attributes	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. value	CHOICE				x + 0x08
Méthodes spécifiques	m/o				

Description d'attribut

logical_name	Identifie l'instance de l'objet "Data". Voir l'Article 6 et l'IEC 62056-6-1:.		
value	Contient les données. CHOICE <pre>{ -- types de données simples null-data [0], boolean [3], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], utf8-string [12], bcd [13], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], enum [22], float32 [23], float64 [24], date-time [25], date [26], time [27], -- types de données complexes array [1], structure [2], compact-array [19] }</pre> Le type de données dépend de l'instanciation définie par le <i>logical_name</i> et éventuellement provenant du fabricant. Il est à choisir de manière à rendre possible, avec le <i>logical_name</i>, une interprétation non ambiguë. Un type de données simple ou complexe énuméré en 4.5 peut être utilisé, sauf en cas de limitation du choix par l'Article 6.		

5.2.2 Register (class_id = 3, version = 0)

Cette IC permet la modélisation d'une valeur issue d'un processus ou d'un état avec le scalaire et l'unité. Les objets "Register" ont une notion de la nature de la valeur issue du processus ou de l'état. Elle est identifiée par l'attribut *logical_name*.

Register (Registre)		0...n				class_id = 3, version = 0			
Attributs		Type de données		Min.	Max.	Def.	Nom court		
1.	logical_name (static)	octet-string					x		
2.	value	CHOICE					x + 0x08		
3.	scaler_unit (static)	scal_unit_type					x + 0x10		
Méthodes spécifiques		m/o							
1.	reset (data)	f					x + 0x28		

Description d'attribut

logical_name	Identifie l'instance de l'objet "Register" (Registre). Voir l'Article 6 et l'IEC 62056-6-1:.	
value	Contient la valeur traitée ou d'état courant. CHOICE <pre>{ -- types de données simples null-data [0], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], utf8-string [12], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], enum [22], float32 [23], float64 [24], }</pre>	Le type de données de la valeur dépend de l'instanciation définie par le <i>logical_name</i> et éventuellement du choix du fabricant. Il est à choisir de manière à rendre possible, avec le <i>logical_name</i>, une interprétation non ambiguë de la valeur. Lorsqu'en lieu et place d'un objet "Data", un objet "Register" est utilisé, (sans utilisation de l'attribut <i>scaler_unit</i> ou avec <i>scaler</i> = 0, <i>unit</i> = 255), les types de données autorisés pour l'attribut <i>value</i> de l'IC "Data" sont autorisés.
scaler_unit	Fournit des informations relatives à l'unité et scalaire de la valeur. <pre>scal_unit_type ::= structure { scaler, unit } scaler: integer</pre> Il s'agit de l'exposant (en base 10) du facteur de multiplication. REMARQUE Si la valeur n'est pas numérique, scaler (l'échelle de comptage) doit être mis à 0. <pre>unit: enum</pre> Énumération définissant l'unité physique. Pour les détails, voir le Tableau 4 ci-dessous.	
Description de la méthode		
reset (data)	Force une réinitialisation de l'objet. En appelant cette méthode, la valeur est mise à la valeur par défaut. La valeur par défaut est une constante spécifique à l'instance. data ::= integer (0)	

Tableau 4 – Valeurs énumérées pour les unités physiques

unit::=enum	Unité	Grandeur	Nom de l'unité	Définition SI (commentaire)
(1)	a	time	year (c'est-à-dire année)	
(2)	mo	time	month (c'est-à-dire mois)	
(3)	wk	time	week (c'est-à-dire semaine)	7*24*60*60 s
(4)	d	time	day (c'est-à-dire jour)	24*60*60 s
(5)	h	time	heure	60*60 s
(6)	min.	time	min	60 s
(7)	s	temps (<i>t</i>)	seconde	s
(8)	°	(phase) angle	degré	rad*180/π
(9)	°C	température (<i>T</i>)	degré-celsius	K-273.15
(10)	currency	monnaie (locale)		
(11)	m	longueur (<i>l</i>)	mètre	m
(12)	m/s	vitesse (<i>v</i>)	mètre par seconde	m/s
(13)	m ³	volume (<i>V</i>) r _v , constante du compteur ou valeur d'impulsion (volume)	mètre cube	m ³
(14)	m ³	volume corrigé	mètre cube	m ³
(15)	m ³ /h	débit volumique	mètre cube par heure	m ³ /(60*60s)
(16)	m ³ /h	débit volumique corrigé	mètre cube par heure	m ³ /(60*60s)
(17)	m ³ /d	débit volumique		m ³ /(24*60*60s)
(18)	m ³ /d	débit volumique corrigé		m ³ /(24*60*60s)
(19)	l	volume	litre	10 ⁻³ m ³
(20)	kg	masse (<i>m</i>)	kilogramme	
(21)	N	force (<i>F</i>)	newton	
(22)	Nm	énergie	newton-mètre	J = Nm = Ws
(23)	Pa	pression (<i>p</i>)	pascal	N/m ²
(24)	bar	pression (<i>p</i>)	bar	10 ⁵ N/m ²
(25)	J	énergie	joule	J = Nm = Ws
(26)	J/h	puissance thermique	joule par heure	J/(60*60s)
(27)	W	puissance active (<i>P</i>)	watt	W = J/s
(28)	VA	puissance apparente (<i>S</i>)	volts-ampères	
(29)	var	puissance réactive (<i>Q</i>)	var	
(30)	Wh	énergie active r _w , constante d'énergie active du compteur ou valeur d'impulsion	watt-heure	W*(60*60s)
(31)	VAh	énergie apparente r _s , constante d'énergie apparente du compteur ou valeur d'impulsion	volt-ampère-heure	VA*(60*60s)
(32)	varh	énergie réactive r _B , constante d'énergie réactive du compteur ou valeur d'impulsion	var-heure	var*(60*60s)
(33)	A	courant (<i>I</i>)	ampère	A
(34)	C	charge électrique (<i>Q</i>)	coulomb	C = As
(35)	V	tension (<i>U</i>)	volt	V
(36)	V/m	intensité du champ électrique (<i>E</i>)	volt par mètre	V/m
(37)	F	capacité (<i>C</i>)	farad	C/V = As/V
(38)	Ω	résistance (<i>R</i>)	ohm	Ω = V/A

unit::=enum	Unité	Grandeur	Nom de l'unité	Définition SI (commentaire)
(39)	$\Omega\text{m}^2/\text{m}$	résistivité (ρ)		Ωm
(40)	Wb	flux magnétique (ϕ)	weber	$\text{Wb} = \text{Vs}$
(41)	T	induction magnétique (B)	tesla	Wb/m^2
(42)	A/m	intensité du champ magnétique (H)	ampère par mètre	A/m
(43)	H	inductance (L)	henry	$\text{H} = \text{Wb}/\text{A}$
(44)	Hz	fréquence (f, ω)	hertz	1/s
(45)	1/(Wh)	R_W , constante d'énergie active du compteur ou valeur d'impulsion		
(46)	1/(varh)	R_B , constante d'énergie réactive du compteur ou valeur d'impulsion		
(47)	1/(VAh)	R_S , constante d'énergie apparente du compteur ou valeur d'impulsion		
(48)	V^2h	volt carré-heure, r_{U2h} , constante volt carré-heure du compteur ou valeur d'impulsion	volt carré-heures	$\text{V}^2(60 \cdot 60\text{s})$
(49)	A^2h	ampère carré-heure, r_{I2h} , constante ampère carré-heure du compteur ou valeur d'impulsion	ampère carré-heures	$\text{A}^2(60 \cdot 60\text{s})$
(50)	kg/s	débit massique	kilogramme par seconde	kg/s
(51)	S, mho	conductance	siemens	$1/\Omega$
(52)	K	température (T)	kelvin	
(53)	1/(V^2h)	R_{U2h} , constante volt carré heure du compteur ou valeur d'impulsion		
(54)	1/(A^2h)	R_{I2h} , constante ampère carré heure du compteur ou valeur d'impulsion		
(55)	$1/\text{m}^3$	R_V , constante du compteur ou valeur d'impulsion (volume)		
(56)		pourcentage	%	
(57)	Ah	ampèreheures	Ampère-heure	
...				
(60)	Wh/m^3	énergie volumique	$3,6 \cdot 10^3 \text{ J}/\text{m}^3$	
(61)	J/m^3	valeur calorifique, wobbe		
(62)	Mol %	fraction molaire de la composition gazeuse	mole pourcent	(unité de composition gazeuse de base)
(63)	g/m^3	masse volumique, quantité de matière		(analyse du gaz, accompagnant les éléments)
(64)	Pa s	viscosité dynamique	pascal seconde	(caractéristique du flux gazeux)
(65)	J/kg	énergie spécifique NOTE La quantité d'énergie par unité de masse d'une substance	Joule par kilogramme	$\text{m}^2 \cdot \text{kg} \cdot \text{s}^{-2} / \text{kg} = \text{m}^2 \cdot \text{s}^{-2}$
....				
(70)	dBm	Intensité de signal, dB milliwatt (du système radio GSM, par exemple)		
(71)	dbµV	Intensité du signal, dB microvolt		
(72)	dB	Unité logarithmique qui exprime le rapport entre deux valeurs d'une grandeur physique		
...				
(253)		reserved (réservé)		
(254)	autre	autre unité		
(255)	count	sans unité, sans dimension, compte		

Des exemples sont présentés au Tableau 5 ci-dessous.

Tableau 5 – Exemples pour scaler_unit

Valeur	Scalaire	Unité	Données
263788	-3	m ³	263,788 m ³
593	3	Wh	593 kWh
3467	-1	V	346,7
3467	0	V	3 467 V
3467	1	V	34 670 V

5.2.3 Extended register (class_id = 4, version = 0)

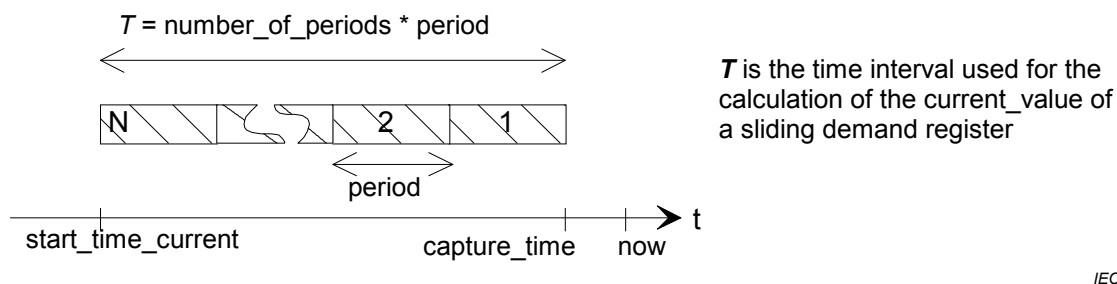
Cette IC permet de modéliser une valeur issue d'un processus avec ses informations associées relatives au scalaire, à l'unité, à l'état et à l'instant d'acquisition. Les objets "Extended register" (Registre étendu) ont une notion de la nature de la valeur issue du processus. Elle est décrite par l'attribut *logical_name*.

Extended register (registre étendu)		0...n	class_id = 4, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	value (dyn.)	CHOICE				x + 0x08
3.	scaler_unit (static)	scal_unit_type				x + 0x10
4.	status (dyn.)	CHOICE				x + 0x18
5.	capture_time (dyn.)	octet-string				x + 0x20
Méthodes spécifiques		m/o				
1.	reset (data)	f				x + 0x38

Description d'attribut	
logical_name	Identifie l'instance de l'objet "Extended register" (Registre étendu). Voir 6.3.1.
value	Voir la spécification de l'IC "Register" en 5.2.2.
scaler_unit	Voir la spécification de l'IC "Register" en 5.2.2.
status	Fournit des informations d'état spécifiques à "Extended register". La signification des éléments de statut doit être fournie pour chaque instance de l'objet. CHOICE <pre> { -- types de données simples null-data [0], bit-string [4], double-long-unsigned [6], octet-string [9], visible-string [10], utf8-string [12], unsigned [17], long-unsigned [18], long64-unsigned [21] } </pre> Le type de données et le codage dépendent de l'instanciation et éventuellement du choix du fabricant. Pour l'interprétation, des informations complémentaires fournies par le fabricant peuvent être nécessaires. Def. Dépendant de la définition du type de statut.
capture_time	Fournit les informations de date et d'heure spécifiques au registre étendu "Extended register" montrant le moment où la valeur de l'attribut <i>value</i> a été saisie. octet-string, mis en forme comme indiqué en 4.6.1 pour <i>date-time</i>.
Description de la méthode	
reset (data)	Cette méthode force une réinitialisation de l'objet. En appelant cette méthode, la valeur par défaut est affectée à l'attribut <i>value</i>. La valeur par défaut est une constante spécifique à l'instance. L'attribut <i>capture_time</i> est défini à l'heure de l'exécution de la réinitialisation. data := integer (0)

5.2.4 Demand register (class_id = 5, version = 0)

Cette IC permet de modéliser une valeur de puissance avec ses informations associées relatives au scalaire, aux unités, à l'état et aux instants d'acquisition. Un objet "Demand register" mesure et calcule périodiquement une valeur moyenne courante (*current_average_value*) et stocke une dernière valeur moyenne (*last_average_value*). L'intervalle de temps *T* sur lequel la demande est mesurée ou calculée est défini en spécifiant *number_of_periods* (nombre de périodes) et *period* (période). Voir la Figure 7.

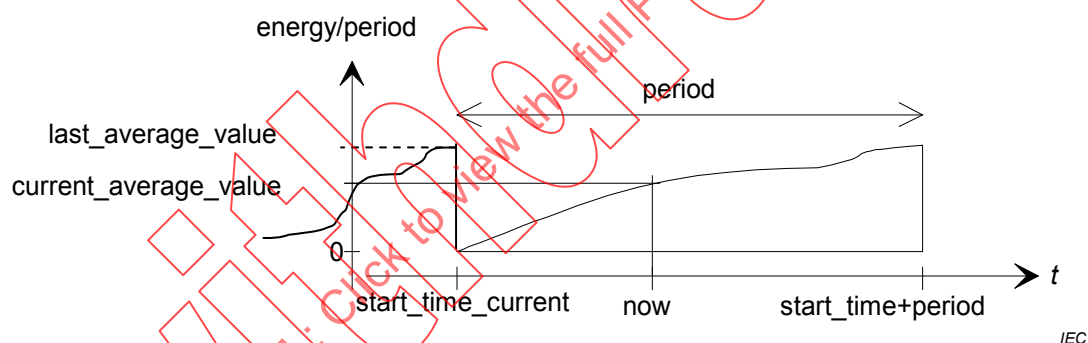


Anglais	Français
T is the time interval used for the calculation of the current_value of a sliding demand register	T est l'intervalle de temps utilisé pour le calcul de la current_value d'un Registre de puissance glissante

Figure 7 – Attributs de temps pour mesurer une puissance glissante

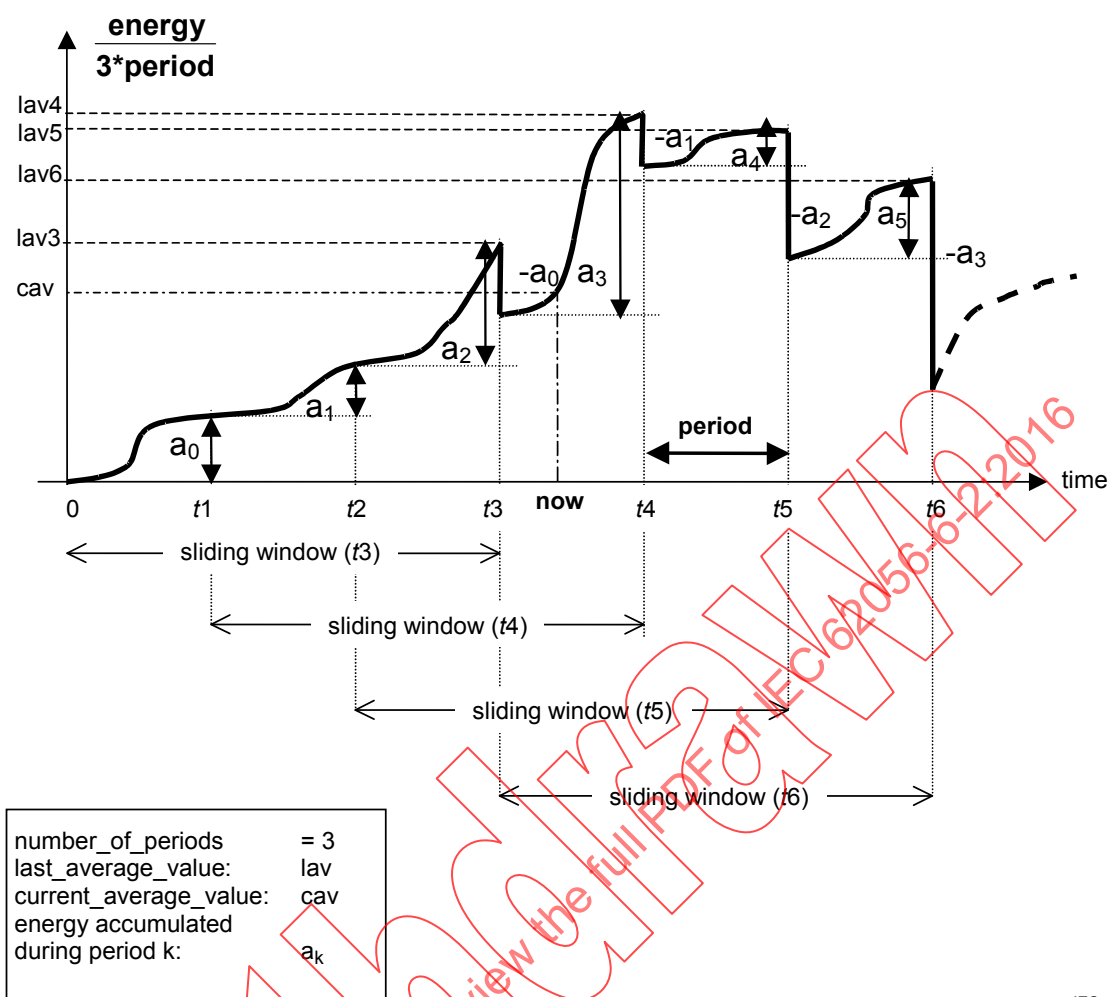
Le registre de puissance délivre deux types de puissance: *current_average_value* et *last_average_value* (voir Figure 8 et Figure 9).

Les objets "Demand register" ont une notion de la nature de la valeur issue du processus qui est décrite par l'attribut *logical_name*.



Anglais	Français
energy/period	énergie/période
period	période
now	maintenant

Figure 8 – Attributs dans le cas de la puissance en bloc



Anglais	Français
energy/period	énergie/période
period	période
now	maintenant
time	temps
sliding window	fenêtre glissante
energy accumulated during period k	énergie accumulée au cours de la période k

Figure 9 – Attributs dans le cas de la puissance glissante (nombre de périodes = 3)

Demand register (Registre de puissance)		0...n	class_id = 5, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	current_average_value (dyn.)	CHOICE			0	x + 0x08
3.	last_average_value (dyn.)	CHOICE			0	x + 0x10
4.	scaler_unit (static)	scal_unit_type				x + 0x18
5.	status (dyn.)	CHOICE				x + 0x20
6.	capture_time (dyn.)	octet-string				x + 0x28
7.	start_time_current (dyn.)	octet-string				x + 0x30
8.	period (static)	double-long-unsigned	1			x + 0x38
9.	number_of_periods (static)	long-unsigned	1		1	x + 0x40
Méthodes spécifiques		m/o				
1.	reset (data)	f				x + 0x48
2.	next_period (data)	f				x + 0x50

Description d'attribut

logical_name Identifie l'instance de l'objet "Demand register" (Registre de puissance). Voir 6.3.1 et l'IEC 62056-6-1..

current_average_value Fournit la valeur courante (puissance en cours) de l'énergie accumulée depuis *start_time*, divisée par *number_of_periods*period*.

Le type de données de la valeur dépend de l'instanciation définie par le *logical_name* et éventuellement du choix du fabricant. Le type est à choisir de manière à rendre possible, avec le *logical_name*, une interprétation non ambiguë de la valeur.

Si une grandeur autre que l'énergie est mesurée, d'autres méthodes de calcul peuvent s'appliquer (par exemple pour calculer les valeurs moyennes de la tension ou du courant).

CHOICE Pour les types de données, voir "Register" en 5.2.2, attribut "value".

last_average_value Fournit la valeur de l'énergie accumulée (sur les dernières *number_of_periods*period*) divisée par *number_of_periods*period*. L'énergie de la période en cours (non achevée) n'est pas prise en compte dans le calcul.

Si une grandeur autre que l'énergie est mesurée, d'autres méthodes de calcul peuvent s'appliquer (par exemple pour calculer les valeurs moyennes de la tension ou du courant).

CHOICE Pour les types de données, voir l'IC "Register", attribut "value".

scaler_unit Voir la spécification de l'IC "Register" en 5.2.2.

status Fournit des informations d'état spécifiques à "Demand register". Le type de données et le codage dépendent de l'instanciation et éventuellement du choix du fabricant. Pour l'interprétation, des informations complémentaires fournies par le fabricant peuvent être nécessaires.

CHOICE Pour les types de données, voir l'IC "Extended register" en 5.2.3, attribut "status".

Def. Dépendant de la définition du type de statut.

capture_time Fournit la date et l'heure du calcul de *last_average_value*.

	octet-string, mis en forme comme indiqué en 4.6.1 pour <i>date-time</i> .
start_time_current	Fournit la date et l'heure du début du mesurage de <i>current_average_value</i> . octet-string, mis en forme comme indiqué en 4.6.1 pour <i>date-time</i> .
period	"Period" est l'intervalle entre deux mises à jour successives de <i>last_average_value</i> . (<i>number_of_periods*period</i> est le dénominateur pour le calcul de la demande). double-long-unsigned Période de mesure en secondes Le comportement du compteur après écriture d'une nouvelle valeur dans cet attribut doit être spécifié par le fabricant.
number_of_periods	Le nombre de périodes utilisées pour calculer <i>last_average_value</i> . <i>number_of_periods</i> >= 1 – <i>number_of_periods</i> > 1 indique que <i>last_average_value</i> représente une "puissance glissante". – <i>number_of_periods</i> = 1 indique que <i>last_average_value</i> représente une "puissance en bloc". Le comportement du compteur après écriture d'une nouvelle valeur dans cet attribut doit être spécifié par le fabricant.
Description de la méthode	
reset (data)	Cette méthode force une réinitialisation de l'objet. L'activation de cette méthode provoque les actions suivantes: – il est mis fin à la période courante; – <i>current_average_value</i> et <i>last_average_value</i> sont mises à leurs valeurs par défaut; – <i>capture_time</i> et <i>start_time_current</i> sont mis à l'heure de l'exécution de <i>reset (data)</i> . data::= integer (0)
next_period (data)	Cette méthode est utilisée pour déclencher la fin régulière (et le redémarrage) d'une période. Cloture (met fin à) la période de mesure en cours. Met à jour <i>capture_time</i> et <i>start_time</i> et copie <i>current_average_value</i> dans <i>last_average_value</i> , met <i>current_average_value</i> à sa valeur par défaut. Démarre la prochaine période de mesure. REMARQUE L'ancien <i>old last_average_value</i> (et <i>capture_time</i>) peut être lu pendant la durée "period". L'ancien <i>current_average_value</i> n'est plus disponible à l'interface. data::= integer (0)

5.2.5 Register activation (class_id = 6, version = 0)

Cette IC permet de modéliser la gestion des différentes structures de tarification. Pour chaque objet "Register activation", des groupes d'objets "Register", "Extended register" ou "Demand register", modélisation de différentes sortes de grandeurs (par exemple: énergie active, puissance active, énergie réactive, etc.) sont attribués. Des sous-groupes de ces registres, définis par les *activation masks* (masques d'activation) définissent différentes structures de tarifs (par exemple: tarif de jour, tarif de nuit). L'un de ces masques d'activation, *active_mask* (masque actif), définit quel sous-ensemble des registres, attribués à l'instance d'objet "Register activation", est actif. Les registres qui ne sont pas inclus dans l'attribut *register_assignment* de tout objet "Register activation" sont toujours activés par défaut.

Register activation (activation de registre)		0...n	class_id = 6, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	register_assignment (static)	array				x + 0x08
3.	mask_list (static)	array				x + 0x10
4.	active_mask (dyn.)	octet-string				x+ 0x18
Méthodes spécifiques		m/o				
1.	add_register (data)	f				x + 0x30
2.	add_mask (data)	f				x + 0x38
3.	delete_mask (data)	f				x + 0x40

Description d'attribut

logical_name Identifie l'instance de l'objet "Register activation" (Activation de registre). Voir la 6.2.11.

register_assignment Spécifie une liste ordonnée d'objets COSEM attribués à l'objet "Register activation". La liste peut contenir différentes sortes d'objets COSEM (par exemple des instances de "Register", "Extended register" ou "Demand register").

array object_definition

```
object_definition ::= structure
{
    class_id: long-unsigned,
    logical_name: octet-string
}
```

mask_list Spécifie une liste de masques d'activation de registres. Chaque entrée (masque) est identifiée par son mask_name et contient un tableau d'indices renvoyant aux registres attribués au masque (le premier objet dans register_assignment est référencé par l'indice 1, le deuxième objet par l'indice 2,...).

array register_act_mask

```
register_act_mask ::= structure
{
    mask_name: octet-string,
    index_list: index_array
}
```

mask_name est à définir de manière unique au sein de l'objet.

index_array ::= array unsigned

active_mask Définit le masque actuellement actif. Le masque est défini par son mask_name (voir mask_list).

active_mask définit les registres qui sont actuellement activés. Tous les autres registres énumérés dans register_assignment sont désactivés.

Description de la méthode

add_register (data)	Ajoute un registre de plus à l'attribut <i>register_assignment</i>. Le nouveau registre est ajouté à la fin du tableau, à savoir le registre nouvellement ajouté possède l'indice le plus élevé. Les indices des registres existants ne sont pas modifiés. <pre>data ::= structure { class_id: long-unsigned, logical_name: octet-string }</pre>
add_mask (data)	Ajoute un autre masque à l'attribut <i>mask_list</i>. S'il existe déjà un masque portant le même nom, le masque existant est écrasé par le nouveau masque. <pre>data ::= register_act_mask (voir ci-dessus)</pre>
delete_mask (data)	Supprime un masque de l'attribut <i>mask_list</i>. Le masque est défini par son nom de masque. <pre>data ::= octet-string (mask_name)</pre>

5.2.6 Profile generic (class_id = 7, version = 1)

Cette IC fournit un concept généralisé permettant de stocker, de trier et d'accéder à des groupes de données ou séries de données, appelés *capture objects* (objets de saisie). Les objets de saisie sont des attributs ou éléments appropriés d'un ou de plusieurs attributs d'objets COSEM. Les objets de saisie sont recueillis de façon périodique ou occasionnelle.

Un profil est un *buffer* (tampon) pour stocker les données saisies. Pour récupérer une partie seulement du tampon, il est permis de spécifier soit une plage de valeurs, soit une plage d'entrées, demandant de récupérer toutes les entrées qui s'inscrivent dans la plage spécifiée.

La liste des *capture objects* définit la valeur à stocker dans le *buffer* (en utilisant la saisie automatique ou la méthode *capture*). La liste est définie de façon statique afin d'assurer des entrées de tampon homogènes (toutes les entrées ont la même taille et la même structure). Si la liste d'objets de saisie est modifiée, le *buffer* est vidé. Si le tampon est saisi par d'autres objets "Profile generic", leur *buffer* est également vidé, afin de garantir l'homogénéité de leurs entrées de *buffer*.

Le *buffer* peut être défini en étant trié suivant l'un des *capture objects* (objets de saisie), par exemple l'horloge ou bien les entrées sont empilées (dans l'ordre "dernier entré premier sorti"). Par exemple, il est très facile de construire un "registre de puissance maximum" avec une saisie de profil trié d'une profondeur d'une entrée et trié par un attribut *last_average_value* de "Demand register". Il est tout aussi simple de définir un profil retenant les trois valeurs les plus élevées d'une certaine période.

La taille des données du profil est déterminée par trois paramètres:

- le nombre d'entrées remplies (*entries_in_use*). Il sera de zéro après effacement du profil;
- le nombre maximum d'entrées à conserver (*profile_entries*). Si toutes les entrées sont remplies et une demande capture () se produit, l'entrée la moins importante (selon la méthode de tri demandée) sera perdue. Ce nombre maximum d'entrées peut être spécifié. Lorsqu'il change, le tampon sera réajusté;
- la limite physique du tampon. Cette limite dépend typiquement des objets à saisir. L'objet rejettera une tentative d'établissement du nombre maximum d'entrées à une valeur supérieure à celle physiquement possible.

Profile generic (Profil générique)		0...n	class_id = 7, version = 1			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	buffer (dyn.)	compact-array ou array				x + 0x08
3.	capture_objects (static)	array				x + 0x10
4.	capture_period (static)	double-long-unsigned				x + 0x18
5.	sort_method (static)	enum				x + 0x20
6.	sort_object (static)	capture_object_definition				x + 0x28
7.	entries_in_use (dyn.)	double-long-unsigned	0		0	x + 0x30
8.	profile_entries (static)	double-long-unsigned	1		1	x + 0x38
Méthodes spécifiques		m/o				
1.	reset (data)	o				x + 0x58
2.	capture (data)	o				x + 0x60
3.	reserved from previous versions (réservé provenant de versions précédentes)	o				
4.	reserved from previous versions (réservé provenant de versions précédentes)	o				

Description d'attribut

logical_name Identifie l'instance de l'objet "Profile generic" (Profil générique). Pour des exemples, voir 6.2.17, 6.2.19, 6.2.20, 6.2.33, 6.2.36, 6.2.38, 6.2.49, 6.2.51 6.2.52, 6.3.2, etc.

buffer Contient une séquence d'entrées. Chaque entrée contient les valeurs des objets saisis.

compact-array ou array entrée

entry ::= structure

```
{
    CHOICE
    {
        -- types de données simples
        null-data                              [0],
        boolean                                [3],
        bit-string                             [4],
        double-long                           [5],
        double-long-unsigned                [6],
        octet-string                          [9],
        visible-string                        [10],
        utf8-string                           [12],
        bcd                                    [13],
        integer                                [15],
        long                                   [16],
        unsigned                              [17],
        long-unsigned                        [18],
        long64                                [20],
        long64-unsigned                      [21],
        enum                                  [22],
        float32                               [23],
        float64                               [24],
        date-time                             [25],
        date                                   [26],
        time                                   [27],
        -- types de données complexes
        array                                  [1],
        structure                             [2],
        compact-array                        [19]
    }
}
```

Le nombre et l'ordre des éléments de la structure contenant les entrées sont les mêmes que dans la définition des *capture_objects*. Le *buffer* est rempli par des saisies automatiques ou par des appels ultérieurs à la méthode *capture*. La séquence des entrées dans le tableau est ordonnée selon la *sort_method* spécifiée.

Par défaut: Le *buffer* est vide après une réinitialisation.

REMARQUE 1 La lecture du *buffer* tout entier ne donne que les entrées qui sont "en utilisation".

REMARQUE 2 La valeur d'un objet saisi peut être remplacée par "null-data" si elle peut être récupérée sans ambiguïté à partir de la valeur précédente (par exemple pour le temps: si elle peut être calculée à partir de la valeur précédente et de *capture_period* ou pour une valeur: si elle est égale à la valeur précédente).

l'accès *sélectif* (voir 4.4) à l'attribut *buffer* (tampon) peut être disponible (facultatif). Les paramètres d'accès sélectif sont tels que définis ci-dessous.

capture_objects	Spécifie la liste des objets de saisie qui sont attribués à l'instance d'objet. Sur appel de la méthode <i>capture</i> (data) ou automatiquement dans des intervalles définis, les attributs sélectionnés sont copiés dans le <i>buffer</i> du profil. array capture_object_definition capture_object_definition ::= structure <pre> { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer, data_index: long-unsigned } </pre> - où <i>attribute_index</i> est un pointeur vers l'attribut au sein de l'objet identifié par son <i>class_id</i> et son <i>logical_name</i>. <i>Attribute_index</i> 1 fait référence au 1^{er} attribut (c'est-à-dire <i>logical_name</i>), <i>attribute_index</i> 2 au 2^{ème} attribut, etc.; <i>attribute_index</i> 0 fait référence à tous les attributs publics; - où <i>data_index</i> est un pointeur sélectionnant un élément spécifique de l'attribut. Le premier élément dans l'attribut structure est identifié par <i>data_index</i> 1. Si l'attribut n'est pas une structure, le <i>data_index</i> n'a pas de signification. Si l'objet de saisie est le tampon d'un profil, le <i>data_index</i> identifie l'objet saisi du tampon (c'est-à-dire la colonne) du profil interne; <i>data_index</i> 0: référence l'attribut tout entier.
capture_period	>= 1: Saisie automatique supposée. Spécifie la période de saisie en secondes. 0: Absence de saisie automatique; la saisie est déclenchée extérieurement ou des événements de saisie se produisent de manière asynchrone.
sort_method	Si le profil n'est pas trié, il fonctionne comme un tampon "premier entré, premier sorti" (il est donc trié par saisie, et pas nécessairement par le temps maintenu dans l'objet "Clock"). Si le <i>buffer</i> est plein, l'appel suivant de <i>capture</i> () expulsera la première entrée (la plus ancienne) du <i>buffer</i> pour dégager de la place pour la nouvelle entrée. Si le profil est trié, un appel de <i>capture</i> () stockera la nouvelle entrée à l'emplacement approprié dans le tampon, déplaçant toutes les entrées suivantes et perdant probablement l'entrée la moins intéressante. Si la nouvelle entrée entraine dans le <i>buffer</i> après la dernière entrée, et si le <i>buffer</i> est déjà plein, la nouvelle entrée ne sera pas conservée du tout. enum: (1) fifo (first in first out, c'est-à-dire première entrée, première sortie) (2) lifo (last in first out, c'est-à-dire dernière entrée, première sortie) (3) largest (c'est-à-dire la plus grande), (4) smallest (c'est-à-dire la plus petite), (5) nearest_to_zero (c'est-à-dire la plus proche de zéro), (6) farthest_from_zero (la plus éloignée de zéro) Def. fifo
sort_object	Si le profil est trié, cet attribut spécifie le registre ou l'horloge servant de base à l'ordre de classement. capture_object_definition Voir ci-dessus.

Def. aucun objet par lequel trier (possible seulement possible avec `sort_method` `fifo` ou `lifo`)

NOTE 1 Si la `sort_method` est `FIFO` ou `LIFO`, tous les éléments de la `capture_object_definition` spécifiant l'objet de tri peuvent être zéro.

entries_in_use Compte le nombre d'entrées stockées dans le tampon. Après un appel de la méthode `reset` (), le tampon ne contient pas d'entrées, et cette valeur est zéro. Sur chaque appel ultérieur de la méthode `capture` (), cette valeur sera incrémentée jusqu'au nombre maximum d'entrées qui seront stockées (voir `profile_entries`).

double-long-unsigned 0...`profile_entries`

Def. 0

profile_entries Spécifie le nombre des entrées qui doivent être conservées dans le tampon.

double-long-unsigned 1...(limité par la taille physique)

Def. 1

Paramètres pour l'accès sélectif à l'attribut tampon

Sélecteur d'accès	Paramètre d'accès	Commentaire
1	<code>range_descriptor</code>	Seuls les éléments de tampon correspondant au <code>range_descriptor</code> doivent être retournés dans la réponse.
2	<code>entry_descriptor</code>	Seuls les éléments de tampon correspondant à l' <code>entry_descriptor</code> doivent être retournés dans la réponse.

range_descriptor ::= structure

```
{
    restricting_  capture_object_definition  Définit le capture_object limitant la plage des
    object:                                           entrées à récupérer. Seuls les types de
                                                    données simples sont autorisés.

    from_value:                                     L'entrée à récupérer la plus ancienne ou la
                                                    plus petite

                                                    CHOICE
                                                    {
                                                    -- types de données simples
                                                    double-long [5],
                                                    double-long-unsigned [6],
                                                    octet-string [9],
                                                    visible-string [10],
                                                    utf8-string [12],
                                                    integer [15],
                                                    long [16],
                                                    unsigned [17],
                                                    long-unsigned [18],
                                                    long64 [20],
                                                    long64-unsigned [21],
                                                    float32 [23],
                                                    float64 [24],
                                                    date-time [25],
                                                    date [26],
                                                    time [27]
                                                    }

    to_value: CHOICE                               Entrée à récupérer la plus récente ou la plus
    {voir ci-dessus}                               grande

    selected_  array                               Liste de colonnes à récupérer. Si le tableau
    values:    capture_object_definition            est vide (n'a pas d'entrées), toutes les
                                                    données saisies sont retournées. Autrement,
                                                    seules les colonnes spécifiées dans le
                                                    tableau sont retournées. Le type
                                                    capture_object_definition est spécifié ci-
                                                    dessus (capture_objects).
}
```

entry_descriptor ::= structure

```
{
    from_entry: double-long-unsigned  première entrée à récupérer,
    to_entry:   double-long-unsigned  dernière entrée à récupérer
                                                    to_entry == 0: l'entrée la plus élevée
                                                    possible,
    from_selected_value long-unsigned  indice de la première valeur à récupérer,
    :
    to_selected_value:  long-unsigned  indice de la dernière valeur à récupérer
                                                    to_selected_value == 0: selected_value la plus
                                                    élevée possible
}
```

NOTE 2 from_entry et to_entry identifient les lignes, from_selected_value et to_selected_value identifient les colonnes du tampon à récupérer.

NOTE 3 La numérotation des entrées et des valeurs sélectionnées commence à 1.

Description de la méthode

reset (data)	Vide le <i>buffer</i>. Il n'a pas d'entrées valides après cela. <i>entries_in_use</i> est zéro après cet appel. Cet appel ne déclenche pas d'opérations supplémentaires sur les objets de saisie. Précisément, il ne réinitialise pas les attributs saisis. <i>data</i>::= integer (0)
capture (data)	Copie les valeurs des objets à saisir dans le <i>buffer</i> en lisant chaque objet de saisie. En fonction de la <i>sort_method</i> et de l'état réel du <i>buffer</i>, elle produit une nouvelle entrée ou un remplacement pour l'entrée la moins significative. Tant que toutes les entrées n'ont pas été déjà utilisées, l'attribut <i>entries_in_use</i> sera incrémenté. Cet appel ne déclenche pas d'opérations supplémentaires au sein des objets de saisie telles que <i>capture ()</i> ou <i>reset ()</i>. A noter que, s'il est nécessaire de saisir plusieurs attributs d'un objet, ces attributs sont à définir un à un sur la liste <i>capture_objects</i>. Si <i>attribute_index</i> = 0, tous les attributs sont saisis. <i>data</i>::= integer (0)

Comportement de l'objet après modification de certains attributs

Toute modification de l'un des *capture_objects* décrivant la structure statique du *buffer* appellera automatiquement un *reset ()*, et cet appel se propagera à tous les autres profils saisissant ce profil.

En cas de tentative d'écriture dans *profile_entries* d'une valeur trop grande pour le tampon, celle-ci sera rejetée.

Restrictions

Lors de la définition des *capture_objects*, toute référence circulaire au profil doit être évitée.

Profil utilisé pour définir un sous-ensemble de valeurs de lecture préférentielle

En attribuant la valeur 1 à *profile_entries*, un objet "Profile generic" peut être utilisé pour définir un ensemble de valeurs de lecture préférentielle. Voir également 6.2.17. L'attribution de la valeur 1 à *capture_period* assure que les valeurs sont mises à jour toutes les secondes.

5.2.7 Utility tables (class_id = 26, version = 0)

Cette IC permet d'encapsuler des données en tables de l'ANSI C12.19. Chaque "table" est représentée par une instance de cette IC, identifiée par son *nom logique*.

Utility tables (Tables de fournisseurs de service d'énergie)	0...n	class_id = 26, version = 0			
Attributs	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. table_ID (static)	long-unsigned				x + 0x08
3. length	double-long-unsigned				x + 0x10
4. buffer	octet-string				x + 0x18
Méthodes spécifiques	m/o				

Description d'attribut	
logical_name	Identifie l'instance de l'objet "Utility tables" (Table de fournisseurs de service d'énergie). Voir 6.2.32.
table_ID	Numéro de table. Ce numéro de table est tel que spécifié dans la norme ANSI et peut être soit une table standard, soit une table de fabricant.
length	Nombre d'octets dans le tampon de la table.
buffer	Contenu de table. <i>L'accès sélectif (voir 4.4) à l'attribut buffer peut être disponible (facultatif). Les paramètres d'accès sélectif sont définis ci-dessous.</i>

Paramètres pour l'accès sélectif à l'attribut tampon

Sélecteur d'accès	Paramètre	Commentaire
1	offset_access	Accès à la table par offset (c'est-à-dire décalage) et count (c'est-à-dire compteur) utilisant l'offset_selector pour les données de paramètre.
2	index_access	Accès à la table par id d'élément et nombre d'éléments en utilisant index_selector pour les données de paramètre.

offset_selector::=
structure

{

Offset: double-long-unsigned Décalage en octets au début de la zone d'accès, par rapport au début de la table.

Count: long-unsigned Nombre d'octets demandés ou transférés

}

index_selector::=
structure

{

Index: array long-unsigned Séquence d'indices pour identifier des éléments dans la hiérarchie de la table.

Count: long-unsigned Nombre d'éléments demandés ou transférés. Les valeurs de "count" supérieures à 1 retournent autant d'éléments au maximum. Une valeur de zéro, lorsqu'elle est donnée dans le contexte d'une demande, se réfère au sous-arbre entier de la hiérarchie commençant au point de sélection.

}

5.2.8 Register table (class_id = 61, version = 0)

Cette IC permet de regrouper des entrées homogènes, des attributs identiques d'objets multiples, qui sont tous des instances de la même IC. De plus, dans leur *logical_name* (code OBIS), la valeur dans les groupes de valeurs A à D et F est identique. Les valeurs possibles dans le groupe de valeurs E sont définies dans l'IEC 62056-6-1: sous forme de tableau: l'entête de tableau définit la partie commune du code OBIS et chaque cellule du tableau définit une valeur possible du groupe de valeurs E. Un objet "Register table" (Tableau de registre) peut saisir les attributs de tout ou partie de ces objets.

NOTE 1 Certains exemples sont le tableau "Extended phase angle measurement" (Mesure d'angle de phase étendu), voir l'IEC 62056-6-1:; Tableau 16 ou le tableau "UNIPED voltage dip quantities" (Grandeurs de creux de tension UNIPED), voir l'IEC 62056-6-1:; Tableau 18.

NOTE 2 Si une fonctionnalité plus complexe est nécessaire, l'IC "Profile generic" peut être utilisée.

Register table (Tableau de registre)	0...n	class_id = 61, version = 0			
Attributes	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. table_cell_values (dyn.)	compact-array ou array				x + 0x08
3. table_cell_definition (static)	structure				x + 0x10
4. scaler_unit (static)	scaler_unit_type				x + 0x18
Méthodes spécifiques	m/o				
1. reset (data)	f				x + 0x28
2. capture (data)	f				x + 0x30

Description d'attribut

logical_name	Identifie l'instance de l'objet "Register table" (Tableau de registre). Lorsque le format du <i>logical_name</i> est A.B.C.D.255.F, les valeurs A à D et F définissent la partie commune du nom logique des objets, dont les attributs sont saisis. Un seul attribut des objets concernés peut être saisi (par exemple l'attribut <i>value</i>). Lorsque le format du <i>logical_name</i> est A.B.98.10.x.255, plusieurs instances de l'IC "Register table" peuvent être utilisées pour saisir différents attributs des objets concernés. Le groupe de valeurs E numérote les instances. Voir 6.4 de l'IEC 62056-6-1:.
table_cell_values	Contient la valeur des attributs saisis, telle qu'elle serait retournée sur une GET ou Read request aux attributs individuels. compact array ou array table_cell_entry


```

table_cell_entry::= CHOICE
{
    -- types de données simples
    null-data          [0],
    bit-string         [4],
    double-long        [5],
    double-long-unsigned [6],
    octet-string       [9],
    visible-string     [10],
    utf8-string        [12],
    bcd                [13],
    integer            [15],
    long               [16],
    unsigned           [17],
    long-unsigned      [18],
    long64             [20],
    long64-unsigned    [21],
    float32            [23],
    float64            [24],
    -- types de données complexes
    structure          [2]
}

```

Si l'attribut saisi est `attribute_0`, les valeurs redondantes peuvent être remplacées par "null-data", si leur valeur peut être récupérée sans ambiguïté (par exemple: `scaler_unit`).

table_cell_definition

Spécifie la liste des attributs saisis dans le tableau de registre.

```

structure
{
    class_id:      long-unsigned,
    logical_name:  octet-string,
    group_E_values: array unsigned,
    attribute_index: integer
}

```

où:

- `class_id` définit le `class_id` commun à tous les objets dont les attributs sont saisis;
- `logical_name` contient le nom logique commun à tous les objets, avec E = 255 (caractère générique);
- les `group_E_values` contiennent la liste des identifiants de cellules, de type *unsigned*, tels que définis dans le tableau correspondant de l'IEC 62056-6-1;
- `attribute_index` est un pointeur sur l'attribut au sein de l'objet. `attribute_index 0` se réfère à tous les attributs publics.

Si le *logical_name* de l'objet "Register table" est au format A.B.C.D.255.F et les attributs définis de tous les objets identifiés dans le tableau correspondant de l'IEC 62056-6-1: sont saisis, l'attribut 3 peut ne pas être accessible. Dans ce cas:

- le `class_id` doit être 1 "Data", 3 "Register" ou 4 "Extended register";
- le *logical_name* des objets à saisir est défini par le *logical_name* de l'objet "Register table" et le tableau correspondant de l'IEC 62056-6-1;
- l'indice d'attribut doit être 2 (value).

scaler_unit	Voir la description de l'IC "Register". Si les attributs "value" des objets "Register" ou "Extended register" sont saisis, l'attribut <i>scaler_unit</i> doit être commun à tous les objets et cet attribut doit contenir une copie. Si d'autres attributs ou IC sont saisi(e)s, l'attribut <i>scaler_unit</i> n'a pas de signification et doit être inaccessible.

Description de la méthode

reset (data)	Efface les <i>table_cell_values</i> . Il n'a pas d'effet sur les attributs saisis. data::= integer (0)
capture (data)	Copie les valeurs des attributs dans les <i>table_cell_values</i> . Si attribute_index = 0, tous les attributs sont saisis.

Comportement de l'objet après modification de l'attribut *table_cell_definition*

Toute modification apportée à cet attribut appellera automatiquement la méthode *reset (data)* et elle sera propagée à tous les profils capturant cet objet.

En cas de tentative d'écriture dans *table_cell_definition* d'une valeur trop grande pour le tampon contenant l'attribut *table_cell_values*, elle sera rejetée.

5.2.9 Status mapping (class_id = 63, version = 0)

Cette IC permet de modéliser la mise en correspondance des bits dans un mot d'état à des entrées dans un tableau de référence.

Status mapping (Mise en correspondance d'état)		0...n	class_id = 63, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	status_word (dyn.)	CHOICE				x + 0x08
3.	mapping_table (static)	structure				x + 0x10
Méthodes spécifiques		m/o				

Description d'attribut

logical_name Identifie les instances de l'objet "Status mapping". Voir 6.2.36, 6.2.38, 6.2.44 et 6.3.7

status_word Contient la valeur courante du mot d'état.

CHOICE

```
{
    bit-string [4],
    double-long-unsigned [6],
    octet-string [9],
    visible-string, [10],
    utf8-string [12],
    unsigned [17],
    long-unsigned [18],
    long64-unsigned [21]
}
```

La taille du *status_word* est de $n \times 8$ bits, la taille maximale est de 65 536 bits.

NOTE Les fabricants peuvent choisir n'importe lesquels des types énumérés ci-dessus. Cependant, le mot d'état est toujours interprété comme un bit-string (chaîne binaire).

Contient la mise en correspondance du *status_word* avec les positions dans le tableau de référence.

structure

```
{
    ref_table_id: unsigned,
    ref_table_mapping: CHOICE
    {
        long-unsigned [18],
        array long-unsigned [1]
    }
}
```

où:

- *ref_table_id* identifie le tableau de statuts de référence;
- si "long-unsigned" est choisi, la valeur pointe vers une entrée du tableau de référence. Cette entrée est mise en correspondance avec le premier bit du mot d'état. L'entrée suivante est mise en correspondance avec le bit suivant, et ainsi de suite. La dernière entrée mise en correspondance avec le dernier bit est déterminée par la longueur du mot d'état;
- si "array" est choisi, les éléments du tableau pointent vers les entrées du tableau de statut de référence. L'ordre des éléments dans le tableau correspond à la position du mot d'état. Le premier élément du tableau met en correspondance l'entrée de tableau référencée avec le premier bit, et le dernier élément avec le dernier bit.

5.3 Classes d'interfaces pour contrôle d'accès et gestion

5.3.1 Vue d'ensemble

Les classes d'interfaces de cette catégorie modélisent la structure logique du serveur DLMS/COSEM, ce qui permet de configurer et gérer l'accès à ses ressources, de mettre à jour le firmware et de gérer la sécurité:

- les classes "Association SN" class (voir 5.3.3) et "Association LN" (voir 6.3.4) modélisent les AA. Leurs instances, les objets d'association, fournissent la liste des objets

accessibles dans chaque AA, gèrent et contrôlent les droits d'accès à leurs attributs et méthodes. Ils gèrent également l'authentification des partenaires de communication;

- la classe "SAP Assignment" (voir 6.3.5) modélise la structure logique du serveur;
- la classe "Image transfer" (voir 6.3.6) modélise le processus de mise à jour du firmware;
- la classe "Security setup" (voir 5.3.7) modélise les éléments du contexte de sécurité. Les objets "Security setup" sont référencés à partir des objets "Association" et permettent de configurer les suites et règles de sécurité, et de gérer le matériel de sécurité.
- La classe "Push setup" (voir 5.3.8) modélise l'opération "Push" du serveur

5.3.2 Identification de l'utilisateur du client

Cette nouvelle fonction permet au serveur de distinguer les différents utilisateurs côté client et de consigner leurs activités d'accès au compte.

Chaque AA établie entre un client et un serveur peut être utilisée par plusieurs utilisateurs côté client. Les propriétés de l'AA sont configurées dans le serveur à l'aide des objets "Association" et "Security setup". Tous les utilisateurs d'une AA côté client utilisent ces mêmes propriétés.

NOTE 1 Les clés de sécurité sont connues du client et du serveur, mais elles n'ont pas besoin d'être connues des utilisateurs du client.

La liste des utilisateurs (identifiés par leur `user_id` et `user_name`) est connue du client et du serveur. Elle est gérée dans le serveur par l'attribut `user_list` des objets "Association".

NOTE 2 La manière dont un client authentifie un utilisateur qui se connecte au système client est hors du domaine d'application de la présente spécification.

Lors de l'établissement de l'AA, l'`user_id` (appartenant au `user_name`) se trouve dans le champ calling-AE-invocation-id de l'AARQ APDU. Si l'`user_id` fourni se trouve dans l'`user_list`, l'AA peut être établie (à condition que toutes les autres conditions soient satisfaites) et l'attribut `current_user` est mis à jour. La valeur de cet attribut peut être consignée.

Si le serveur ne "connait" pas l'utilisateur, l'AA ne doit pas être établie. Le serveur peut écarter en silence la requête d'établissement de l'AA ou renvoyer un message d'erreur approprié.

Le processus d'identification d'utilisateur est facultatif: si l'`user_list` est vide (c'est-à-dire un tableau de 0 élément), la fonction est désactivée.

5.3.3 Association SN (`class_id` = 12, `version` = 3)

NOTE 1 Cette nouvelle version 3 de l'IC "Association SN" prend en charge le processus d'identification d'utilisateur du client, voir 5.3.2.

Les dispositifs logiques COSEM capables d'établir des AA au sein d'un contexte COSEM en utilisant le référencement par SN modélisent les AA à l'aide des instances de l'IC "Association SN". Un dispositif logique COSEM peut avoir une instance de cette IC pour chaque AA que le dispositif est capable de prendre en charge.

Le **short_name** de l'objet "Association SN" lui-même est fixé dans le contexte COSEM. Voir 4.3.

Association SN (SN d'association)		0...n	class_id = 12, version = 3			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	object_list (static)	objlist_type				x + 0x08
3.	access_rights_list (static)	access_rights_type				x + 0x10
4.	security_setup_reference (static)	octet-string				x + 0x18
5.	user_list (static)	array				x + 0x20
6.	current_user	structure				x + 0x28
Méthodes spécifiques		m/o				
1.	reserved from previous versions (réservé provenant de versions précédentes)	o				
2.	reserved from previous versions (réservé provenant de versions précédentes)	o				
3.	read_by_logicalname (data)	o				x + 0x30
4.	reserved from previous versions (réservé provenant de versions précédentes)	o				
5.	change_secret (data)	o				x + 0x40
6.	reserved from previous versions (réservé provenant de versions précédentes)	o				
7.	reserved from previous versions (réservé provenant de versions précédentes)					
8.	reply_to_HLS_authentication (data)	o				x + 0x58
9.	add_user (data)	o				x + 0x60
10.	remove_user (data)	o				x + 0x68

Description d'attribut

logical_name Identifie l'instance de l'objet "Association SN" (SN d'association). Voir 6.2.27.

object_list Contient la liste de tous les objets avec leurs base_name (short_name), class_id, version et logical_name. Le base_name est l'objectName DLMS du premier attribut (logical_name).

objlist_type ::= array objlist_element

objlist_element ::= structure

```
{
    base_name:      long,
    class_id:       long-unsigned,
    version:        unsigned,
    logical_name:   octet-string
}
```

l'accès sélectif (voir 4.4) à l'attribut object_list peut être disponible. Les valeurs de sélecteur d'accès et leurs paramètres sont tels que définis ci-dessous.

**access_
rights_list**

Contient les droits d'accès aux attributs et méthodes.

La liaison entre l'*object_list* et l'*access_rights_list* est le *base_name*, présent à la fois dans la structure *objlist_element* et la structure *access_right_element*. Par conséquent, les *base_names* sur les deux listes doivent être les mêmes. Le nombre (et de préférence, l'ordre) des éléments doit être le même dans le tableau d'*objlist_element* et dans le tableau d'*access_right_element*.

access_rights_type::= array *access_rights_element*

access_rights_element::= structure

```
{
    base_name:                long,
    attribute_access:          attribute_access_descriptor,
    method_access:            method_access_descriptor
}
```

attribute_access_descriptor::= array *attribute_access_item*

attribute_access_item::= structure

```
{
    attribute_id:              integer,
    access_mode:               enum:
        (0) no_access,
        (1) read_only,
        (2) write_only,
        (3) read_and_write,
        (4) authenticated_read_only,
        (5) authenticated_write_only,
        (6) authenticated_read_and_write

    access_selectors:          CHOICE
    {
        null_data              [0],
        array integer          [1]
    }
}
```

method_access_descriptor::= array *method_access_item*

method_access_item::= structure

```
{
    method_id:                integer,
    access_mode:               enum:
        (0) no_access,
        (1) access,
        (2) authenticated_access
}
```

l'accès sélectif (voir 4.4) à l'attribut *access_rights_list* peut être disponible (facultatif). Les valeurs de sélecteur d'accès et leurs paramètres sont tels que définis ci-dessous.

**security_
setup_
reference**

Référence l'objet "Security setup" par son *logical_name*. L'objet référencé gère la sécurité pour une instance d'objet donnée de l'"Association SN".

user_list	Contient la liste des utilisateurs autorisés à utiliser l'AA gérée par l'instance donnée de l'IC "Association SN". array user_list_entry user_list_entry ::= structure { user_id: unsigned, user_name: visible-string } où: – user_id est l'identifiant de l'utilisateur (cette valeur se trouve dans le champ calling-AE-invocation-id de l'AARQ); – user_name est le nom de l'utilisateur. Si l'attribut user_list est vide (c'est-à-dire tableau de 0 élément), tous les utilisateurs peuvent utiliser l'AA, c'est-à-dire que le champ calling-AE-invocation-id de l'AARQ est ignoré. Si l'attribut user_list n'est pas vide, seuls les utilisateurs présents dans la liste peuvent établir l'AA, c'est-à-dire que le champ calling-AE-invocation-id de l'AARQ doit être présent et que sa valeur doit correspondre à l'un des user_ids de l'user_list, sinon, l'AA n'est pas établie.
current_user	Contient l'identifiant de l'utilisateur en cours. current_user ::= user_list_entry (voir ci-dessus) Si l'user_list est vide, le current_user doit être une structure {user_id: unsigned 0, user_name: chaîne visible de 0 élément}

Paramètres pour accès sélectif à l'attribut *object_list* et *access_rights_list*

Valeur de sélecteur d'accès	Paramètre	Disponible avec l'attribut	Commentaire
1	class_id: long-unsigned	2	Délivre le sous-ensemble de l'object_list pour une class_id spécifique. Pour la réponse: data::= objlist_type
2	structure { class_id: long-unsigned, logical_name: octet-string }	2	Délivre l'entrée de l'object_list pour un class_id et un logical_name spécifiques. Pour la réponse: data::= objlist_element
3	base_name: long	2, 3	Dans le cas de l'attribut 2, délivre l'entrée de l'object_list pour un base_name spécifique. Pour la réponse: data::= objlist_element Dans le cas de l'attribut 3, délivre l'entrée de l'access_rights_list pour un base_name spécifique. Pour la réponse: data::=access_rights_element

Description de la méthode

read_by_logicalname (data) Lit les attributs pour les objets sélectionnés. Les objets sont spécifiés par leur *class_id* et leur *logical_name*. Avec cette méthode, la caractéristique d'accès paramétré peut aussi être utilisée.

data::= array *attribute_identification*

```
attribute_identification::= structure
{
    class_id:          long-unsigned,
    logical_name:      octet-string,
    attribute_index:   integer
}
```

où *attribute_index* est un pointeur (à savoir, le décalage) vers l'attribut au sein de l'objet.

attribute_index 0 délivre tous les attributs; *attribute_index* 1 délivre le premier attribut (c'est-à-dire: *logical_name*), etc.).

Pour la réponse: les données sont selon le type de l'attribut.

NOTE 2 Si au moins un attribut n'a pas de droit d'accès en lecture dans le cadre de l'association courante, un *read_by_logicalname* () à l'indice d'attribut 0 révèle le message d'erreur "scope-of-access-violated" ("champ d'accès violé". Voir l'Article 8 de l'IEC 62056-5-3:).

change_secret (data) Change le secret LLS ou HLS (par exemple: mot de passe).

data::= octet-string *new secret*

NOTE 3 La structure de "new secret" dépend du mécanisme de sécurité mis en œuvre. Le "new secret" peut contenir des bits de contrôle supplémentaires et il peut être chiffré.

NOTE 4 Dans le cas de HLS avec GMAC, le (HLS_) secret est détenu par l'objet "Security setup" référencé dans l'attribut.

reply_to_HLS_authentication (data) L'appel à distance de cette méthode délivre au serveur le résultat du traitement de secret par le client du défi du serveur au client, *f*(StoC), comme le paramètre de service *data* (donnée) de la primitive *Read.request* appelée avec l'accès paramétré.

data::= octet-string réponse du client au défi

Si l'authentification est acceptée, la réponse (primitive *Read.confirm*) contient *Result* == OK et le résultat du traitement de secret par le serveur du défi du client au serveur, *f*(CtoS) dans le paramètre de service *data* (données) du service *Read.response*.

data::= octet-string réponse du serveur au défi

Si l'authentification n'est pas acceptée, le paramètre résultat dans la réponse doit contenir une valeur non-OK et aucune donnée ne doit être renvoyée.

add_user (data) Ajoute un utilisateur à l'*user_list*.

data::= *user_list_entry* (voir ci-dessus)

remove_user (data) Supprime un utilisateur de l'*user_list*.

data::= *user_list_entry* (voir ci-dessus)

5.3.4 Association LN (*class_id* = 15, *version* = 2)

NOTE 1 Cette nouvelle version 2 de l'IC "Association LN" prend en charge le processus d'identification d'utilisateur du client, voir 5.3.2.

Les dispositifs logiques COSEM capables d'établir des AA au sein d'un contexte COSEM en utilisant le référencement par LN modélisent les AA à travers les instances de l'IC "Association LN". Un dispositif logique COSEM a une instance de cette IC pour chaque AA que le dispositif est capable de prendre en charge.

Association LN (LN d'association)	0...MaxNbOfAss.	class_id = 15, version = 2			
Attributs	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. object_list (static)	object_list_type				x + 0x08
3. associated_partners_id	associated_partners				x + 0x10
4. application_context_name	context_name_type				x + 0x18
5. xDLMS_context_info	xDLMS_context_type				x + 0x20
6. authentication_mechanism_name	mechanism_name_type				x + 0x28
7. secret	octet-string				x + 0x30
8. association_status	enum				x + 0x38
9. security_setup_reference (static)	octet-string				x + 0x40
10. user_list (static)	array				x + 0x48
11. current_user	structure				x + 0x50
Méthodes spécifiques	m/o				
1. reply_to_HLS_authentication (data)	o				x + 0x60
2. change_HLS_secret (data)	o				x + 0x68
3. add_object (data)	o				x + 0x70
4. remove_object (data)	o				x + 0x78
5. add_user (data)	o				x + 0x80
6. remove_user (data)	o				x + 0x88

Description d'attribut

logical_name Identifie l'instance de l'objet "Association LN" (LN d'association). Voir 6.2.27.

object_list Contient la liste des objets COSEM visibles avec leurs class_id, version, logical_name et les droits d'accès à leurs attributs et méthodes au sein de l'AA donnée.

object_list_type ::= array object_list_element

object_list_element ::= structure

```
{
    class_id:          long-unsigned,
    version:           unsigned,
    logical_name:      octet-string,
    access_rights:     access_right
}
```

object_list (suite)	<pre> access_right ::= structure { attribute_access: attribute_access_descriptor, method_access: method_access_descriptor } attribute_access_descriptor ::= array attribute_access_item attribute_access_item ::= structure { attribute_id: integer, access_mode: enum: (0) no_access, (1) read_only, (2) write_only, (3) read_and_write, (4) authenticated_read_only, (5) authenticated_write_only, (6) authenticated_read_and_write access_selectors: CHOICE { null-data [0], array integer [1] } } method_access_descriptor ::= array method_access_item method_access_item ::= structure { method_id: integer, access_mode: enum: (0) no_access, (1) access, (2) authenticated_access } </pre> où: - les éléments <code>attribute_access_descriptor</code> et <code>method_access_descriptor</code> contiennent toujours tous les attributs ou toutes les méthodes mis(es) en œuvre; - les <code>access_selectors</code> contiennent une liste des valeurs de sélecteur prises en charge. l'accès <i>sélectif</i> (voir 4.4) à l'attribut <i>object_list</i> peut être disponible (facultatif). Les paramètres d'accès sélectif sont tels que définis ci-dessous.

**associated_
partners_id**

Contient les identifiants des AP (de dispositif logique) client et serveur COSEM au sein des dispositifs physiques hébergeant ces AP, qui appartiennent à l'AA modélisée par l'objet "Association LN" (LN d'association).

associated_partners_type ::= structure

```
{
    client_SAP:      integer,
    server_SAP:      long-unsigned
}
```

La plage pour le client_SAP est 0...0x7F.

La plage pour le server_SAP est 0x0000...0x3FFF.

Les SAP doivent se situer dans la plage permise par le type de données et les supports.

**application_
context_name**

Dans l'environnement COSEM, il est prévu qu'un contexte d'application préexiste et soit référencé par son nom pendant l'établissement d'une AA. Cet attribut contient le nom du contexte d'application pour l'AA en question.

```
context_name_type ::= CHOICE
{
    context_name_structure [2],
    octet-string           [9]
}
```

Le nom du contexte d'application est spécifié comme appartenant au type OBJECT IDENTIFIER dans l'IEC 62056-5-3, 7.2.2.2.

Si le context_name_type est codé comme une structure, il contient les "arc labels" (étiquettes d'arc) de l'OBJECT IDENTIFIER.

context_name_structure ::= structure

```
{
    joint_iso_ctt_element:      unsigned,
    country_element:           unsigned,
    country_name_element:      long-unsigned,
    identified_organization_element: unsigned,
    DLMS_UA_element:          unsigned,
    application_context_element: unsigned,
    context_id_element:        unsigned
}
```

Exemple 1: Dans le cas du context_id(1), le codage A-XDR est: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 01 11 01 (toutes les valeurs sont hexadécimales).

Si le context_name_type est codé comme une chaîne d'octets, il contient la valeur l'OBJECT IDENTIFIER. Voir l'IEC 62056-5-3, Article C.4.

Exemple 2: Dans le cas du context_id(1), le codage A-XDR est: 09 07 60 85 74 05 08 01 01 (toutes les valeurs sont hexadécimales).

**xDLMS_
context_info**

Contient toutes les informations nécessaires sur le contexte xDLMS pour l'AA donnée.

xDLMS_context_type ::= structure

```
{
    conformance:                bit-string,
    max_receive_pdu_size:        long-unsigned,
    max_send_pdu_size:           long-unsigned,
    dlms_version_number:         unsigned,
    quality_of_service:           integer,
    cyphering_info:              octet-string
}
```

où:

- l'élément conformance (conformité) contient le bloc de conformité xDLMS pris en charge par le serveur. La longueur de la chaîne binaire est de 24 bits.
- l'élément max_receive_pdu_size contient la longueur maximale pour une APDU xDLMS, exprimée en octets, que le client peut envoyer. Il est le même que le paramètre server-max-receive-pdu-size de l'APDU initiateResponse xDLMS;
- l'élément max_send_pdu_size, dans une AA active, contient la longueur maximale pour une APDU xDLMS, exprimée en octets, que le serveur peut envoyer. Il est le même que le paramètre client-max-receive-pdu-size de l'APDU initiateResponse xDLMS;
- l'élément dlms_version_number contient le numéro de version de DLMS pris en charge par le serveur;
- l'élément quality_of_service n'est pas utilisé;
- l'élément cyphering_info, dans une AA active, contient le paramètre clé dédié de l'APDU initiateRequest xDLMS. Voir l'Article 8 de l'IEC 62056-5-3:.

authentication_mechanism_name	Contient le nom du mécanisme d'authentification pour l'AA. <pre> mechanism_name_type ::= CHOICE { mechanism_name_structure [2], octet-string [9] } </pre> Le nom du mécanisme d'authentification est spécifié comme appartenant au type OBJECT IDENTIFIER dans l'IEC 62056-5-3:, 7.2.2.3. Si le <code>mechanism_name_type</code> est codé comme une structure, il contient les "arc labels" (étiquettes d'arc) de l'OBJECT IDENTIFIER. <pre> mechanism_name_structure ::= structure { joint_iso_ctt_element: unsigned, country_element: unsigned, country_name_element: long- unsigned, identified_organization_element: unsigned, DLMS_UA_element: unsigned, authentication_mechanism_name_element: unsigned, mechanism_id_element: unsigned } </pre> Exemple 3 Dans le cas du <code>mechanism_id(1)</code>, le codage A-XDR est: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 02 11 01 (toutes les valeurs sont hexadécimales): Si le <code>mechanism_name_type</code> est codé comme une chaîne d'octets, il contient la valeur l'OBJECT IDENTIFIER. Voir l'IEC 62056-5-3:, Article C.4. EXEMPLE 4: Dans le cas du <code>mechanism_id(1)</code>, le codage A-XDR est: 09 07 60 85 74 05 08 02 01 (toutes les valeurs sont hexadécimales). Aucun <code>mechanism_name</code> n'est exigé lorsqu'il n'est pas utilisé d'authentification.
secret	Contient le secret pour le processus d'authentification LLS ou HLS. NOTE 2 Dans le cas de HLS avec GMAC, le (HLS_)secret est détenu par l'objet "Security setup" référencé dans l'attribut 9, <i>security_setup_reference</i>.
association_status	Indique l'état courant de l'association, qui est modélisé par l'objet. <pre> enum: (0) non-associated, (1) association-pending, (2) associated </pre>
security_setup_reference	Référence un objet "Security setup" par son nom logique. L'objet référencé gère la sécurité pour une instance d'objet donnée de l'"Association LN".

user_list	Contient la liste des utilisateurs autorisés à utiliser l'AA gérée par l'instance donnée de l'IC "Association LN". <pre>array user_list_entry user_list_entry ::= structure { user_id: unsigned, user_name: visible-string } où: – user_id est l'identifiant de l'utilisateur (cette valeur se trouve dans le champ calling-AE-invocation-id de l'AARQ); – user_name est le nom de l'utilisateur.</pre> Si l'attribut <i>user_list</i> est vide (c'est-à-dire tableau de 0 élément), tous les utilisateurs peuvent utiliser l'AA, c'est-à-dire que le champ calling-AE-invocation-id de l'AARQ est ignoré. Si l'attribut <i>user_list</i> n'est pas vide, seuls les utilisateurs présents dans la liste peuvent établir l'AA, c'est-à-dire que le champ calling-AE-invocation-id de l'AARQ doit être présent et que sa valeur doit correspondre à l'un des <i>user_ids</i> de l'<i>user_list</i>, sinon, l'AA n'est pas établie.
current_user	Contient l'identifiant de l'utilisateur en cours. <pre>current_user ::= user_list_entry (voir ci-dessus)</pre> Si l'<i>user_list</i> est vide, le <i>current_user</i> doit être une structure {<i>user_id</i>: unsigned 0, <i>user_name</i>: chaîne visible de 0 élément}

Paramètres pour l'accès sélectif à l'attribut *object_list*

- Si aucun accès sélectif n'est demandé (aucun paramètre *Access_Selection_Parameters* n'est présent dans la primitive de service GET.request (.indication) pour l'attribut *object_list*), le service.response (.confirmation) correspondant doit contenir tous les *object_list_elements* de l'attribut *object_list*.
- Lorsqu'un accès sélectif est demandé à l'attribut *object_list* (le paramètre *Access_Selection_Parameter* est présent), la réponse doit contenir une liste "filtrée" d'*object_list_elements*, comme suit:

Sélecteur d'accès	Paramètre d'accès	Commentaire
1	NULL	Toutes les informations, à l'exclusion des <code>access_rights</code> , doivent être incluses dans la réponse.
2	<code>class_list</code>	Accès par <code>class_id</code> . Dans ce cas, seuls doivent être inclus dans la réponse les <code>object_list_elements</code> de l' <i>object_list</i> qui ont un <code>class_id</code> égal à l'un des <code>class_id</code> de la <code>class_list</code> . Aucune information <code>access_right</code> n'est incluse. <code>class_list::= array class_id</code> <code>class_id: long-unsigned</code>
3	<code>object_id_list</code>	Accès par objet. L'enregistrement d'informations complet des instances d'objet sur l' <code>object_id_list</code> doit être retourné. <code>object_id_list::= array object_id</code> <code>object_id::= structure</code> { <code>class_id:</code> long-unsigned, <code>logical_name:</code> octet-string }
4	<code>object_id</code>	L'enregistrement d'informations complet de l'instance d'objet COSEM requise doit être retourné. <code>object_id::= structure</code> Voir ci-dessus.

Description de la méthode

reply_to_HLS_authentication (data)

L'appel à distance de cette méthode délivre au serveur le résultat du traitement de secret par le client du défi du serveur au client, `f(StoC)`, comme le paramètre de service *data* (données) de la primitive ACTION.request appelée.

`data::= octet-string` réponse du client au défi

Si l'authentification est acceptée, la réponse (primitive ACTION.confirm) contient `Result == OK` et le résultat du traitement de secret par le serveur du défi du client au serveur, `f(CtoS)` dans le paramètre de service *data* (données) du service de réponse.

`data::= octet-string` réponse du serveur au défi

Si l'authentification n'est pas acceptée, le paramètre résultat dans la réponse doit contenir une valeur non-OK et aucune donnée ne doit être renvoyée.

change_HLS_secret (data)

Change le secret HLS (par exemple: clé de chiffrement).

`data::= octet-string` nouveau secret HLS

La structure du "new secret" dépend du mécanisme de sécurité mis en œuvre. Le "new secret" peut contenir des bits de contrôle supplémentaires et il peut être chiffré.

add_object (data)

Ajoute l'objet référencé à l'*object_list*.

`data::= object_list_element` (voir ci-dessus)

remove_object (data)

Retire l'objet référencé de l'*object_list*.

`data::= object_list_element` (voir ci-dessus)

add_user (data)

Ajoute un utilisateur à l'*user_list*.

`data::= user_list_entry` (voir ci-dessus)

remove_user (data) Supprime un utilisateur de l'*user_list*.

data::= user_list_entry (voir ci-dessus)

5.3.5 SAP assignment (class_id = 17, version = 0)

Cette IC permet de modéliser la structure logique des dispositifs physiques, en fournissant des informations relatives à l'affectation des dispositifs logiques à leurs SAP. Voir l'IEC 62056-5-3:, Annexe A.

SAP assignment (Affectation de SAP)		0...1	class_id = 17, version = 0			
Attribut(s)		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. SAP_assignment_list	(static)	asslist_type			0	x + 0x08
Méthodes spécifiques		m/o				
1. connect_logical_device (data)		f				x + 0x20

Description d'attribut

logical_name Identifie l'instance des objets "SAP assignment" (affectation de SAP). Voir la 6.2.28.

SAP_assignment_list Contient la liste de tous les dispositifs logiques et leurs adresses de SAP au sein du dispositif physique.

asslist_type::= array asslist_element

asslist_element::= structure

```
{
  SAP: long-unsigned,
  logical_device_name: CHOICE
  {
    octet-string [9],
    visible-string [10],
    utf8-string [12]
  }
}
```

REMARQUE: L'adressage effectif est accompli par les couches communication en place.

Description de la méthode

connect_logical_device (data) Connecte un dispositif logique à un SAP. La connexion au SAP 0 déconnectera le dispositif. Plusieurs dispositifs ne peuvent pas être connectés à un SAP (à l'exception du SAP 0).

data::= asslist_element

5.3.6 Image transfer (Transfert d'Image)

5.3.6.1 Généralités

Les instances de l'IC "Image transfer" (transfert d'Image) modélisent le processus de transfert de fichiers binaires, appelés Images, vers les serveurs COSEM.

NOTE La présente spécification inclut certaines améliorations et précisions apportées au texte. Les principales modifications sont les suivantes:

- La description du processus de transfert d'image est donnée dans un seul exemple. Le texte et l'organigramme sont mis à jour;
- L'échange de données entre le client et un serveur d'image conceptuel est hors du domaine d'application;

- L'image transférée et l'image à activer sont clairement distinguées;
- Les étapes 1 et 6 sont à présent facultatives;
- La taille de la chaîne binaire *image_transferred_blocks_status* peut être dynamique;
- Désormais, il est indiqué que le fait d'attribuer la valeur FALSE à l'attribut *image_transfer_enabled* désactive le processus de transfert d'image;
- Désormais, il est indiqué que le fait de réinitialiser le processus de transfert d'image réinitialise l'ensemble du processus;
- Certaines précisions ont été ajoutées aux effets de l'appel des méthodes;
- Voir également les parties mises en évidence du texte.

5.3.6.2 Étapes du processus de transfert d'image

Le transfert d'image se déroule en général en plusieurs étapes:

- Étape 1: (facultative): Récupérer l'ImageBlockSize;
- Étape 2: le client lance le transfert d'image;
- Étape 3: le client transfère ImageBlocks;
- Étape 4: le client vérifie l'exhaustivité de l'image;
- Étape 5: le serveur vérifie l'image (initiée par le client ou de son propre chef);
- Étape 6 (facultative): le client vérifie les informations relatives aux images à activer;
- Étape 7: le serveur active la/les image(s) (initières par le client ou de son propre chef);

Pour obtenir un exemple avec des explications plus détaillées, voir 5.3.6.4.

5.3.6.3 Image transfer (class_id = 18, version = 0)

Image transfer (Transfert d'image)		0...n	class_id = 18, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	image_block_size (static)	double-long-unsigned				x + 0x08
3.	image_transferred_blocks_status (dyn.)	bit-string				x + 0x10
4.	image_first_not_transferred_block_number (dyn.)	double-long-unsigned				x + 0x18
5.	image_transfer_enabled (static)	boolean				x + 0x20
6.	image_transfer_status (dyn.)	enum				x + 0x28
7.	image_to_activate_info (dyn.)	array				x + 0x30
Méthodes spécifiques		m/o				
1.	image_transfer_initiate	m				x + 0x40
2.	image_block_transfer	m				x + 0x48
3.	image_verify	m				x + 0x50
4.	image_activate	m				x + 0x58

Description d'attribut

logical_name	Identifie l'instance de l'objet "Image transfer" (Transfert d'Image). Voir 6.2.31.
image_block_size	Contient l'ImageBlockSize, en octets, qui peut être géré par le serveur. ImageBlockSize ne doit pas dépasser le ServerMaxReceivePduSize négocié. NOTE 1 <i>image_block_size</i> est une propriété du serveur.
image_transferred_blocks_status	Fournit les informations relatives à l'état du transfert de chaque ImageBlock. Chaque bit dans la chaîne binaire fournit des informations relatives à un ImageBlock pris séparément: 0 = Non transféré, 1 = Transféré NOTE 2 La taille de l'attribut peut être dynamique, c'est-à-dire qu'elle peut croître à la réception de nouveaux ImageBlocks.
image_first_not_transferred_block_number	Fournit l'ImageBlockNumber du premier ImageBlock non transféré. Une fois que l'Image est complète, il convient que la valeur retournée soit supérieure ou égale au nombre de blocs calculé à partir de la taille de l'Image et de l'ImageBlockSize.
image_transfer_enabled	Contrôle l'activation du processus de transfert d'image. boolean: FALSE = Désactivé, TRUE = Activé Les méthodes de transfert d'image peuvent être appelées uniquement si la valeur de cet attribut est TRUE. Si la valeur FALSE lui est attribuée, toutes les méthodes sont désactivées (échec de l'appel).
image_transfer_status	Contient l'état du processus de transfert d'Image. enum: (0) Transfert d'Image non déclenché, (1) Transfert d'Image déclenché, (2) Vérification d'Image déclenchée, (3) Vérification d'Image réussie, (4) Vérification d'Image échouée, (5) Activation d'Image déclenchée, (6) Activation d'Image réussie, (7) Activation d'Image échouée

image_to_activate_info Fournit les informations de l'Image ou des Images prête(s) à l'activation. Il est généré comme le résultat du processus de vérification de l'Image. Le client peut vérifier ces informations avant d'activer les Images.

array image_to_activate_info_element

image_to_activate_info_element::= structure

```
{  
    image_to_activate_size:                    double-long-unsigned,  
    image_to_activate_identification:    octet-string,  
    image_to_activate_signature:            octet-string  
}
```

où:

- image_to_activate_size est la taille de l'Image à activer, exprimée en octets;
- image_to_activate_identification est l'identification de l'Image à activer et peut contenir des informations telles que fabricant, le type de dispositif, les informations de version, etc.;
- image_to_activate_signature est la signature de l'Image à activer.

NOTE 3 L'algorithme de génération de la signature est hors du domaine d'application de la présente spécification.

Description de la méthode

image_transfer_initiate (data)	Initialise le processus de transfert de l'Image. data ::= structure <pre>{ image_identifier: octet-string, image_size: double-long-unsigned }</pre> où: - image_identifier identifie l'Image à transférer; - image_size contient l'ImageSize, en octets. NOTE 4 L'<i>image_identifier</i> identifie l'Image à transférer (conteneur); mais n'est pas nécessairement lié à son contenu, c'est-à-dire les Images qui seront activées. Ces informations peuvent être récupérées de l'attribut <i>image_to_activate</i> après vérification de l'Image transférée. Suite à l'appel réussi de la méthode, la valeur (1) est affectée à l'attribut <i>image_transfer_status</i> et la valeur 0 à <i>image_first_not_transferred_block_number</i>. Tous les appels subséquents de la méthode réinitialisent l'ensemble du processus de transfert d'image, et tous les ImageBlocks sont de nouveau à transférer.
image_block_transfer (data)	Transfère un bloc de l'Image vers le serveur. data ::= structure <pre>{ image_block_number: double-long-unsigned, image_block_value: octet-string }</pre> Suite à l'appel réussi de la méthode, la valeur 1 est affectée au bit correspondant de l'attribut <i>image_transferred_blocks_status</i> et l'attribut <i>image_first_not_transferred_block_number</i> est mis à jour.
image_verify (data)	Vérifie l'intégrité de l'Image avant activation. data ::= integer (0) Le résultat de l'appel de cette méthode peut être "success", "temporary_failure" ou "other_reason". S'il n'est pas "success", le résultat de la vérification peut être déterminé en récupérant la valeur de l'attribut <i>image_transfer_status</i>. En cas de succès, l'attribut <i>image_to_activate_info</i> contient les informations relatives aux images à activer. NOTE 5 Pour les services xDLMS, les codes Action-Result/Data-Access-Result sont spécifiés dans l'Article 8 de l'IEC 62056-5-3:.
image_activate (data)	Active l'Image. data ::= integer (0) Si l'Image transférée n'a pas été vérifiée au préalable, cela est fait comme partie intégrante de l'activation de l'Image. Le résultat de l'appel de cette méthode peut être "success", "temporary-failure" ou "other-reason". S'il n'est pas "success", le résultat de l'activation peut être connu en récupérant la valeur de l'attribut <i>image_transfer_status</i>. NOTE 6 Pour les services xDLMS, les codes Action-Result/Data-Access-Result sont spécifiés dans l'IEC 62056-5-3:., Article 8.

5.3.6.4 Exemple de processus de transfert d'image normalisé

Dans le présent paragraphe, un exemple usuel de processus de transfert d'image à partir d'un client présumé est donné, accompagné d'un organigramme, voir Figure 10. À noter qu'il n'est pas obligatoire de suivre ce processus. Selon le cas d'utilisation, certaines étapes peuvent être ignorées ou d'autres peuvent être nécessaires.

Condition préalable: Le transfert d'image doit être activé. *image_transfer_enabled* = TRUE.

Le fait d'attribuer la valeur FALSE à *image_transfer_enabled* à tout moment désactive toutes les méthodes (l'appel de ces méthodes échoue). La valeur des attributs d'état n'est pas définie.

Étape 1 (facultative): Récupérer l'ImageBlockSize

Si le client ne connaît pas la taille des blocs d'image, le serveur cible du transfert d'image peut la traiter. Il doit lire l'attribut *image_block_size* de l'objet "Image transfer" correspondant de chaque serveur vers lequel l'image est à transférer, et ce avant de commencer le processus. Le client peut alors transférer les ImageBlocks de la bonne taille.

Si les ImageBlock sont envoyés, par diffusion, vers un groupe de serveurs COSEM, l'ImageBlockSize doit être le même dans chaque membre du groupe.

Étape 2: le client lance le transfert d'image

Le client lance le processus de transfert d'image individuellement ou par diffusion dans tous les serveurs, en appelant la méthode *image_transfer_initiate*. Le paramètre d'appel de la méthode contient l'identifiant et la taille de l'image à transférer. Le serveur doit rendre disponible l'espace mémoire nécessaire pour contenir l'image.

À l'issue du lancement, la valeur de l'attribut *image_transfer_status* est (1). L'attribut *image_transferred_blocks_status* doit être réinitialisé, la valeur 0 doit être affectée à l'attribut *image_first_not_transferred_block_number*, et il convient de réinitialiser la valeur de l'attribut *image_to_activate_info*. Le processus de transfert d'image est lancé, et le serveur COSEM se prépare à accepter les ImageBlocks.

Étape 3: le client transfère ImageBlocks

Le client transfère les ImageBlocks vers (un groupe de) serveur(s) en appelant la méthode *image_block_transfer* individuellement ou par diffusion. Les paramètres d'appel de la méthode comprennent l'ImageBlockNumber et un ImageBlock. Les ImageBlock sont acceptés par les seuls serveurs COSEM dans lesquels le processus de transfert d'image a été déclenché avec succès. Les autres serveurs rejettent silencieusement tous les éventuels ImageBlock reçus.

Étape 4: le client vérifie l'exhaustivité de l'image

Le client vérifie (avec chaque serveur individuellement) l'exhaustivité de l'image transférée. Si l'image n'est pas complète, il transfère les ImageBlock non (encore) transférés. Il s'agit d'un processus itératif qui se poursuit jusqu'au transfert réussi de l'image entière.

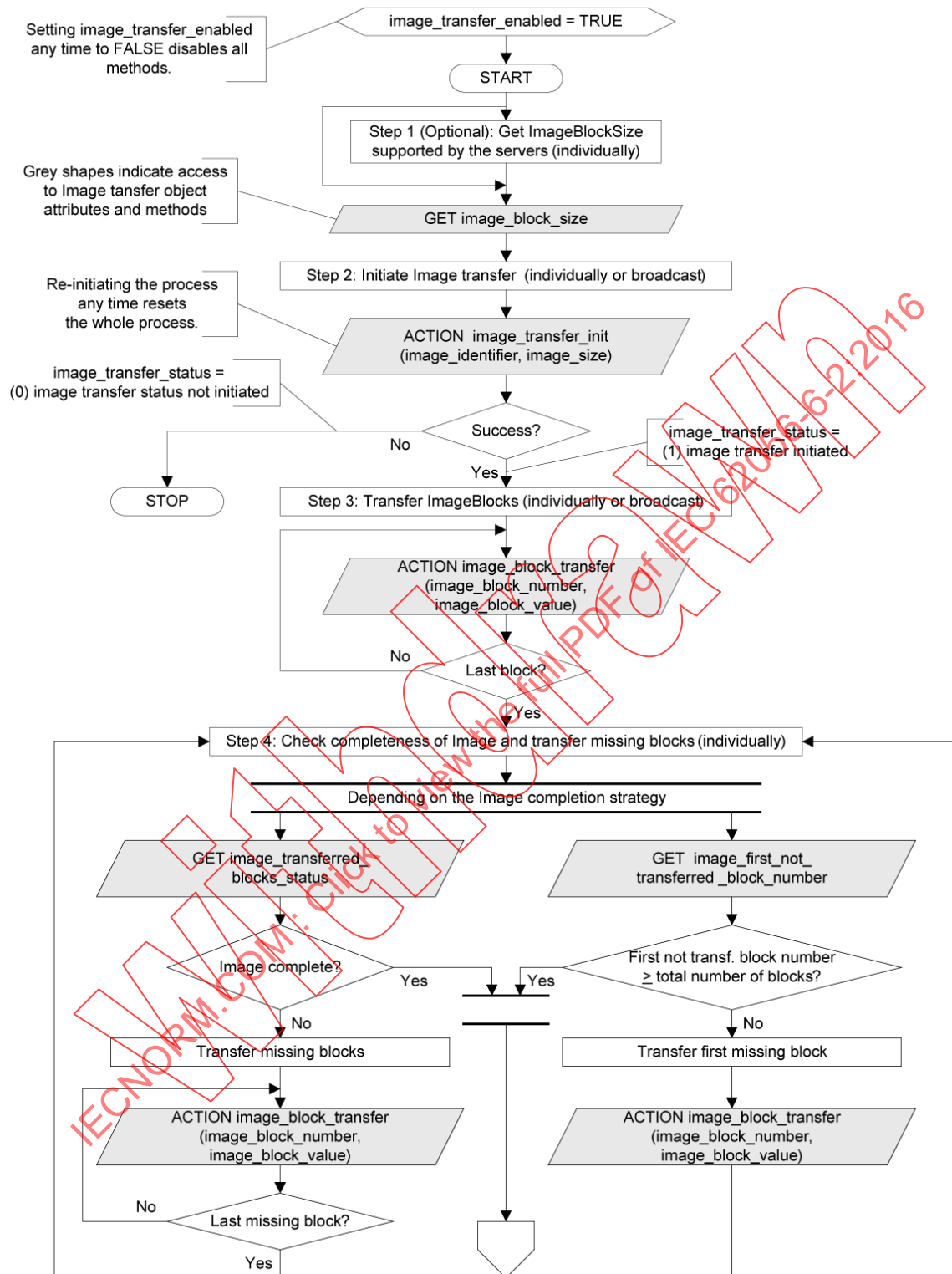
Pour identifier et transférer les ImageBlock non transférés, deux mécanismes sont disponibles:

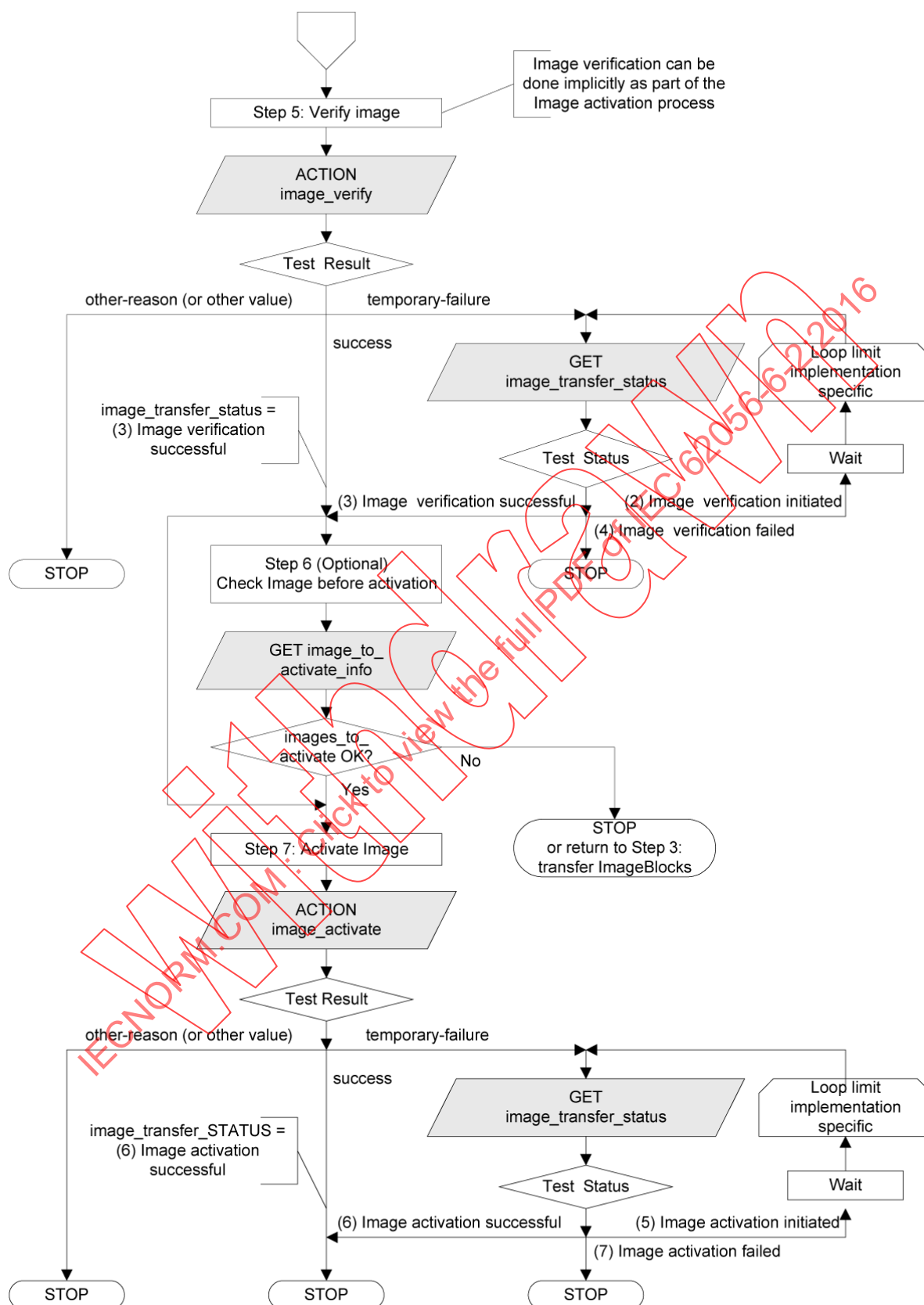
- le client peut récupérer l'état de chaque ImageBlock: "non transféré" ou "transféré". Cela est réalisé en récupérant la valeur de l'attribut *image_transferred_blocks_status*. Le client transfère ensuite les ImageBlock non (encore) transférés;

- en variante, le client peut récupérer l'ImageBlockNumber du premier bloc non transféré. Pour ce faire, il récupère la valeur de l'attribut *image_first_not_transferred_block_number*. Le client transfère alors l'ImageBlock qui n'est pas (encore) transféré;
- ensuite, le client vérifie de nouveau l'exhaustivité de l'Image.

NOTE 1 Les deux mécanismes peuvent être combinés librement.

IECNORM.COM: Click to view the full PDF of IEC 62056-6-2:2016





Anglais	Français
Setting image_transfer-enabled any time to FALSE disables all methods	Le fait d'attribuer la valeur FALSE à image_transfer_enabled à tout moment désactive toutes les méthodes
Grey shapes indicate access to image transfer object attributes and methods	Les formes grises indiquent l'accès aux attributs et méthodes de l'objet transfert d'image
Step 1: (Optional): Get ImageBlockSize supported by the servers (individually)	Étape 1: (Facultative) Get ImageBlockSize pris en charge par les serveurs (individuellement)
Step 2: initiate image transfer (individually or broadcast)	Étape 2: Déclencher le transfert d'image (individuellement ou en diffusion)
Re-initiating the process any time resets the whole process	Le fait de redéclencher le processus à tout moment réinitialise tout le processus
Image_transfer_status= (0) image transfer status not initiated	Image_transfer_status = (0) statut de transfert d'image non déclenché
Success ?	Succès ?
No	Non
Yes	Oui
Image_transfer_status= (1) image transfer initiated	Image_transfer_status= (1) transfert d'image déclenché
Step 3: Transfer ImageBlocks (individually or broadcast)	Étape 3: Transférer les ImageBlocks (individuellement ou en diffusion)
Last block ?	Dernier bloc ?
Step 4: Check completeness of image and transfer missing blocks (individually)	Étape 4: Vérifier l'exhaustivité de l'image et transférer les blocs manquants (individuellement)
Depending on the Image completion strategy	En fonction de la stratégie d'achèvement d'image
Image complete ?	Image complète ?
First not transf. block number > total number of blocks?	Nombre de premiers blocs non transférés > nombre total de blocs ?
Transfer missing blocks	Transférer les blocs manquants
Transfert first missing block	Transférer le premier bloc manquant
Last missing block ?	Dernier bloc manquant ?
Step 5: Verify image	Étape 5: Vérifier l'image
Image verification can be done implicitly as part of the image activation process	La vérification de l'image peut être réalisée de manière implicite dans le cadre du processus d'activation de l'image
Test result	Résultat de l'essai
Other-reason (or other value)	Autre raison (ou autre valeur)
Success	Succès
Loop limit implementation specific	Limite de boucle spécifique à la mise en œuvre
Image_transfer_status= (3) image verification successful	Image_transfer_status = (3) vérification d'image réussie
Test status	Statut de l'essai
Wait	Attendre
(2) image verification initiated	(2) vérification d'image lancée
(3) image verification successful	(3) vérification d'image réussie
(4) image verification failed	(4) vérification d'image échouée
Step 6: (Optional) Check Image before activation	Étape 6: (Facultative) Vérifier l'Image avant activation
Or return to Step 3: Transfer Imageblocks	Ou retourner à l'Étape 3: Transférer des blocs d'image
Step 7: Activate Image	Étape 7: Activer l'image

Anglais	Français
Image_transfer_STATUS= (6) Image activation successful	Image_transfer_STATUS= (6) activation de l'Image réussie
(5) image activation initiated	(5) activation d'image lancée
(6) image activation successful	(6) activation d'image réussie
(7) image activation failed	(7) activation d'image échouée

Figure 10 – Organigramme du processus de transfert d'image

Étape 5: Le serveur vérifie l'Image

L'Image est vérifiée par le serveur. Pour ce faire, le client appelle la méthode *image_verify* ou le serveur peut lancer la vérification. Le résultat peut être:

- success (succès), si la vérification a pu être achevée;
- temporary-failure (échec temporaire), si la vérification n'a pas pu être achevée;
- other-reason (autre raison), si la vérification a échoué.

NOTE 2 Les conditions de vérification de l'Image ne relèvent pas du domaine d'application de la présente norme.

Étape 6 (facultative): le client vérifie les informations relatives aux images à activer

Le résultat de la vérification de l'Image peut être vérifié par le client en récupérant la valeur de l'attribut *image_transfer_status*. La valeur de cet attribut est mise à jour suite à la vérification de l'image.

Une Image transférée peut contenir une ou plusieurs Images à activer. Pour chaque Image à activer, cet attribut détient les paramètres: {image_to_activate_size, image_to_activate_identification, image_to_activate_signature}.

Si ces informations ne sont pas ce qui est attendu, le client peut recommencer le transfert de l'image.

Autrement, il passe à l'étape suivante, activation de l'Image.

Étape 7: le serveur active la/les Image(s)

L'Image est activée par le serveur. Pour ce faire, le client appelle la méthode *image_activate* ou le serveur peut lancer l'activation.

Si l'activation est faite sans vérification préalable, une vérification est faite implicitement en tant que partie intégrante de l'activation. Le résultat de l'appel de la méthode *Image_activate* peut être:

- success (succès), si l'activation d'Image a abouti;
- temporary-failure (échec temporaire), si la vérification/activation n'a pas pu être achevée;
- other-reason (autre raison), si l'activation a échoué.

En cas de succès, le serveur procède à l'activation de la/des nouvelle(s) Image(s). Au cours de ce processus, il n'est pas accessible. Après l'activation de la/des Image(s), le résultat peut être vérifié par le client, en récupérant la valeur de l'attribut *image_transfer_status* ou en lisant le contenu des objets COSEM appropriés contenant l'identifiant, la version et la signature numérique du firmware actif.

5.3.7 Security setup (class_id = 64, version = 0)

Les instances de cette IC contiennent les informations nécessaires relatives à la politique de sécurité applicable et à la suite de sécurité utilisée au sein d'une AA particulière, entre deux systèmes identifiés respectivement par leur titre système client et leur titre système serveur. Elles contiennent aussi les méthodes pour augmenter le niveau de sécurité et transférer les clés globales. Voir l'Article 5 de l'IEC 62056-5-3.

Security setup (établissement de la sécurité)		0...n	class_id = 64, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	security_policy (static)	enum	0	3	0	x + 0x08
3.	security_suite (static)	enum	0	0	0	x + 0x10
4.	client_system_title (dyn.)	octet-string				x + 0x18
5.	server_system_title (static)	octet-string				x + 0x20
Méthodes spécifiques		m/o				
1.	security_activate	o				x + 0x28
2.	global_key_transfer	o				x + 0x30

Description d'attribut

logical_name Identifie l'instance de l'objet "Security setup" (Établissement de la sécurité). Voir 6.2.30.

security_policy Impose l'algorithme d'authentification et/ou de chiffrement fourni avec security_suite.
enum:
(0) aucune
(1) tous les messages devant être authentifiés,
(2) tous les messages devant être chiffrés,
(3) tous les messages devant être authentifiés et chiffrés
(4)...(15) réservé

security_suite Spécifie l'algorithme d'authentification, de chiffrement et de transport de clé.
enum:
(0) AES-GCM-128 pour chiffrement authentifié et AES-128 pour l'enveloppement de clé
(1) ...(15) réservé

client_system_title Détient le titre du système client (courant):

- Dans l'environnement PLC S-FSK, l'initiateur actif envoie son titre système en utilisant le protocole CIASE;

NOTE 1 Il est également détenu par l'attribut *active_initiator* de l'objet S-FSK Active initiator. Voir 5.8.4;

- au cours de l'établissement d'AA confirmé ou non confirmé, il est transporté par le champ calling-AP-title de l'AARQ APDU;
- Si un titre système client a déjà été envoyé au cours d'un processus d'enregistrement, comme dans le cas du profil PLC S-FSK, il convient que le titre système client transporté par l'AARQ APDU soit le même. Autrement, l'AA doit être rejetée et des informations de diagnostic appropriées doivent être envoyées.
- Dans une AA préétablie, il peut être écrit par le client avec l'aide d'un service SET / Write non sécurisé.

server_system_title	Transporte le titre système serveur. – Dans l'environnement PLC S-FSK, le serveur envoie son titre système au cours du processus de découverte, en utilisant le protocole CIASE; – au cours de l'établissement d'AA confirmé, il est transporté par le champ responding-AP-title de l'AARE APDU. Cet attribut doit être en lecture seulement.
Description de la méthode	
security_activate (data)	Active et renforce la politique de sécurité: enum: (0) aucune (1) tous les messages devant être authentifiés, (2) tous les messages devant être chiffrés, (3) tous les messages devant être authentifiés et chiffrés La nouvelle politique de sécurité s'applique dès que l'appel de la méthode a été confirmé avec succès. NOTE 2 La politique de sécurité peut seulement être renforcée.
global_key_transfer (data)	Met à jour une ou plusieurs clés globales. Le paramètre <i>data</i> comprend des données de clé enveloppée. Les données de clé comprennent les identifiants de clés et les clés elles-mêmes. array key_data key_data ::= structure { key_id: enum: (0) clé globale de chiffage monodiffusion, (1) clé globale de chiffage en diffusion, (2) clé d'authentification key_wrapped: octet-string } L'algorithme d'enveloppement de clé est tel que spécifié par la suite de sécurité. La KEK est la clé principale. La ou les nouvelles clés sont valides dès que l'appel de la méthode a été confirmé avec succès.

5.3.8 Classes d'interfaces et objets poussés

5.3.8.1 Vue d'ensemble

Les messages DLMS peuvent être "poussés" dans plusieurs cas vers une destination sans être demandés de manière explicite, par exemple

- si un temps de référence a été atteint;
- si une valeur surveillée en local a dépassé un seuil;
- au déclenchement par un événement local (mise sous/hors tension, pression sur le bouton-poussoir, couvercle du compteur ouvert, par exemple).

Le mécanisme de poussée DLMS/COSEM suit le modèle d'édition/abonnement:

Le modèle éditer/abonner est un modèle de messagerie dans lequel les émetteurs (éditeurs) des messages ne programment pas les messages à envoyer directement aux destinataires spécifiques (abonnés). Les messages édités se caractérisent plutôt par des classes, sans savoir à quels abonnés, le cas échéant, elles appartiennent. Les abonnés expriment leur intérêt dans une ou plusieurs classes et reçoivent uniquement les messages qui les intéressent, sans savoir de quels éditeurs, le cas échéant, ils proviennent. [Wikipedia]

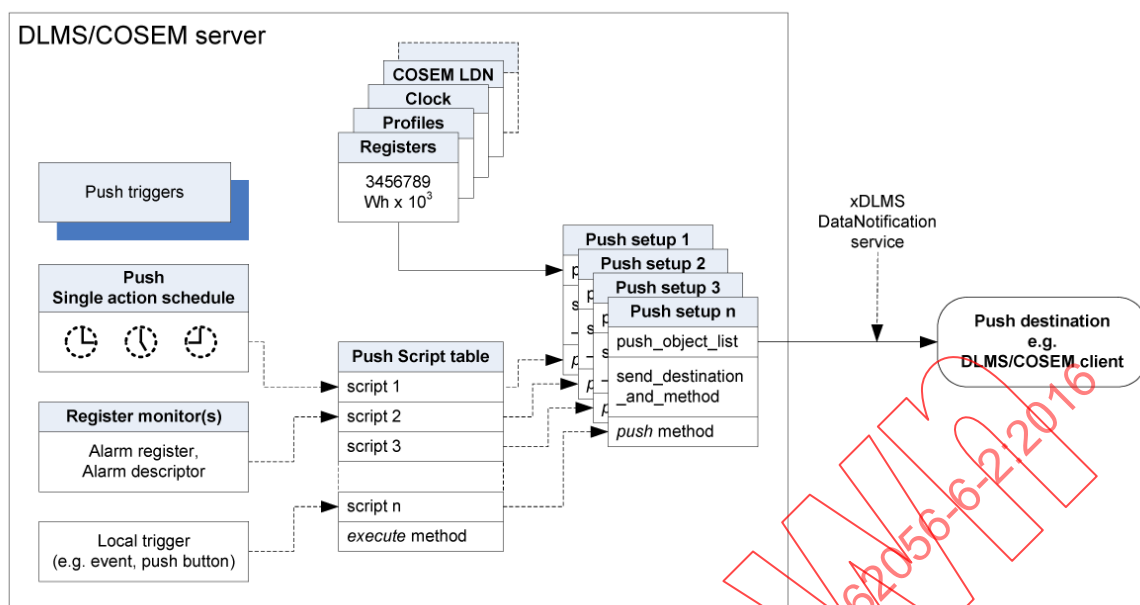
Dans DLMS/COSEM, l'édition est modélisée par l'attribut *object_list* des objets "Association" fournissant la liste des objets COSEM et de leurs attributs accessibles dans une AA donnée. L'abonnement est modélisé par l'écriture des attributs appropriés des objets "Push setup". Les données exigées sont envoyées (lors du déclenchement spécifié) à l'aide du service xDLMS DataNotification.

Le modèle COSEM d'opération Push est représenté à la Figure 11.

L'élément central de la modélisation de l'opération Push est l'IC "Push setup". L'attribut *push_object_list* contient une liste des références aux attributs d'objet COSEM à pousser.

Les différents déclencheurs (programmeurs, moniteurs, déclencheurs locaux, etc.) appellent une entrée de script dans un objet Push "Script table" (nouvelles instances de l'IC "Script table") qui appelle alors la méthode Push de l'objet "Push setup" correspondant. La destination, le support de communication, le protocole, le codage, la synchronisation et les nouvelles tentatives d'opération Push sont déterminés par les autres attributs de l'objet "Push setup".

Chaque déclencheur peut être à l'origine de l'envoi des données à une destination dédiée. Par conséquent, pour chaque déclencheur ou groupe de déclencheurs, un objet "Push setup" individuel est disponible, qui définit le contenu et la destination du message Push, ainsi que le support de communication utilisé.



IEC

Anglais	Français
DLMS/COSEM server	Serveur DLMS/COSEM
Push triggers	Déclencheurs Push
Local trigger (e.g. event, push button)	Déclencheur local (événement, bouton-poussoir, par exemple)
execute method	Méthode <i>execute</i>
xDLMS DataNotification service	Service xDLMS DataNotification
Push destination e.g. DLMS/COSEM client.	Destination de la poussée (client DLMS/COSEM, par exemple)
push method	Méthode <i>Push</i>

Figure 11 – Modèle COSEM d'opération Push

Pour les besoins de la poussée des données lorsqu'une alarme se déclenche, les objets Alarm monitor (nouvelles instances de l'IC "Register monitor") sont disponibles.

Les alarmes sont détenues par les objets *Alarm register* ou *Alarm descriptor*, voir 6.2.52.

La structure de l'objet *Alarm descriptor* et les conditions d'alarme nécessitent d'être définies dans une spécification d'accompagnement.

Les objets *Alarm descriptor* sont surveillés par les objets "Register monitor" de l'alarme. Voir 6.2.13. L'attribut *actions* fournit le lien vers les scripts *action_up* et peut-être *action_down* de l'objet Push "Script table" (voir 6.2.7) qui appelle alors la méthode *push* de l'objet "Push setup" souhaité. Lorsqu'une alarme se déclenche, l'ensemble de données préalablement défini (qui peut contenir, entre autres, les objets *Alarm register* et *Alarm descriptor*) est poussé.

Lorsque les conditions de poussée sont satisfaites, les données poussées sont envoyées à l'aide du service xDLMS de type de serveur/non client non sollicité: le service DataNotification. Si les données poussées sont longues, elles peuvent être envoyées en blocs.

Le processus de poussée a lieu dans le contexte de l'application de l'AA, dans laquelle l'objet "Push setup" est visible. Le contexte de sécurité est déterminé par l'objet "Security setup" référencé depuis l'objet "Association".

NOTE Les détails font l'objet de spécifications d'accompagnement.

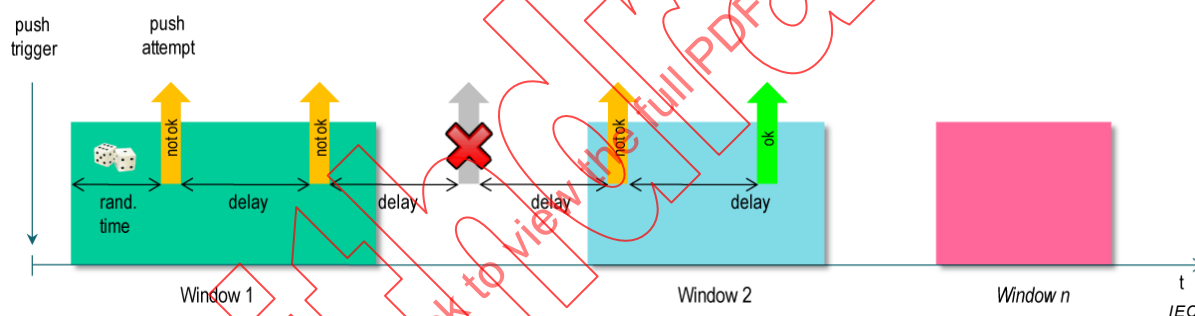
Toutes les informations nécessaires au traitement des données reçues par le client doivent être:

- récupérées par le client (en lisant l'attribut *push_object_list attribute*, par exemple); ou
- doivent faire partie des données poussées; ou
- doivent être préalablement définies dans l'application du client.

5.3.8.2 Push setup (class_id = 40, version = 0)

L'IC "Push setup" contient une liste de références aux attributs de l'objet COSEM à pousser. Elle contient également la destination et la méthode de poussée, ainsi que les fenêtres de temps de communication et la gestion des nouvelles tentatives.

La poussée a lieu à l'appel de la méthode *push*, déclenchée par un objet Push "Single action schedule", par un objet d'alarme "Register monitor", par un événement interne dédié ou externe. Suite au déclenchement de l'opération Push, elle est exécutée selon les paramètres donnés dans l'objet "Push setup" donné. Selon les paramètres de la fenêtre de communication, la poussée est exécutée immédiatement ou dès qu'une fenêtre de communication devient active, après un délai aléatoire. Si la poussée n'a pas abouti, de nouvelles tentatives ont lieu. Les fenêtres de poussée, les délais et les nouvelles tentatives sont représentés à la Figure 12.



Anglais	Français
push trigger	Déclencheur Push
Push attempt	Tentative de poussée
Rand. time	Durée aléatoire
delay	retard
Window 1	Fenêtre 1
Window 2	Fenêtre 2
Window n	Fenêtre n

Figure 12 – Fenêtres de poussée et délais

Push setup (établissement de Push)		0...n	class_id = 40, version = 0			
Attribut(s)		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. push_object_list	(static)	array				x + 0x08
3. send_destination_and_method	(static)	structure				x + 0x10
4. communication_window	(static)	array				x + 0x18
5. randomisation_start_interval	(static)	long-unsigned				x + 0x20
6. number_of_retries	(static)	unsigned				x + 0x28
7. repetition_delay	(static)	long-unsigned				x + 0x30
Méthodes spécifiques		m/o				
1. push (data)		m				x + 0x38

Description d'attribut

logical_name Identifie l'instance de l'objet "Push setup". Voir 6.2.21.

push_object_list Définit la liste des attributs à pousser.
 À l'appel de la méthode *push (data)*, les éléments sont envoyés à la destination définie dans l'attribut *send_destination_and_method*.
 array object_definition

object_definition ::= structure
 {
 class_id: long-unsigned,
 logical_name: octet-string,
 attribute_index: integer,
 data_index: long-unsigned
 }
 où:
 – attribute_index est un pointeur vers l'attribut au sein de l'objet, identifié par class_id et logical_name: attribute_index 1 fait référence au 1^{er} attribut (c'est-à-dire *logical_name*), attribute_index 2 au 2^{ème} attribut, etc.; attribute_index 0 fait référence à tous les attributs publics;
 – data_index est un pointeur qui sélectionne un ou plusieurs éléments particuliers d'un attribut avec un type de données complexe (structure ou tableau).

data_index:	MS-Byte		LS-Byte
	Quartet supérieur	Quartet inférieur	

**push_
object_list**

(suite)

Si le `data_type` de l'attribut est simple, alors `data_index` n'a aucune signification.

Si l'attribut est une structure ou un tableau, `data_index` pointe vers un élément de la structure ou du tableau. Le premier élément dans l'attribut complexe est identifié par `data_index 1`.

Si l'attribut est le *buffer* d'un objet "Profile generic", `data_index` contient des paramètres d'accès sélectif.

- 0x0000 = identifie tout l'attribut;
- 0x0001 à 0x0FFF = identifie un élément dans l'attribut complexe;
- 0x1000 à 0xFFFF = accès sélectif au tableau contenant le *buffer* d'un objet "Profile generic". Le `data-index` sélectionne les entrées dans un certain nombre de périodes récentes ou un certain nombre d'entrées récentes, ainsi que les colonnes du tableau.

Le codage est spécifié au Tableau 6.

NOTE 1 Si le tableau `push_object_list` est vide, l'opération Push est désactivée.

NOTE 2 L'attribut `push_object_list` lui-même peut être poussé, pour identifier clairement les données poussées.

NOTE 3 Comme pour l'IC "Profile generic", tous les attributs inclus dans l'attribut `push_object_list` sont poussés, quels que soient les droits d'accès dont ils font l'objet. Par conséquent, il convient que l'écriture de l'attribut `push_object_list` soit limitée aux clients détenant les droits d'accès appropriés.

**send_
destination_
and_method**

Contient l'adresse de destination (numéro de téléphone, adresse électronique, adresse IP, par exemple) à laquelle les données spécifiées par *push_object_list* sont à envoyer, ainsi que la méthode d'envoi.

send_destination_and_method ::= structure

```
{
    transport_service:    transport_service_type,
    destination:          octet-string,
    message:              message_type
}
```

où:

- l'élément transport_service définit le type de service utilisé pour pousser les données:

transport_service_type ::= enum:

(0)	TCP,
(1)	UDP,
(2)	réservé pour FTP,
(3)	réservé pour SMTP,
(4)	SMS,
(5)	HDLC
(6)	réservé pour M-Bus
(7)	réservé pour ZigBee®
(200...255)	spécifique au fabricant

- l'élément de destination contient l'adresse cible vers laquelle les données sont à envoyer. Les éléments de l'adresse cible dépendent du service de transport utilisé.

Chaque instance d'objet "Push setup" spécifie une seule destination. Si les données sont à pousser vers plusieurs destinations, plusieurs objets "Push setup" sont à instancier,

- l'élément message_type identifie le codage de l'APDU xDLMS utilisée.

message_type ::= enum:

(0)	APDU xDLMS codée A-XDR,
(1)	APDU xDLMS codée XML,
(128...255)	spécifique au fabricant

- Toutes les autres valeurs de transport_service_type et de message_type sont réservées à une utilisation ultérieure.

communication_ window	Définit les points dans le temps auxquels la/les fenêtre(s) de communication pour la poussée devien(nen)t active(s) (<i>start_time</i>) et inactive(s) (<i>end_time</i>). Voir la Figure 12. array window_element window_element ::= structure { <i>start_time</i>: octet-string, <i>end_time</i>: octet-string } <i>start_time</i> et <i>end_time</i> sont formatés comme indiqué en 4.6.1 pour <i>date-time</i>, y compris les caractères génériques. Si la fin d'une fenêtre de communication est atteinte, une opération Push déjà commencée se termine. Si aucune fenêtre de communication n'est définie (<i>array</i> [0]), l'opération Push est toujours possible.
randomisation_ start_ interval	Pour éviter qu'un grand nombre de dispositifs ne procèdent à des opérations Push en même temps, un intervalle de randomisation, en secondes, peut être défini. Cela signifie que l'opération Push ne démarre pas immédiatement après l'appel de la méthode <i>push</i>, mais de manière aléatoire dans l'intervalle. L'attribut <i>randomisation_start_interval</i> définit la valeur maximale qui peut être obtenue à partir d'un algorithme de randomisation. La valeur obtenue permet de retarder la première opération Push. Elle n'est plus utilisée en cas de nouvelles tentatives. Si aucune fenêtre de communication n'est active lors de l'appel de la méthode <i>push</i>, le délai commence au début de la fenêtre de communication suivante. Si la méthode <i>push</i> est appelée pendant une fenêtre de communication active, l'opération Push est encore retardée. L'attribut <i>randomisation_start_interval</i> est uniquement actif pour la première tentative de poussée. Si aucun attribut <i>communication_window</i> n'est défini, l'attribut <i>randomisation_start_interval</i> est actif pour chaque tentative de poussée. Si la valeur 0 est attribuée à <i>randomisation_start_interval</i>, aucun délai n'est actif. Si la durée de randomisation est plus longue que la première fenêtre de communication, la première tentative de poussée a lieu pendant une fenêtre ultérieure.
number_of_ retries	Définit le nombre maximal de nouvelles tentatives, en cas de tentatives de poussée non réussies ou ignorées. Après une opération Push réussie, aucune autre tentative de poussée n'est réalisée tant que l'opération Push ne s'est pas de nouveau déclenchée. Une poussée est considérée comme ayant abouti si une confirmation de transmission de couche inférieure a été reçue. Les conditions de détection d'une tentative de poussée non réussie peuvent dépendre du profil de communication utilisé et de la mise en œuvre. Elles n'entrent donc pas dans le domaine d'application de la présente Spécification technique.

repetition_delay Délai, exprimé en secondes, jusqu'au début de la tentative de poussée suivante, après une poussée non réussie.

NOTE 4 Le délai de répétition lui-même n'est pas influencé par la fenêtre de communication. Toutefois, une nouvelle tentative peut uniquement être réalisée si une fenêtre de communication est active à cet instant. Sinon, elle est traitée comme une tentative de poussée non réussie.

NOTE 5 Les données de poussée ne sont pas enregistrées dans un tampon intermédiaire. En cas de nouvelles tentatives, les valeurs en cours des attributs peuvent changer à chaque tentative de poussée.

Description de la méthode

push (data) Active le processus de poussée donnant lieu à l'élaboration et l'envoi des données de poussée, en tenant compte des valeurs des attributs définis dans l'instance donnée de cette IC.

data::= integer (0)

Les données de poussée renvoyées doivent être une structure. Cette structure doit contenir le même nombre d'éléments que le tableau de l'attribut *push_object_list*.

Tableau 6 – Codage des paramètres d'accès sélectif avec data_index

	Quartet supérieur MS_Byte: Sélectionne les périodes de temps ou les entrées.
0xF	Dernier nombre de mois complet: Accès sélectif au <i>tampon</i> , donnant toutes les entrées du dernier nombre de mois complet et la première entrée à minuit du mois en cours.
0xE	Dernier nombre de jours complet: Accès sélectif au <i>tampon</i> du profil, donnant toutes les entrées du dernier nombre de jours complet et la première entrée à minuit le jour même.
0xD	Dernier nombre d'heures complet: Accès sélectif au <i>tampon</i> , donnant toutes les entrées du dernier nombre d'heures complet et la première entrée de l'heure en cours.
0xC	Dernier nombre de minutes complet: Accès sélectif au <i>tampon</i> , donnant toutes les entrées du dernier nombre de minutes complet et la première entrée de la minute en cours.
0xB	Dernier nombre de secondes: Accès sélectif au <i>tampon</i> , donnant lieu à toutes les entrées du dernier nombre de secondes.
0xA	Dernier nombre de mois complet, y compris le mois en cours: Identique à 0xF ci-dessus, sauf que toutes les entrées sont désormais récupérées.
0x9	Dernier nombre de jours complet, y compris le jour en cours: Identique à 0xE ci-dessus, sauf que toutes les entrées sont désormais récupérées.
0x8	Dernier nombre d'heures complet, y compris l'heure en cours: Identique à 0xD ci-dessus, sauf que toutes les entrées sont désormais récupérées.
0x7	Dernier nombre de minutes complet, y compris la minute en cours: Identique à 0xC ci-dessus, sauf que toutes les entrées sont désormais récupérées.
0x6...0x2	Réservé
0x1	Dernier nombre d'entrées
0x0	Un ensemble d'attributs ou un seul élément d'un attribut avec type de données complexes est sélectionné (quartet inférieur MS-Byte 0x0...0xF, LS-Byte 0x00...0xFF).
	Quartet inférieur MS-Byte: Définit le nombre de colonnes d'un <i>tampon</i> "Profile generic" sélectionné.
0x0	Toutes les colonnes
0x1 à 0xF	Nombre de colonnes, en commençant par la colonne 1.

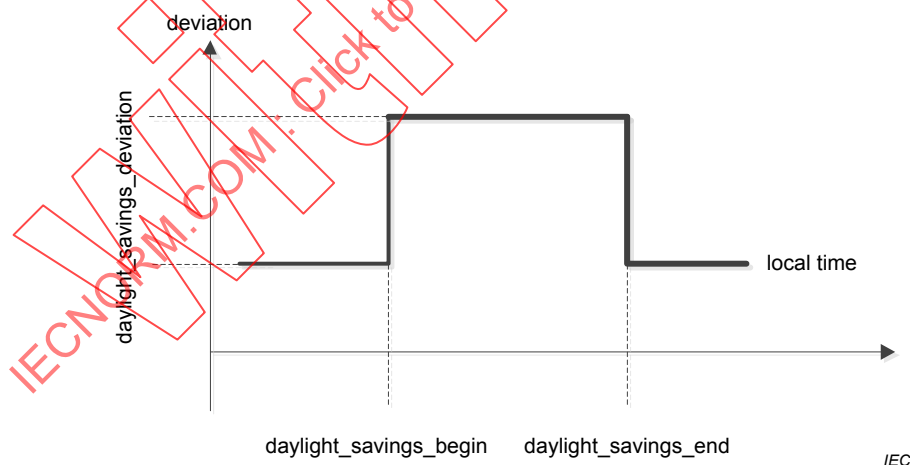
0x00...0xFF	LS-Byte: - en cas d'accès sélectif par période (c'est-à-dire le nombre de mois, de jours, d'heures, etc.), définit le nombre de périodes complètes récentes (1 à 255), - en cas d'accès sélectif par entrée, définit le nombre d'entrées récentes.
Exemple 1)	0xE401: Les entrées du dernier jour complet sont sélectionnées. Les 4 premières colonnes sont incluses.
Exemple 2)	0xA300: Les entrées pour aucun dernier mois complet et le mois en cours sont sélectionnées. Les 3 premières colonnes sont incluses.
Exemple 3)	0x800C: Les entrées des 12 dernières heures complètes sont sélectionnées. Toutes les colonnes sont incluses.
Exemple 4)	0x1080: Les 128 dernières entrées sont sélectionnées. Toutes les colonnes sont incluses.

5.4 Classes d'interfaces pour commande à limite temporelle et événementielle

5.4.1 Clock (class_id = 8, version = 0)

Cette IC modélise l'horloge, gérant toutes les informations relatives à la date et à l'heure, y compris les décalages de l'heure locale par rapport à une référence temporelle généralisée (TUC), en raison des fuseaux horaires et des schémas de changement d'horaire légaux. L'IC propose également diverses méthodes pour régler l'horloge.

Les informations de date comprennent les éléments year (année), month (mois), day of month (jour dans le mois) et day of week (jour dans la semaine). Les informations d'heure comprennent les éléments hour (heure), minutes, seconds (secondes), hundredths of seconds (centièmes de seconde) et le décalage de l'heure locale par rapport au temps TUC. La fonction heure (été/hiver) de changement légal modifie le décalage de l'heure locale par rapport au TUC en fonction des attributs (voir la Figure 13). Le point initial et le point final de la fonction en question sont normalement fixés une seule fois. Un algorithme interne calcule le point réel de commutation en fonction de ces valeurs fixées.



Anglais	Français
deviation	décalage
local time	heure locale

Figure 13 – Concept de temps généralisé

Clock (Horloge)		0...n	class_id = 8, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	time (dyn.)	octet-string				x + 0x08
3.	time_zone (static)	long		± 720		x + 0x10
4.	status (dyn.)	unsigned				x + 0x18
5.	daylight_savings_begin (static)	octet-string				x + 0x20
6.	daylight_savings_end (static)	octet-string				x + 0x28
7.	daylight_savings_deviation (static)	integer		± 120		x + 0x30
8.	daylight_savings_enabled (static)	boolean				x + 0x38
9.	clock_base (static)	enum				x + 0x40
Méthodes spécifiques		m/o				
1.	adjust_to_quarter (data)	f				x + 0x60
2.	adjust_to_measuring_period (data)	f				x + 0x68
3.	adjust_to_minute (data)	f				x + 0x70
4.	adjust_to_preset_time (data)	f				x + 0x78
5.	preset_adjusting_time (data)	f				x + 0x80
6.	shift_time (data)	f				x + 0x88

Description d'attribut

logical_name	Identifie l'instance de l'objet "Clock" (Horloge). Voir 6.2.5.
time	Contient la date et l'heure locales du compteur, son écart par rapport au TUC et l'état. octet-string, mis en forme comme indiqué en 4.6.1 pour <i>date-time</i>. Si cet attribut est défini, tous les champs de la chaîne d'octets représentant <i>date-time</i> doivent être évalués, et la date et l'heure locales du compteur doivent être définies selon les règles définies en 4.6.1. Seuls les champs spécifiés du <i>date-time</i> sont modifiés. EXEMPLE Pour régler la <i>date</i> sans modifier l'<i>heure</i>, tous les octets relatifs à l'heure dans la chaîne d'octets représentant <i>date-time</i> doivent être mis à "not specified" (non spécifié).
time_zone	L'écart de l'heure locale normale par rapport au TUC en minutes. La valeur dépend de la position géographique du compteur.
status	clock_status géré par le compteur. L'élément clock_status indiqué dans l'attribut <i>time</i> est égal à la valeur de cet attribut. unsigned (non signé), formaté tel que spécifié en 4.6.1 pour <i>clock_status</i>
daylight_savings_begin	Définit la date et l'heure locales de basculement lorsque l'heure locale commence à être décalée par rapport à l'heure normale. Pour des définitions génériques, il est permis d'utiliser des caractères génériques. octet-string, mis en forme comme indiqué en 4.6.1 pour <i>date-time</i>.

daylight_savings_end	Définit la date et l'heure locales de basculement lorsque l'heure locale finit d'être décalée par rapport à l'heure normale. octet-string, mis en forme comme indiqué en 4.6.1 pour <i>date-time</i> .
daylight_savings_deviation	Contient le nombre de minutes dont l'écart d'heure généralisée doit être corrigé au daylight_savings_begin (moment du début d'heure (été/hiver)). integer: Écart dans la plage de ± 120 min
daylight_savings_enabled	boolean: TRUE = DST enabled (heure hiver/été activée), FALSE = DST disabled (heure hiver/été désactivée) NOTE Ce paramètre permet d'activer et de désactiver la fonction d'heure d'été/hiver. L'état en cours est indiqué dans l'attribut status.
clock_base	Définit la provenance des informations de temporisation de base. enum: (0) non définie, (1) quartz interne, (2) fréquence secteur de 50 Hz, (3) fréquence secteur de 60 Hz, (4) GPS (système de positionnement global), (5) radiocommandée
Description de la méthode	
adjust_to_quarter (data)	Règle l'heure du compteur sur le plus proche (+/-) quart d'heure (*:00, *:15, *:30, *:45). data::= integer(0)
adjust_to_measuring_period (data)	Règle l'heure du compteur sur le plus proche (+/-) point de départ d'une période de mesure. data::= integer(0)
adjust_to_minute (data)	Règle l'heure du compteur sur la minute la plus proche. Si second_counter < 30 s, second_counter est donc mis à 0. Si second_counter $\geq$ 30 s, second_counter est mis à 0 et les valeurs de minute_counter et toutes les valeurs d'horloge dépendantes sont incrémentées si besoin est. data::= integer(0)
adjust_to_preset_time (data)	Cette méthode est utilisée conjointement à la méthode preset_adjusting_time. Si l'heure du compteur s'inscrit entre validity_interval_start et validity_interval_end, l'heure est mise à preset_time. data::= integer(0)
preset_adjusting_time (data)	Prérègle l'heure sur une nouvelle valeur (preset_time) et définit un validity_interval dans lequel la nouvelle heure peut être activée. } data::= structure { preset_time: octet-string, validity_interval_start: octet-string, validity_interval_end: octet-string } tous les octet-string, formatés comme indiqué en 4.6.1 pour <i>date-time</i>
shift_time (data)	Décale l'heure de n ($-900 \leq n \leq 900$) s. data::= long

5.4.2 Script table (class_id = 9, version = 0)

Cette IC permet de modéliser le déclenchement d'une série d'actions en exécutant des scripts par la méthode *execute* (data).

Les objets "Script table" contiennent un tableau d'entrées de scripts. Chaque entrée consiste en un "script identifier" (identifiant de script) et une série de "action specifications" (spécifications d'action). Une spécification d'action active une méthode ou modifie un attribut d'un objet COSEM au sein d'un dispositif logique.

Un certain script peut être activé par d'autres objets COSEM au sein du même dispositif logique ou depuis l'extérieur.

Si deux scripts sont à exécuter dans la même instance temporelle, c'est celui qui a l'indice le plus petit qui est exécuté le premier.

Script table (tableau de scripts)		0...n	class_id = 9, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. scripts	(static)	array				x + 0x08
Méthodes spécifiques		m/o				
1. execute(data)		m				x + 0x20

Description d'attribut

logical_name	Identifie l'instance de l'objet "Script table " (Tableau de scripts). Voir 6.2.7.
scripts	Spécifie les différents scripts, c'est-à-dire la liste d'actions. array script script::= structure <pre>{ script_identifier: long-unsigned, actions: array action_specification }</pre> Le script_identifier 0 est réservé. S'il est spécifié avec une méthode <i>execute</i>, il se traduit en un script vide (aucune action à accomplir). action_specification::= structure <pre>{ service_id: enum, class_id: long-unsigned, logical_name: octet-string, index: integer, parameter: spécifique au service }</pre> où: - l'élément service_id définit l'action à appliquer à l'objet référencé: (1) écrire un attribut, (2) exécuter une méthode spécifique - l'élément index définit (avec service_id 1) quel attribut de l'objet sélectionné est affecté ou (avec service_id 2) quelle méthode spécifique est à exécuter. Le premier attribut (logical_name) a l'indice 1, la première méthode spécifique a l'indice 1 également. NOTE 1- L'action_specification se limite à activer des méthodes qui ne produisent pas de réponse (du serveur au client). NOTE 2- Une spécification d'action "dummy" (fictive) avec tous les éléments ayant la valeur 0 signifie que l'action n'est pas configurée.

Description de la méthode

execute (data)	Exécute le script spécifié dans les données de paramètre. data::= long-unsigned Si data concorde avec l'un des script_identifier dans le tableau de scripts, l'action_specification correspondante est exécutée.

5.4.3 Schedule (class_id = 10, version = 0)

Cette IC, avec l'IC "Special days" (jours spéciaux), permet de modéliser des activités qui dépendent de l'heure et de la date au sein d'un dispositif. Le Tableau 7 donne une vue générale et montre les interactions entre les deux IC. Voir aussi le Tableau 8.

Tableau 7 – Schedule (programme)

Index	enable	action (script)	Switch _time	validity _window	exec_weekdays							exec_specdays					plage de dates	
					Lu n	M ar	M er	Je u	Ve n	Sa m	Di m	S1	S2	...	S8	S9	begin_ date	end_ date
120	Oui	xxxx:y y	06:00	0xFFFF	x	x	x	x	x	x							xx-04-01	xx-09-30
121	Oui	xxxx:y y	22:00	15	x	x	x	x	x								xx-04-01	xx-09-30
122	Oui	xxxx:y y	12:00	0						x							xx-04-01	xx-09-30
200	Non	xxxx:y y	06:30		x	x	x	x	x	x							xx-04-01	xx-09-30
201	Non	xxxx:y y	21:30		x	x	x	x	x								xx-04-01	xx-09-30
202	Non	xxxx:y y	11:00							x							xx-04-01	xx-09-30

**Tableau 8 – Special days table
(Tableau de jours spéciaux)**

Index	special_day_date	day_id
12	xx-12-24	S1
33	xx-12-25	S3
77	97-03-31	S3

Schedule (programme)		0...n	class_id = 10, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	entries (static)	array				x + 0x08
Méthodes spécifiques		m/o				
1.	enable/disable (data)	f				x + 0x20
2.	insert (data)	f				x + 0x28
3.	delete (data)	f				x + 0x30

Description d'attribut

logical_name	Identifie l'instance de l'objet "Schedule" (Programme). Voir 6.2.9.
entries	Spécifie les scripts devant être exécutés à des instants donnés. Un seul script peut être exécuté par entrée. array schedule_table_entry <pre> schedule_table_entry ::= structure { index: long-unsigned, enable: boolean, script_logical_name: octet-string, script_selector: long-unsigned, switch_time: octet-string, validity_window: long-unsigned, exec_weekdays: bit-string, exec_specdays: bit-string, begin_date: octet-string, end_date: octet-string } </pre> où: – script_logical_name: définit le nom logique de l'objet "Script table"; – script_selector: définit le script identifier du script à exécuter; – switch_time accepte les caractères génériques pour définir les entrées répétitives. Le format de l'octet-string suit les règles établies en 4.1.6.1 pour time; – validity_window définit une période, en minutes, dans laquelle une entrée doit être traitée après une coupure secteur. (temps entre le switch_time défini et le power_up effectif) 0xFFFF: le script est traité à tout instant; – exec_weekdays définit les jours de la semaine pendant lesquels l'entrée est valide; – exec_specdays établit le lien à l'IC "Special days table", day_id; – begin_date et end_date définissent la date à laquelle l'entrée est valide (les caractères génériques sont permis). Le format suit les règles établies en 4.6.1 pour date.

Description de la méthode

enable/disable (data)	Met le bit désactivé des entrées de la page A à true (vrai) puis active les entrées de la page B.

```

data ::= structure
{
    firstIndexA: long-unsigned,
    lastIndexA: long-unsigned,
    firstIndexB: long-unsigned,
    lastIndexB: long-unsigned
}

```

firstIndexA	premier indice de la page qui est désactivée,
lastIndexA	dernier indice de la page qui est désactivée,
firstIndexB	premier indice de la page qui est activée,

	lastIndexB	dernier indice de la plage qui est activée,
	firstIndexA/B < lastIndexA/B:	toutes les entrées de la plage A/B sont désactivées/activées,
	firstIndexA/B = lastIndexA/B:	une entrée est désactivée/activée,
	firstIndexA/B > lastIndexA/B:	rien n'est désactivé/activé,
	firstIndexA/B et lastIndexA/B > 9999:	aucune entrée n'est désactivée/activée
insert (data)	Insère une nouvelle entrée dans le tableau. Si l'indice de l'entrée existe déjà, l'entrée existante est écrasée par la nouvelle entrée. entry (entrée): schedule_table_entry data::= correspondant à l'entrée	
delete (data)	Efface une plage d'entrées dans le tableau. data::= structure { firstIndex: long-unsigned, lastIndex: long-unsigned }	
	firstIndex:	premier indice de la plage qui est effacée,
	lastIndex:	dernier indice de la plage qui est effacée,
	firstIndex < lastIndex:	toutes les entrées de la plage A/B sont effacées,
	firstIndex = lastIndex:	une entrée est effacée,
	firstIndex > lastIndex:	rien n'est effacé

Remarques concernant les "incohérences" dans les entrées du tableau:

S'il convient d'exécuter plusieurs fois le même script à une instance de time spécifique, il n'est exécuté qu'une seule fois.

S'il convient d'exécuter des scripts différents dans la même instance de time, l'ordre d'exécution est selon l'indice. Le script à l'indice le plus bas est exécuté le premier.

Récupération après une coupure secteur

Après une coupure secteur, le programme tout entier est traité pour exécuter tous les scripts nécessaires qui seraient perdus pendant une coupure secteur. Pour cela, les entrées qui n'ont pas été exécutées pendant la coupure secteur sont à détecter. En fonction de la fenêtre de validité, elles sont exécutées dans l'ordre correct (selon lequel elles auraient été exécutées en fonctionnement normal).

Gestion des changements de temps

Il existe quatre "actions" différentes de changements de temps:

- réglage de l'heure en avance (en écrivant dans l'attribut *time* de l'objet "Clock");
- réglage de l'heure en retard (en écrivant dans l'attribut "time" de l'objet "Clock");
- synchronisation du temps (en utilisant la méthode *adjust_to_quarter* de l'objet "Clock");
- action d'économie d'énergie (heure d'été/hiver).

Ces quatre actions nécessitent une gestion différente exécutée par le programme en interaction avec l'activité de réglage de l'heure.

Réglage de l'heure en avance

Il est géré de la même façon qu'une coupure secteur. Toutes les entrées manquées sont exécutées en fonction de `validity_window`. Un réglage de temps court (défini et spécifique au fabricant) peut être géré comme synchronisation du temps.

Réglage de l'heure en retard

Il se traduit par une répétition des entrées qui sont activées pendant le temps répété. Un réglage de temps court (défini et spécifique au fabricant) peut être géré comme synchronisation du temps.

Synchronisation du temps

La synchronisation du temps est utilisée pour corriger de petits écarts entre l'horloge maîtresse et l'horloge locale. L'algorithme est spécifique au fabricant. Il doit garantir qu'aucune entrée du programme ne se perd ou n'est exécutée deux fois. L'attribut `validity_window` n'a aucun effet, car toutes les entrées sont à exécuter en fonctionnement normal.

Heure (été/hiver) de changement d'horaire légal

Si l'horloge est mise en avance, tous les scripts qui s'inscrivent dans l'intervalle d'avance (et seraient donc perdus) sont exécutés. Si l'horloge est remise en arrière, la réexécution des scripts qui s'inscrivent dans l'intervalle de retard est arrêtée.

5.4.4 Special days table (class_id = 11, version = 0)

Cette IC permet de définir des dates spéciales. À ces dates, un comportement de commutation spécial prévaut sur le comportement normal. L'IC fonctionne conjointement à la classe "Schedule" ou "Activity calendar". L'élément de donnée de liaison est `day_id`.

Special days table (tableau de jours spéciaux)	0...n	class_id = 11, version = 0			
Attributs	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. entries (static)	array				x + 0x08
Méthodes spécifiques	m/o				
1. insert (data)	f				x + 0x10
2. delete (data)	f				x + 0x18

Description d'attribut

logical_name	Identifie l'instance de l'objet "Special days table" (Tableau de jours spéciaux). Voir 6.2.8.
entries	Spécifie un identifiant de jour spécial pour une date donnée. La date peut comporter des caractères génériques pour les jours spéciaux qui se répètent, comme Noël. <pre>array spec_day_entry spec_day_entry ::= structure { index: long-unsigned, specialday_date: octet-string, day_id: unsigned }</pre> où: - le formatage de specialday_date suit les règles établies en 4.6.1 pour <i>date</i>; - la plage du day_id doit concorder avec la longueur de la chaîne binaire exec_specdays dans l'objet connexe de l'IC "Schedule".

Description de la méthode

insert (data)	Insère une nouvelle entrée dans le tableau. <pre>entry ::= spec_day_entry</pre> Si un jour spécial ayant le même indice ou la même date qu'un jour déjà défini est inséré, l'ancienne entrée sera écrasée.
delete (data)	Efface une entrée dans le tableau. <pre>index Indice de l'entrée qui doit être supprimée. data ::= long-unsigned</pre>

5.4.5 Activity calendar (class_id = 20, version = 0)

Cette IC permet de modéliser la gestion de diverses structures de tarification dans le compteur. L'IC fournit une liste d'actions programmées, suivant le format classique des programmes basés sur un calendrier en définissant des saisons, des semaines, etc.

Un objet "Activity calendar" peut coexister avec l'objet plus général "Schedule" et peut même se superposer à celui-ci. Si des actions dans un objet "Schedule" sont programmées pour la même heure d'activation que dans un objet "Activity calendar", les actions déclenchées par l'objet "Schedule" sont exécutées les premières.

Après une coupure secteur, la "last action" (dernière action) manquée issue de l'objet "Activity calendar" est exécutée (différée). Le but est d'assurer une tarification correcte après une mise sous tension. Si un objet "Schedule" est présent, la "dernière action" manquée de l'objet "Activity calendar" doit être exécutée à l'heure correcte dans la séquence d'actions demandées par l'objet "Schedule".

L'objet "Activity calendar" définit l'activation de certains scripts, qui peuvent accomplir différentes activités à l'intérieur du dispositif logique. L'interface à l'IC "Script table" est la même que dans le cas de l'IC "Schedule" (voir 5.4.3).

Si une instance de l'IC "Special days table" (voir 5.4.4) est disponible, les entrées pertinentes prévalent sur la sélection d'un profil de jour commandée par l'objet "Activity calendar". Le

profil de jour référencé dans le "Special days table" active le day_schedule du day_profile_table dans l'objet "Activity calendar" par référencement par le biais du day_id.

Activity calendar (calendrier d'activités)		0...n	class_id = 20, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	calendar_name_active (static)	octet-string				x + 0x08
3.	season_profile_active (static)	array				x + 0x10
4.	week_profile_table_active (static)	array				x + 0x18
5.	day_profile_table_active (static)	array				x + 0x20
6.	calendar_name_passive (static)	octet-string				x + 0x28
7.	season_profile_passive (static)	array				x + 0x30
8.	week_profile_table_passive (static)	array				x + 0x38
9.	day_profile_table_passive (static)	array				x + 0x40
10.	activate_passive_calendar_time (static)	octet-string				x + 0x48
Méthodes spécifiques		m/o				
1.	activate_passive_calendar (data)	f				x + 0x50

Description d'attribut

Les attributs appelés ..._active sont actuellement actifs. ceux appelés ..._passive seront activés par la méthode spécifique activate_passive_calendar.

logical_name Identifie l'instance de l'objet "Activity calendar" (Calendrier d'activités). Voir 6.2.10.

calendar_name Contient typiquement un identifiant, qui est descriptif du jeu de scripts activés par l'objet.

season_profile Contient une liste de saisons définies par leur date de début et un week_profile spécifique devant être exécuté. La liste est triée selon season_start (par ordre croissant):

array season

season ::= structure

```
{
    season_profile_name: octet-string,
    season_start:        octet-string,
    week_name:           octet-string
}
```

où:

- `season_profile_name` est un nom défini par l'utilisateur qui identifie le `season_profile` courant;
- `season_start` définit l'heure de début de la saison, formatée comme spécifié en 4.6.1 pour *date-time*. Les caractères génériques sont autorisés dans `season_start`. Si tous les champs sont des caractères génériques, la saison ne commencera jamais. L'utilisation de caractères génériques exige un soin particulier pour éviter toute paramétrisation conflictuelle, c'est-à-dire éviter que deux saisons différentes aient le même `season_start`;

NOTE La saison en cours est terminée par le `season_start` de la saison suivante.

- `week_name` définit le `week_profile` actif dans cette saison.

**week_profile_
table**

Contient un tableau de `week_profiles` à utiliser dans différentes saisons. Pour chaque `week_profile`, le `day_profile` pour chaque jour d'une semaine est identifié.

array `week_profile`

```

week_profile ::= structure
{
    week_profile_name:  octet_string,
    monday:             day_id,
    tuesday:            day_id,
    wednesday:          day_id,
    thursday:           day_id,
    friday:             day_id,
    saturday:           day_id,
    sunday:             day_id
}
day_id: unsigned

```

où:

- `week_profile_name` est un nom défini par l'utilisateur qui identifie le `week_profile` courant;
 - Monday (lundi) définit le `day_profile` valide le lundi;
 - ...
 - Sunday (dimanche) définit le `day_profile` valide le dimanche.
-

day_profile_table	Contient un tableau de day_profiles, identifiés par leur day_id. Pour chaque day_profile, une liste d'actions programmées est définie par un script à exécuter et l'heure d'activation (start_time) correspondante. La liste est triée selon start_time. array day_profile <pre> day_profile ::= structure { day_id: unsigned, day_schedule: array day_profile_action } day_profile_action ::= structure { start_time: octet-string, script_logical_name: octet-string, script_selector: long-unsigned } </pre> où: - day_id est un identifiant défini par l'utilisateur, qui identifie le day_profile courant; - start_time: définit l'heure à laquelle le script est à exécuter (aucun caractère générique); Le format suit les règles établies en 4.6.1 pour <i>time</i>; - script_logical_name: définit le <i>logical_name</i> de l'objet "Script table"; - script_selector: définit le script_identifiant du script à exécuter.
activate_passive_calendar_time	Définit l'heure à laquelle l'objet lui-même appelle la méthode spécifique <i>activate_passive_calendar</i>. Une définition avec une notation "not specified" (c'est-à-dire: non spécifié) dans tous les champs de l'attribut désactivera cet automatisme. Une notation "not specified" partielle, juste dans quelques champs de la date et de l'heure, n'est pas autorisée. octet-string, mis en forme comme indiqué en 4.6.1 pour <i>date-time</i>.
Description de la méthode	
activate_passive_calendar (data)	Cette méthode copie tous les attributs appelés ..._passive dans les attributs correspondants appelés ..._active. data ::= integer (0)

5.4.6 Register monitor (class_id = 21, version = 0)

Cette IC permet de modéliser la fonction de surveillance de valeurs modélisées par les objets "Data", "Register", "Extended register" ou "Demand register". Elle permet de spécifier des seuils, la valeur surveillée et un jeu de scripts (voir 5.4.2) qui sont exécutés lorsque la valeur surveillée franchit un seuil.

L'IC "Register monitor" requiert une instanciation de l'IC "Script table" dans le même dispositif logique.

Register monitor (Moniteur de registre)		0...n				class_id = 21, version = 0			
Attributs		Type de données		Min.	Max.	Def.	Nom court		
1.	logical_name (static)	octet-string					x		
2.	thresholds (seuils) (static)	array					x + 0x08		
3.	monitored_value (static)	value_definition					x + 0x10		
4.	actions (static)	array				0	x + 0x18		
Méthodes spécifiques		m/o							

Description d'attribut

logical_name Identifie l'instance de l'objet "Register monitor" (Moniteur de registre). Voir 6.2.13 et 6.3.10.

thresholds Fournit les valeurs de seuil auxquelles l'attribut du registre référencé est comparé.
array threshold
threshold: Le seuil est du même type que l'attribut surveillé de l'objet référencé.

monitored_value Définit quel attribut d'un objet est à surveiller. Seules sont autorisées les valeurs ayant des types de donnée simples.

```
value_definition ::= structure
{
    class_id:          long-unsigned,
    logical_name:      octet-string,
    attribute_index:   integer
}
```

actions Définit les scripts à exécuter lorsque l'attribut surveillé de l'objet référencé franchit le seuil correspondant. L'attribut *actions* a exactement le même nombre d'éléments que l'attribut *thresholds*. L'ordre de classement des *action_items* correspond à l'ordre de classement des *thresholds* (seuils, voir ci-dessus).

```
array      action_set
action_set ::= structure
{
    action_up:        action_item,
    action_down:      action_item
}
```

où:

- *action_up* définit l'action lorsque la valeur d'attribut du registre surveillé franchit le seuil dans le sens montant;
- *action_down* définit l'action lorsque la valeur d'attribut du registre surveillé franchit le seuil dans le sens descendant;

```
action_item ::= structure
{
    script_logical_name: octet-string,
    script_selector:     long-unsigned
}
```

5.4.7 Single action schedule (class_id = 22, version = 0)

Cette IC permet de modéliser l'exécution d'actions périodiques au sein d'un compteur. Ces actions ne sont pas nécessairement liées à la tarification (voir "Activity calendar" ou "Schedule").

Single action schedule (programme d'action unique)	0...n	class_id = 22, version = 0			
Attributs	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. executed_script (static)	script				x + 0x08
3. type (static)	enum				x + 0x10
4. execution_time (static)	array				x + 0x18
Méthodes spécifiques	m/o				

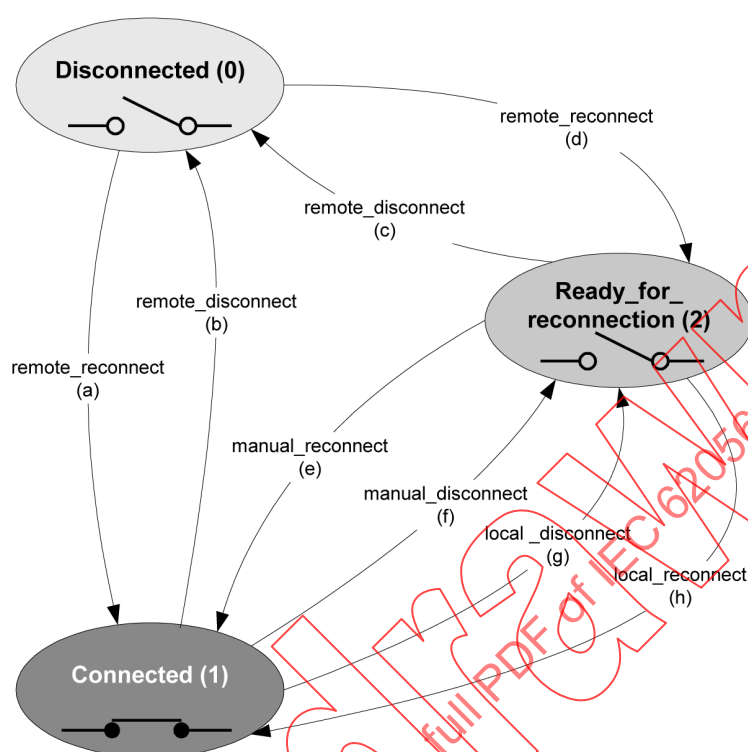
Description d'attribut

logical_name	Identifie l'instance d'objet "Single action schedule" (programme d'action unique). Voir 6.2.12.
executed_script	Contient le nom logique de l'objet "Script table" et le sélecteur de script du script à exécuter. script ::= structure { script_logical_name: octet-string, script_selector: long-unsigned } Script_logical_name et script_selector définissent le script à exécuter.
type	enum: (1) taille de execution_time = 1; caractère générique autorisé dans date, (2) taille de execution_time = n; toutes les valeurs de time sont les mêmes, les caractères génériques sont interdits dans date, (3) taille de execution_time = n; toutes les valeurs de time sont les mêmes, les caractères génériques sont autorisés dans date, (4) taille de execution_time = n; les valeurs de time peuvent être différentes, les caractères génériques sont interdits dans date, (5) taille de execution_time = n; les valeurs de time peuvent être différentes, les caractères génériques sont autorisés dans date,
execution_time	Spécifie l'heure et la date auxquelles le script est exécuté. array execution_time_date execution_time_date ::= structure { time: octet-string, date: octet-string } les deux octet-strings contiennent time et date, dans cet ordre; time et date sont formatées comme spécifié en 4.6.1. Les centièmes de seconde doivent être zéro.

5.4.8 Disconnect control (class_id = 70, version = 0)

Les instances de l'IC "Disconnect control" (commande de déconnexion) gèrent une unité de déconnexion interne ou externe du compteur (disjoncteur, robinet de gaz, par exemple) afin de brancher ou débrancher (partiellement ou totalement) les locaux du consommateur à/de

l'alimentation. Le diagramme d'états et les possibles transitions d'état sont montrés à la Figure 14.



IEC

Figure 14 – Diagramme d'états de l'IC "Disconnect control"

Une déconnexion et une reconnexion peuvent être demandées:

- À distance, via une voie de communication: remote_disconnect, remote_reconnect;
- Manuellement, à l'aide d'un bouton-poussoir, par exemple: manual_disconnect, manual_reconnect;
- Localement, par une fonction du compteur, par exemple limiteur, prépaiement: local_disconnect, local_reconnect.

Les états et les transitions d'état de l'IC "Disconnect control" sont montrés au Tableau 9. Les transitions d'état possibles dépendent du mode de commande. L'objet "Disconnect control" ne comporte pas de mémoire, c'est-à-dire que toute commande est exécutée immédiatement.

Afin de définir le comportement de l'objet "commande de déconnexion" pour chaque déclencheur, le mode de commande doit être configuré.

Tableau 9 – IC "Disconnect control" – états et transitions d'état

États		
Numéro d'état	Nom d'état	Description d'état
0	Disconnected (Déconnecté)	L' <i>output_state</i> est mis à FALSE et le consommateur est déconnecté.
1	Connected (Connecté)	L' <i>output_state</i> est mis à TRUE et le consommateur est connecté.
2	Ready_for_reconnection (prêt pour la reconnexion)	L' <i>output_state</i> est mis à FALSE et le consommateur est déconnecté.
Transitions d'état		
Transition	Nom de transition	Description d'état
a	remote_reconnect	Déplace l'objet "Disconnect control" de l'état Disconnected (0) directement à l'état Connected (1) sans intervention manuelle.
b	remote_disconnect	Déplace l'objet "Disconnect control" de l'état Connected (1) à l'état Disconnected (0).
c	remote_disconnect	Déplace l'objet "Disconnect control" de l'état Ready_for_reconnection (2) à l'état Disconnected (0).
d	remote_reconnect	Déplace l'objet "Disconnect control" de l'état Disconnected (0) à l'état Ready_for_reconnection (2). À partir de cet état, il est possible de passer à l'état Connected (2) via la transition manual_reconnect (e) ou la transition local_reconnect (h).
e	manual_reconnect	Déplace l'objet "Disconnect control" de l'état Ready_for_reconnection (2) à l'état Connected (1).
f	manual_disconnect	Déplace l'objet "Disconnect control" de l'état Connected (1) à l'état Ready_for_reconnection (2). À partir de cet état, il est possible de revenir à l'état Connected (2) via la transition manual_reconnect (e) ou la transition local_reconnect (h).
g	local_disconnect	Déplace l'objet "Disconnect control" de l'état Connected (1) à l'état Ready_for_reconnection (2). À partir de cet état, il est possible de revenir à l'état Connected (2) via la transition manual_reconnect (e) ou la transition local_reconnect (h). NOTE 1 Les transitions f) et g) sont essentiellement les mêmes, mais leur déclencheur est différent.
h	local_reconnect	Déplace l'objet "Disconnect control" de l'état Ready_for_reconnection (2) à l'état Connected (1). NOTE 2 Les transitions e) et h) sont essentiellement les mêmes, mais leur déclencheur est différent.

Disconnect control (commande de déconnexion)		0...n				class_id = 70, version = 0			
Attributs		Type de données		Min.	Max.	Def.	Nom court		
1.	logical_name (static)	octet-string					x		
2.	output_state (dyn.)	boolean					x + 0x08		
3.	control_state (dyn.)	enum					x + 0x10		
4.	control_mode (static)	enum					x + 0x18		
Méthodes spécifiques		m/o							
1.	remote_disconnect()	m					x + 0x20		
2.	remote_reconnect()	m					x + 0x28		

Description d'attribut

logical_name	Identifie l'instance de l'objet "Disconnect control" (commande de déconnexion). Voir 6.2.20 et 6.2.37.
output_state	Montre l'état physique réel de la connexion du dispositif à l'alimentation. boolean: TRUE = (BOOLEAN 1) Connecté, FALSE = (BOOLEAN 0) Déconnecté NOTE 1 En comptage d'électricité, l'alimentation est connectée lorsque le dispositif déconnecteur est fermé. NOTE 2 En comptage de gaz et d'eau, l'alimentation est connectée lorsque le robinet est ouvert.
control_state	Montre l'état interne de l'objet de commande de déconnexion. enum: (0) Disconnected (déconnecté), (1) Connected (connecté), (2) Ready_for_reconnection (prêt pour la reconnexion)
control_mode	Configure le comportement de l'objet commande de déconnexion pour tous les déclencheurs, c'est-à-dire les transitions d'état possibles.

control_ mode	Déconnexion				Reconnexion			
	Distante		Manuelle	Locale	Distante		Manuelle	Locale
enum:	(b)	(c)	(f)	(g)	(a)	(d)	(e)	(h)
(0)	—	—	—	—	—	—	—	—
(1)	x	x	x	x	—	x	x	—
(2)	x	x	x	x	x	—	x	—
(3)	x	x	—	x	—	x	x	—
(4)	x	x	—	x	x	—	x	—
(5)	x	x	x	x	—	x	x	x
(6)	x	x	—	x	—	x	x	x

NOTE 3 En Mode (0), l'objet commande de déconnexion est toujours à l'état "connecté".

NOTE 4 La déconnexion locale est toujours possible, sauf si le déclencheur correspondant est inhibé.

Description de la méthode

remote_disconnect (data)	Force l'objet "Disconnect control" à l'état "disconnected" si la déconnexion distante est activée (control mode > 0). data::= integer (0)
remote_reconnect (data)	Force l'objet "Disconnect control" à l'état "ready_for_reconnection" si une reconnexion distante directe est désactivée (control_mode = 1, 3, 5, 6). Force l'objet "Disconnect control" à l'état "connected" si une reconnexion distante directe est activée (control_mode = 2, 4). data::= integer (0)

5.4.9 Limiter (class_id = 71, version = 0)

Les instances de l'IC "Limiter" permettent de définir un jeu d'actions qui sont exécutées lorsque la valeur d'un attribut *value* d'un objet surveillé "Data", "Register", "Extended

Register", "Demand Register", etc., franchit la valeur seuil pendant au moins la durée minimale.

La valeur seuil peut être un seuil normal ou un seuil d'urgence. Le *seuil d'urgence* est activé via l'*emergency_profile* défini par l'*id de profil d'urgence*, le *temps d'activation d'urgence* et la *durée d'urgence*. L'élément *emergency profile id* (id de profil d'urgence) est mis en concordance avec un *emergency profile group id* (id de groupe de profils d'urgence): ce mécanisme permet l'activation du seuil d'urgence uniquement pour un groupe spécifique d'urgences.

Limiter (Limiteur)		0...n	class_id = 71, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. monitored_value	(static)	value_definition				x + 0x08
3. threshold_active	(dyn.)	threshold (seuil)				x + 0x10
4. threshold_normal	(static)	threshold (seuil)				x + 0x18
5. threshold_emergency	(static)	threshold (seuil)				x + 0x20
6. min_over_threshold_duration	(static)	double-long-unsigned				x + 0x28
7. min_under_threshold_duration	(static)	double-long-unsigned				x + 0x30
8. emergency_profile	(static)	emergency_profile				x + 0x38
9. emergency_profile_group_id_list	(static)	array				x + 0x40
10. emergency_profile_active	(dyn.)	boolean				x + 0x48
11. actions	(static)	action				x + 0x50
Méthodes spécifiques		m/o				

Description d'attribut

logical_name Identifie l'instance de l'objet "Limiter" (Limiteur). Voir 6.2.15.

monitored_value Définit un attribut d'un objet à surveiller. Seuls sont autorisés les attributs ayant des types de donnée simples.

value_definition ::= structure

{

class_id: long-unsigned,

logical_name: octet-string,

attribute_index: integer

}

threshold_active Fournit la valeur de seuil active à laquelle l'attribut surveillé est comparé.

threshold: Le seuil est du même type que l'attribut surveillé.

threshold_normal Fournit la valeur de seuil à laquelle l'attribut surveillé est comparé en fonctionnement normal.

threshold: voir ci-dessus.

threshold_ emergency	Fournit la valeur de seuil à laquelle l'attribut surveillé est comparé lorsqu'un profil d'urgence est actif. threshold: voir ci-dessus.
min_over_ threshold_duration	Définit la durée minimale au-dessus du seuil, en secondes, requise pour exécuter l'action au-dessus du seuil.
min_under_ threshold_duration	Définit la durée minimale en dessous du seuil, en secondes, requise pour exécuter l'action en dessous du seuil.
emergency_profile	Un <i>emergency_profile</i> est défini par trois éléments: <i>emergency_profile_id</i>, <i>emergency_activation_time</i>, <i>emergency_duration</i>. Un profil d'urgence est activé si l'élément <i>emergency_profile_id</i> concorde avec l'un des éléments de l'<i>emergency_profile_group_id_list</i>, et si le temps concorde avec l'<i>emergency_activation_time</i> et l'élément <i>emergency_duration</i>: <pre>emergency_profile::= structure { emergency_profile_id: long-unsigned, emergency_activation_time: octet-string, emergency_duration: double-long-unsigned }</pre> où: – l'élément <i>emergency_activation_time</i> définit la date et l'heure auxquelles l'<i>emergency_profile</i> s'est activé. L'octet-string est codé comme spécifié en 4.6.1 pour <i>date_time</i>, – l'élément <i>emergency_duration</i> définit la durée, en secondes, pour laquelle l'<i>emergency_profile</i> est activé. Lorsqu'un profil d'urgence est actif, l'attribut <i>emergency_profile_active</i> est mis à TRUE.
emergency_profile _group_id_list	Définit une liste des id de groupe du profil d'urgence. Le profil d'urgence peut être activé uniquement si l'élément <i>emergency_profile_id</i> de l'attribut <i>emergency_profile</i> concorde avec l'un des éléments de l'<i>emergency_profile_group_id_list</i>: <pre>array emergency_profile_group_id</pre> <pre>emergency_profile_group_id::= long-unsigned</pre>
emergency_profile _active	Indique que l' <i>emergency_profile</i> est actif.

actions	Définit les scripts à exécuter lorsque la valeur surveillée franchit le seuil pendant la durée minimale. <pre> action ::= structure { action_over_threshold: action_item, action_under_threshold: action_item } où: – action_over_threshold définit l'action lorsque la valeur de l'attribut surveillé franchit le seuil dans le sens montant et reste au-dessus du seuil pendant la durée minimale au-dessus du seuil; – action_under_threshold définit l'action lorsque la valeur de l'attribut surveillé franchit le seuil dans le sens descendant et reste en dessous du seuil pendant la durée minimale en dessous du seuil. action_item ::= structure { script_logical_name: octet-string, script_selector: long-unsigned } </pre>

5.4.10 Parameter monitor (class_id = 65, version = 0)

Les instances de l'IC "Parameter monitor" gèrent une liste d'attributs d'objet COSEM contenant des paramètres.

Les paramètres peuvent être modifiés, comme d'habitude. Si la valeur d'un attribut change et que cet attribut est présent dans l'attribut *parameter_list*, l'identifiant et la valeur de cet attribut sont automatiquement saisis dans l'attribut *changed_parameter*. L'heure de la modification du paramètre est saisie dans l'attribut *capture_time*. Ces attributs peuvent alors être saisis par un objet "Profile generic". De cette manière, un journal de toutes les modifications de paramètre peut être généré. Pour le code OBIS des objets "Parameter monitor log", voir 6.5 de l'IEC 62056-6-1.

NOTE 1 En cas de modifications de paramètre simultanées ou quasi simultanées, les modifications de l'ordre de saisie et de consignment des paramètres modifiés sont à gérer par l'application.

Plusieurs objets "Parameter monitor" et objets "Profile generic" correspondants peuvent être instanciés afin de gérer un certain nombre de groupes de paramètres. Le lien entre l'objet "Parameter monitor" et l'objet "Profile generic" correspondant est assuré par l'intermédiaire de l'attribut *capture_object* de l'objet "Profile generic".

NOTE 2 Les différents paramètres pouvant être de type et de longueur différents, les entrées de la colonne du profil contenant les paramètres peuvent également l'être. Cela peut être géré, par exemple, en saisissant différents paramètres dans différents objets "Profile generic" et journaux de paramètres d'une autre liste de paramètres.

NOTE 3 L'objet "Profile generic" contenant le journal de modification de paramètre peut saisir d'autres attributs d'objet pertinents, comme l'attribut *time* de l'objet "Clock", et d'autres valeurs appropriées.

Parameter monitor (moniteur de paramètre)		0...n				class_id = 65, version = 0			
Attributs		Type de données		Min.	Max.	Def.	Nom court		
1.	logical_name (static)	octet-string					x		
2.	changed_parameter	structure					x + 0x08		
3.	capture_time	date-time					x + 0x10		
4.	parameter_list	array					x + 0x18		
Méthodes spécifiques		m/o							
1.	add_parameter (data)	o					x + 0x20		
2.	delete_parameter (data)	o					x + 0x28		

Description d'attribut

logical_name Identifie l'instance de l'objet "Parameter monitor" (Moniteur de paramètre). Voir 6.2.14.

changed_parameter Contient l'identifiant et la valeur du dernier paramètre modifié.

structure

```
{
    class_id:      long-unsigned,
    logical_name:  octet-string,
    attribute_index: integer,
    attribute_value: CHOICE
    -- CHOICE comme indiqué dans la classe d'interfaces "Data"
}
```

capture_time Fournit les données et les informations relatives à l'heure de saisie de la valeur de l'attribut *changed_parameter*.

date-time formaté comme spécifié en 4.6.1.

parameter_list Contient la liste des paramètres gérés par une instance donnée de l'IC "Parameter monitor".

parameter_list::= array *parameter_list_element*

parameter_list_element::= structure

```
{
    class_id:      long-unsigned,
    logical_name:  octet_string,
    attribute_index: integer
}
```

NOTE 4 La liste des paramètres surveillés peut être modifiée grâce à la méthode *add_parameter* ou *delete_parameter* ou en écrivant cet attribut.

Description de la méthode

add_parameter (data) Ajoute un paramètre à *parameter_list*.

data::= *parameter_list_element*

NOTE 5 Un paramètre peut être consigné dès qu'il a été ajouté à la liste. L'ajout d'un élément à la liste n'a aucun impact sur le *buffer* de l'objet "Profile generic" qui saisit l'attribut *changed_parameter*.

delete_parameter (data) Supprime un paramètre de *parameter_list*.

data::= *parameter_list_element*

NOTE 6 Si un paramètre est supprimé de la liste de paramètres, ses modifications ne sont plus consignées. La suppression d'un élément de la liste n'a aucun impact sur le *buffer* de l'objet "Profile generic" qui saisit l'attribut *changed_parameter*.

5.4.11 Classe d'interfaces Sensor manager

5.4.11.1 Généralités

NOTE Cette classe d'interfaces est essentiellement utilisée dans le comptage des types d'énergie autres que l'électricité.

La plupart des instruments de mesure relevant du domaine d'application de la MID fonctionnent avec des capteurs dédiés (transducteurs et transmetteurs) reliés à l'unité de traitement. Ces capteurs sont à surveiller en permanence quant à leur fonctionnement et leurs limites pour satisfaire aux exigences métrologiques en vue d'un calcul ultérieur de valeurs monétaires.

De plus, les valeurs mesurées sont à surveiller. Ces valeurs peuvent être reliées à une grandeur physique – valeurs brutes de tension, de courant, de résistance, de fréquence, de sortie numérique – fournie par le capteur et aux grandeurs mesurées résultant du traitement des informations fournies par le capteur.

Il est nécessaire de surveiller et de journaliser souvent les valeurs pertinentes afin d'obtenir des informations de diagnostic qui permettent:

- l'identification du dispositif capteur;
- la connexion et l'état de scellé du capteur;
- la configuration des capteurs;
- la surveillance du fonctionnement des capteurs;
- la surveillance du résultat du traitement.

La classe d'interfaces "Sensor manager" permet de gérer les informations détaillées relatives à un capteur au moyen d'un seul objet.

Pour des capteurs/dispositifs plus simples, les objets COSEM déjà existants – identifiant les capteurs, détenant les valeurs de mesure et surveillant ces valeurs de mesure – peuvent être utilisés.

5.4.11.2 Sensor manager (class_id = 67, version = 0)

Les instances de l'IC "Sensor manager" gèrent des informations complexes relatives aux capteurs. Elles permettent aussi de surveiller les données brutes et la valeur traitée, dérivée par traitement des données brutes à l'aide d'algorithmes appropriés tels que requis par l'application particulière. Cette IC inclut un certain nombre de fonctions:

- données de plaque signalétique du capteur et informations de site (attributs 2 à 6);
- une fonction "Extended register" pour les *données brutes* (attributs 7 à 10);
- une fonction "Register monitor" pour les *données brutes* (attributs 11-12);

NOTE 1 Toutes les données brutes (par exemple la tension de sortie d'un capteur de pression) n'ont pas leur propre code/objet OBIS. C'est la raison pour laquelle les données brutes sont incluses dans la classe "Sensor manager".

- une fonction "Register monitor" pour la *processed_value* (attributs 13 à 15).

NOTE 2 Tous les "modules" ne sont pas nécessairement présents. Les attributs non utilisés peuvent ne pas être mis en œuvre ou ne pas être accessibles.

Sensor manager (Gestionnaire de capteur)		0...n	class_id = 67, version = 0			
Attribut(s)		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. serial_number	(dyn.)	octet-string				x + 0x08
3. metrological_identification	(dyn.)	octet-string				x + 0x10
4. output_type	(dyn.)	enum				x + 0x18
5. adjustment_method	(dyn.)	octet-string				x + 0x20
6. sealing_method	(dyn.)	enum				x + 0x28
7. raw_value	(dyn.)	CHOICE				x + 0x30
8. scaler_unit	(dyn.)	structure				x + 0x38
9. status	(dyn.)	CHOICE				x + 0x40
10. capture_time	(dyn.)	date-time				x + 0x48
11. raw_value_thresholds	(dyn.)	array				x + 0x50
12. raw_value_actions	(dyn.)	array				x + 0x58
13. processed_value	(dyn.)	processed_value_definition				x + 0x60
14. processed_value_thresholds	(dyn.)	array				x + 0x68
15. processed_value_actions	(dyn.)	array				x + 0x70
Méthodes spécifiques		m/o				
1. reset (data)		o				x + 0x80

Description d'attribut

logical_name Identifie l'instance de l'objet "Sensor manager" (Gestionnaire de capteur). Voir DLMS UA 1000-1, the "Blue Book" Ed. 12,0: 20143, 6.4.4.

serial_number Identifie le capteur (->informations de site).

NOTE 3 Pour un capteur simple, le numéro de série peut être détenu par les objets "Data" avec des codes OBIS appropriés s'il s'agit de la seule information exigée. Voir les objets d'usage général liés aux gaz, les objets de configuration à la DLMS UA 1000-1, the "Blue Book" Ed. 12,0: 20143, Tableau 77.

metrological_identification Décrit les informations du capteur pertinentes du point de vue métrologique, par exemple: identifiant métrologique, date d'étalonnage.

output_type décrit la sortie physique du capteur (->informations de site)

enum:

- (0) non spécifié,
- (1) 4–20 mA,
- (2) 0–20 mA
- (3) 0-5 V,
- (4) 0-10 V,
- (5) Pt100,
- (6) Pt500,
- (7) Pt1000,
- (8) HART

128 spécifique au fabricant

Toutes les autres valeurs sont réservées.

adjustment_method Décrit la méthode d'ajustement du capteur, par exemple par l'équation à trois points de mesure.

sealing_method Type de scellés appliqués au capteur.

	enum: (0) aucun, (1) mécanique (fil métallique ou étiquette de protection, par exemple); (2) électronique (contact, par exemple); (3) logiciel (protection par mot de passe, par exemple);
raw_value	Valeur physique provenant du capteur (par exemple: tension provenant d'un convertisseur de pression en tension). Pour les choix possibles de types de données, voir l'IC "Register" (5.2.2).
scaler_unit	Scalaire et l'unité de <i>raw_value</i>. Pour la définition de <i>scaler_unit</i>, voir l'IC "Register" (5.2.2).
status	Statut de la dernière <i>raw_value</i> saisie (pour la définition de <i>status</i>, voir l'IC "Extended register"). L'état conserve les informations comme: - défaillance de capteur, - capteur activé/non activé. La signification des éléments de <i>status</i> doit être fournie pour chaque instance de l'objet "Sensor manager".
capture_time	Fournit les informations de date et d'heure spécifiques à l'objet "Sensor manager" montrant le moment où la valeur de l'attribut <i>raw_value</i> a été saisie. <i>date_time</i> est formaté comme spécifié en 4.6.1. Pour les choix possibles de types de données, voir la classe d'interfaces "Extended register" (5.2.3).
raw_value_thresholds	Fournit les valeurs de seuil auxquelles l'attribut <i>raw_value</i> est comparé. array threshold threshold: Le "threshold" (seuil) est du même type que les "raw-data" (données brutes).
raw_value_actions	Définit les scripts à exécuter lorsque la <i>raw_value</i> franchit le seuil correspondant. L'attribut <i>raw_value_actions</i> a exactement le même nombre d'éléments que l'attribut <i>raw_value_thresholds</i>. L'ordre de classement des <i>action_items</i> correspond à l'ordre de classement des seuils. Pour la spécification des actions, voir l'IC <i>Register monitor</i> en 5.4.6.
processed_value	Référence l'attribut détenant la valeur traitée des données brutes fournies par le capteur. <i>processed_value_definition</i>::= structure <pre> { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer } </pre> À noter que plusieurs objets "Sensor manager" et objets de valeur traitée peuvent appartenir au même capteur.
processed_value_thresholds	Fournit les valeurs de seuil auxquelles la valeur traitée est comparée. array threshold threshold: Le seuil est du même type que la valeur traitée. (référéncée par l'attribut <i>processed-value</i>).

processed_value_actions Définit les scripts à exécuter lorsque la valeur traitée franchit le seuil correspondant.

L'attribut *processed_value_actions* a exactement le même nombre d'éléments que l'attribut *processed_value_thresholds*. L'ordre de classement des *action_items* correspond à l'ordre de classement des seuils.

Pour la spécification des actions, voir l'IC *Register monitor* en 5.4.6.

Description de la méthode

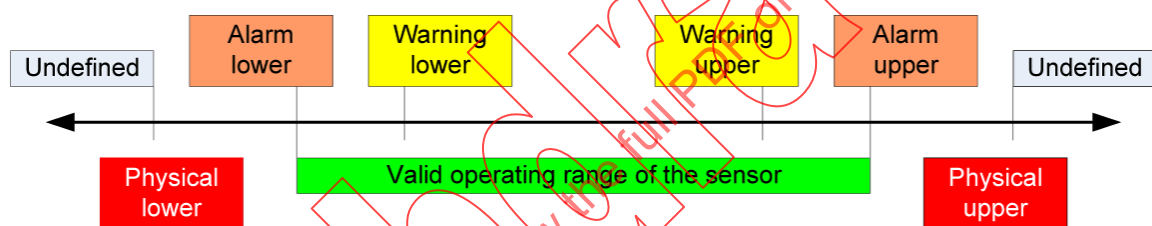
reset (data) Cette méthode réinitialise la *raw_value* à la valeur par défaut. La valeur par défaut est une constante spécifique à l'instance.

L'attribut *capture_time* est défini à l'heure de l'exécution de la réinitialisation.

data::= integer (0)

5.4.11.3 Exemple pour capteur de pression absolue

La Figure 15 représente la définition des seuils supérieur et inférieur applicables.



IEC

Anglais	Français
Undefined	Non défini
Alarm lower	Alarme (inférieur)
Warning lower	Avertissement (inférieur)
Warning upper	Avertissement (supérieur)
Alarm upper	Alarme (supérieur)
Physical lower	Physique (inférieur)
Valid operating range of the sensor	Plage de fonctionnement valide du capteur
Physical upper	Physique (supérieur)

Figure 15 – Définition des seuils supérieur et inférieur

Le Tableau 10 et le Tableau 11 montrent un exemple des divers seuils et des actions à exécuter lorsque les seuils sont franchis.

Tableau 10 – Présentation explicite des tableaux de valeurs seuils

Threshold (Seuil)	Physique (inférieur)	Physique (supérieur)	Alarme (inférieur)	Alarme (supérieur)	Avertissement (inférieur)	Avertissement (supérieur)
Value (Valeur)	1,0	5,5	1,2	5,0	1,4	4,8
scaler_unit	1, Volt	1, Volt	1, bar	1, bar	1, bar	1, bar

Tableau 11 – Présentation explicite d'action_sets

action_set	Physique (inférieur)	Physique (supérieur)	Alarme (inférieur)	Alarme (supérieur)	Avertissement (inférieur)	Avertissement (supérieur)
action_up	clr_phy_alarm_bit	set_phy_alarm_bit	clear_alarm_bit	set_alarm_bit	clear_warn_bit	set_warn_bit
action_down	set_phy_alarm_bit	clr_phy_alarm_bit	set_alarm_bit	clear_alarm_bit	set_warn_bit	clear_warn_bit

5.5 Classes d'interfaces pour l'établissement de l'échange de données via des ports locaux et des modems

5.5.1 IEC local port setup (class_id = 19, version = 1)

Cette IC permet de modéliser la configuration de ports de communication en utilisant les protocoles spécifiés dans l'IEC 62056-21:2002. Plusieurs ports peuvent être configurés.

IEC local port setup (Établissement de port local IEC)		0...n	class_id = 19, version = 1			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	default_mode (static)	enum				x + 0x08
3.	default_baud (static)	enum				x + 0x10
4.	prop_baud (static)	enum				x + 0x18
5.	response_time (static)	enum				x + 0x20
6.	device_addr (static)	octet-string				x + 0x28
7.	pass_p1 (static)	octet-string				x + 0x30
8.	pass_p2 (static)	octet-string				x + 0x38
9.	pass_w5 (static)	octet-string				x + 0x40
Méthodes spécifiques		m/o				

Description d'attribut	
logical_name	Identifie l'instance d'objet "IEC local port setup". Voir 6.2.16.
default_mode	Définit le protocole utilisé par le compteur sur le port. enum: (0) protocole selon l'IEC 62056-21:2002 (modes A...E), (1) protocole selon l'IEC 62056-46:2007. En utilisant cette valeur d'énumération, tous les autres attributs de cette IC ne sont pas applicables. (2) protocole non spécifié. En utilisant cette valeur d'énumération, l'attribut 4), <i>prop_baud</i> , est utilisé pour régler la vitesse de communication sur le port. Tous les autres attributs ne sont pas applicables.
default_baud	Définit le débit en bauds pour la séquence d'ouverture. enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud
prop_baud	Définit la vitesse en bauds devant être proposée par le compteur. enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud
response_time	Définit le temps minimum entre la réception d'une requête (fin de télégramme de requête) et l'émission de la réponse (début de télégramme de réponse). enum: (0) 20 ms, (1) 200 ms
device_addr	Adresse de dispositif selon l'IEC 62056-21:2002.
pass_p1	Mot de passe 1 selon l'IEC 62056-21:2002.
pass_p2	Mot de passe 2 selon l'IEC 62056-21:2002.
pass_w5	Mot de passe W5 réservé pour des applications nationales.

5.5.2 IEC HDLC setup (class_id = 23, version = 1)

Cette IC permet de modéliser et configurer des voies de communication selon l'IEC 62056-46:2007.

Plusieurs voies de communication peuvent être configurées.

IEC HDLC setup (Établissement de HDLC IEC)		0...n	class_id = 23, version = 1			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. comm_speed	(static)	enum	0	9	5	x + 0x08
3. window_size_transmit	(static)	unsigned	1	7	1	x + 0x10
4. window_size_receive	(static)	unsigned	1	7	1	x + 0x18
5. max_info_field_length_transmit	(static)	long-unsigned	32	2030	128	x + 0x20
6. max_info_field_length_receive	(static)	long-unsigned	32	2030	128	x + 0x28
7. inter_octet_time_out	(static)	long-unsigned	20	6000	25	x + 0x30
8. inactivity_time_out	(static)	long-unsigned	0		120	x + 0x38
9. device_address	(static)	long-unsigned	0x0010	0x3FFD		x + 0x40
Méthodes spécifiques		m/o				

NOTE 1 La valeur maximale des attributs *max_info_field_length_transmit* et *max_info_field_length_receive* a été augmentée de 128 à 2 030 pour des raisons d'efficacité.

NOTE 2 Un *max_info_field_length_receive* de 128 octets est nécessaire afin d'assurer une performance minimale.

NOTE 3 La valeur maximale de l'attribut *inter_octet_time_out* a été augmentée de 1 000 ms à 6 000 ms afin de permettre l'utilisation de supports de communications, lorsqu'il peut se produire de longs retards. La valeur par défaut a été changée à 25 ms pour l'aligner avec 6.4.4.3.4 de l'IEC 62056-46:2007.

Description d'attribut

logical_name	Identifie l'instance de l'objet "IEC HDLC setup" (Établissement de HDLC IEC). Voir 6.2.18.
comm_speed	La vitesse de communication prise en charge par le port correspondant. enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud Cette vitesse de communication peut être neutralisée si le passage au mode HDLC d'un dispositif se fait par le biais d'un mode spécial d'un autre protocole.
window_size_transmit	Le nombre maximal de trames qu'un dispositif ou système peut émettre avant qu'il ait besoin de recevoir un acquittement provenant de la station correspondante. D'autres valeurs peuvent être négociées pendant l'ouverture de session.
window_size_receive	Le nombre maximal de trames qu'un dispositif ou système peut recevoir avant qu'il ait besoin d'émettre un acquittement vers la station correspondante. D'autres valeurs peuvent être négociées pendant l'ouverture de session.
max_info_length_transmit	La longueur maximale du champ information qu'un dispositif peut émettre. Une valeur inférieure peut être négociée pendant l'ouverture de session.

max_info_length_receive	La longueur maximale du champ information qu'un dispositif peut recevoir. Une valeur inférieure peut être négociée pendant l'ouverture de session.																				
inter_octet_time_out	Définit le temps, en millisecondes, au-delà duquel, si aucun autre caractère n'est reçu en provenance de la station primaire, le dispositif traitera les données déjà reçues comme une trame complète.																				
inactivity_time_out	Définit le temps, en secondes, au-delà duquel, si aucune trame n'est reçue en provenance de la station primaire, le dispositif traitera une déconnexion. Lorsque cette valeur est mise à 0, cela signifie que l'inactivity_time_out n'est pas opérationnel.																				
device_address	Contient l'adresse de dispositif physique d'un dispositif. Dans le cas de l'adressage d'un octet: <table> <tr> <td>0x00</td><td>Adresse NO_STATION,</td></tr> <tr> <td>0x01...0x0F</td><td>Réservé pour utilisation ultérieure,</td></tr> <tr> <td>0x10...0x7D</td><td>Espace d'adresses utilisable,</td></tr> <tr> <td>0x7E</td><td>Adresse de dispositif CALLING,</td></tr> <tr> <td>0x7F</td><td>Adresse de diffusion</td></tr> </table> Dans le cas de l'adressage de deux octets: <table> <tr> <td>0x0000</td><td>Adresse NO_STATION,</td></tr> <tr> <td>0x0001...0x000F</td><td>Réservé pour utilisation ultérieure,</td></tr> <tr> <td>0x0010...0x3FFD</td><td>Espace d'adresses utilisable,</td></tr> <tr> <td>0x3FFE</td><td>Adresse de dispositif physique CALLING,</td></tr> <tr> <td>0x3FFF</td><td>Adresse de diffusion</td></tr> </table>	0x00	Adresse NO_STATION,	0x01...0x0F	Réservé pour utilisation ultérieure,	0x10...0x7D	Espace d'adresses utilisable,	0x7E	Adresse de dispositif CALLING,	0x7F	Adresse de diffusion	0x0000	Adresse NO_STATION,	0x0001...0x000F	Réservé pour utilisation ultérieure,	0x0010...0x3FFD	Espace d'adresses utilisable,	0x3FFE	Adresse de dispositif physique CALLING,	0x3FFF	Adresse de diffusion
0x00	Adresse NO_STATION,																				
0x01...0x0F	Réservé pour utilisation ultérieure,																				
0x10...0x7D	Espace d'adresses utilisable,																				
0x7E	Adresse de dispositif CALLING,																				
0x7F	Adresse de diffusion																				
0x0000	Adresse NO_STATION,																				
0x0001...0x000F	Réservé pour utilisation ultérieure,																				
0x0010...0x3FFD	Espace d'adresses utilisable,																				
0x3FFE	Adresse de dispositif physique CALLING,																				
0x3FFF	Adresse de diffusion																				

5.5.3 IEC twisted pair (1) setup (class_id = 24, version = 1)

5.5.3.1 Vue d'ensemble

Le support de communication *paire torsadée avec signal de porteuse* est largement utilisé dans le domaine des compteurs. L'utilisation de ce support présente l'avantage de faciliter l'installation et d'augmenter la fiabilité des communications grâce au signal de porteuse. Le support peut être utilisé:

- entre les points d'accès au réseau local (LNAP – Local Network Access Point) et les dispositifs d'extrémité du compteur (interface M);
- entre les points d'accès au réseau local (LNAP) et les points d'accès aux réseaux avoisinants (NNAP – Neighbourhood Network Access Point); et
- pour une connexion directe entre un HHU et le dispositif d'extrémité du compteur.

L'IEC 62056-3-1:2013 spécifie trois profils de communication utilisant le support à paire torsadée avec signal de porteuse:

- sans DLMS;
- avec DLMS; et
- avec DLMS/COSEM.

L'IEC 62056-31:1999 prend en charge uniquement les deux premiers profils.

Le nouveau profil DLMS/COSEM introduit une entité Couche gestion du support permettant d'initialiser le bus, la gestion des reconnaissances, la gestion des alarmes et la négociation de la vitesse de communication. Il permet également d'augmenter le débit en bauds jusqu'à 9 600 Bd. La Couche de transport prend en charge la segmentation et le réassemblage.

L'IC "IEC Twisted pair (1) set up" (class_id = 24, version = 0) prend en charge les deux premiers profils de communication spécifiés dans l'IEC 62056-3-1:1999.

La nouvelle version 1 prend en charge le profil DLMS/COSEM. Avec cette introduction, l'utilisation de la version 0 est devenue obsolète.

L'utilisation des profils de communication spécifiés dans l'IEC 62056-3-1:2013 exige d'utiliser les services d'enregistrement fournis par Euridis Association: www.euridis.org.

Les objets d'interface COSEM suivants sont indispensables à l'établissement de l'échange de données sur le support *Paire torsadée avec signal de porteuse*:

- "IEC Twisted pair (1) setup": class_id = 24, version = 1;
- "MAC address": class_id = 43, version = 0;
- "Data" class_id = 1, version = 0.

Pour les codes OBIS, voir 6.2.19.

5.5.3.2 IEC twisted pair (1) setup (class_id = 24, version = 1)

Les instances de cette IC permettent de configurer l'échange de données sur le support *Paire torsadée avec signal de porteuse* comme indiqué dans l'IEC 62056-3-1:2013. Plusieurs voies de communication peuvent être configurées.

IEC twisted pair (1) setup (établissement de paire torsadée (1) IEC)		0...n	class_id = 24, version = 1			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	mode (static)	enum	0		1	x + 0x08
3.	comm_speed (static)	enum	(2)	(7)	(2)	x + 0x10
4.	primary_address_list (static)	primary_address_list_type				x + 0x18
5.	tabi_list (static)	tabi_list_type				x + 0x20
Méthodes spécifiques		m/o				

Description d'attribut													
logical_name	Identifie l'instance d'objet "IEC twisted pair setup". Voir 6.2.19.												
mode	Cet attribut spécifie le mode de fonctionnement de cette interface enum: <table> <tr> <td>(0)</td><td>inactive. L'interface ignore toutes les trames</td></tr> <tr> <td>(1)</td><td>reçues,</td></tr> <tr> <td>(2) à (127)</td><td>toujours active,</td></tr> <tr> <td>(128) à (250)</td><td>réservé,</td></tr> <tr> <td></td><td>spécifique au fabricant.</td></tr> </table>	(0)	inactive. L'interface ignore toutes les trames	(1)	reçues,	(2) à (127)	toujours active,	(128) à (250)	réservé,		spécifique au fabricant.		
(0)	inactive. L'interface ignore toutes les trames												
(1)	reçues,												
(2) à (127)	toujours active,												
(128) à (250)	réservé,												
	spécifique au fabricant.												
comm_speed	Contient la vitesse de communication prise en charge par le port. enum: <table> <tr> <td>(2)</td><td>1 200 baud,</td></tr> <tr> <td>(3)</td><td>2 400 baud,</td></tr> <tr> <td>(4)</td><td>4 800 baud,</td></tr> <tr> <td>(5)</td><td>9 600 baud,</td></tr> <tr> <td>(6)</td><td>19 200 baud,</td></tr> <tr> <td>(7)</td><td>38 400 baud</td></tr> </table> NOTE L'IEC 62056-3-1:2013 prend en charge des débits en bauds compris entre 1 200 et 9 600.	(2)	1 200 baud,	(3)	2 400 baud,	(4)	4 800 baud,	(5)	9 600 baud,	(6)	19 200 baud,	(7)	38 400 baud
(2)	1 200 baud,												
(3)	2 400 baud,												
(4)	4 800 baud,												
(5)	9 600 baud,												
(6)	19 200 baud,												
(7)	38 400 baud												
primary_address_list	Contient la liste des ADP (Primary Station Address) pour lesquelles chaque dispositif logique de l'équipement réel (la station secondaire) a été programmé. Voir IEC 62056-3-1:2013, 5.2.4. primary_address_list_type ::= array unsigned												
tabi_list	Représente la liste des TAB(I) pour lesquels l'équipement réel (la station secondaire) a été programmé dans le cas d'un appel de station publié (voir l'IEC 62056-3-1:2013). Lors de l'utilisation du profil de l'IEC 62056-3-1 avec DLMS/COSEM, l'attribut <i>tabi_list</i> est composé d'un seul élément de valeur 0, qui est la valeur utilisée pour le processus de reconnaissance. tabi_list_type ::= array tabi_element tabi_element: integer												

5.5.3.3 MAC address (adresse MAC)

Les stations secondaires (en général des dispositifs d'extrémité de compteur) contiennent une adresse de station secondaire (ADS). L'ADS doit être composée de 6 octets et des éléments présentés au Tableau 12. Cette adresse doit être unique dans le monde entier et doit être attribuée par le fabricant à la station secondaire. L'ADS attribuée est valide sur toute la durée de vie de la station secondaire.

Tableau 12 – Éléments de l'adresse ADS

Éléments de l'ADS	Description	Longueur (octet)	Type	Plage
manufacturer_id	Attribué à la demande d'Euridis Association au fabricant. Il doit être utilisé dans tous les dispositifs utilisant le support <i>Paire torsadée avec signal de porteuse</i> et fabriqué par ce fabricant.	1	BCD	0-99
year_of_manufacture	Contient l'année de fabrication du dispositif (les deux nombres inférieurs uniquement).	1		0-99
equipment_id	Relatif au type de matériel concerné et donne des informations relatives aux fonctions disponibles. Comme manufacturer_id, il est attribué par Euridis Association.	1		0-99
device_serial_number	Numéro de fabrication du dispositif attribué par le fabricant. Il convient que sa valeur soit identique à celle contenue par l'objet Device ID 1 (voir IEC 62056-6-1 :2015, Tableau 7).	3		000001-999999 000000 est réservé
EXEMPLE 031267123456: 03: ITRON International; 12: Année 2012; 67: Compteur intelligent LINKY monophasé 123456: Numéro de série commençant chaque année à partir de 000001.				

5.5.3.4 Registre des erreurs fatales

Chaque dispositif mettant en œuvre le profil de communication DLMS/COSEM spécifié dans IEC 62056-3-1:2013 doit fournir un registre des erreurs contenant le résultat de la dernière communication avec la station primaire. La structure du registre des erreurs fatales doit être conforme au Tableau 13.

Tableau 13 – Registre des erreurs fatales

Réf	Nom	Description
Bit 0	EP-3F	Erreurs de transmission Le TOE d'expiration du délai s'est écoulé sans envoi d'octet, donnant lieu à une impossibilité d'envoyer le reste de la trame.
Bit 1	EP-4F	Erreur de réception. Le nombre d'octets reçus est supérieur à la valeur maximale prévue.
Bit 2	EP-5F	Délai TARSO écoulé en recevant une trame RSO. Ne concerne pas les stations secondaires (serveur). Concerne la station primaire seulement (client).
Bit 3	EL-1F	Indication d'alarme reçue pendant une association. Non pertinent. Concerne la station primaire seulement (client).
Bit 4	EL-2F	Réponse incorrecte de la Station Secondaire après transmissions répétées MaxRetry d'une demande.
Bit 5	EA-1F	TAB incorrect depuis le serveur. Ne concerne pas les stations secondaires (serveur). Concerne la station primaire seulement (client).
Bit 6	EA-2F	Erreur d'authentification sur les données reçues du serveur. Ne concerne pas les stations secondaires (serveur). Concerne la station primaire seulement (client).
Bit 7	EA-3F	Erreur d'authentification détectée par la station secondaire.

5.5.4 Modem configuration (class_id = 27, version = 1)

Cette IC permet de modéliser la configuration et l'initialisation des modems utilisés pour le transfert de données en provenance/à destination d'un dispositif. Plusieurs modems peuvent être configurés.

Modem configuration (configuration de modem)	0...n	class_id = 27, version = 1			
Attributs	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. comm_speed (static)	enum	0	9	5	x + 0x08
3. initialization_string (static)	array				x + 0x10
4. modem_profile (static)	array				x + 0x18
Méthodes spécifiques	m/o				

Description d'attribut

logical_name Identifie l'instance de l'objet "Modem configuration" (Configuration de modem). Voir 6.2.6.

comm_speed La vitesse de communication entre le dispositif et le modem, pas nécessairement la vitesse de communication sur le réseau étendu.

enum:

- (0) 300 baud,
- (1) 600 baud,
- (2) 1 200 baud,
- (3) 2 400 baud,
- (4) 4 800 baud,
- (5) 9 600 baud,
- (6) 19 200 baud,
- (7) 38 400 baud,
- (8) 57 600 baud,
- (9) 115 200 baud

initialization_string Contient toutes les commandes d'initialisation nécessaires devant être envoyées au modem afin de le configurer correctement. Cela peut inclure la configuration de caractéristiques spéciales du modem.

array initialization_string_element

```
initialization_string_element ::= structure
{
    request:                octet-string,
    response:               octet-string,
    delay_after_response:   long-unsigned
}
```

Si le tableau contient plus d'un initialization_string_element, les demandes peuvent être envoyées en séquence. La demande suivante est envoyée après la réponse attendue correspondant à la demande précédente et après une attente d'un temps delay_after_response [ms], pour permettre au modem d'exécuter la demande.

NOTE Il est supposé que le modem est préconfiguré et accepte, de ce fait, l'initialization_string (c'est-à-dire chaîne d'initialisation). Si l'initialisation n'est pas nécessaire, la chaîne d'initialisation est vide.

modem_profile Définit la mise en correspondance des commandes/réponses de la norme Hayes avec les chaînes spécifiques au modem.

array modem_profile_element

modem_profile_element: octet-string

Le tableau modem_profile doit contenir les chaînes correspondantes pour le modem utilisé dans l'ordre suivant:

Élément 0: OK,
 Élément 1: CONNECT,
 Élément 2: RING,
 Élément 3: NO CARRIER,
 Élément 4: ERROR,
 Élément 5: CONNECT 1 200,
 Élément 6: NO DIAL TONE,
 Élément 7: BUSY,
 Élément 8: NO ANSWER,
 Élément 9: CONNECT 600,
 Élément 10: CONNECT 2 400,
 Élément 11: CONNECT 4 800,
 Élément 12: CONNECT 9 600,
 Élément 13: CONNECT 14 400,
 Élément 14: CONNECT 28 800,
 Élément 15: CONNECT 33 600,
 Élément 16: CONNECT 56 000

5.5.5 Auto answer (class_id = 28, version = 2)

NOTE 1 La version 1 de la classe "Auto answer" était une version provisoire.

La version 0 de la classe "Auto answer" modélise la manière dont le dispositif gère les appels entrants pour demander la connexion du modem.

Dans la version 2, de nouvelles capacités sont ajoutées pour gérer les demandes de réactivation qui peuvent se présenter sous la forme d'un appel ou d'un message de réactivation (un message SMS (vide), par exemple). Suite à une demande de réactivation réussie, le dispositif se connecte au réseau. Voir également l'Annex A.

Pour les deux fonctions, une sécurité supplémentaire est assurée en ajoutant la possibilité de vérification du numéro appelant: les appels ou les messages sont acceptés uniquement en fonction d'une liste préalablement définie d'appelants. Cette fonction exige la présence d'un service d'identification de la ligne appelante (CLI) dans le réseau de communication utilisé.

NOTE 2 Le processus de réactivation est totalement découplé des services AL, c'est-à-dire qu'un message de réactivation ne peut pas contenir de demandes de service xDLMS. Il s'agit d'éviter la création d'une porte dérobée. Les messages xDLMS peuvent être échangés dans les messages SMS à l'issue du processus de réactivation

Auto answer (réponse automatique)		0...n	class_id = 28, version = 2			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. mode	(static)	enum				x + 0x08
3. listening_window	(static)	array				x + 0x10
4. status	(dyn.)	enum				x + 0x18
5. number_of_calls	(static)	unsigned				x + 0x20
6. number_of_rings	(static)	nr_rings_type				x + 0x28
7. list_of_allowed_callers	(static)	array				x + 0x30
Méthodes spécifiques		m/o				

Description d'attribut

logical_name	Identifie l'instance de l'objet "Auto answer" (Réponse automatique). Voir 6.2.6.
mode	Définit le mode de fonctionnement de la ligne lorsque le dispositif est en mode réponse automatique. enum: (0) ligne dédiée au dispositif, (1) gestion de ligne partagée avec un nombre limité d'appels autorisés. Une fois que le nombre d'appels est atteint, l'état de la fenêtre devient "inactif" jusqu'à la prochaine date de début, quel que soit le résultat de l'appel, (2) gestion de ligne partagée avec un nombre limité d'appels réussis autorisés. Une fois que le nombre de communications réussies est atteint, l'état de la fenêtre devient "inactif" jusqu'à la prochaine date de début, (3) actuellement aucun modem connecté, (200...255) modes spécifiques au fabricant
listening_window	Contient l'instant de début où la fenêtre de communication devient active (start_time) et inactive (end_time). Le start_time définit implicitement la période. EXEMPLE Lorsque le jour du mois n'est pas spécifié (égal à 0xFF), cela signifie qu'il s'agit de la gestion d'une ligne d'écoute quotidienne. Une gestion de fenêtre quotidienne, mensuelle ... peut être définie. array window_element <pre> window_element ::= structure { start_time: octet-string, end_time: octet-string } </pre> start_time et end_time sont formatés comme spécifié en 4.6.1 pour <i>date-time</i>.

status	Ici est défini l'état de la fenêtre. enum: (0) Inactive: le dispositif ne gérera aucun nouvel appel entrant. Cet état est automatiquement réinitialisé à "Active" lorsque la fenêtre d'écoute suivante commence, (1) Active: le dispositif peut répondre au nouvel appel entrant, (2) Locked (verrouillée): Cette valeur peut être fixée automatiquement par le dispositif ou par un client spécifique lorsque ce client a terminé sa session de lecture et souhaite rendre la ligne au client avant la fin de la durée de la fenêtre. Cet état est automatiquement réinitialisé à "Active" lorsque la fenêtre d'écoute suivante commence.
number_of_calls	Ce nombre est la référence utilisée dans les modes 1 et 2. Lorsqu'il est mis à 0, cela signifie qu'il n'y a pas de limite.
number_of_rings	Définit le nombre de sonneries avant que le compteur ne connecte le modem. Deux cas sont distingués: nombre de sonneries dans la fenêtre définie par l'attribut <i>listening_window</i> et nombre de sonneries à l'extérieur de la <i>listening_window</i>. <pre>nr_rings_type::= structure { nr_rings_in_window: unsigned (0 = pas de connexion dans la fenêtre), nr_rings_out_of_window: unsigned (0 = pas de connexion hors de la fenêtre) }</pre> Si le nombre de sonneries à l'intérieur et à l'extérieur de la fenêtre est le même, le modem se connecte toujours quels que soient les paramètres de <i>listening_window</i>.

list_of_allowed_callers

Contient une liste (facultative) de numéros appelants qui limitent encore plus la connectivité du modem selon le numéro appelant. Il contrôle également l'acceptation des appels ou messages de réactivation (SMS, par exemple) provenant d'un numéro appelant.

Cela exige la présence d'un service d'identification de la ligne appelante (CLI) dans le réseau de communication utilisé.

list_of_allowed_callers::= array list_of_allowed_callers_element

list_of_allowed_callers_element::= structure

```
{
    caller_id:  octet-string,
    call_type:  enum
}
```

- l'élément caller_id contient un numéro appelant à partir duquel les appels ou messages (SMS, par exemple) sont acceptés. Les caractères génériques "?" et "*" sont pris en charge. Avec "?" un seul caractère correspond. Avec "*", toutes les chaînes de caractères correspondent. "*" peut uniquement être utilisé au début ou à la fin d'un numéro, mais jamais au milieu, ni seul.

Exemple 1: "+994193500" = seuls les appels provenant du numéro "+994193500" sont acceptés.

Exemple 2: "+9941935?????" = les appels provenant de tous les numéros compris entre "+99419350000" et "+99419359999" sont acceptés.

Exemple 3: "7777*" = les appels provenant de tous les numéros commençant par "7777" sont acceptés.

Exemple 4: "*9000" = les appels provenant de tous les numéros se terminant par "9000" sont acceptés.

**list_of_
allowed_
callers**

(suite)

- l'élément `call_type` définit l'objet de l'appel, c'est-à-dire s'il s'agit d'un appel CSD normalisé ou d'un appel/message de réactivation.

enum:

- (0) = appel CSD normal; le modem se connecte uniquement si le numéro appelant correspond à une entrée de la liste. Il est soumis à essai en plus de tous les autres attributs (*number_of_rings*, *listening_windows*, par exemple).
- (1) = requête de réactivation; les appels ou messages provenant de ce numéro appelant sont gérés comme des requêtes de réactivation. La requête de réactivation est immédiatement traitée, quels que soient les autres attributs comme *number_of_rings* et *listening_window* (sauf si le numéro appelant est également présent dans la liste des appels CSD normaux. Voir la Note 3 ci-dessous).

Le message reçu doit être complètement vide. Sinon, il n'est pas traité comme un message de réactivation.

Si le message contient une xDLMS APDU valide provenant d'un client d'une AA préalablement établie, le service xDLMS correspondant est exécuté à la place du traitement de la requête de réactivation.

Si le message n'est pas vide, mais qu'il ne contient pas d'APDU xDLMS valide provenant d'un client d'une AA préalablement établie, le serveur ne réagit pas.

Pour un appel de type (1), le modem *ne se connecte pas*, quel que soit le résultat du processus de réactivation.

Pour un appel de type (1) et de type (0): voir Note 3.

NOTE 3 Si le même numéro appelant est défini comme étant à l'origine d'un appel CSD normal (type (0)) et d'une requête de réactivation (type (1)), la règle suivante s'applique pour les appels entrants: la requête de réactivation est uniquement traitée si l'appelant déconnecte la ligne avant d'atteindre le critère *number_of_rings* (en fonction de *listening_window*). Si le critère *number_of_rings* est satisfait, une connexion modem est établie.

La valeur du paramètre *number_of_rings* doit être suffisamment élevée pour permettre à l'initiateur de l'appel de contrôler le comportement du récepteur de l'appel sans connaître les instances temporelles des sonneries côté récepteur.

NOTE 4 Si *list_of_allowed_callers* est vide (= tableau [0]), la fonction de réponse automatique fonctionne dans le type (0) = appel CSD normal, et le modem se connecte quel que soit le numéro appelant.

5.5.6 Auto connect (class_id = 29, version = 2)

La version 1 de la classe "Auto connect" modélise la manière dont le dispositif procède à la numérotation automatique ou à l'envoi des messages à l'aide de différents services.

Dans la version 2, de nouvelles capacités ont été ajoutées afin de modéliser la connexion du dispositif à un réseau de communication. La connexion au réseau peut être permanente, à l'intérieur d'une fenêtre de temps ou à l'appel de la méthode de connexion.

Auto connect (connexion automatique)		0...n	class_id = 29, version = 2			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	mode (static)	enum				x + 0x08
3.	repetitions (static)	unsigned				x + 0x10
4.	repetition_delay (static)	long-unsigned				x + 0x18
5.	calling_window (static)	array				x + 0x20
6.	destination_list (static)	array				x + 0x28
Méthodes spécifiques		m/o				
1.	connect (data)	o				x + 0x30

Description d'attribut	
logical_name	Identifie l'instance de l'objet "Auto connect" (Connexion automatique). Voir 6.2.6.
mode	Commande la fonction de connexion automatique en termes de synchronisation, de type de message et d'infrastructure à utiliser. Les modes (1) à (3) sont dédiés aux services CSD. Les modes (4) à (6) sont dédiés à l'envoi de messages particuliers à l'aide d'une infrastructure particulière. Les modes (101) à (104) s'appliquent aux connexions de réseau à commutation de paquets uniquement (GPRS, par exemple). enum: (0) pas de connexion automatique; le dispositif ne se connecte jamais, (1) numérotation automatique autorisée à tout moment, les valeurs définies dans <i>calling_window</i> sont ignorées, (2) numérotation automatique autorisée dans la limite du temps de validité de <i>calling_window</i>, (3) numérotation automatique "régulière" autorisée dans la limite du temps de validité de <i>calling_window</i>; numérotation automatique déclenchée par une "alarme" autorisée à tout moment, (4) envoi de SMS via les réseaux mobiles terrestres publics (PLMN), (5) envoi de SMS via le RPTC, (6) envoi de courriel (email), (7...99) réservé, (101) le dispositif est connecté en permanence au réseau de communication, (102) le dispositif est connecté en permanence au réseau de communication dans la durée de validité de la fenêtre d'appel. Le dispositif est déconnecté hors de la fenêtre d'appel. Aucune connexion possible hors de la fenêtre d'appel, (103) le dispositif est connecté en permanence au réseau de communication dans la durée de validité de la fenêtre d'appel. Le dispositif est déconnecté hors de la fenêtre d'appel, mais il se connecte au réseau de communication dès que la méthode de connexion est appelée, (104) le dispositif est en général déconnecté. Il se connecte au réseau de communication dès que la méthode de connexion est appelée, (105...199) réservé, (200...255) modes spécifiques au fabricant.
repetitions	Nombre maximal de nouvelles tentatives, en cas de tentatives de connexion non réussies.
repetition_delay	Délai, en secondes, à observer avant qu'une tentative de connexion infructueuse puisse être répétée. repetition_delay = 0 signifie que le retard n'est pas spécifié.

calling_window	Contient les points dans le temps auxquels la fenêtre devient active (start_time) et inactive (end_time). Le start_time définit implicitement la période. EXEMPLE Lorsque le jour du mois n'est pas spécifié (égal à 0xFF), cela signifie que la fenêtre d'appel est gérée quotidiennement. Une gestion de fenêtre quotidienne, mensuelle ...peut être définie. array window_element window_element ::= structure <pre>{ start_time: octet-string, end_time: octet-string }</pre> start_time et end_time sont formatés comme spécifié en 4.6.1 pour <i>date-time</i>.
destination_list	Contient la liste de destinations (par exemple numéros de téléphone, adresses email ou leurs combinaisons) vers lesquelles le(s) message(s) est/sont à envoyer dans certaines conditions. Les conditions et leur lien aux éléments du tableau ne sont pas définis ici. array destination destination ::= octet-string

Description de la méthode

connect (data)	Lance le processus de connexion au réseau de communication conformément aux règles définies par l'intermédiaire de l'attribut de mode. data ::= integer (0)

5.5.7 GPRS modem setup (class_id = 45, version = 0)

Cette IC permet d'installer des modems GPRS en gérant toutes les données nécessaires à la gestion du modem.

GPRS modem setup (établissement de modem GPRS)		0...n				class_id = 45, version = 0			
Attributs		Type de données		Min.	Max.	Def.	Nom court		
1.	logical_name (static)	octet-string					x		
2.	APN (static)	octet-string					x + 0x08		
3.	PIN_code (static)	long-unsigned					x + 0x10		
4.	quality_of_service (static)	structure					x + 0x18		
Méthodes spécifiques		m/o							

Description d'attribut	
logical_name	Identifie l'instance de l'objet "GPRS modem setup" (Établissement de modem GPRS). Voir 6.2.21.
APN	Définit le nom de point d'accès du réseau.
PIN_code	Contient le numéro d'identification personnel.
quality_of_service	Spécifie les paramètres de qualité de service. Il s'agit d'une structure de 2 éléments: – le premier élément définit les caractéristiques par défaut ou minimales du réseau concerné. Ces paramètres sont à régler à la valeur de meilleur effort; – le second élément définit les paramètres demandés. <pre> quality_of_service ::= structure { default: qos_element, requested: qos_element } qos_element ::= structure { precedence: unsigned, delay: unsigned, reliability: unsigned, peak throughput: unsigned, mean throughput: unsigned } </pre>

5.5.8 GSM diagnostic (class_id = 47, version = 0)

Le réseau GSM/GPRS fait l'objet de modifications constantes en termes d'état d'enregistrement, de qualité du signal, etc. Il est nécessaire de surveiller et de consigner les paramètres pertinents afin d'obtenir des informations de diagnostic permettant d'identifier des problèmes de communication dans le réseau.

Une instance de la classe "GSM diagnostic" enregistre les paramètres du réseau GSM/GPRS nécessaires à l'analyse du fonctionnement du réseau.

Un objet "Profile generic" de diagnostic GSM permet également de saisir les attributs de l'objet de diagnostic GSM. Voir 6.5 de l'IEC 62056-6-1:.

GSM diagnostic (diagnostic GSM)		0...n				class_id = 47, version = 0			
Attributs		Type de données		Min.	Max.	Def.	Nom court		
1.	logical_name (static)	octet-string					x		
2.	operator (dyn.)	visible-string					x + 0x08		
3.	status (dyn.)	enum		0	255	0	x + 0x10		
4.	cs_attachment (dyn.)	enum		0	255	0	x + 0x18		
5.	ps_status (dyn.)	enum		0	255	0	x + 0x20		
6.	cell_info (dyn.)	cell_info_type					x + 0x30		
7.	adjacent_cells (dyn.)	array					x + 0x38		
8.	capture_time (dyn.)	date-time					x + 0x40		
Méthodes spécifiques		m/o							

Description d'attribut	
logical_name	Identifie l'instance de l'objet "GSM Diagnostic". Pour les noms logiques, voir 6.2.21.
operator	Contient le nom de l'opérateur réseau ("YourNetOp", par exemple)
status	Indique l'état d'enregistrement du modem. enum: (0) non enregistré, (1) enregistré, réseau domestique (2) non enregistré, mais MT recherche actuellement un nouvel opérateur auprès duquel s'enregistrer, (3) enregistrement refusé, (4) inconnu, (5) enregistré, itinérance (6) ... (255) réservé
cs_attachment	Indique l'état commuté du circuit courant. enum: (0) inactif, (1) appel entrant, (2) actif, (3) ... (255) réservé
ps_status	Le champ de valeur <i>ps_status</i> indique l'état commuté par paquets du modem. enum: (0) inactif, (1) GPRS, (2) EDGE, (3) UMTS, (4) HSDPA, (5) ... (255) réservé

cell_info	Représente les informations de cellule: <pre>cell_info_type ::= structure { cell_ID: long-unsigned, location_ID: long-unsigned, signal_quality: unsigned, ber: unsigned }</pre> où: - cell_ID: ID de cellule à deux octets au format hexadécimal; - location_ID: Code à deux octets de la zone de position (LAC) au format hexadécimal ("00C3" équivaut à 195 en notation décimale, par exemple); - signal_quality: Représente la qualité du signal: (0) –113 dBm maximum, (1) –111 dBm, (2...30) –109...-53 dBm, (31) –51 au moins, (99) inconnu ou indétectable; - ber: Mesure TEB (taux d'erreurs sur les bits) en pourcentage: (0...7) comme les valeurs RXQUAL_n spécifiées dans l'ETSI GSM 05.08 8.2.4. (99) inconnu ou indétectable.
adjacent_cells	array adjacent_cell_info <pre>adjacent_cell_info ::= structure { cell_ID: long-unsigned, signal_quality: unsigned }</pre> où: - cell_ID: ID de cellule à deux octets au format hexadécimal; - signal_quality: Représente la qualité du signal: (0) –113 dBm maximum, (1) –111 dBm, (2...30) –109...-53 dBm, (31) –51 au moins, (99) inconnu ou indétectable.
capture_time	Contient la date et l'heure de la dernière saisie des données. <i>date-time</i> formaté comme spécifié en 4.6.1.

5.6 Classes d'interfaces pour l'établissement d'échange de données via le M-Bus

5.6.1 M-Bus slave port setup (class_id = 25, version = 0)

NOTE Le nom de cette IC a été changé de "M-BUS port setup" en "M-Bus slave port setup" afin d'indiquer qu'elle sert à établir l'échange de données lorsqu'un serveur COSEM communique avec un client COSEM en utilisant le M-Bus câblé.

Cette IC permet de modéliser et configurer des voies de communication selon l'EN 13757-2:2004. Plusieurs voies de communication peuvent être configurées.

M-Bus slave port setup (établissement de port de M-Bus esclave)		0...n		class_id = 25, version = 0		
Attributs		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. default_baud	(static)	enum	0	5	0	x + 0x08
3. avail_baud	(static)	enum	0	7		x + 0x10
4. addr_state	(static)	enum				x + 0x18
5. bus_address	(static)	unsigned				x + 0x20
Méthodes spécifiques		m/o				

Description d'attribut

logical_name Identifie l'instance d'objet "M-Bus slave port setup" (établissement de port de M-Bus esclave). Voir 6.2.20.

default_baud Définit le débit en bauds pour la séquence d'ouverture.

enum: (0) 300 baud,
(3) 2 400 baud,
(5) 9 600 baud

avail_baud Définit les débits en bauds qui peuvent être négociés après démarrage.

enum: (0) 300 baud,
(1) 600 baud,
(2) 1 200 baud,
(3) 2 400 baud,
(4) 4 800 baud,
(5) 9 600 baud,
(6) 19 200 baud,
(7) 38 400 baud

addr_state Définit si, oui ou non, le dispositif a eu une adresse assignée depuis la dernière mise sous tension du dispositif.

enum:
(0) N'a pas encore reçu d'adresse attribuée,
(1) a une adresse attribuée, soit par réglage manuel, soit par une méthode automatisée.

bus_address L'adresse actuellement attribuée sur le bus pour le dispositif.

NOTE Si aucune adresse de bus n'est attribuée, la valeur est 0.

5.6.2 M-Bus client (class_id = 72, version = 1)

Les instances de "M-Bus client" permettent d'installer des dispositifs de M-Bus esclaves en utilisant le M-Bus câblé et d'échanger des données avec eux. Chaque objet "M-Bus client" commande un dispositif M-Bus esclave. Pour les détails relatifs à la couche d'application M-Bus dédiée voir l'EN 13757-3:2013.

NOTE La version 1 de l'IC "M-Bus client" est conforme à l'EN 13757-3:2013.

Le dispositif client M-Bus peut avoir une ou plusieurs interfaces physiques M-Bus, qui peuvent être configurées à l'aide d'instances de l'IC "M-Bus master port setup" (voir 5.6.4).

Un dispositif de M-Bus esclave est identifié avec son adresse primaire Primary Address, son numéro d'identification Identification Number, son identifiant de fabricant Manufacturer ID, etc., comme défini à l'Article 5 de l'EN 13757-3:2013, Variable Data Send (Envoi et Réponse de données variables). Ces paramètres sont contenus dans les attributs respectifs de l'IC M-Bus client.

Les valeurs à saisir à partir d'un dispositif M-Bus esclave sont identifiées par l'attribut *capture_definition*, contenant une liste d'identifiants de données (DIB, VIB) pour le dispositif M-Bus esclave. Les données sont saisies périodiquement ou sur un déclencheur approprié. Chaque élément de donnée est stocké dans un objet "M-Bus value" de l'IC "Extended register". Les objets "M-Bus value" peuvent être saisis dans les objets génériques M-Bus "Profile generic", éventuellement avec d'autres objets, non spécifiques à M-Bus.

En utilisant les méthodes des objets "M-Bus client", les dispositifs M-Bus esclaves peuvent être installés et désinstallés.

Il est également possible d'envoyer des données à des dispositifs M-Bus esclaves et d'exécuter des opérations telles que la réinitialisation d'alarmes, la synchronisation d'horloge, le transfert d'une clé de chiffrement, etc.

Le champ de configuration défini en 5.12 de l'EN 13757-3:2013 donne des informations relatives au mode de chiffrement et au nombre d'octets chiffrés.

L'état de la clé de chiffrement indique si la clé a été définie, transférée vers le dispositif M-Bus esclave et utilisée avec ce dernier.

M-Bus client (client M-Bus)		0...n	class_id = 72, version = 1			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	mbus_port_reference (static)	octet-string				x + 0x08
3.	capture_definition (static)	array			vide	x + 0x10
4.	capture_period (static)	double-long-unsigned			0	x + 0x18
5.	primary_address	unsigned			0	x + 0x20
6.	identification_number (dyn.)	double-long-unsigned			0	x + 0x28
7.	manufacturer_id (dyn.)	long-unsigned			0	x + 0x30
8.	version (dyn.)	unsigned			0	x + 0x38
9.	device_type (dyn.)	unsigned			0	x + 0x40
10.	access_number (dyn.)	unsigned			0	x + 0x48
11.	status (dyn.)	unsigned			0	x + 0x50
12.	alarm (dyn.)	unsigned			0	x + 0x58
13.	configuration (dyn.)	long-unsigned			0	x + 0x60
14.	encryption_key_status (dyn.)	enum			0	x + 0x68
Méthodes spécifiques		m/o				
1.	slave_install (data)	o				x + 0x70
2.	slave_deinstall (data)	o				x + 0x78
3.	capture (data)	o				x + 0x80
4.	reset_alarm (data)	o				x + 0x88
5.	synchronize_clock (data)	o				x + 0x90
6.	data_send (data)	o				x + 0x98
7.	set_encryption_key (data)	o				x + 0xA0
8.	transfer_key (data)	o				x + 0xA8

Description d'attribut	
logical_name	Identifie l'instance de l'objet "M-Bus client" (client M-Bus). Voir 6.2.20.
mbus_port_reference	Fournit une référence à un objet "M-Bus master port setup", utilisé pour configurer un port M-Bus, chaque interface permettant d'échanger des données avec un ou plusieurs dispositifs M-Bus esclaves.
capture_definition	Fournit la <i>capture_definition</i> pour les dispositifs M-Bus esclaves. NOTE 1 Cet attribut peut être préalablement configuré ou écrit dans le cadre de la procédure d'installation. array capture_definition_element capture_definition_element ::= structure <pre> { data_information_block: octet-string, value_information_block: octet-string } </pre> NOTE 2 Les éléments <i>data_information_block</i> et <i>value_information_block</i> correspondent respectivement à Data Information Block (DIB) et à Value Information Block (VIB) décrits dans l'EN 13757-3:2013, 6.2 et Article 7.
capture_period	>= 1: Saisie automatique supposée. Spécifie la période de saisie en secondes. 0: Absence de saisie automatique; la saisie est déclenchée extérieurement ou des événements de saisie se produisent de manière asynchrone.
primary_address	Contient l'adresse primaire du dispositif M-Bus esclave. La plage est 0...250. Chaque dispositif M-bus est lié à une voie du M-Bus maître. Toutefois, il n'existe aucun lien direct entre l'adresse primaire et le numéro de voie. NOTE 3 La spécification du champ B des codes OBIS limite la plage de 1...64 à l'intérieur d'un dispositif logique. Voir 5.2 de l'IEC 62056-6-1:. Si le dispositif esclave est déjà configuré et donc que son adresse primaire est différente de 0, cela doit être écrit dans l'attribut <i>primary_address</i>. À partir de cet instant, l'échange de données avec le dispositif de M-Bus esclave est possible. Autrement, la méthode <i>slave_install</i> doit être utilisée (voir ci-dessous). NOTE 4 L'attribut <i>primary_address</i> ne peut pas être utilisé pour stocker une adresse primaire souhaitée pour un dispositif esclave non configuré. Si l'attribut <i>primary_address</i> est défini, cela signifie que le client M-Bus peut opérer immédiatement avec cette adresse primaire, ce qui n'est pas le cas avec un dispositif esclave non configuré.
identification_number	Contient l'élément "Identification Number" (numéro d'identification) de l'en-tête de donnée tel que spécifié au 5.5 de l'EN 13757-3:2013. Cet attribut, ainsi que les attributs 7, 8 et 9, contiennent les valeurs trouvées dans le premier message reçu après l'installation. Si ces valeurs sont différentes dans les messages subséquents, le message est ignoré.
manufacturer_id	Contient l'élément "Manufacturer Identification" (identification du fabricant) de l'en-tête de donnée tel que spécifié au 5.6 de l'EN 13757-3:2013.
version	Contient l'élément "Version" de l'en-tête de donnée tel que spécifié au 5.7 de l'EN 13757-3:2013.

device_type	Contient l'élément "Device type identification " (identification du type de dispositif) de l'en-tête de donnée tel que spécifié au 5.8 de l'EN 13757-3:2013, Tableau 6.
access_number	Contient l'élément "Access Number " (numéro d'accès) de l'en-tête de donnée tel que spécifié au 5.9 de l'EN 13757-3:2013.
status	Contient l'élément "Status byte" (octet d'état) de l'en-tête de donnée tel que spécifié au 5.10 de l'EN 13757-3:2013, Tableau 7 et Tableau 8. Il est mis à jour à chaque lecture du dispositif M-Bus esclave.
alarm	Contient l'état Alarm (Alarme) spécifié à l'Annexe D de l'EN 13757-3:2013. Il est mis à jour à chaque lecture du dispositif M-Bus esclave.
configuration	Contient le champ Configuration (précédemment: champ Signature) tel que spécifié au 5.12 de l'EN 13757-3:2013. Il contient des informations relatives au mode de chiffrement et au nombre d'octets chiffrés. Il est mis à jour à chaque lecture du dispositif M-Bus esclave.
encryption_key_status	Donne des informations relatives à l'état de la clé de chiffrement. Voir également l'Annex B. enum: (0) pas de clé de chiffrement, (1) encryption_key défini, (2) encryption_key transféré, (3) encryption_key défini et transféré, (4) encryption_key en cours d'utilisation Voir également l'Annex B.

Description de la méthode

slave_install (data)	Installe un dispositif esclave, qui est encore non configuré (son adresse primaire est 0). data := unsigned Cette méthode ne peut être appelée avec succès que si la valeur en cours de l'attribut <i>primary_address</i> est 0. Les actions suivantes sont entreprises: – L'adresse M-Bus 0 est vérifiée pour détecter la présence d'un nouveau dispositif. – Si aucun M-Bus esclave non installé n'est trouvé, l'appel de la méthode échoue; – Si la méthode <i>slave_install</i> est appelée avec "0" comme paramètre, l'adresse primaire est automatiquement attribuée. Pour ce faire, l'attribut <i>primary_address</i> de tous les objets "M-Bus client" est vérifié dans le dispositif DLMS/COSEM et en sélectionnant ensuite le premier numéro non utilisé. L'attribut <i>primary_address</i> est mis à cette adresse et est ensuite transféré au dispositif M-Bus esclave; – Si la méthode <i>slave_install</i> est appelée avec une adresse primaire (différente de 0) comme paramètre, l'attribut <i>primary_address</i> est défini sur cette valeur, puis transféré au dispositif M-Bus esclave.

NOTE 5 Les dispositifs esclaves non configurés sont configurés avec l'adresse primaire telle que spécifiée dans l'EN 13757-3:2013, Annexe E.5.

slave_deinstall (data)	Désinstalle le dispositif esclave. Le but principal de ce service est de désinstaller le dispositif M-Bus esclave et de préparer le maître pour l'installation d'un nouveau dispositif. Les actions suivantes sont entreprises: – l'adresse M-Bus est mise à 0 dans le dispositif M-Bus esclave; – la clé de chiffrement transférée précédemment au dispositif M-Bus esclave est détruite; la clé par défaut n'est pas altérée; – <i>encryption_key_status</i> est défini sur (0): pas d'<i>encryption_key</i>; – l'attribut <i>primary_address</i> est également mis à 0. NOTE 6 Un nouvel esclave M-Bus peut être installé seulement une fois que la valeur de l'attribut <i>primary_address</i> est 0. data::= unsigned (0)
capture (data)	Capture des valeurs (telles que spécifiées par l'attribut <i>capture_definition</i>) provenant du dispositif M-Bus esclave. data::= integer (0)
reset_alarm (data)	Réinitialise l'état d'alarme du dispositif M-Bus esclave. data::= integer (0)
Synchronize_clock (data)	Synchronise l'horloge du dispositif M-Bus esclave avec celle du dispositif de M-Bus client. data::= integer (0)
data_send (data)	Envoie des données au dispositif M-Bus esclave. data::= array data_definition_element data_definition_element::= structure { data_information_block: octet-string, value_information_block: octet-string, data: CHOICE }
data_send (data) (suite)	– types de données simples null-data [0], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], utf8-string [12], bcd [13], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], float32 [23], float64 [24] }

set_encryption Définit la clé de chiffrement dans le client de M-Bus et assure la communication chiffrée avec le dispositif M-Bus esclave.

key (data)

data::= octet-string (encryption_key)

Après l'installation de l'esclave M-Bus, le client de M-Bus détient une clé de chiffrement vide. Ainsi, le chiffrement des trames M-Bus est désactivé.

Le chiffrement peut être désactivé en appelant la méthode *set_encryption_key* avec un octet_string de longueur nulle.

La modification de la clé de chiffrement exige deux étapes:

- en premier lieu, la clé est envoyée à l'esclave M-Bus, chiffrée avec la clé par défaut, en utilisant la méthode *transfer_key*;
- en second lieu, la clé est définie dans le M-Bus maître en utilisant la méthode *set_encryption_key*.

transfer_key Transfère une clé de chiffrement au dispositif M-Bus esclave.

(data)

data::= octet-string (encrypted_key)

Avant de pouvoir utiliser des trames M-Bus chiffrées, une clé de chiffrement opérationnelle est à envoyer à l'esclave M-Bus, par appel de la méthode *transfer_key*. Le paramètre d'appel de la méthode est la clé de chiffrement opérationnelle chiffrée avec la clé par défaut du dispositif de M-Bus esclave. La trame de M-Bus envoyée n'est pas chiffrée. Après l'exécution, le chiffrement est activé et tous les télégrammes ultérieurs sont chiffrés.

Chaque dispositif M-Bus esclave doit être délivré avec une clé de chiffrement par défaut.

Une nouvelle clé de chiffrement peut être définie dans le client de M-Bus en appelant la méthode *set_encryption_key* avec la nouvelle clé de chiffrement comme paramètre.

Avec des appels ultérieurs de la méthode *transfer_key*, de nouvelles clés de chiffrement peuvent être envoyées au M-Bus esclave. Le paramètre d'appel de la méthode est la nouvelle clé de chiffrement chiffrée avec la clé par défaut. La trame de M-Bus est chiffrée.

Lorsqu'un M-Bus esclave est désinstallé, la clé de chiffrement est détruite, mais la clé par défaut n'est pas altérée. Le chiffrement reste désactivé jusqu'à ce qu'une nouvelle clé de chiffrement soit transférée.

5.6.3 Wireless Mode Q channel (class_id = 73, version = 1)

Les instances de cette IC définissent les paramètres opérationnels pour la communication en utilisant les interfaces de mode Q. Voir également l'EN 13757-5:2015.

Wireless Mode Q channel (canal en mode Q sans fil)	0...n	class_id = 73, version = 1			
Attributs	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. addr_state (static)	enum				x + 0x08
3. device_address (static)	octet-string				x + 0x10
4. address_mask (static)	octet-string				x + 0x18
Méthodes spécifiques	m/o				

Description d'attribut

logical_name	Identifie l'instance d'objet "Wireless Mode Q channel" (canal en mode Q sans fil). Voir 6.2.20.
addr_state	Définit si, oui ou non, le dispositif a eu une adresse assignée depuis la dernière mise sous tension du dispositif. enum: (0) n'a pas encore reçu d'adresse attribuée, (1) a une adresse attribuée, soit par réglage manuel, soit par une méthode automatisée
device_address	L'adresse actuellement attribuée du dispositif sur le réseau
address_mask	L'adresse de groupe à laquelle le dispositif répondra lorsque l'adressage de forme courte est utilisé

5.6.4 M-Bus master port setup (class_id = 74, version = 0)

Les instances de cette IC définissent les paramètres opérationnels pour la communication en utilisant les interfaces de l'EN 13757-2 si le dispositif agit comme un M-Bus maître.

M-Bus master port setup (établissement de port de M-Bus maître)		0...n	class_id = 74, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. comm_speed	(static)	enum	0	7	3	x + 0x08
Méthodes spécifiques		m/o				

Description d'attribut

logical_name	Identifie l'instance d'objet "M-Bus master port setup" (établissement de port M-Bus maître). Voir 6.2.20
comm_speed	La vitesse de communication prise en charge par le port enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud

5.7 Classes d'interfaces pour l'établissement d'échange de données sur Internet**5.7.1 TCP-UDP setup (class_id = 41, version = 0)**

Cette IC permet de modéliser l'établissement de la sous-couche TCP ou UDP de la couche transport COSEM basée sur TCP ou UDP d'un profil de communication basé sur TCP-UDP/IP.

Dans les profils de communication basés sur TCP-UDP/IP, toutes les AA entre un dispositif physique hébergeant un ou plusieurs processus d'application client COSEM et un dispositif physique hébergeant un ou plusieurs AP serveur COSEM s'appuient sur une seule connexion TCP ou UDP. L'entité TCP ou UDP est enveloppée dans la couche transport COSEM basée sur TCP-UDP. Au sein d'un dispositif physique, chaque Processus d'Application – AP client

ou dispositif logique serveur – est lié à un Wrapper Port (WPort), port conteneur. La liaison se fait avec l'aide de l'objet "SAP Assignment". Voir 5.3.5.

D'autre part, une couche transport COSEM basée sur TCP ou sur UDP peut être capable de prendre en charge plus d'une connexion TCP ou UDP, entre un dispositif physique et plusieurs dispositifs physiques homologues hébergeant des AP COSEM.

Lorsqu'un dispositif physique COSEM prend en charge diverses couches liaisons de données – par exemple Ethernet et PPP – une instance de l'objet "TCP-UDP setup" est nécessaire pour chacune d'elles.

TCP-UDP setup (établissement de TCP-UDP)		0...n	class_id = 41, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	TCP-UDP_port (static)	long-unsigned				x + 0x08
3.	IP_reference (static)	octet-string				x + 0x10
4.	MSS (static)	long-unsigned	40	65 535	576	x + 0x18
5.	nb_of_sim_conn (static)	unsigned	1			x + 0x20
6.	inactivity_time_out (static)	long-unsigned			180	x + 0x28
Méthodes spécifiques		m/o				

Description d'attribut

logical_name Identifie l'instance de l'objet "TCP-UDP setup" (établissement de TCP-UDP). Voir 6.2.21.

TCP-UDP_port Détient le numéro de port TCP-UDP sur lequel le dispositif physique est à l'écoute pour détecter l'application DLMS/COSEM.
Pour DLMS/COSEM, les numéros de port suivants ont été enregistrés par l'IANA. Voir <http://www.iana.org/assignments/port-numbers>

dlms/cosem	4059/TCP	DLMS/COSEM
dlms/cosem	4059/UDP	DLMS/COSEM

IP_reference Référence un objet "IP setup" (établissement d'IP) par son nom logique. L'objet référencé contient des informations relatives aux réglages de l'adresse IP de la couche IP prenant en charge la couche TCP-UDP.

MSS Avec l'aide de l'option Taille de segment maximale (MSS – Maximum Segment Size) une couche TCP peut indiquer la taille maximale de segment de réception à son partenaire. À noter que:

- cette option doit seulement être envoyée dans la demande de connexion initiale (c'est-à-dire dans des segments avec le bit de commande SYN envoyé);
- Si cette option est absente, la MSS est conventionnellement considérée comme sa valeur par défaut, 576;
- La MSS n'est pas négociable; sa valeur est indiquée par son attribut.

nb_of_sim_conn Le nombre maximal de connexions simultanées que la couche transport COSEM basée sur TCP-UDP est capable de prendre en charge.

inactivity_time_out Définit le temps, en secondes, sur lequel, si aucune trame n'est reçue en provenance du client COSEM, la connexion TCP inactive doit être abandonnée.

Lorsque cette valeur est mise à 0, cela signifie que l'*inactivity_time_out* n'est pas opérationnel. En d'autres termes, une connexion TCP, une fois établie, dans des conditions normales – pas de coupure secteur, etc. – ne sera jamais abandonnée par le serveur COSEM.

À noter que toutes les actions liées à la gestion de la fonction de dépassement de délai d'inactivité – mesure du temps d'inactivité, abandon de la connexion TCP si la temporisation est terminée, etc. – sont gérées dans la mise en œuvre de la couche TCP-UDP.

5.7.2 IPv4 setup (class_id = 42, version = 0)

NOTE 1 Comparée aux éditions antérieures de cette norme, la présente spécification apporte des améliorations dans la présentation des attributs. Ne s'agissant pas de modifications techniques, la version de l'IC reste 0.

Cette IC permet de modéliser l'établissement de la couche IPv4, de gérer toutes les informations liées aux réglages d'adresses IP associés à un dispositif donné et à une connexion de couche inférieure sur laquelle ces réglages sont utilisés.

Il doit y avoir une instance de cette IC dans un dispositif pour chaque interface réseau différente mise en œuvre. Par exemple, si un dispositif a deux interfaces (utilisant le profil TCP-UDP/IPv4 sur les deux), il doit y avoir deux instances de l'IC "IPv4 setup" dans le dispositif en question: une pour chacune de ces interfaces.

IPv4 setup (établissement de IPv4)		0...n	class_id = 42, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	DL_reference (static)	octet-string				x + 0x08
3.	IP_address	double-long-unsigned				x + 0x10
4.	multicast_IP_address	array				x + 0x18
5.	IP_options	array				x + 0x20
6.	subnet_mask	double-long-unsigned				x + 0x28
7.	gateway_IP_address	double-long-unsigned				x + 0x30
8.	use_DHCP_flag (static)	boolean				x + 0x38
9.	primary_DNS_address	double-long-unsigned				x + 0x40
10.	secondary_DNS_address	double-long-unsigned				x + 0x48
Méthodes spécifiques		m/o				
1.	add_mc_IP_address (data)	o				x + 0x60
2.	delete_mc_IP_address (data)	o				x + 0x68
3.	get_nbof_mc_IP_addresses (data)	o				x + 0x70

Description d'attribut

logical_name Identifie l'instance de l'objet "IPv4 setup" (Établissement de IPv4). Voir 6.2.21.

DL_reference	Référence un objet d'établissement de couche liaison de données (par exemple Ethernet ou PPP) par son nom logique. L'objet référencé contient des informations relatives aux réglages spécifiques de la couche liaison de données prenant en charge la couche IP.
IP_address	Contient la valeur de l'adresse IP (IPv4) de ce dispositif physique sur le réseau auquel le dispositif est connecté via l'interface de couche inférieure référencée. Il peut être (statique) ou (dynamique). Dans le second cas, l'affectation d'adresse IP dynamique (par exemple DHCP) est utilisée. Si aucune adresse IP n'est assignée, la valeur est 0. EXEMPLE L'adresse IPv4 192.168.0.1 (en notation décimale à point) correspond à C0A80001 (hexa), ce qui donne 3232235521 (double-long-unsigned).
multicast_IP_address	Contient un tableau d'adresses IP. Les adresses IP dans ce tableau doivent s'inscrire dans la plage d'adresses de groupe de multidiffusion (adresses de "Class D", y compris les adresses IP dans la plage de 224.0.0.0 à 239.255.255.255). Lorsqu'un dispositif reçoit un datagramme IP avec l'une de ces adresses IP dans le champ adresse IP de destination, il doit considérer que ce datagramme lui est adressé. multicast_IP_address::= array double-long-unsigned

IP_options Contient les paramètres nécessaires à la prise en charge des options IP sélectionnées – par exemple horodatage de datagrammes ou services de sécurité (IPSec).

IP_options ::= array IP_options_element

IP_options_element ::= structure

```
{
    IP_Option_Type:      unsigned,
    IP_Option_Length:    unsigned,
    IP_Option_Data:      octet-string
}
```

NOTE 2 Dans tous les cas, comme cela est spécifié dans le RFC 791, le champ IP_Option_Length inclut la longueur totale des trois champs: IP_Option_Type, IP_Option_Length et IP_Option_Data.

IP_Option_Types admis:

- Security: IP_Option_Type = 0x82, IP_Option_Length = 11

Si cette option est présente, le dispositif doit être autorisé à envoyer des paramètres de sécurité, de compartimentage, de restrictions de gestion et de TCC (groupe d'utilisateurs fermé) dans ses datagrammes IP. IP_Option_Data doit contenir la valeur des données de sécurité cloisonnement, gestion des restrictions et le code de pays de la transmission tels que spécifiés dans le RFC 791.

- Loose Source et Record Route: IP_Option_Type = 0x83

Si cette option est présente, le dispositif doit fournir des informations de routage devant être utilisées par les passerelles pour transmettre le datagramme à la destination et pour enregistrer les informations de chemin. Les valeurs d'IP_Option_Length et d'IP_Option_Data sont spécifiées dans le RFC 791.

- Strict Source et Record Route: IP_Option_Type = 0x89

Si cette option est présente, le dispositif doit fournir des informations de routage devant être utilisées par les passerelles pour transmettre le datagramme à la destination et pour enregistrer les informations de chemin. Les valeurs d'IP_Option_Length et d'IP_Option_Data sont spécifiées dans le RFC 791.

- Record Route: IP_Option_Type = 0x07

Si cette option est présente, le dispositif doit également:

- envoyer des datagrammes IP émis avec cette option, fournissant des moyens d'enregistrer le chemin de ces datagrammes;
- comme routeur, envoyer des datagrammes IP acheminés avec l'option de chemin réglée en fonction de cette option.

Les valeurs d'IP_Option_Length et d'IP_Option_Data sont spécifiées dans le RFC 791.

- Internet Timestamp: IP_Option_Type = 0x44

Si cette option est présente, le dispositif doit également:

- envoyer des datagrammes IP émis avec cette option, fournissant des moyens pour horodater le datagramme dans le chemin vers sa destination;
- comme routeur, envoyer des datagrammes IP acheminés avec l'option horodatage réglée en fonction de cette option.

Les valeurs d'IP_Option_Length et d'IP_Option_Data sont spécifiées dans le RFC 791.

subnet_mask	Contient le masque de sous-réseau. Lorsque le sous-réseautage est utilisé dans un segment de réseau, chaque dispositif concerné doit se comporter conformément aux règles de sous-réseautage. Pour ce faire, le dispositif, outre son adresse IP, a besoin de connaître aussi comment l'adresse IP est structurée au sein du segment mis en sous-réseau. L'attribut <i>subnet_mask</i> contient cette information. Avec IPv4, le <i>subnet_mask</i> est un mot de 32 bits, exprimé exactement dans le même format qu'une adresse IP (par exemple 255.255.255.0), mais a une autre signification: les bits "0" du <i>subnet_mask</i> indiquent la partie de l'adresse IP qui est encore utilisée comme Device_ID sur un réseau IP mis en sous-réseau ⁴.
gateway_IP_address	Contient l'adresse IP du dispositif passerelle. Dans la plupart des mises en œuvre IP, il existe un code dans le module qui gère les datagrammes sortants pour décider si un datagramme peut être envoyé directement à la destination sur le réseau local ou s'il doit être envoyé vers une passerelle. Afin de pouvoir envoyer des datagrammes non locaux vers la passerelle, le dispositif doit connaître l'adresse IP du dispositif passerelle assignée au segment de réseau donné. Si aucune adresse IP n'est assignée, la valeur est 0.
use_DHCP_flag	Si ce fanion est mis à TRUE, le dispositif utilise le protocole DHCP (Dynamic Host Configuration Protocol – protocole de configuration de serveur dynamique) pour déterminer dynamiquement les paramètres <i>IP_address</i>, <i>subnet_mask</i> et <i>gateway_IP_address</i>. D'autre part, si le fanion est mis à FALSE, les paramètres <i>IP_address</i>, <i>subnet_mask</i> et <i>gateway_IP_address</i> doivent être définis localement.
primary_DNS_address	L'adresse IP du Serveur de noms de domaine (DNS) primaire. Si aucune adresse IP n'est assignée, la valeur est 0.
secondary_DNS_address	L'adresse IP du Serveur de noms de domaine (DNS) secondaire. Si aucune adresse IP n'est assignée, la valeur est 0.
Description de la méthode	
add_mc_IP_address (IP_Address)	Ajoute une adresse IP de multidiffusion au tableau <i>multicast_IP_address</i>. IP_Address ::= double-long-unsigned
delete_mc_IP_address (IP_Address)	Supprime une adresse IP du tableau <i>multicast_IP_Address</i>. L'adresse IP à effacer est identifiée par sa valeur. IP_Address ::= double-long-unsigned
get_nbof_mc_IP_addresses (data)	Retourne le nombre d'adresses IP contenues dans le tableau <i>multicast_IP_Address</i>. data ::= unsigned

5.7.3 IPv6 setup (class_id = 48, version = 0)

NOTE 1 Voir également l'Annex C.

⁴ Voir plus de détails concernant le sous-réseautage dans les documents RFC 940 et RFC 950.

L'IC IPv6 setup permet de modéliser l'établissement de la couche IPv6, de gérer toutes les informations liées aux réglages d'adresse IPv6 associés à un dispositif donné et à une connexion de couche inférieure sur laquelle ces réglages sont utilisés.

Il doit y avoir une instance de cette IC dans un dispositif pour chaque interface réseau différente mise en œuvre. Par exemple, si un dispositif a deux interfaces (utilisant le profil UDP/IP et/ou TCP/IP sur les deux), il doit y avoir deux instances de l'IC "IPv6 setup" dans le dispositif en question: une pour chacune de ces interfaces.

IPv6 setup (établissement de IPv6)		0...n	class_id = 48, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	DL_reference (static)	octet-string				x + 0x08
3.	address_config_mode (static)	enum			0	x + 0x10
4.	unicast_IPv6_addresses	array				x + 0x18
5.	multicast_IPv6_addresses (static)	array				x + 0x20
6.	gateway_IPv6_addresses (static)	array			0	x + 0x28
7.	primary_DNS_address (static)	octet-string			0	x + 0x30
8.	secondary_DNS_address (static)	octet-string			0	x + 0x38
9.	traffic_class (static)	unsigned	0	63	0	x + 0x40
10.	neighbor_discovery_setup (static)	array				x + 0x48
Méthodes spécifiques		m/o				
1.	add_IPv6_address (data)	o				x + 0x60
2.	remove_IPv6_address (data)	o				x + 0x68

Description d'attribut

logical_name Identifie l'instance de l'objet "IPv6 setup" (établissement de IPv6). Voir 6.2.21.

DL_reference Référence un objet "Data link layer setup" par son nom logique. L'objet référencé contient des informations relatives aux réglages spécifiques de la couche liaison de données prenant en charge la couche IPv6.

address_config_mode Définit le mode de configuration de l'adresse IPv6
enum:
(0) Auto-configuration (par défaut),
(1) DHCPv6,
(2) Manuel,
(3) ND (Neighbour Discovery)

NOTE 2 L'*address_config_mode* est commun à toutes les adresses IPv6 gérées par une instance de la classe "IPv6 setup".

unicast_IPv6_addresses Contient les adresses IPv6 unicast attribuées à l'interface concernée du dispositif physique sur le réseau (Unique Local Unicast, Link Local Unicast et/ou Global Unicast). Une adresse IPv6 peut être (statique) et/ou (dynamique).

unicast_IPv6_addresses ::= array octet-string

Le format de chaque adresse IPv6 unicast doit être tel que spécifié dans le RFC 3513.

Si aucune adresse IPv6 unicast n'est attribuée à l'interface, un tableau ne contenant aucun élément est présent (par défaut).

Pour réinitialiser toutes les adresses IPv6 unicast configurées, un tableau ne contenant aucun élément doit être écrit.

multicast_IPv6_addresses	Contient un tableau d'adresses IPv6 utilisées pour la diffusion multicast. multicast_IPv6_addresses::= array octet-string Le format de chaque adresse IPv6 multicast doit être tel que spécifié dans le RFC 3513. Si aucune adresse IPv6 multicast n'est attribuée à l'interface, un tableau ne contenant aucun élément est présent (par défaut). Pour réinitialiser toutes les adresses IPv6 multicast configurées, un tableau ne contenant aucun élément doit être écrit.
gateway_IPv6_addresses	Contient les adresses IPv6 du dispositif de passerelle IPv6. gateway_IPv6_addresses::= array octet-string Le format de chaque adresse IPv6 de passerelle doit être tel que spécifié dans le RFC 3513. Si aucune adresse IPv6 de passerelle n'est attribuée à l'interface, un tableau ne contenant aucun élément est présent (par défaut). Pour réinitialiser toutes les adresses IPv6 de passerelle configurées, un tableau ne contenant aucun élément doit être écrit.
primary_DNS_address	Contient l'adresse IPv6 du Serveur de noms de domaine (DNS) primaire. Si aucune adresse IPv6 n'est attribuée, la longueur de l'octet-string doit être 0.
secondary_DNS_address	Contient l'adresse IPv6 du Serveur de noms de domaine (DNS) secondaire. Si aucune adresse IPv6 n'est attribuée, la longueur de l'octet-string doit être 0.
traffic_class	Contient l'élément de classe de trafic de l'en-tête IPv6. Les 6 bits de poids fort sont utilisés pour DSCP (Differentiated Services Codepoint – code d'accès aux services différenciés), utilisé pour classer les paquets conformément au RFC 2474:1998, Article 3.

neighbor_discovery_setup Contient la configuration à utiliser pour les routeurs et les hôtes pour prendre en charge le protocole Neighbor Discovery pour IPv6 (RFC 4861).

```
array neighbour_discovery_setup
neighbor_discovery_setup:= structure
{
    RS_max_retry:      unsigned,
    RS_retry_wait_time: long-unsigned,
    RA_send_period:    double-long-unsigned
}
```

où:

RS_max_retry Donne le nombre maximal de nouvelles tentatives de sollicitation de routeur à réaliser par un nœud si l'annonce de routeur prévue n'a pas été reçue.

Plage: 1-255

Valeur par défaut: 3

RS_retry_wait_time Donne le temps d'attente en millisecondes entre deux nouvelles tentatives successives de sollicitation de routeur.

Plage: 0-65 535

Valeur par défaut: 10 000

RA_send_period Donne la période de transmission de l'annonce de routeur, en secondes.

Plage: 0-4 294 967 295

Description de la méthode

add_IPv6_address (data) Ajoute une adresse IPv6 de l'interface physique au tableau d'adresse IPv6.

```
data:= structure
{
    IPv6_address_type: enum,
    IPv6_address:      octet_string
}
```

Où IPv6_address_type définit le type d'adresse IPv6 à ajouter:

enum: (0) unicast,
(1) multicast,
(2) passerelle

IPv6_address spécifie l'adresse IPv6 à ajouter. La nouvelle adresse est automatiquement ajoutée à la fin de la liste. Si une adresse est à ajouter à une position spécifique, cela peut être fait en écrivant l'ensemble du tableau.

remove_IPv6_address (data) Retire l'adresse IPv6 de l'interface physique au tableau d'adresse IPv6.

```
data:= structure
{
    IPv6_address_type: enum,
    IPv6_address:      octet-string
}
```

Où IPv6_address_type définit le type d'adresse IPv6 à retirer:

enum: (0) unicast,
(1) multicast,
(2) passerelle

IPv6_address spécifie l'adresse IPv6 à retirer.

5.7.4 MAC address setup (class_id = 43, version = 0)

NOTE 1 Le nom et l'usage de cette classe d'interfaces ont été changés de "Ethernet setup" en "MAC address setup" pour permettre une utilisation plus générale, sans changer la version.

Les instances de cette IC contiennent l'adresse MAC du dispositif physique (ou, plus généralement, un dispositif ou un logiciel). Il doit y avoir une instance de cette IC pour chaque interface réseau d'un dispositif physique.

NOTE 2 Dans le cas du profil de communication basé sur HDLC à trois couches, l'adresse MAC (adresse HDLC inférieure) est contenue dans un objet "IEC HDLC setup" (configuration de HDLC IEC).

NOTE 3 Dans le cas du profil de communication S-FSK PLC, l'adresse MAC est contenue dans un objet "S-FSK Phy&MAC setup".

MAC address setup (établissement d'adresse MAC)	0...n	class_id = 43, version = 0			
Attributs	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. MAC_address	octet-string				x + 0x08
Méthodes spécifiques	m/o				

Description d'attribut

logical_name Identifie l'instance de l'objet "MAC address setup" (Établissement d'adresse MAC). Voir 6.2.19, 6.2.21 et 6.2.24.

mac_address Contient l'adresse MAC.

5.7.5 PPP setup (class_id = 44, version = 0)

NOTE 1 Comparée aux éditions antérieures de cette norme, la présente spécification apporte des améliorations dans la présentation des attributs. Ne s'agissant pas de modifications techniques, la version de l'IC reste 0.

Cette IC permet de modéliser l'établissement d'interfaces utilisant le protocole PPP, en gérant toutes les informations liées aux réglages de PPP associés à un dispositif physique donné et à une connexion de couche inférieure sur laquelle ces réglages sont utilisés. Il doit y avoir une instance de cette IC pour chaque interface réseau d'un dispositif physique.

PPP setup (établissement de PPP)	0...n	class_id = 44, version = 0			
Attributs	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. PHY_reference (static)	octet-string				x + 0x08
3. LCP_options (static)	LCP_options_type				x + 0x10
4. IPCP_options (static)	IPCP_options_type				x + 0x18
5. PPP_authentication (static)	PPP_auth_type				x + 0x20
Méthodes spécifiques	m/o				

Description d'attribut

logical_name Identifie l'instance de l'objet "PPP setup" (Établissement de PPP). Voir 6.2.21.

PHY_reference Référence un autre objet par son nom logique. L'objet référencé contient

	des informations relatives à l'interface de couche physique spécifique, prenant en charge la couche PPP.
LCP_options	Cet attribut contient les paramètres de prise en charge des options de configuration LCP sélectionnées. LCP_options_type ::= array LCP_options_type_element LCP_options_type_element ::= structure <pre> { LCP_Option_Type: unsigned, LCP_Option_Length: unsigned, LCP_Option_Data: CHOICE { structure [2] -- pour Callback-data boolean [3] -- pour ProtF-Compr et AdCtr- Compr, double-long-unsigned [6] -- pour ACCM et Mag-Num, unsigned [17] -- pour FCS-Alternatives, long-unsigned [18] -- pour MRU et Auth-Prot } } </pre> NOTE 2 Dans tous les cas, comme cela est spécifié dans le IETF STD 51 / RFC 1661, le champ LCP_Option_Length inclut la longueur totale des trois champs: LCP_Option_Type, LCP_Option_Length et LCP_Option_Data. NOTE 3 Pour les valeurs attribuées, voir <i>Point-to-Point (PPP) Protocol Field Assignments</i>, à l'adresse suivante: http://www.iana.org/assignments/ppp-numbers/ppp-numbers.xml Les LCP_Option_Types pris en charge sont les suivants: – Maximum-Receive-Unit (MRU), LCP_Option_Type = 1. Voir IETF STD 51 / RFC 1661. Cette option de configuration peut être envoyée pour informer l'homologue que l'implémentation réalisée peut recevoir des paquets plus gros ou pour demander à l'homologue d'envoyer des paquets plus petits. La valeur par défaut est 1 500 octets; Async-Control-Character-Map (ACCM), LCP_Option_Type = 2. Voir : RFC 2153, obsolètes: RFC 1548 – IETF STD 51 / RFC 1662: RFC 2153, obsolètes: RFC 1548 – IETF STD 51 / RFC 1662. Cette option de configuration fournit une méthode pour négocier l'utilisation de la transparence de caractères de contrôle sur des liaisons asynchrones; – Authentication-Protocol, LCP_Option_Type = 3. Voir IETF STD 51 / RFC 1661. Cette option de configuration fournit une méthode pour négocier l'utilisation d'un protocole spécifique pour l'authentification. Par défaut, l'authentification n'est pas exigée. La valeur indique le protocole d'authentification utilisé sur la liaison PPP donnée. Les valeurs possibles sont: 0x0000 – Aucun protocole d'authentification n'est utilisé, 0xc023 – Le protocole PAP est utilisé, 0xc223 – Le protocole CHAP est utilisé, 0xc227 – Le protocole EAP est utilisé. – Magic-Number, LCP_Option_Type = 5. Voir IETF STD 51 / RFC 1661. Cette option de configuration fournit une méthode pour détecter des

	liaisons fonctionnant en boucle et autres anomalies de la couche liaison de données;
	– Protocol-Field-Compression (PFC), LCP_Option_Type = 7. Voir IETF STD 51 / RFC 1661.
LCP_options (suite)	Cette option de configuration fournit une méthode pour négocier la compression des champs de protocole PPP; – Address-and-Control-Field-Compression (ACFC), LCP_Option_Type = 8. Voir IETF STD 51 / RFC 1661. Cette option de configuration fournit une méthode pour négocier la compression des champs adresse et commande de couche liaison de données; – FCS-Alternatives, LCP_Option_Type = 9. Voir RFC 1570. Cette option de configuration fournit une méthode pour une mise en œuvre pour spécifier un autre format de FCS devant être envoyé par l'homologue ou pour négocier la FCS également. La valeur du champ "FCS-Alter (FCS Alternatives) options" identifie la FCS utilisée. Ce champ contient un seul octet. Il est composé de la "valeur logique ou" des valeurs suivantes: - bit 1 FCS Null, - bit 2 FCS 16 bits de CCITT, - bit 4 FCS 32 bits de CCITT. – Callback, LCP_Option_Type = 13. Voir RFC 1570. Cette option de configuration fournit une méthode pour une mise en œuvre pour demander à un homologue joint de retourner l'appel. Cela procure une sécurité renforcée en assurant que le site distant peut se connecter seulement à partir d'un seul emplacement tel que défini par le numéro de rappel. <pre> callback_data ::= structure { callback_active: boolean, // default: false, callback_data_length: unsigned, callback_operation: unsigned, callback_message: octet-string } ou: - le champ callback-active indique si l'option de rappel est active sur cette liaison PPP; - le champ callback_operation indique le contenu du champ Message; callback_operation: unsigned (0) Emplacement déterminé par l'authentification d'utilisateur, (1) Chaîne de numérotation, (2) Identifiant d'emplacement, (3) Numéro E.164, (4) Nom distinctif X.500, (5) Non attribué, (6) L'emplacement est déterminé lors de la négociation CBCP. le champ callback_message est zéro ou plus d'octets, et son contenu général est déterminé par le champ callback_operation. Le format réel de l'information est spécifique au site ou à l'application. </pre>
IPCP_options	Contient les paramètres nécessaires pour le protocole de commande IP – le module de protocole de commande de réseau du PPP – permettant de négocier les paramètres de protocole Internet souhaités. Pour plus de

détails sur IPCP, voir RFC 1332.

IPCP_options_type ::= array IPCP_options_type_element

IPCP_options (suite)

```

IPCP_options_type_element ::= structure
{
    IPCP_Option_Type          unsigned,
    IPCP_Option_Length        unsigned,
    IPCP_Option_Data          CHOICE
    {
        array                  [1] -- pour Pref-Peer-IP,
        -- chaque adresse IP est de type double-long-unsigned
        boolean                 [3] -- pour GAO et USIP,
        double-long-unsigned    [6] -- pour Pref-Local-IP,
        long-unsigned           [18] -- pour IP-Comp-Prot
    }
}

```

NOTE 4 Dans tous les cas, comme cela est spécifié dans le RFC 1332, le champ IPCP_Option_Length inclut la longueur totale des trois champs: IPCP_Option_Type, IPCP_Option_Length et IPCP_Option_Data.

Les IPCP_Option_Types pris en charge sont les suivants:

- IP-Compression-Protocol (IP-Comp-Prot), IPCP_Option_Type = 2. Voir RFC 1332.

Cette option de configuration fournit une méthode pour négocier l'utilisation d'un protocole de compression spécifique. Par défaut, la compression n'est pas activée. Les valeurs possibles sont:

0x0000 – Aucune compression IP n'est utilisée (valeur par défaut),
 0x002d – Van Jacobson Compressed TCP/IP (RFC 1332),
 0x0003 – Robust Header Compression (ROHC) (RFC 3241),
 0x0061 – IP Header Compression (RFC 2507, RFC 3544)

- Preferred-Local-IP-Address (Pref-Local-I), IPCP_Option_Type = 3. Voir RFC 1332.

Cette option de configuration fournit une façon de négocier l'adresse IP à utiliser sur l'extrémité locale de la liaison. Elle permet à l'expéditeur de la Configure-Request d'énoncer quelle adresse IP est souhaitée ou de demander que l'homologue fournisse les informations. L'homologue peut fournir cette information en acquittant négativement (NAK) l'option et en retournant une adresse IP valide;

NOTE 5 Dans le RFC 1332, le nom du Type 3 de l'option est "IP-Address".

NOTE 6 Les types d'option 20, 21 et 22 ont été spécifiés pour les besoins de DLMS/COSEM. Ils ne sont pas spécifiés dans le RFC 1332.

- Preferred-Peer-IP-Addresses (Pref-Peer-IP), IPCP_Option_Type = 20.

Cette option de configuration fournit une façon de négocier l'adresse IP à utiliser sur l'extrémité distante de la liaison. Si la valeur TRUE est attribuée au paramètre Grant-Access-Only-to-Pref-Peer-on-List (GAO), le dispositif doit accepter la connexion PPP uniquement avec un dispositif distant dont l'une des adresses IP figure dans cette liste. Si la valeur TRUE est attribuée au paramètre Use-Static-IP-Pool (USIP), le dispositif Server COSEM doit tenter d'attribuer l'une de ces adresses IP au dispositif distant;

- Grant-Access-Only-to-Pref-Peer-on-List (GAO), IPCP_Option_Type = 21.

Cette option de configuration indique si le dispositif peut accepter la connexion PPP uniquement avec des dispositifs homologues dont l'adresse IP figure sur la liste ou pas. Sa valeur par défaut est FALSE;

- Use-Static-IP-Pool (USIP), IPCP_Option_Type = 22.

	Cette option de configuration indique si, oui ou non, il convient que le dispositif tente d'attribuer l'une des adresses IP des Preferred-Peer-IP-Addresses au dispositif situé à l'extrémité distante au cours de la phase de négociation d'adresses IP.
PPP_authentication	Contient les paramètres requis par la procédure d'authentification de PPP utilisée. <pre> PPP_auth_type ::= CHOICE { null-data [0] -- utilisé si aucune authentification n'est exigée, structure [2] -- PAP_login, CHAP_algorithm ou EAP_params } PAP_login ::= structure { user-name: octet-string, PAP-password: octet-string } CHAP_algorithm ::= structure { user-name: octet-string, algorithm_id: unsigned } </pre> Les valeurs possibles pour le paramètre CHAP-algorithm-id sont aujourd'hui les suivantes: 0x05: CHAP avec MD5 (par défaut). Voir RFC 1994. 0x06 – SHA-1, 0x80 – MS-CHAP, voir RFC 2433 0x81 – MS-CHAP-2, voir RFC 2759. NOTE 7 De nouvelles valeurs peuvent être utilisées à mesure qu'elles sont attribuées. Lorsque CHAP est utilisé, un "secret" est également requis pour vérifier le "défi" envoyé par le client. Ce "secret" n'est pas accessible dans l'objet "PPP setup". NOTE 8 Le PPP Challenge Handshake Authentication Protocol (CHAP) est spécifié dans le RFC 1994. <pre> EAP_params ::= structure { md5_challenge (Type 4): boolean, one_time_password (OTP, Type 5): boolean, generic_token_card (GTC, Type 6): boolean } </pre> NOTE 9 Le protocole EAP (Extensible Authentication Protocol) est spécifié dans le RFC 3748.

5.7.6 SMTP setup (class_id = 46, version = 0)

Cette IC permet d'établir un échange de données utilisant le protocole SMTP.

SMTP modem setup (établissement de modem SMTP)		0...n				class_id = 46, version = 0			
Attributs		Type de données		Min.	Max.	Def.	Nom court		
1.	logical_name (static)	octet-string					x		
2.	server_port (static)	long-unsigned				25	x + 0x08		
3.	user_name (static)	octet-string					x + 0x10		
4.	login_password (static)	octet-string					x + 0x18		
5.	server_address (static)	octet-string					x + 0x20		
6.	sender_address (static)	octet-string					x + 0x28		
Méthodes spécifiques		m/o							

Description d'attribut

logical_name	Identifie l'instance de l'objet "SMTP setup" (Établissement de SMTP). Voir 6.2.21.
server_port	Définit la valeur du port TCP-UDP lié à ce protocole. Par défaut, cette valeur est le numberID de port SMTP attribué par l'IANA: – smtp 25/tcp, smtp 25/udp Simple Mail Transfer (transfert simple de courrier)
user_name	Définit le nom d'utilisateur à utiliser pour la connexion au serveur SMTP.
login_password	Mot de passe à utiliser pour la connexion. Si la chaîne est vide, cela signifie qu'il n'y a pas d'authentification.
server_address	Définit l'adresse de serveur comme une chaîne d'octets. Cette adresse de serveur peut être un nom qui doit pouvoir être résolu par le DNS primaire ou le DNS secondaire. S'il s'agit directement de l'adresse IP du serveur, qui est spécifiée ici, elle doit être une chaîne au format comportant des points. EXEMPLE: 163.187.45.87.
sender_address	Définit l'adresse de l'expéditeur comme une chaîne d'octets. Cette adresse d'expéditeur peut être un nom. S'il s'agit directement de l'adresse IP de l'expéditeur, qui est spécifiée ici, elle sera une chaîne au format comportant des points.

5.8 Classes d'interfaces pour l'établissement d'échange de données en utilisant le PLC à modulation S-FSK

5.8.1 Généralités

Ce paragraphe spécifie les classes d'interfaces COSEM pour établir et gérer les couches protocolaires du profil de communications DLMS/COSEM S-FSK PLC:

- la couche physique S-FSK et la sous-couche MAC telles que définies dans l'IEC 61334-5-1:2001 et l'IEC 61334-4-512:2001.
- la sous-couche LLC telle que spécifiée dans l'IEC 61334-4-32:1996.

Les variables MIB variables/paramètres de liaison logique respectivement spécifiés dans l'IEC 61334-4-512:2001, l'IEC 61334-5-1:2001, l'IEC 61334-4-32:1996 et l'ISO/IEC 8802-2:1998 ont été mis en correspondance avec les attributs et/ou méthodes des IC de la spécification COSEM. La spécification de ces éléments a été prise dans les normes ci-dessus et le texte a été adapté à l'environnement DLMS/COSEM.

NOTE L'IEC 61334-4-512:2001 spécifie également un certain nombre de variables de gestion à utiliser du côté Client. Cependant, le modèle d'objet du côté Client n'est pas couvert par la présente norme.

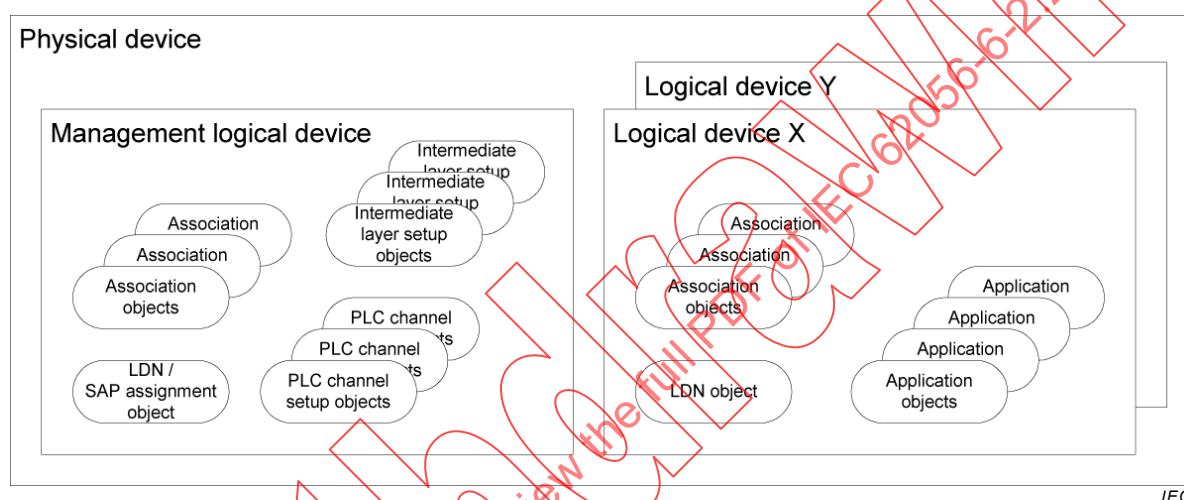
Pour des définitions relatives au profil S-FSK PLC, voir 3.2.

5.8.2 Vue d'ensemble

Les objets COSEM pour établir le canal PLC S-FSK et la couche LLC, si elle est mise en œuvre, doivent être situés dans le Dispositif logique de gestion des serveurs COSEM.

La Figure 16 montre un exemple avec un dispositif physique COSEM comprenant trois dispositifs logiques. Chaque dispositif logique doit contenir un objet Logical Device Name (LDN, Nom de dispositif logique). Chaque dispositif logique contient un ou plusieurs objets Association, un par client pris en charge.

NOTE Comme dans l'exemple, il y a plus d'un dispositif logique, le Dispositif logique de gestion obligatoire contient un objet "SAP Assignment" (Affectation de SAP) en lieu et place d'un objet "Logical Device Name" (Nom de dispositif logique).



Anglais	Français
Physical device	Dispositif physique
Logical device	Dispositif logique
Management logical device	Dispositif logique de gestion
Association objects	Objets «Association»
Intermediate layer setup objects	Objets établissement de couche intermédiaire
LDN /SAP assignment objet	Objet d'affectation de SAP /LDN
PLC channel setup objects	Objets d'établissement de canal PLC
LDN object	Objet LDN
Application objects	Objets d'application»

Figure 16 – Modèle d'objet des serveurs DLMS/COSEM

Le dispositif logique de gestion contient les objets d'établissement des couches physiques et MAC du canal PLC, ainsi que des objets d'établissement pour la/les couche(s) intermédiaire(s). Il peut contenir des objets d'application supplémentaires.

Les autres dispositifs logiques contiennent, en plus des objets Association et Logical Device Name susmentionnés, d'autres objets d'applications, détenant des paramètres et des valeurs de mesure.

L'IEC 61334-4-512:2001 utilise des variables nommées selon DLMS pour modéliser les objets MIB et spécifie leur nom DLMS dans la page 8...184. Pour la compatibilité avec des mises en œuvre existantes, les noms courts 8...400 [sic] sont réservés aux dispositifs utilisant les

profils PLC S-FSK de l'IEC 61334-5-1:2001 sans COSEM. Par conséquent, cette page ne doit pas être utilisée lors de la mise en correspondance des attributs et méthodes des objets COSEM spécifiés dans la présente norme avec les variables nommées selon DLMS (mise en correspondance par SN).

Le Tableau 14 montre la mise en correspondance des variables MIB avec les attributs et/ou méthodes des IC de la COSEM.

À noter que toutes les variables MIB spécifiées dans l'IEC 61334-4-512:2001 n'ont pas été mises en correspondance avec les attributs et méthodes des IC de COSEM. D'autre part, la présente norme spécifie un certain nombre de nouvelles variables de gestion.

**Tableau 14 – Mise en correspondance des variables MIB
de l'IEC 61334-4-512:2001 avec les attributs/méthodes des IC de la COSEM**

Nom	Référence (sauf indication contraire)	Classe d'interfaces	class_id / attribute / method
S-FSK Physical layer management (gestion de couche physique S-FSK)			
delta-electrical-phase	variable 1	S-FSK Phy&MAC set-up (class_id = 50, version = 1)	50 / Attr. 3
max-receiving-gain	variable 2		50 / Attr. 4
max-transmitting-gain	–		50 / Attr. 5
search-initiator-threshold	–		50 / Attr. 6
frequencies	–		50 / Attr. 7
transmission-speed	–		50 / Attr. 15
MAC layer management (gestion de couche MAC)			
mac-address	variable 3	S-FSK Phy&MAC set-up (class_id = 50, version = 1)	50 / Attr. 8
mac-group-addresses	variable 4		50 / Attr. 9
repeater	variable 5		50 / Attr. 10
repeater-status	–		50 / Attr. 11
search-initiator time-out	–	S-FSK MAC synchronization timeouts (class_id = 52, version = 0)	52 / Attr. 2
synchronization-confirmation- time-out	variable 6		52 / Attr. 3
time-out-not-addressed	variable 7		52 / Attr. 4
time-out-frame-not-OK	variable 8		52 / Attr. 5
min-delta-credit	variable 9	S-FSK Phy&MAC set-up (class_id = 50, version = 1)	50 / Attr. 12
initiator-mac-address	IEC 61334-5-1:2001 4.3.7.6		50 / Attr. 13
synchronization-locked	variable 10		50 / Attr. 14
IEC 61334-4-32 LLC layer management (gestion de couche LLC de l'IEC 61334-4-32)			
max-frame-length	IEC 61334-4-32:1996 5.1.4	IEC 61334-4-32 LLC setup (class_id = 55, version = 1)	55 / Attr. 2
reply-status-list	variable 11		55 / Attr. 3
broadcast-list	variable 12	–	–
L-SAP-list	variable 13	NOTE Dans DLMS/COSEM, les L-SAP des dispositifs logiques sont détenus par l'objet "SAP Assignment" (Affectation de SAP)	

ACSE management (gestion ACSE)			
application-context-list	variable 14	NOTE Dans DLMS/COSEM, les objets Association jouent un rôle similaire.	
Application management (gestion d'application)			
active-initiator	variable 15	S-FSK Active initiator (class_id = 51, version = 0)	51 / Attr. 2
MIB system objects (objets système MIB)			
reporting-system-list	variable 16	S-FSK Reporting system list (class_id = 56, version = 0)	56 / Attr. 2
Other MIB objects (autres objets MIB)			
reset-NEW-not-synchronized	variable 17	S-FSK Active initiator (class_id = 51, version = 0)	51 / Méthode 1
new-synchronization	IEC 61334-5-1:2001 4.3.7.6	–	
initiator-electrical-phase	variable 18		50 / Attr. 2
broadcast-frames-counter	variable 19	S-FSK MAC counters (class_id = 53, version = 0)	53 / Attr. 4
repetitions-counter	variable 20		53 / Attr. 5
transmissions-counter	variable 21		53 / Attr. 6
CRC-OK-frames-counter	variable 22		53 / Attr. 7
CRC-NOK-frames-counter	–		53 / Attr. 8
synchronization-register	variable 23		53 / Attr. 2
desynchronization-listing	variable 24		53 / Attr. 3

5.8.3 S-FSK Phy&MAC set-up (class_id = 50, version = 1)

NOTE 1 L'utilisation de la version 0 de cette classe d'interfaces est déconseillée.

Une instance de la classe "S-FSK Phy&MAC set-up" stocke les données nécessaires pour établir et gérer la couche physique et la couche MAC du profil de couche inférieure PLC S-FSK.

S-FSK Phy&MAC setup (établissement de Phy et MAC S-FSK)		0...n				class_id = 50, version = 1			
Attributs		Type de données		Min.	Max.	Def.	Nom court		
1.	logical_name (static)	octet-string					x		
2.	initiator_electrical_phase (static)	enum		0	3		x + 0x08		
3.	delta_electrical_phase (dyn.)	enum		0	6		x + 0x10		
4.	max_receiving_gain (static)	unsigned					x + 0x18		
5.	max_transmitting_gain (static)	unsigned					x + 0x20		
6.	search_initiator_threshold (static)	unsigned				98	x + 0x28		
7.	frequencies (static)	frequencies_type					x + 0x30		
8.	mac_address (dyn.)	long-unsigned				FFE	x + 0x38		
9.	mac_group_addresses (static)	array					x + 0x40		
10.	repeater (static)	enum					x + 0x48		
11.	repeater_status (dyn.)	boolean					x + 0x50		
12.	min_delta_credit (dyn.)	unsigned					x + 0x58		
13.	initiator_mac_address (dyn.)	long-unsigned					x + 0x60		
14.	synchronization_locked (dyn.)	boolean					x + 0x68		
15.	transmission_speed (static)	enum		0	6	3	x + 0x70		
Méthodes spécifiques		m/o							

Description d'attribut

logical_name Identifie l'instance de l'objet "S-FSK Phy&MAC setup" (Établissement de Phy et MAC S-FSK). Voir 6.2.22.

initiator_electrical_phase Contient la variable MIB *initiator-electrical-phase* (variable 18) spécifiée dans l'IEC 61334-4-512:2001, 5.8.

Il est inscrit par le système client pour indiquer la phase à laquelle il est connecté.

enum:

- (0) Non définie (par défaut),
- (1) Phase 1,
- (2) Phase 2,
- (3) Phase 3

NOTE 2 Cette énumération diffère de celle de l'IEC 61334-4-512.

delta_electrical_phase Contient la variable MIB *delta-electrical-phase* (variable 1) spécifiée dans l'IEC 61334-4-512:2001, 5.2 et l'IEC 61334-5-1:2001, 3.5.5.3.

Il indique la différence de phases entre la phase de connexion du client et la phase de connexion du serveur. Les valeurs suivantes sont prédéfinies:

enum:

- (0) Non définie: le serveur est temporairement incapable de déterminer la différence de phases,
- (1) Le système serveur est connecté à la même phase que le système client.

La différence de phases entre la phase de connexion du serveur et la phase de connexion du client est égale à:

- (2) 60 degrés,
- (3) 120 degrés,
- (4) 180 degrés,
- (5) -120 degrés,
- (6) -60 degrés,

max_receiving_gain	Contient la variable MIB <i>max-receiving-gain</i> (variable 2) spécifiée dans l'IEC 61334-4-512:2001, 5.2 et l'IEC 61334-5-1:2001, 3.5.5.3. Correspond au gain autorisé maximal destiné à être utilisé par le système serveur dans le mode réception. L'unité par défaut est dB. NOTE 3 L'IEC 61334-4-512:2001 ne spécifie pas d'unités. Les valeurs possibles du gain peuvent dépendre du matériel. Par conséquent, après écriture d'une valeur dans cet attribut, il convient de relire la valeur pour connaître la valeur réelle.
max_transmitting_gain	Contient la valeur du <i>max-transmitting-gain</i>. Correspond à l'affaiblissement maximal destiné à être utilisé par le système serveur dans le mode émission. L'unité par défaut est dB. Les valeurs possibles du gain peuvent dépendre du matériel. Par conséquent, après écriture d'une valeur dans cet attribut, il convient de relire la valeur pour connaître la valeur réelle.
search_initiator_threshold	Cet attribut est utilisé dans le processus d'initiateur de recherche intelligente. Si la valeur du signal de l'initiateur se situe au-dessus de la valeur de cet attribut, un processus de synchronisation rapide est possible. La valeur par défaut est 98 dBµV.
frequencies	Contient les fréquences requises par la modulation S-FSK. <pre>frequencies_type ::= structure { mark_frequency: double-long-unsigned, space_frequency: double-long-unsigned }</pre> L'unité par défaut est Hz.
mac_address	Contient la variable MIB <i>mac-address</i> (variable 3) spécifiée dans l'IEC 61334-4-512:2001, 5.3 et l'IEC 61334-5-1:2001, 4.3.7.6. NOTE 4 Les adresses MAC sont exprimées sur 12 bits. Contient la valeur de l'adresse de la jonction physique (adresse MAC) associée au système local. Dans l'état non configuré, l'adresse MAC est "NEW-address". L'attribut est écrit localement par le CIASE lorsque le système est enregistré (auprès d'un service Register). La valeur est utilisée dans chaque trame sortante ou entrante. La valeur par défaut est "NEW-address". Cet attribut est mis à NEW: – par la sous-couche MAC, une fois que le retard time-out-not-addressed est dépassé; – lorsqu'un système client "réinitialise" le système serveur. Voir 5.8.4. Lorsque cet attribut est mis à NEW: – le système perd sa synchronisation (fonction de la sous-couche MAC) – l'attribut <i>mac_group_address</i> est réinitialisé (tableau de 0 élément); – le système libère automatiquement toutes les AA qui peuvent être libérées. NOTE 5 Le second élément n'est pas présent dans l'IEC 61334-4-512:2001. Les adresses MAC prédéfinies sont présentées au Tableau 15.

mac_group_addresses	Contient la variable MIB <i>mac-group-address</i> (variable 4) spécifiée dans l'IEC 61334-4-512:2001, 5.3 et dans l'IEC 61334-5-1:2001, 4.3.7.6. Contient un jeu d'adresses de groupe MAC utilisées à des fins de diffusion. array mac-address mac-address::= long-unsigned Les adresses ALL-configured-address, ALL-physical-address et NO-BODY ne sont pas incluses dans cette liste. Celles-ci sont des valeurs prédéfinies internes. Cet attribut doit être écrit par l'initiateur utilisant les services DLMS pour déclarer des adresses de groupe MAC spécifiques sur un système serveur. Cet attribut est lu localement par la sous-couche MAC lors de la vérification du champ adresse de destination d'une trame MAC non reconnue comme adresse individuelle ou comme l'une des trois valeurs prédéfinies (ALL-configured-address, ALL-physical-address et NO-BODY).
repeater	Contient la variable MIB <i>repeater</i> (variable 5) spécifiée dans l'IEC 61334-4-512:2001, 5.3 et dans l'IEC 61334-5-1:2001, 4.3.7.6. Il spécifie si, oui ou non, le système serveur répète effectivement toutes les trames. enum: (0) never repeater, (1) always repeater, (2) dynamic repeater Si la variable <i>repeater</i> est égale à 0, il convient que le système serveur ne répète jamais les trames. Si elle est mise à 1, le système serveur est un répéteur: il est tenu de répéter toutes les trames reçues sans erreur et avec un crédit courant supérieur à zéro. Si elle est mise à 2, l'état de répéteur peut être modifié de façon dynamique par le serveur lui-même. NOTE 6 La valeur 2 n'est pas spécifiée dans l'IEC 61334-4-512. Cet attribut est lu en interne par la sous-couche MAC chaque fois qu'une trame est reçue. La valeur par défaut doit être spécifiée dans des spécifications d'accompagnement spécifiques au projet.
repeater_status	Contient l'état <i>repeater</i> courant du dispositif. boolean: (0) FALSE = pas de répéteur, (1) TRUE = répéteur
min_delta_credit	Contient la variable MIB <i>min-delta-credit</i> (variable 9) spécifiée dans l'IEC 61334-4-512:2001, 5.3 et l'IEC 61334-5-1:2001, 4.3.7.6. NOTE 7 Seuls les trois bits de poids faible sont utilisés. Le Delta Credit (DC) est la soustraction des champs Initial Credit (IC) et Current Credit (CC) d'une trame MAC reçue correcte. La valeur minimum de delta-credit pour une trame MAC reçue correcte, dirigée vers un système serveur, est contenue dans cette variable. La valeur par défaut est mise au crédit initial maximal (voir l'IEC 61334-5-1:2001, 4.2.3.1 pour de plus amples explications concernant le crédit et la valeur de MAX_INITIAL_CREDIT). Un système client peut réinitialiser cette variable en mettant cette valeur au crédit initial maximal.

initiator_mac_address	Contient la variable MIB <i>initiator-mac-address</i> spécifiée dans l'IEC 61334-5-1:2001, 4.3.7.6. Sa valeur est soit l'adresse MAC de l'active-initiator, soit l'adresse NO-BODY, en fonction de la valeur de l'attribut <i>synchronization_locked</i> (voir ci-dessous). Voir également l'IEC 61334-5-1:2001, 3.5.3, 4.1.6.3 et 4.1.7.2.																									
synchronization_locked	Contient la variable MIB <i>synchronization-locked</i> (variable 10) spécifiée dans l'IEC 61334-4-512:2001, 5.3. Commande l'état locked/unlocked (verrouillé/déverrouillé) de la synchronisation. Voir l'IEC 61334-5-1:2001 pour plus de détails. Si la valeur de cet attribut est égale à TRUE, le système est dans l'état <i>synchronization-locked</i>. Dans cet état, l'<i>initiator-mac-address</i> est toujours égale au champ d'adresse MAC de l'objet MIB active-initiator. Voir l'attribut 2 de l'IC "S-FSK Active initiator" en 5.8.4. Si la valeur de cet attribut est FALSE, le système est à l'état <i>synchronization-unlocked</i>. Dans cet état, l'attribut <i>initiator_mac_address</i> est toujours mis à la valeur NO-BODY: un changement de valeur dans le champ d'adresse MAC de l'objet MIB active-initiator n'a pas d'incidence sur le contenu de l'attribut <i>initiator_mac_address</i>, qui reste à la valeur NO-BODY. La valeur par défaut de cette variable doit être spécifiée dans les spécifications de mise en œuvre. NOTE 8 À l'état <i>synchronization-unlocked</i> (synchronisation déverrouillée), le serveur se synchronise sur n'importe quelle trame valide. À l'état synchronisation verrouillée, le serveur se synchronise seulement sur les trames produites ou dirigées vers le système client dont l'adresse MAC est égale à la valeur de l'attribut <i>initiator_mac_address</i>.																									
transmission_speed	La vitesse de transmission prise en charge par le dispositif physique. Voir également l'IEC 61334-5-1:2001, 3.2.2. <table> <tr> <td>enum:</td><td>50Hz</td><td>60 Hz</td></tr> <tr> <td>(0)</td><td>300 baud</td><td>360 baud</td></tr> <tr> <td>(1)</td><td>600 baud</td><td>720 baud</td></tr> <tr> <td>(2)</td><td>1 200 baud</td><td>1 440 baud</td></tr> <tr> <td>(3) -- par défaut</td><td>2 400 baud</td><td>2 880 baud</td></tr> <tr> <td>(4)</td><td>4 800 baud</td><td>5 760 baud</td></tr> <tr> <td>(5)</td><td>7 200 baud</td><td>8 640 baud</td></tr> <tr> <td>(6)</td><td>9 600 baud</td><td>11 520 baud</td></tr> </table>		enum:	50Hz	60 Hz	(0)	300 baud	360 baud	(1)	600 baud	720 baud	(2)	1 200 baud	1 440 baud	(3) -- par défaut	2 400 baud	2 880 baud	(4)	4 800 baud	5 760 baud	(5)	7 200 baud	8 640 baud	(6)	9 600 baud	11 520 baud
enum:	50Hz	60 Hz																								
(0)	300 baud	360 baud																								
(1)	600 baud	720 baud																								
(2)	1 200 baud	1 440 baud																								
(3) -- par défaut	2 400 baud	2 880 baud																								
(4)	4 800 baud	5 760 baud																								
(5)	7 200 baud	8 640 baud																								
(6)	9 600 baud	11 520 baud																								

Tableau 15 – Adresses MAC dans le profil S-FSK

Adresses	Value (Valeur)
NO-BODY	000
Local MAC (MAC locale)	001...FIMA-1
Initiator	FIMA...LIMA
MAC group address (adresse de groupe MAC)	LIMA + 1...FFB
All configured	FFC
NEW	FFE
All Physical	FFF

NOTE Les adresses MAC sont exprimées sur 12 bits. Ces adresses sont spécifiées dans l'IEC 61334-5-1:2001, 4.2.3.2, 4.3.7.5.1, 4.3.7.5.2 et 4.3.7.5.3.

FIMA = First Initiator MAC address (Adresse Mac du premier initiateur); C00

LIMA = Last Initiator MAC address (Adresse Mac du dernier initiateur); DEF

5.8.4 S-FSK Active initiator (class_id = 51, version = 0)

Une instance de l'IC "S-FSK Active initiator" stocke les données de l'initiateur actif. L'initiateur actif est le système client, qui a enregistré le dernier le système serveur avec une demande d'enregistrement CIASE. Voir l'IEC 61334-4-511:2000, 7.2.

S-FSK Active initiator (Initiateur actif S-FSK)	0...n	class_id = 51, version = 0			
Attributs	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. active_initiator (dyn.)	initiator_descriptor				x + 0x08
Méthodes spécifiques	m/o				
1. reset_NEW_not_synchronized					x + 0x10

Description d'attribut

logical_name Identifie l'instance d'objet "S-FSK Active initiator" (initiateur actif S-FSK). Voir 6.2.22.

active_initiator Contient la variable MIB *active-initiator* (variable 15) spécifiée dans l'IEC 61334-4-512:2001, 5.6.

Contient les identifiants du dernier initiateur actif qui a enregistré le système avec une demande d'enregistrement. Voir l'IEC 61334-4-511:2000, 7.2.

Le système initiateur est identifié avec son System Title (titre système), MAC address (adresse MAC) et L-SAP selector (sélecteur de L-SAP):

initiator_descriptor ::= structure

```
{
    system_title:      octet-string,
    MAC_address:      long-unsigned,
    L_SAP_selector:    unsigned
}
```

La taille et la structure du titre système peuvent être spécifiées dans les spécifications du système. Lorsque le titre système est utilisé comme partie intégrante du vecteur d'initialisation d'algorithmes cryptographiques, la taille doit satisfaire aux exigences applicables pour le vecteur d'initialisation.

L'élément MAC_address est utilisé pour mettre à jour la variable de gestion MAC *initiator-mac-address* lorsque le système est configuré dans l'état *synchronization-locked*. Voir la spécification des attributs *initiator_mac_address* et *synchronization_locked* de l'IC "S-FSK Phy&MAC setup" en 5.8.3.

Tant que le serveur n'est pas enregistré par un initiateur actif, le champ L_SAP_selector est mis à 0 et le champ system_title est égal à une chaîne d'octet de zéros (0).

La valeur par défaut de l'initiator-descriptor est: system_title = octet-string de zéros, MAC_address = NO-BODY et L_SAP_selector = 0.

La valeur de cet attribut peut être mise à jour par l'appel de la méthode *reset_NEW_not_synchronized* ou par le service "CIASE Register".

Description de la méthode

reset_NEW_not_synchronized (data)	Contient la variable MIB <i>reset-NEW-not-synchronized</i> (variable 17) spécifiée dans l'IEC 61334-4-512:2001, 5.8. Permet à un système client de "réinitialiser" le système serveur. La valeur proposée correspond à une adresse MAC de client. L'écriture est refusée si: – la valeur ne correspond pas à une adresse MAC de client valide ou à l'adresse NO-BODY prédéfinie; – la valeur proposée est différente de l'adresse NO-BODY et l'attribut <i>synchronization_locked</i> n'est pas égal à TRUE. Pour la description du processus d'initiateur de recherche intelligente (Intelligent Search Initiator), voir l'IEC 62056-8-3:2013, 10.7. Lorsque cette méthode est appelée, les actions suivantes sont exécutées: – le système retourne à l'état non configuré (UNC: MAC-address est égale à NEW-address). Cette transition provoque automatiquement la perte de synchronisation (fonction de la sous-couche MAC); – le système modifie la valeur de l'attribut <i>active_initiator</i>: la MAC_address est mise à la valeur proposée, le L-SAP_selector est mis à la valeur 0 et le system_title est mis à une chaîne d'octets de zéros; – toutes les AA qui peuvent être libérées sont libérées.

5.8.5 S-FSK MAC synchronization timeouts (class_id = 52, version = 0)

Une instance de l'IC "S-FSK MAC synchronization timeouts" (temps d'expirations de la synchronisation S-FSK) stocke les temps d'expirations liées au processus de synchronisation.

S-FSK MAC synchronization timeouts (temps d'expiration de synchronisation MAC S-FSK)		0...n	class_id = 52, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	search_initiator_timeout (static)	long-unsigned				x + 0x08
3.	synchronization_confirmation_timeout (static)	long-unsigned				x + 0x10
4.	time_out_not_addressed (static)	long-unsigned				x + 0x18
5.	time_out_frame_not_OK (static)	long-unsigned				x + 0x20
Méthodes spécifiques		m/o				

Description d'attribut

logical_name	Identifie l'instance d'objet "S-FSK synchronization timeouts" (temps d'expirations de synchronisation S-FSK). Voir 6.2.22.
search_initiator_timeout	Ce temps d'expiration prend en charge la fonction d'initiateur de recherche intelligente. Il définit la valeur du temps, en secondes, pendant lequel le système serveur recherche l'initiateur ayant le signal le plus fort. Au cours du temps d'expiration, tous les initiateurs, qui peuvent être entendus par les serveurs, sont censés parler. Après expiration de ce temps, le serveur acceptera une demande d'enregistrement issue de l'initiateur ayant le signal le plus fort et verrouillera sur l'initiateur en question. Si la valeur du temps d'expiration est égale à 0, cela signifie que la caractéristique n'est pas utilisée. Le temps d'expiration est lancé au début de la phase de recherche d'initiateur (Search Initiator Phase) lorsque le serveur reçoit la première trame avec une adresse MAC d'initiateur valide. Le temps d'expiration est relancé lorsque la "Search Initiator Phase" est terminée et le serveur verrouille sur l'initiateur. Au cours de la phase de vérification d'initiateur (Check Initiator Phase), elle est relancée à la réception de chaque trame valide. NOTE Une synchronisation rapide peut être accomplie si le niveau de signal est suffisamment bon (Niveau du signal d'initiateur $\geq$ Search-Initiator-Threshold) et l'une des adresses MAC (Source ou Destination) est une adresse MAC d'initiateur. Cela signifie que le module (le compteur) est proche d'un DC ou proche d'un module qui est déjà verrouillé sur le DC en question. Le module verrouille dans ce cas sur l'initiateur en question.

synchronization_confirmation_timeout	Contient la variable MIB <i>synchronization-confirmation-timeout</i> (variable 6) spécifiée dans l'IEC 61334-4-512:2001, 5.3 et l'IEC 61334-5-1:2001, 4.3.7.6. Définit la valeur du temps, en secondes, après lequel un système serveur qui devient juste synchronisé à une trame (détection d'un chemin de données égal à AAAA54C7 hex) perdra automatiquement sa synchronisation de trame si la sous-couche MAC n'identifie pas une trame MAC valide. Le temps d'expiration démarre après la réception des quatre premiers octets d'une trame physique. La valeur de cette variable peut être modifiée par un système client. Ce temps d'expiration assure une rapide désynchronisation d'un système qui s'est synchronisé sur une trame physique incorrecte. Voir l'IEC 61334-5-1:2001, 3.5.3 pour plus de détails. Il convient de spécifier la valeur par défaut de cette variable dans les spécifications de mise en œuvre. Une valeur égale à 0 équivaut à l'annulation de l'utilisation du compteur de <i>synchronization_confirmation_timeout</i> connexe.
time_out_not_addressed	Contient la variable MIB <i>time-out-not-addressed</i> (variable 7) spécifiée dans l'IEC 61334-4-512:2001, 5.3 et l'IEC 61334-5-1:2001, 4.3.7.6. Définit le temps, en minutes, après lequel un système serveur qui n'a pas été adressé individuellement: – retourne à l'état non configuré (UNC: MAC-address est égale à NEW-address): cette transition implique automatiquement la perte de synchronisation (fonction de la sous-couche MAC) et la libération de toutes les AA qui peuvent être libérées; – perd son initiateur actif: la MAC_address de l'initiateur actif est mise à NO-BODY, le sélecteur LSAP est mis à la valeur 00 et le Titre système est mis à une chaîne d'octets de zéros; Parce que les adresses de diffusion ne sont pas des adresses système individuelles, le temporisateur associé au retard <i>time-out-not-addressed</i> assure qu'un système oublié retournera tôt ou tard à l'état non configuré. Il sera alors redécouvert. Un système oublié est un système qui n'a pas été adressé individuellement pendant un temps supérieur à la quantité de temps "<i>time-out-not-addressed</i>". Il convient de spécifier la valeur par défaut de cette variable dans les spécifications de mise en œuvre. Une valeur égale à 0 équivaut à annuler l'utilisation du compteur de <i>time-out-not-addressed</i> connexe.
time_out_frame_not_OK	Contient la variable MIB <i>time-out-frame-not-OK</i> (variable 8) spécifiée dans l'IEC 61334-4-512:2001, 5.3 et l'IEC 61334-5-1:2001, 4.3.7.6. Définit le temps, en secondes, après lequel un système serveur qui n'a pas reçu une trame MAC correctement formée (champ NS incorrect, nombre incohérent de sous-trames reçues, faux contrôle de code de redondance cyclique) perd sa synchronisation de trame. La valeur par défaut de cette variable doit être spécifiée dans les spécifications de mise en œuvre. Une valeur égale à 0 équivaut à annuler l'utilisation du compteur de <i>time-out-frame-not-OK</i> connexe.

5.8.6 S-FSK MAC counters (class_id = 53, version = 0)

Une instance de l'IC "S-FSK MAC counters" stocke les compteurs liés aux phases d'échange de trames, d'émission et de répétition.

S-FSK MAC counters (compteurs S-FSK MAC)		0...n	class_id = 53, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	synchronization_register (dyn.)	array				x + 0x08
3.	desynchronization_listing (dyn.)	structure				x + 0x10
4.	broadcast_frames_counter (dyn.)	array				x + 0x18
5.	repetitions_counter (dyn.)	double-long-unsigned			0	x + 0x20
6.	transmissions_counter (dyn.)	double-long-unsigned			0	x + 0x28
7.	CRC_OK_frames_counter (dyn.)	double-long-unsigned			0	x + 0x30
8.	CRC_NOK_frames_counter (dyn.)	double-long-unsigned				x + 0x38
Méthodes spécifiques		m/o				
1.	reset (data)					x + 0x50

Description d'attribut

logical_name Identifie l'instance de l'objet "S-FSK MAC counters" (Compteurs S-FSK MAC). Voir 6.2.22.

synchronization_register Contient la variable MIB *synchronization-register* (variable 23) spécifiée dans l'IEC 61334-4-512:2001, 5.8.

array *synchronization_couples*

synchronization_couples ::= structure

```
{
    mac_address:                long-unsigned,
    synchronizations_counter:    double-long-unsigned
}
```

Cette variable compte le nombre de processus de synchronisation exécutés par le système. Les processus qui conduisent à une perte de synchronisation en raison de la détection d'un initiateur erroné sont enregistrés. Les autres processus qui conduisent à une perte de synchronisation (temps d'expiration, écriture de gestion) **ne sont pas enregistrés**.

Cette variable fournit un bilan des différents systèmes sur lesquels le système serveur est "potentiellement" capable de se synchroniser.

synchronization – register
(suite)

Un processus de synchronisation est initialisé lorsque l'entité d'application de gestion (gestionnaire de connexion) reçoit une primitive MA_Sync.indication (Synchronization State = SYNCHRO_FOUND) provenant de l'entité de sous-couche MAC. Ce processus est enregistré dans la variable *synchronization-register* uniquement si la primitive MA_Sync.indication (Synchronization State = SYNCHRO_FOUND) est suivie de l'une des trois primitives:

- 1) primitive MA_Data.indication (DA, SA, MSDU);
- 2) MA_Sync.indication (Synchronization State = SYNCHRO_CONF, SA, DA);
- 3) MA_Sync.indication (Synchronization State = SYNCHRO_LOSS, Synchro Loss Cause = wrong_initiator, SA, DA).

NOTE La troisième primitive n'est générée que si le système serveur est configuré à l'état *synchronization-locked*. Voir 5.8.3.

Les processus qui conduisent à la génération de primitives MA_Sync.indication (Synchronization State = SYNCHRO_LOSS) indiquant une perte de synchronisation due:

- à la couche physique;
- au compteur *time-out-not-addressed*;
- à la mise de l'attribut *mac_address* de l'objet "S-FSK Phy&MAC setup" à NEW; voir 5.8.3; ou
- à l'appel de la méthode *reset_NEW_not_synchronized* de l'objet "S-FSK Active initiator"; voir 5.8.4 (cela est appelé Management Writing, c'est-à-dire écriture de gestion)

ne sont pas pris en compte dans cette variable.

Pour les détails relatifs à la primitive de service MA_Sync.indication, voir l'IEC 61334-5-1:2001, 4.1.7.1.

Si le processus de synchronisation se termine avec l'une des trois primitives énumérées ci-dessus, la variable "*synchronization-register*" est mise à jour en tenant compte des champs SA et DA de la primitive.

La mise à jour de la variable *synchronization-register* est accomplie comme suit:

En premier lieu, l'entité de gestion (Management Entity) vérifie les champs SA et DA.

- Si l'un de ces champs correspond à une adresse MAC de client (CMA, client MAC address), l'entité:
 - vérifie si l'adresse MAC de client (CMA) apparaît dans l'un des couples contenus dans la variable *synchronization-register*;
 - si elle y apparaît, le sous-champ *synchronizations-counter* correspondant est incrémenté;
 - si elle n'y apparaît pas, un nouveau couple (*mac-address*, *synchronizations-counter*) est ajouté. Ce couple est initialisé à la valeur (CMA, 1).
- Si aucun des champs SA et DA ne correspond à une adresse MAC de client, il est supposé que le système a trouvé sa référence de synchronisation sur une trame de type *DiscoverReport*. Dans ce cas, la *mac-address* qu'il convient d'enregistrer dans la variable *synchronization-register* est la valeur NEW prédéfinie (FFE). La mise à jour de la variable "*synchronization-register*" est accomplie de la même manière qu'elle est faite pour une adresse MAC de client (CMA) normale.

Lorsqu'un champ *synchronizations-counter* atteint la valeur maximale, il retourne automatiquement à 0 sur le prochain incrément.

Il convient de spécifier dans les spécifications de mise en œuvre le nombre maximal de couples de synchronisation {mac-address, synchronizations-counter} contenus dans cette variable. Lorsque le maximum est atteint, la mise à jour de la variable suit un mécanisme de premier entré, premier sorti (First-In-First-Out, FIFO): seules les adresses MAC sources les plus récentes sont mémorisées.

La valeur par défaut de cette variable est un tableau vide.

desynchronization_listing Contient la variable MIB *desynchronization-listing* (variable 24) spécifiée dans l'IEC 61334-4-512:2001, 5.8.

structure

```
{
    nb_physical_layer_desynchronization:
        double_long_unsigned,
    nb_time_out_not_addressed_desynchronization:
        double_long_unsigned,
    nb_timeout_frame_not_OK_desynchronization:
        double_long_unsigned,
    nb_write_request_desynchronization:
        double_long_unsigned,
    nb_wrong_initiator_desynchronization:
        double_long_unsigned
}
```

Cette variable compte le nombre de désynchronisations qui se sont produites en fonction de leur cause. A la réception d'une notification de perte de synchronisation, l'entité de gestion (Management Entity) met à jour cet attribut en incrémentant le compteur lié à la cause de la désynchronisation.

Lorsque l'un des compteurs atteint la valeur maximale, il retourne automatiquement à 0 sur le prochain incrément.

La valeur par défaut de cette variable est une structure dont tous les éléments qui sont tous égaux à 0.

broadcast_frames_counter Contient la variable MIB *broadcast-frames-counter* (variable 19) spécifiée dans l'IEC 61334-4-512:2001, 5.8.

array broadcast-couples

broadcast-couples::= structure

```
{
    source-mac-address: long-unsigned,
    frames-counter:     double-long-unsigned
}
```

Il compte les trames de diffusion reçues par le système serveur et produites par un système client (source-mac-address = n'importe quelle client-mac-address valide, destination-mac address = ALL-physical). Le nombre de trames est classé en fonction de l'origine de l'émetteur. Le compteur est incrémenté même si la LLC-destination-address n'est pas valide sur le système serveur. Lorsqu'un champ frames-counter atteint sa valeur maximale, il retourne automatiquement à 0 sur le prochain incrément.

Il convient de spécifier dans les spécifications de mise en œuvre le nombre maximal de couples de diffusion "broadcast-couples" {source-mac-address, frames-counter} contenus dans cette variable. Lorsque le maximum est atteint, la mise à jour de la variable suit un mécanisme de premier entré, premier sorti (First-In-First-Out, FIFO): seules les adresses MAC sources les plus récentes sont mémorisées.

repetitions_counter	Contient la variable MIB <i>repetitions-counter</i> (variable 20) spécifiée dans l'IEC 61334-4-512:2001, 5.8. Compte le nombre de phases de répétition. Les phases de répétition suivant une émission ne sont pas comptées. Si la sous-couche MAC est configurée dans le mode "non-répéteur", cette variable n'est pas mise à jour. Le compteur de répétitions mesure l'activité du système comme un répéteur. Une trame reçue répétée cinq fois (de CC=4 à CC=0) est comptée seulement une seule fois dans le compteur de répétitions, car elle correspond à une phase de répétition. Le compteur est incrémenté au début de chaque phase de répétition. Lorsque le compteur de répétitions atteint la valeur maximale, il retourne automatiquement à 0 sur le prochain incrément. La valeur par défaut est 0.
transmissions_counter	Contient la variable MIB <i>transmissions-counter</i> (variable 21) spécifiée dans l'IEC 61334-4-512:2001, 5.8. Compte le nombre de phases d'émission. Une phase d'émission est caractérisée par l'émission et la répétition d'une trame. Une phase de répétition qui suit la réception d'une trame n'est pas comptée. Le compteur d'émissions est incrémenté au début de chaque phase d'émission. Un système client peut écrire cette variable pour mettre à jour le compteur. Lorsque le compteur d'émissions atteint la valeur maximale, il retourne automatiquement à 0 sur le prochain incrément. La valeur par défaut est 0.
CRC_OK_frames_counter	Contient la variable MIB <i>CRC-OK-frames-counter</i> (variable 22) spécifiée dans l'IEC 61334-4-512:2001, 5.8. Compte le nombre de trames reçues avec un champ Frame Check Sequence correct. Lorsque le champ "CRC OK frames counter" atteint la valeur maximale, il retourne automatiquement à 0 sur le prochain incrément. La valeur par défaut est 0.
CRC_NOK_frames_counter	Compte le nombre de trames reçues avec un champ Frame Check Sequence incorrect. Lorsque le champ "CRC NOK frames counter" atteint la valeur maximale, il retourne automatiquement à 0 sur le prochain incrément. La valeur par défaut est 0.
Description de la méthode	
reset (data)	Efface tous les compteurs. data::= integer (0)

5.8.7 IEC 61334-4-32 LLC setup (class_id = 55, version = 1)

Une instance de l'IC "IEC 61334-4-32 LLC setup" contient les paramètres nécessaires pour établir et gérer la couche LLC comme spécifié dans l'IEC 61334-4-32.

IEC 61334-4-32 LLC setup (établissement de LLC de l'IEC 61334-4-32)		0...n				class_id = 55, version = 1			
Attributs		Type de données		Min.	Max.	Def.	Nom court		
1.	logical_name (static)	octet-string					x		
2.	max_frame_length (static)	long-unsigned					x + 0x08		
3.	reply_status_list (dyn.)	array					x + 0x10		
Méthodes spécifiques		m/o							

Description d'attribut

logical_name	Identifie l'instance de l'objet "IEC 61334-4-32 LLC setup" (établissement de LLC de l'IEC 61334-4-32). Voir 6.2.22.
max_frame_length	Contient la longueur de la trame de LLC en octets. Voir l'IEC 61334-4-32:1996, 5.1.4. Pour le profil PLC S-FSK, les valeurs minimum/par défaut / maximale sont respectivement 26/134/242 octets. Voir l'IEC 61334-5-1:2001, 4.2.2. NOTE Pour d'autres profils de couches inférieures, voir les valeurs correspondantes dans la spécification pertinente.
reply_status_list	Contient la variable MIB <i>reply-status-list</i> (variable 11) spécifiée dans l'IEC 61334-4-512:2001, 5.4. Énumère les L-SAP qui ont un tampon RDR (Reply Data on Request, Données de réponse sur demande) non vide, qui n'a pas été déjà lu. La longueur d'une L-SDU en attente est spécifiée en nombre de sous-trames (différent de zéro). La variable est générée localement par la sous-couche LLC.
reply_status_list (suite)	array reply_status reply_status ::= structure <pre> { L_SAP_selector: unsigned, length_of_waiting_L_SDU: unsigned } </pre> length_of_waiting_L_SDU dans le cas du profil S-FSK est en nombre de sous-trames. Les valeurs valides sont 1 à 7.

5.8.8 S-FSK Reporting system list (class_id = 56, version = 0)

Une instance de l'IC "S-FSK Reporting system list" contient la liste des systèmes de comptes-rendus.

S-FSK Reporting system list (liste de systèmes de comptes-rendus S-FSK)	0...n	class_id = 56, version = 0			
Attributs	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. reporting_system_list (dyn.)	array				x + 0x08
Méthodes spécifiques	m/o				

Description d'attribut

logical_name	Identifie l'instance d'objet "S-FSK Reporting system list" (liste de systèmes de comptes-rendus S-FSK). Voir 6.2.22.
reporting_system_list	Contient la variable MIB <i>reporting-system-list</i> (variable 16) spécifiée dans l'IEC 61334-4-512:2001, 5.7. array system-title system-title::= octet-string Contient les titres systèmes des systèmes serveurs qui ont fait une demande de DiscoverReport et qui n'ont pas été déjà enregistrés. La liste a une taille finie et est triée à l'arrivée. Le premier élément est le plus récent. Une fois qu'elle est pleine, les éléments les plus anciens sont remplacés par les nouveaux. La liste de systèmes de comptes-rendus est mise à jour: – lorsqu'une CI_PDU DiscoverReport est reçue par le système serveur (quel que soit son état: non configuré ou configuré): le CIASE ajoute le titre du système de compte-rendu au début de la liste et vérifie qu'il n'existe nulle part ailleurs dans la liste; s'il existe, il détruit l'ancien. Un titre système ne peut exister qu'une seule fois dans la liste; – lorsqu'une CI_PDU Register est reçue par le système serveur (quel que soit son état: non configuré ou configuré): le CIASE vérifie la liste de systèmes de comptes-rendus. Si un titre système est présent dans la liste de systèmes de comptes-rendus et dans la CI-PDU Register, le CIASE supprime le titre système dans la la liste de systèmes de comptes-rendus: ce système n'est plus considéré comme un système de compte-rendu.

5.9 Classes d'interfaces pour l'établissement de la couche LLC pour l'ISO/IEC 8802-2

5.9.1 Généralités

Le présent paragraphe spécifie les IC disponibles pour l'établissement de la couche LLC selon l'ISO/IEC 8802-2, utilisée dans certains profils de communication DLMS/COSEM, dans les divers types de fonctionnement.

Pour les définitions liées à la couche LLC selon l'ISO/IEEE 8802-2, voir 1.4.2 de l'ISO/IEC 8802-2:1998.

5.9.2 ISO/IEC 8802-2 LLC Type 1 setup (class_id = 57, version = 0)

Une instance de l'IC "ISO/IEC 8802-2 LLC Type 1 setup" contient les paramètres nécessaires pour établir la couche LLC de l'ISO/IEC 8802-2 dans le fonctionnement de Type 1.

ISO/IEC 8802-2 LLC Type 1 setup (établissement de LLC de Type 1 selon l'ISO/IEC 8802-2)		0...n		class_id = 57, version = 0		
Attributs		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. max_octets_ui_pdu	(static)	long unsigned			128	x + 0x08
Méthodes spécifiques		m/o				

Description d'attribut

logical_name	Identifie l'instance de l'objet "ISO/IEC 8802-2 LLC Type 1 setup". Voir 6.2.23.
max_octets_ui_pdu	Se référer à la spécification de protocole MAC appropriée pour toute limitation sur le nombre maximum d'octets dans une PDU UI. Aucune restriction n'est imposée par la sous-couche LLC. Cependant, dans l'intérêt de disposer d'une valeur sur laquelle tous les utilisateurs de LLC de type 1 peuvent s'appuyer, toutes les MAC doivent au moins être capables de prendre en charge les PDU UI avec des champs d'informations d'une longueur maximale pouvant atteindre 128 octets inclus. Voir l'ISO/IEC 8802-2:1998, 6.8.1 <i>Maximum number of octets in a UI PDU</i> (nombre maximum d'octets dans une PDU UI).

5.9.3 ISO/IEC 8802-2 LLC Type 2 setup (class_id = 58, version = 0)

Une instance de l'IC "ISO/IEC 8802-2 LLC Type 2 setup" contient les paramètres nécessaires pour établir la couche LLC de l'ISO/IEC 8802-2 dans le fonctionnement de Type 2.

ISO/IEC 8802-2 LLC Type 2 setup (établissement de LLC de Type 2 selon l'ISO/IEC 8802-2)		0...n				class_id = 58, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court			
1.	logical_name (static)	octet-string				x			
2.	transmit_window_size_k (static)	unsigned	1	127	1	x + 0x08			
3.	receive_window_size_rw (static)	unsigned	1	127	1	x + 0x10			
4.	max_octets_i_pdu_n1 (static)	long unsigned			128	x + 0x18			
5.	max_number_transmissions_n2 (static)	unsigned				x + 0x20			
6.	acknowledgement_timer (static)	long-unsigned				x + 0x28			
7.	p_bit_timer (static)	long-unsigned				x + 0x30			
8.	reject_timer (static)	long-unsigned				x + 0x38			
9.	busy_state_timer (static)	long-unsigned				x + 0x40			
Méthodes spécifiques		m/o							

Description d'attribut

logical_name	Identifie l'instance de l'objet "ISO/IEC 8802-2 LLC Type 2 setup". Voir 6.2.23.
transmit_window_size_k	La taille de fenêtre d'émission (k) doit être un paramètre de connexion de liaison de données qui ne peut jamais dépasser 127. Elle doit désigner le nombre maximal de I PDU numérotées séquentiellement que la LLC émettrice peut avoir en cours (à savoir, unacknowledged c'est-à-dire non acquittées). La valeur de k est le nombre maximal pour lequel la variable d'état V(S) d'envoi de LLC expéditrice peut dépasser le N(R) de la dernière I PDU reçue. Voir l'ISO/IEC 8802-2:1998, 7.8.4 <i>Transmit window size, k</i> . (Taille de fenêtre d'émission, k)

receive_window_size_rw	La taille de fenêtre de réception (RW) doit être un paramètre de connexion de liaison de données qui ne peut jamais dépasser 127. Elle doit désigner le nombre maximal de I PDU non acquittées numérotées séquentiellement que la LLC locale autorise la LLC distante à avoir en cours. Elle est émise dans le champ information de XID (voir l'ISO/IEC 8802-2:1998, 5.4.1.1.2) et s'applique à l'expéditeur de XID. Le destinataire XID doit mettre sa fenêtre d'émission (k) à une valeur inférieure ou égale à la fenêtre de réception de l'expéditeur XID afin d'éviter d'engorger l'expéditeur XID. Voir l'ISO/IEC 8802-2:1998, 7.8.6 <i>Receive window size, RW</i>. (Taille de fenêtre de réception, RW)
max_octets_i_pdu_n1	N1 est un paramètre de connexion de liaison de données qui désigne le nombre maximal d'octets dans une I PDU. Se référer aux diverses descriptions MAC pour déterminer la valeur précise de N1 pour une méthode donnée d'accès au support. La LLC elle-même n'impose pas de restrictions sur la valeur de N1. Cependant, dans l'intérêt de disposer d'une valeur de N1 sur laquelle tous les utilisateurs de LLC de type 2 peuvent s'appuyer, toutes les MAC doivent au moins être capables de prendre en charge les I PDU avec des champs d'informations d'une longueur maximale pouvant atteindre 128 octets inclus. Voir l'ISO/IEC 8802-2:1998, 7.8.3 <i>Maximum number of octets in an I PDU, N1</i> (Nombre maximal d'octets dans I PDU, N1)
max_number_transmissions_n2	N2 est un paramètre de connexion de liaison de données qui indique le nombre maximal de fois qu'une PDU est envoyée à la suite de l'expiration du temporisateur d'acquiescement, du temporisateur de bit P, du temporisateur de rejet ou du temporisateur d'état occupé. Voir l'ISO/IEC 8802-2:1998, 7.8.2 <i>Maximum number of transmissions, N2</i>. (Nombre maximal d'émissions, N2)
acknowledgement_timer	Le temporisateur d'acquiescement est un paramètre de connexion de liaison de données qui doit définir l'intervalle de temps pendant lequel la LLC doit s'attendre à recevoir un acquiescement à une ou plusieurs I PDU ou une PDU de réponse attendue à une PDU de commande non numérotée envoyée. L'unité est la seconde. Voir l'ISO/IEC 8802-2:1998, 7.8.1.1 <i>Acknowledgement timer</i>. (Temporisateur d'acquiescement)
p_bit_timer	Le temporisateur de bit P est un paramètre de connexion de liaison de données qui doit définir l'intervalle de temps pendant lequel la LLC doit s'attendre à recevoir une PDU avec le bit F mis à "1" en réponse à une commande de type 2 avec le bit P mis à "1". L'unité est la seconde. Voir l'ISO/IEC 8802-2:1998, 7.8.1.2 <i>P-bit timer</i>. (Temporisateur de P-bit)
reject_timer	Le temporisateur de rejet est un paramètre de connexion de liaison de données qui doit définir l'intervalle de temps pendant lequel la LLC doit s'attendre à recevoir une réponse à une REJ PDU envoyée. L'unité est la seconde. Voir l'ISO/IEC 8802-2:1998, 7.8.1.3 <i>Reject timer</i> (Temporisateur de rejet).
busy_state_timer	Le temporisateur d'état occupé est un paramètre de connexion de liaison de données qui doit définir l'intervalle de temps pendant lequel la LLC doit attendre une indication de la suppression d'un état occupé au niveau d'une autre LLC. L'unité est la seconde. Voir l'ISO/IEC 8802-2:1998, 7.8.1.4, <i>Busy-state timer</i> (Temporisateur d'état occupé).

5.9.4 ISO/IEC 8802-2 LLC Type 3 setup (class_id = 59, version = 0)

Une instance de l'IC "ISO/IEC 8802-2 LLC Type 3 setup" contient les paramètres nécessaires pour établir la couche LLC de l'ISO/IEC 8802-2 dans le fonctionnement de Type 3.

ISO/IEC 8802-2 LLC Type 3 setup (établissement de LLC de Type 3 selon l'ISO/IEC 8802-2)		0...n	class_id = 59, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				X
2. max_octets_acn_pdu_n3	(static)	long unsigned				x + 0x08
3. max_number_transmissions_n4	(static)	unsigned				x + 0x10
4. acknowledgement_time_t1	(static)	long unsigned				x + 0x18
5. receive_lifetime_var_t2	(static)	long unsigned				x + 0x20
6. transmit_lifetime_var_t3	(static)	long unsigned				x + 0x28
Méthodes spécifiques		m/o				

Description d'attribut

logical_name	Identifie l'instance de l'objet "ISO/IEC 8802-2 LLC Type 3 setup". Voir 6.2.23.
max_octets_acn_pdu_n3	N3 est un paramètre de liaison logique qui désigne le nombre maximal d'octets dans une PDU de commande ACn. Se référer aux diverses descriptions MAC pour déterminer la valeur précise de N3 pour une méthode donnée d'accès au support. La LLC n'impose pas de restrictions sur la valeur de N3. Voir l'ISO/IEC 8802-2:1998, 8.6.2 <i>Maximum number of octets in an ACn command PDU</i>, N3 (Nombre maximal d'octets dans une commande ACn PDU, N3).
max_number_transmissions_n4	N4 est un paramètre de liaison logique qui indique le nombre maximal de fois qu'une PDU de commande ACn est envoyée par la LLC tentant d'accomplir un échange d'informations réussi. Normalement, N4 est réglé à une valeur suffisamment grande pour surmonter la perte d'une PDU due à des états d'erreur de liaison. Si la sous-couche de Contrôle d'accès au support a sa propre capacité de retransmission, la valeur de N4 peut être mise à un afin que la LLC ne remette pas elle-même en file d'attente une PDU destinée à la sous-couche de Contrôle d'accès au support. Voir l'ISO/IEC 8802-2:1998, 8.6.1 <i>Maximum number of transmissions</i>, N4. (Nombre maximal d'émissions, N4).

acknowledgement_time_t1 Le temps d'acquiescement est un paramètre de liaison logique qui détermine la période des temporisateurs d'acquiescement et, à ce titre, doit définir l'intervalle de temps pendant lequel la LLC doit s'attendre à recevoir une PDU de réponse ACn de la part d'une LLC spécifique dont la LLC attend une PDU de réponse. Le temps d'acquiescement doit prendre en compte tout retard introduit par la sous-couche MAC et la question de savoir si le temporisateur a démarré au début ou à la fin de l'envoi de la PDU de commande ACn par la LLC. Le fonctionnement correct de la procédure doit exiger que le temps d'acquiescement soit supérieur au temps normal entre l'envoi d'une PDU de commande ACn et la réception de la PDU de réponse ACn correspondante. Si la sous-couche de Contrôle d'accès au support effectue ses propres retransmissions et si le paramètre de liaison logique N4 est mis à un pour empêcher que la LLC ne remette en file d'attente une PDU, le temps d'acquiescement T1 peut être mis à l'infini, rendant ainsi inutiles les temporisateurs d'acquiescement.

L'unité est la seconde. L'infini est indiqué par le fait que tous les bits sont mis à 1.

Voir l'ISO/IEC 8802-2:1998, 8.6.4 *Acknowledgement time, T1* (Temps d'acquiescement, T1)

receive_lifetime_var_t2 Cette valeur de temps est un paramètre de liaison logique qui détermine la période de tous les temporisateurs de durée de vie des variables de réception. T2 doit être plus long d'une certaine marge de sécurité que la période la plus longue possible pendant laquelle la première émission et toutes les répétitions d'une simple PDU peuvent se produire. La marge de sécurité doit tenir compte de tout ce qui a une incidence sur la perception des LLC relative à l'heure d'arrivée des PDU, tel que l'heure de réponse de LLC, la résolution du temporisateur et les variations temporelles requises pour que la sous-couche de Contrôle d'accès au support transmette les PDU reçues à la LLC.

Si la destruction des variables d'état reçues n'est pas souhaitée, la valeur du temps T2 peut être mise à l'infini. Dans ce cas, le temporisateur de durée de vie de variable de réception peut ne pas être mis en œuvre.

L'unité est la seconde. L'infini est indiqué par le fait que tous les bits sont mis à 1.

Voir l'ISO/IEC 8802-2:1998, 8.6.5 *Receive lifetime variable, T2* (Variable de durée de vie de réception, T2).

transmit_lifetime_var_t3	Cette valeur de temps est un paramètre de liaison logique qui détermine la durée de vie minimale des variables d'état de séquence d'émission. T3 doit être plus long d'une certaine marge de sécurité que: 1) la variable de liaison logique T2 aux stations vers lesquelles les commandes ACn sont envoyées; et 2) la durée de vie la plus longue possible de la paire commande-réponse Can. La durée de vie d'une paire commande-réponse Can doit prendre en compte la somme du temps de traitement, des retards de mise en file d'attente et du temps d'émission pour les PDU de commande et de réponse au niveau des stations locales et distantes. Si la destruction des variables d'état d'émission n'est pas souhaitée, la valeur du temps T3 peut être mise à l'infini. À noter que, si le paramètre de durée de vie de variable de réception T2 est mis à l'infini en des stations distantes vers lesquelles les commandes ACn sont envoyées, le paramètre T3 doit être mis à l'infini en la station locale. L'unité est la seconde. L'infini est indiqué par le fait que tous les bits sont mis à 1. Voir l'ISO/IEC 8802-2:1998, 8.6.6 <i>Transmit lifetime variable, T3</i> (Variable de durée de vie d'émission, T3).

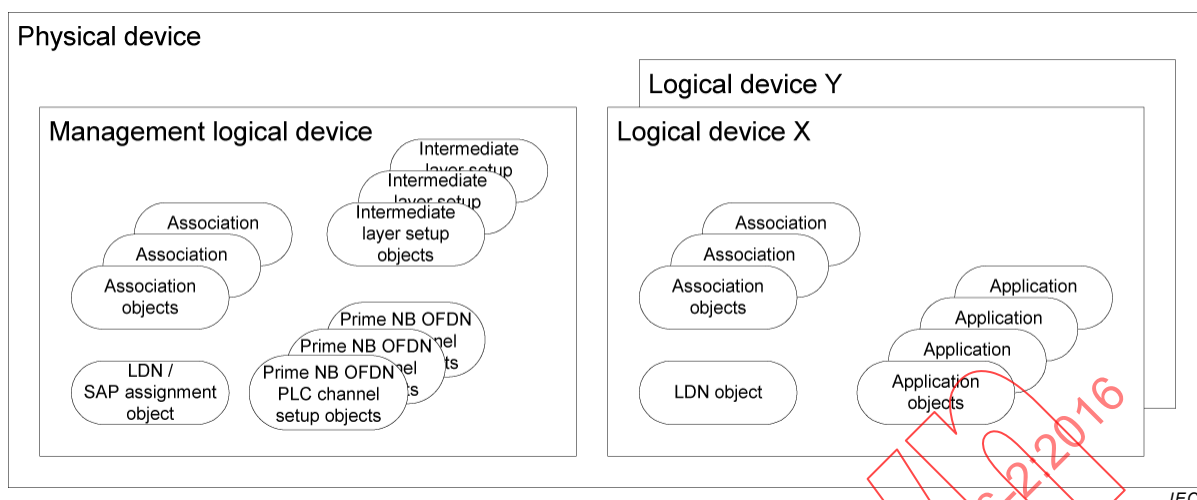
5.10 Classes d'interfaces pour l'établissement et la gestion du profil PLC OFDM à bande étroite DLMS/COSEM pour les réseaux PRIME

5.10.1 Vue d'ensemble

Voir également l'Annex D.

Les objets COSEM pour l'échange de données utilisant le profil PLC OFDM à bande étroite pour les réseaux PRIME, s'il est mis en œuvre, doivent se trouver dans le dispositif logique de gestion des serveurs COSEM.

La Figure 17 montre un exemple avec un dispositif physique COSEM comprenant trois dispositifs logiques.



Anglais	Français
Physical device	Dispositif physique
Logical device	Dispositif logique
Management logical device	Dispositif logique de gestion
Association objects	Objets «Association»
Intermediate layer setup objects	Objets établissement de couche intermédiaire
LDN /SAP assignment objet	Objet d'affectation de SAP /LDN
LDN object	Objet LDN
Application objects	Objets «objet d'application»

Figure 17 – Modèle d'objet des serveurs DLMS/COSEM

Chaque dispositif logique doit contenir un objet Logical Device Name (LDN, Nom de dispositif logique).

NOTE Comme dans cet exemple, il y a plusieurs dispositifs logiques, le Dispositif logique de gestion obligatoire contient un objet "SAP Assignment" (Affectation de SAP) en lieu et place d'un objet "Logical Device" (Dispositif logique).

Chaque dispositif logique contient un ou plusieurs objets "Association", un par client pris en charge.

Le dispositif logique de gestion contient les objets d'établissement des couches physiques et MAC du profil PLC OFDM à bande étroite pour les réseaux PRIME, ainsi que des objets d'établissement pour la/les couche(s) intermédiaire(s). Il peut contenir des objets d'application supplémentaires.

Les autres dispositifs logiques contiennent, en plus des objets "Association" et "Logical Device Name" susmentionnés, d'autres objets d'applications, détenant des paramètres et des valeurs de mesure.

Pour configurer et gérer 61334-4-32 LLC SSCS, une IC est spécifiée:

- "61334-4-32 LLC SSCS setup" (établissement de LLC SSCS 61334-4-32). Voir 5.10.3.

Pour gérer la couche physique (PhL) PRIME NB OFDM PLC, une IC est spécifiée:

- "PRIME NB OFDM PLC Physical layer counters" (Compteurs de couche physique PRIME NB OFDM PLC). Voir 5.10.5;

Pour configurer et gérer la couche PLC PRIME OFDM MAC, quatre IC sont spécifiées:

- "PRIME NB OFDM PLC MAC setup". Voir 5.10.6;
- "PRIME NB OFDM PLC MAC functional parameters". Voir 5.10.7;
- "PRIME NB OFDM PLC MAC counters". Voir 5.10.8;
- "PRIME NB OFDM PLC MAC network administration data". Voir 5.10.9.

Pour l'identification d'application, une IC est spécifiée:

- "PRIME NB OFDM PLC Application identification". Voir 5.10.11.

5.10.2 Mise en correspondance des attributs PRIME NB OFDM PLC PIB avec les attributs d'IC COSEM

L'ITU-T G.9904:2012 définit les variables des attributs PHY PIB (Tableau 10-1 et Tableau 10-2), les attributs MAC PIB (Tableau 10-3 au Tableau 10-8) et les attributs PIB d'application (Tableau 10-9).

Le Tableau 16 présente la mise en correspondance des attributs PRIME NB OFDM PLC PIB avec les attributs des IC COSEM. Seules les variables liées au commutateur et aux nœuds de terminal sont mises en correspondance. Les variables pertinentes pour le nœud de base ne sont pas mises en correspondance, car le nœud de base agit comme un client pour le réseau de distribution.

Tableau 16 – Mise en correspondance des attributs PRIME NB OFDM PLC PIB avec les attributs d'IC COSEM

Nom	Identifiant	Classe d'interfaces	class_id/attribute
Attributs PHY PIB – Variable en lecture seule PHY donnant des informations statistiques ¹			
phyStatsCRCIncorrectCount	0x00A0	PRIME NB OFDM PLC Physical layer counters (class_id = 81, version = 0)	81 / Attr. 2
PhyStatsCRCFailCount	0x00A1		81 / Attr. 3
phyStatsTxDropCount	0x00A2		81 / Attr. 4
phyStatsRxDropCount	0x00A3		81 / Attr. 5
phyStatsRxTotalCount	0x00A4		Non modélisé
phyStatsBlkAvgEvm	0x00A5		Non modélisé
phyEmaSmoothing	0x00A8		Non modélisé
Paramètres en lecture seule PHY, donnant des informations relatives à une mise en œuvre particulière ²			
phyTxQueueLen	0x00B0		Non modélisé
phyRxQueueLen	0x00B1		
phyTxProcessingDelay	0x00B2		
phyRxProcessingDelay	0x00B3		
PhyAgcMinGain	0x00B4		
PhyAgcStepValue	0x00B5		
PhyAgcStepNumber	0x00B6		
Variables en lecture-écriture MAC, variables en lecture seule ³			
macMinSwitchSearchTime	0x0010	PRIME NB OFDM PLC MAC setup (class_id = 82, version = 0)	82 / Attr. 2
macMaxPromotionPdu	0x0011		82 / Attr. 3
macMaxPromotionPduTxPeriod	0x0012		82 / Attr. 4
macBeaconsPerFrame	0x0013		82 / Attr. 5
macSCPMaxTxAttempts	0x0014		82 / Attr. 6
macCtlReTxTimer	0x0015		82 / Attr. 7
macMaxCtlReTx	0x0018		82 / Attr. 8

Nom	Identifiant	Classe d'interfaces	class_id/attribute
macEMASmoothing	0x0019		Non modélisé
macSCPRBO	0x0016		Non modélisé
macSCPChSenseCount	0x0017		Non modélisé
Variables en lecture seule MAC donnant des informations fonctionnelles ⁴			
macLNID	0x0020	PRIME NB OFDM PLC MAC functional parameters (class_id = 83 version = 0)	83 / Attr. 2
macLSID	0x0021		83 / Attr. 3
macSID	0x0022		83 / Attr. 4
macSNA	0x0023		83 / Attr. 5
macState	0x0024		83 / Attr. 6
macSCPLength	0x0025		83 / Attr. 7
macNodeHierarchyLevel	0x0026		83 / Attr. 8
macBeaconSlotCount	0x0027		83 / Attr. 9
macBeaconRxSlot	0x0028		83 / Attr. 10
macBeaconTxSlot	0x0029		83 / Attr. 11
macBeaconRxFrequency	0x002A		83 / Attr. 12
macBeaconTxFrequency	0x002B		83 / Attr. 13
macCapabilities	0x002C		83 / Attr. 14
Variable en lecture seule MAC donnant des informations statistiques ⁵			
macTxDataPktCount	0x0040	PRIME NB OFDM PLC MAC counters (class_id = 84, version = 0)	84 / Attr. 2
macRxDataPktCount	0x0041		84 / Attr. 3
macTxCtrlPktCount	0x0042		84 / Attr. 4
macRxCtrlPktCount	0x0043		84 / Attr. 5
macCSMAFailCount	0x0044		84 / Attr. 6
macCSMAChBusyCount	0x0045		84 / Attr. 7
Listes en lecture seule, que la couche mac a mis a disposition par l'intermediaire de l'interface de gestion ⁶			
macListMcastEntries	0x0052	PRIME NB OFDM PLC MAC network administration data (class_id = 85, version = 0)	85 / Attr. 2
macListSwitchTable	0x0053		85 / Attr. 3
macListDirectTable	0x0055		85 / Attr. 4
macListAvailableSwitches	0x0056		85 / Attr. 5
macListPhyComm	0x0057		85 / Attr. 6
Attributs PIB d'application ⁷			
AppFwVersion	0x0075	PRIME NB OFDM PLC Application identification (class_id = 86, version = 0)	86 / Attr. 2
AppVendorId	0x0076		86 / Attr. 3
AppProductId	0x0077		86 / Attr. 4
1 Voir l'ITU-T G.9904:2012, Tableau 10-1. 2 Voir l'ITU-T G.9904:2012, Tableau 10-2. 3 Voir l'ITU-T G.9904:2012, Tableau 10-3, 10-4. 4 Voir l'ITU-T G.9904:2012, Tableau 10-5. 5 Voir l'ITU-T G.9904:2012, Tableau 10-6. 6 Voir l'ITU-T G.9904:2012, Tableau 10-7. 7 Voir l'ITU-T G.9904:2012, Tableau 10-9.			
NOTE Dans les spécifications des classes d'interfaces COSEM, le trait de soulignement est utilisé, alors que dans la Recommandation ITU-T G.9904:2012 et dans le Tableau 1, la notation camel est utilisée.			

5.10.3 61334-4-32 LLC SSCS setup (class_id = 80, version = 0)

Une instance de l'IC "61334-4-32 LLC SSCS" (Service Specific Convergence Sublayer, 432 CL) contient les adresses fournies par le nœud de base à l'ouverture de la couche de convergence, en réponse à la demande d'établissement du nœud de service. Elles permettent au nœud de service de faire partie intégrante du réseau géré par le nœud de base.

61334-4-32 LLC SSCS setup (établissement de LLC SSCS 61334-4-32)		class_id = 80, version = 0			
Attributs	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. service_node_address (dyn.)	long-unsigned				x + 0x08
3. base_node_address (dyn.)	long-unsigned				x + 0x10
Méthodes spécifiques	m/o				
1. reset (data)	o				x + 0x20

Description d'attribut

logical_name	Identifie l'instance de l'objet "61334-4-32 LLC SSCS setup". Voir 6.2.24.
service_node_address	Contient la valeur de l'adresse attribuée au nœud de service lors de son enregistrement par le nœud de base. Après l'annulation de l'enregistrement, la valeur de cette adresse est NEW, soit 0xFFE.
base_node_address	Contient la valeur de l'adresse du nœud de base à laquelle est enregistré le nœud de service. Après l'annulation de l'enregistrement, la valeur de cette adresse est 0.

Description de la méthode

reset (data)	Cette méthode est utilisée pour annuler l'attribution de l'adresse du nœud de service. La valeur de <i>service_node_address</i> devient NEW, et celle de <i>base_node_address</i> devient 0.

5.10.4 Paramètres de couche physique PRIME NB OFDM PLC

Les paramètres de couche physique ne sont pas modélisés.

5.10.5 PRIME NB OFDM PLC Physical layer counters (class_id = 81, version = 0)

Une instance de l'IC "PRIME NB OFDM PLC Physical layer counters" enregistre les compteurs liés aux échanges des couches physiques. L'objectif de ces compteurs est de donner des informations statistiques pour les besoins de la gestion.

Les attributs des instances de cette IC doivent être en lecture seule. Ils peuvent être réinitialisés à l'aide de la méthode reset.

PRIME NB OFDM PLC Physical layer counters		0...n	class_id = 81, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	phy_stats_crc_incorrect_count (dyn.)	long-unsigned				x + 0x08
3.	phy_stats_crc_fail_count (dyn.)	long-unsigned				x + 0x10
4.	phy_stats_tx_drop_count (dyn.)	long-unsigned				x + 0x18
5.	phy_stats_rx_drop_count (dyn.)	long-unsigned				x + 0x20
Méthodes spécifiques		m/o				
1.	reset (data)	o				

Description d'attribut

logical_name	Identifie l'instance d'objet "PRIME NB OFDM PLC Physical layer counters". Voir 6.2.24.
phy_stats_crc_incorrect_count	Attribut PIB 0x00A0: Nombre de salves reçues sur la couche physique pour lesquelles le CRC était incorrect.
phy_stats_crc_failed_count	Attribut PIB 0x00A1: Nombre de salves reçues sur la couche physique pour lesquelles le CRC était correct, mais dont le champ <i>Protocol</i> de l'entête PHY contenait une valeur non valide. Il s'agit du nombre de réceptions de données corrompues que le CRC n'a pas détectées.
phy_stats_tx_drop_count	Attribut PIB 0x00A2: Nombre de fois que la couche PHY a reçu de nouvelles données à transmettre (PHY_DATA.request) et a dû écraser des données existantes dans sa file d'attente de transmission ou supprimer les données dans une nouvelle requête car la file d'attente était saturée.
phy_stats_rx_drop_count	Attribut PIB 0x00A3: Nombre de fois que la couche PHY a reçu de nouvelles données sur le canal et a dû écraser des données existantes dans sa file d'attente de réception ou supprimer les données qu'elle venait de recevoir car la file d'attente était saturée.
NOTE Si un compteur atteint la valeur maximale (0xFFFF), il est automatiquement arrêté.	

Description de la méthode

reset (data)	Cette méthode est utilisée pour réinitialiser tous les compteurs d'une instance de cette interface.

5.10.6 PRIME NB OFDM PLC MAC setup (class_id = 82, version = 0)

Une instance de l'IC "PRIME NB OFDM PLC MAC setup" contient les paramètres nécessaires à l'établissement et la gestion de la couche PRIME NB OFDM PLC MAC.

Ces attributs influencent le comportement fonctionnel d'une mise en œuvre. Ces attributs peuvent être définis hors de la couche MAC, en général par l'entité de gestion, et les mises en œuvre peuvent permettre de modifier leurs valeurs pendant le fonctionnement normal, c'est-à-dire même après la séquence de démarrage du dispositif.

PRIME NB OFDM PLC MAC setup		0...n	class_id = 82, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	mac_min_switch_search_time (static)	unsigned	16	32	24	x + 0x08
3.	mac_max_promotion_pdu (static)	unsigned	1	4	2	x + 0x10
4.	mac_promotion_pdu_tx_period (static)	unsigned	2	8	5	x + 0x18
5.	mac_beacons_per_frame (static)	unsigned	1	5	5	x + 0x20
6.	mac_scp_max_tx_attempts (static)	unsigned	2	5	5	x + 0x28
7.	mac_ctl_re_tx_timer (static)	unsigned	2	20	15	x + 0x30
8.	mac_max_ctl_re_tx (static)	unsigned	3	5	3	x + 0x38
Méthodes spécifiques		m/o				

Description d'attribut

logical_name	Identifie l'instance d'objet "PRIME NB OFDM PLC MAC setup". Voir 6.2.24.
mac_min_switch_search_time	Attribut PIB 0x0010: Durée minimale pendant laquelle il convient qu'un nœud de service à l'état <i>Disconnected</i> analyse le canal à la recherche d'étiquettes avant de pouvoir diffuser une PNPDU. Cet attribut n'est pas conservé dans les nœuds de base. L'unité de cet attribut est la seconde.
mac_max_promotion_pdu	Attribut PIB 0x0011: Nombre maximal de PNPDU qui peuvent être transmises par un nœud de service dans une période de <i>mac_promotion_pdu_tx_period</i> secondes. Cet attribut n'est pas conservé dans les nœuds de base.
mac_promotion_pdu_tx_period	Attribut PIB 0x0012: Intervalle de temps de limitation du nombre de PNPDU transmises à partir d'un nœud de service. Pas plus de <i>mac_max_promotion_pdu</i> ne peuvent être transmises dans une période <i>mac_promotion_pdu_tx_period</i> . L'unité de cet attribut est la seconde.
mac_beacons_per_frame	Attribut PIB 0x0013: Nombre maximal d'emplacements d'étiquette qui peuvent être prévus dans une trame. Cet attribut est conservé dans les nœuds de base.
mac_scp_max_tx_attempts	Attribut PIB 0x0014: Nombre de fois que l'algorithme CSMA tente de transmettre les données demandées lorsqu'une tentative précédente a été différée, la couche PHY ayant indiqué un canal occupé.
mac_ctl_re_tx_timer	Attribut PIB 0x0015: Nombre de secondes pendant lesquelles une entité MAC attend l'acquiescement de la réception du paquet de commande MAC de la part de son entité homologue. À l'expiration de cette période, l'entité MAC peut retransmettre le paquet de commande MAC. L'unité de cet attribut est la seconde.
mac_max_ctl_re_tx	Attribut PIB 0x0018: Nombre maximal de fois qu'une entité MAC tente de retransmettre un paquet de commande MAC qui n'a pas été acquitté. Si le nombre de retransmissions atteint la valeur maximale, l'entité MAC doit annuler les tentatives supplémentaires de retransmission du paquet de commande MAC.

NOTE Si un compteur atteint la valeur maximale (0xFFFF), il est automatiquement arrêté.

5.10.7 PRIME NB OFDM PLC MAC functional parameters (class_id = 83 version = 0)

Les attributs d'une instance de l'IC "PRIME NB OFDM PLC MAC functional parameters" appartiennent au comportement fonctionnel de MAC. Ils donnent des informations relatives à des aspects spécifiques.

Les attributs des instances de cette IC doivent être en lecture seule.

PRIME NB OFDM PLC MAC functional parameters		0...n	class_id = 83, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1.	logical_name (static)	octet-string				x
2.	mac_LNID (static)	long	0	16 383		x + 0x08
3.	mac_LSID (static)	unsigned	0	255		x + 0x10
4.	mac_SID (static)	unsigned	0	255		x + 0x18
5.	mac_SNA (static)	octet-string				x + 0x20
6.	mac_state (static)	enum	0	3		x + 0x28
7.	mac_scp_length (static)	long				x + 0x30
8.	mac_node_hierarchy_level (static)	unsigned	0	63		x + 0x38
9.	mac_beacon_slot_count (static)	unsigned	0	7		x + 0x40
10.	mac_beacon_rx_slot (static)	unsigned	0	7		x + 0x48
11.	mac_beacon_tx_slot (static)	unsigned	0	7		x + 0x50
12.	mac_beacon_rx_frequency (static)	unsigned	0	31		x + 0x58
13.	mac_beacon_tx_frequency (static)	unsigned	0	31		x + 0x60
14.	mac_capabilities (static)	long-unsigned				x + 0x68
Méthodes spécifiques		m/o				

Description d'attribut

logical_name	Identifie l'instance d'objet "PRIME NB OFDM PLC MAC functional parameters". Voir 6.2.24.
mac_LNID	Attribut PIB 0x0020: LNID attribué à ce nœud au moment de son enregistrement.
mac_LSID	Attribut PIB 0x0021: LSID attribué à ce nœud au moment de sa promotion. Cet attribut n'est pas conservé si le nœud est à l'état fonctionnel <i>Terminal</i> .
mac_SID	Attribut PIB 0x0022: SID du nœud de commutation grâce auquel ce nœud est connecté au sous-réseau. Cet attribut n'est pas conservé dans un nœud de base.
mac_SNA	Attribut PIB 0x0023: Adresse de sous-réseau à laquelle ce nœud est enregistré. Le nœud de base renvoie le SNA qu'il utilise.

mac_state	Attribut PIB 0x0024: État fonctionnel présent du nœud. enum: (0) Disconnected (déconnecté), (1) Terminal, (2) Commutateur, (3) Base
mac_scp_length	Attribut PIB 0x0025: Longueur de SCP, en nombre de symboles, dans la trame en cours.
mac_node_hierarchy_level	Attribut PIB 0x0026: Niveau de ce nœud dans la hiérarchie de sous-réseau.
mac_beacon_slot_count	Attribut PIB 0x0027: Nombre d'emplacements d'étiquettes prévus dans la structure de trame en cours.
mac_beacon_rx_slot	Attribut PIB 0x0028: Emplacement d'étiquettes dans lequel le nœud de commutation de ce dispositif transmet son étiquette. Cet attribut n'est pas conservé dans un nœud de base.
mac_beacon_tx_slot	Attribut PIB 0x0029: Emplacement d'étiquette dans lequel ce dispositif transmet son étiquette. Cet attribut n'est pas conservé dans les nœuds de service à l'état fonctionnel <i>Terminal</i> .
mac_beacon_rx_frequency	Attribut PIB 0x002A: Nombre de trames entre les réceptions de deux étiquettes successives. Une valeur 0x0 indique les étiquettes reçues dans chaque trame. Cet attribut n'est pas conservé dans un nœud de base.
mac_beacon_tx_frequency	Attribut PIB 0x002B: Nombre de trames entre les transmissions de deux étiquettes successives. Une valeur 0x0 indique les étiquettes transmises dans chaque trame. Cet attribut n'est pas conservé dans les nœuds de service à l'état fonctionnel <i>Terminal</i> .
mac_capabilities	Attribut PIB 0x002C: Cet attribut définit les capacités du nœud. Il s'agit d'une carte binaire qui définit une capacité. Bit 0: Capacité de commutation Bit 1: Agrégation de paquets Bit 2: Période sans conflit Bit 3: Connexion directe Bit 4: Multicast Bit 5: Gestion de solidité PHY Bit 6: ARQ Bit 7: Réservé pour une utilisation ultérieure Bit 8: Commutation de connexion directe Bit 9: Capacité de commutation multicast Bit 10: Capacité de commutation de gestion de solidité PHY Bit 11: Capacité de commutation de mise en mémoire tampon ARQ Bit 12 à 15: Réservé pour une utilisation ultérieure

5.10.8 PRIME NB OFDM PLC MAC counters (class_id = 84, version = 0)

Une instance de l'IC "PRIME NB OFDM PLC MAC counters" enregistre des informations statistiques sur le fonctionnement de la couche MAC pour les besoins de la gestion. Les

attributs des instances de cette IC doivent être en lecture seule. Ils peuvent être réinitialisés à l'aide de la méthode reset.

PRIME NB OFDM PLC MAC counters		0...n	class_id = 84, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. mac_tx_data_pkt_count	(dyn.)	double-long-unsigned		4 294 967 295		x + 0x08
3. mac_rx_data_pkt_count	(dyn.)	double-long-unsigned		4 294 967 295		x + 0x10
4. mac_tx_ctrl_pkt_count	(dyn.)	double-long-unsigned		4 294 967 295		x + 0x18
5. mac_rx_ctrl_pkt_count	(dyn.)	double-long-unsigned		4 294 967 295		x + 0x20
6. mac_csma_fail_count	(dyn.)	double-long-unsigned		4 294 967 295		x + 0x28
7. mac_csma_ch_busy_count	(dyn.)	double-long-unsigned		4 294 967 295		x + 0x30
Méthodes spécifiques		m/o				
1. reset (data)		o				x + 0x40

Description d'attribut

logical_name	Identifie l'instance d'objet "PRIME NB OFDM PLC MAC counters". Voir 6.2.24.
mac_tx_data_pkt_count	Attribut PIB 0x0040: Compte les MSDU dont la transmission a abouti.
mac_rx_data_pkt_count	Attribut PIB 0x0041: Compte les MSDU reçues dont l'adresse de destination était ce nœud.
mac_tx_ctrl_pkt_count	Attribut PIB 0x0042: Compte les paquets de commande MAC dont la transmission a abouti.
mac_rx_ctrl_pkt_count	Attribut PIB 0x0043: Compte les paquets de commande MAC reçus dont la destination était ce nœud.
mac_csma_fail_count	Attribut PIB 0x0044: Compte les tentatives de transmission CSMA qui n'ont pas abouti.
mac_csma_ch_busy_count	Attribut PIB 0x0045: Compte le nombre de fois que ce nœud est tenu de réduire la transmission SCP en raison de l'état occupé du canal.

NOTE Si un compteur atteint la valeur maximale (0xFFFFFFFF), il est automatiquement arrêté.

Description de la méthode

reset (data)	Cette méthode est utilisée pour réinitialiser tous les compteurs d'une instance de cette classe d'interfaces. data::= integer (0)

5.10.9 PRIME NB OFDM PLC MAC network administration data (class_id = 85, version = 0)

Cette IC contient les paramètres liés à la gestion des dispositifs connectés au réseau.

PRIME NB OFDM PLC MAC network administration data (données d'administration de réseau PRIME NB OFDM PLC MAC)		0...n	class_id = 85, version = 0			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. mac_list_multicast_entries	(dyn.)	array				x + 0x08
3. mac_list_switch_table	(dyn.)	array				x + 0x10
4. mac_list_direct_table	(dyn.)	array				x + 0x18
5. mac_list_available_switches	(dyn.)	array				x + 0x20
6. mac_list_phy_comm	(dyn.)	array				x + 0x28
Méthodes spécifiques		m/o				
1. reset (data)		o				x + 0x30

Description d'attribut

logical_name	Identifie l'instance d'objet "PRIME NB OFDM PLC MAC network administration data". Voir 6.2.24.
mac_list_multicast_entries	Attribut PIB 0x0052: Liste des entrées dans la table de commutation multicast. Cette liste n'est pas conservée dans les nœuds de service à l'état fonctionnel <i>Terminal</i>. mac_list_multicast_entries_type ::= array mac_list_multicast_entries_element mac_list_multicast_entries_element ::= structure { mcast_entry_LCID: integer, -- LCID du groupe multicast mcast_entry_members: long -- nombre de nœuds enfants } Le nombre de nœuds enfants est le nombre de membres de ce groupe, y compris le Nœud lui-même.
mac_list_switch_table	Attribut PIB 0x0053: Tableau de commutation. Ce tableau n'est pas conservé par les nœuds de service à l'état <i>Terminal</i>. mac switch_table ::= array stbl_entry_LSID stbl_entry_LSID ::= long -- SID du nœud de commutation connexe

mac_list_direct_table	Attribut PIB 0x0055: Tableau direct. mac_direct_table ::= array mac_direct_table_element
	mac_direct_table_element ::= structure <pre> { dconn_entry_src_SID: long, dconn_entry_src_LNID: long, dconn_entry_src_LCID: long, dconn_entry_dst_SID: long, dconn_entry_dst_LNID: long, dconn_entry_dst_LCID: long, dconn_entry_DID: octet-string (6 octets) } </pre> où: – dconn_entry_src_SID est le SID du commutateur par lequel le nœud de service source est connecté; – dconn_entry_src_LNID est le NID attribué au nœud de service source; – dconn_entry_src_LCID est le LCID attribué à cette connexion au niveau de la source; – dconn_entry_dst_SID est le SID du commutateur par lequel le nœud de service de destination est connecté; – dconn_entry_dst_LNID est le NID attribué au nœud de service de destination; – dconn_entry_dst_LCID est le LCID attribué à cette connexion au niveau de la destination; dconn_entry_DID est l'EUI-48 du commutateur direct.
mac_list_available_switches	Attribut PIB 0x0056: Liste des nœuds de commutation dont les étiquettes sont reçues. mac_list_available_switches ::= array mac_list_available_switches_element mac_list_available_switches_element ::= structure <pre> { slist_entry_SNA: octet-string (6 octets), slist_entry_LSID: long, slist_entry_level: integer, slist_entry_rx_level: integer, slist_entry_rx_snr: integer } </pre> où: – slist_entry_SNA est l'EUI-48 du sous-réseau; – slist_entry_LSID est le SID de ce commutateur; – slist_entry_level est le niveau de ce commutateur dans la hiérarchie de sous-réseau; – slist_entry_rx_level est le niveau de signal reçu pour ce commutateur; – slist_entry_rx_snr est le rapport signal-bruit pour ce commutateur.
mac_list_phy_comm	Attribut PIB 0x0057: Liste des paramètres de communication PHY. Il est maintenu dans chaque nœud. Pour les nœuds de terminal, il contient une seule entrée pour le commutateur par lequel le nœud est connecté. Pour les autres nœuds, il contient également des entrées pour chaque nœud enfant directement connecté. mac_list_phy_comm ::= array phy_comm_element

mac_list_	phy_comm_element::= structure
phy_comm	{
(suite)	phy_Comm_EUI: octet-string, phy_Comm_Tx_Pwr: integer, phy_Comm_Tx_Cod: integer, phy_Comm_Rx_Cod: integer, phy_Comm_Rx_Lvl: integer, phy_Comm_SNR: integer, phy_Comm_Tx_Pwr_Mod: integer, phy_Comm_Tx_Cod_Mod: integer, phy_Comm_Rx_Cod_Mod: integer
	}
	où:
	– phy_Comm_EUI est l'EUI-48 de l'autre dispositif;
	– phy_Comm_Tx_Pwr est la puissance de l'émetteur de paquets GPDU envoyés au dispositif;
	– phy_Comm_Tx_Cod est le codage de l'émetteur de paquets GPDU envoyés au dispositif;
	– phy_Comm_Rx_Cod est le codage du récepteur de paquets GPDU reçus du dispositif;
	– phy_Comm_Rx_Lvl est le niveau de puissance du récepteur de paquets GPDU reçus du dispositif;
	– phy_Comm_SNR est le SNR des paquets GPDU reçus du dispositif;
	– phy_Comm_Tx_Pwr_Mod est le nombre de fois que la puissance de l'émetteur a été modifiée;
	– phy_Comm_Tx_Cod_Mod est le nombre de fois que le codage de l'émetteur a été modifié;
	– phy_Comm_Rx_Cod_Mod est le nombre de fois que le codage du récepteur a été modifié.

Description de la méthode

reset (data)	Cette méthode est utilisée pour redéfinir toutes les entrées (d'un tableau de 0 élément) des attributs de 2 à 6 de l'instance de cette classe d'interfaces.
	data::= integer (0)

5.10.10 PRIME NB OFDM PLC MAC address setup (class_id = 43, version = 0)

Une instance de l'IC "MAC address setup" contient l'adresse MAC EUI-48 du dispositif. La taille de cette chaîne d'octets est de 6, compte tenu du fait que cette adresse est un EUI-48 et qu'elle est unique. Voir aussi 5.7.4 et 6.2.24.

5.10.11 PRIME NB OFDM PLC Application identification (class_id = 86, version = 0)

Une instance de l'IC "PRIME NB OFDM PLC Applications identification" contient les informations d'identification relatives à l'administration et la maintenance des dispositifs PRIME NB OFDM PLC. Il ne s'agit pas de paramètres de communication, mais ils permettent néanmoins de gérer le dispositif.

PRIME NB OFDM PLC Application identification	0...n	class_id = 86, version = 0			
Attributs	Type de données	Min.	Max.	Def.	Nom court
1. logical_name (static)	octet-string				x
2. firmware_version (static)	octet-string		128		x + 0x08
3. vendor_Id (static)	long-unsigned				x + 0x10
4. product_Id (static)	long-unsigned				x + 0x18
Méthodes spécifiques	m/o				

Description d'attribut

logical_name Identifie l'instance d'objet d'établissement de dispositif. Voir 6.2.24.

firmware_version Attribut PIB 0x0075: Description textuelle de la version du firmware s'exécutant sur le dispositif.

vendor_Id Attribut PIB 0x0076: Identifiant unique du fournisseur attribué par PRIME Alliance.

product_Id Attribut PIB 0x0077: Identifiant unique attribué par le fournisseur pour un produit spécifique.

5.11 Classes d'interfaces pour l'établissement et la gestion du profil PLC OFDM à bande étroite DLMS/COSEM pour les réseaux G3-PLC

5.11.1 Vue d'ensemble

Le présent paragraphe 5.11 spécifie la version 1 des classes d'interfaces pour l'établissement et la gestion des couches MAC et d'adaptation 6LoWPAN du profil DLMS/COSEM G3-PLC, conformément à l'ITU-T G.9903:2014.

L'utilisation de la version 0 de ces classes d'interfaces basées sur l'ITU-T G.9903 Amd. 1:2013 – voir 7.18 à 7.20 – est obsolète.

A cet effet, les éléments de la PIB (PAN Information Base – base d'informations de PAN) ont été mis en correspondance avec les attributs des IC COSEM.

Les objets COSEM pour l'échange de données utilisant le profil G3 PLC, s'il est mis en œuvre, doivent se trouver dans le dispositif logique de gestion des serveurs COSEM.

Pour établir et gérer les couches de profil DLMS/COSEM G3-PLC (y compris PHY, IEEE 802.15.4:2006 MAC et 6LoWPAN), trois IC sont spécifiées:

- "G3 NB OFDM PLC MAC layer counters", voir 5.11.3;
- "G3-PLC MAC setup", voir 5.11.4;
- "G3-PLC 6LoWPAN adaptation layer setup", voir 5.11.5.

Une instance de la classe d'interfaces COSEM existante "MAC address" (class_id: 43, version: 0) est nécessaire pour indiquer l'adresse MAC EUI-48 du modem G3 PLC (correspondant à la constante aExtendedAddress dans l'IEEE 802.15.4:2006).

La configuration IPv6 est fournie par une instance de la classe "IPv6 setup".

NOTE La couche PHY de l'ITU-T G.9903:2014 est hors du domaine d'application des IC "G3-PLC setup".

5.11.2 Mise en correspondance des attributs G3-PLC PIB avec les attributs d'IC COSEM

Selon les termes de l'IEEE 802.15.4:2006, un compteur est un dispositif aux fonctions réduites (RFD) alors qu'un concentrateur/point d'accès aux réseaux avoisinants (NNAP – Neighbourhood Network Access Point) est un dispositif pleine fonction (FFD)/coordonateur PAN. Selon les termes du DLMS/COSEM, le compteur est le serveur, et le concentrateur/NNAP est le client (ou un agent pour le compte du client).

Le COSEM modélisant uniquement le serveur et pas le client, les classes de configuration G3-PLC concernent uniquement le RFD (Reduced Function Device) et pas le coordonnateur PAN.

Le Tableau 17 présente la mise en correspondance des attributs G3-PLC PIB avec les attributs des classes d'interfaces COSEM.

Tableau 17 – Mise en correspondance des attributs G3-PLC IB avec les attributs d'IC COSEM

Nom	Identifiant	Classe d'interfaces	class_id/attribut
Compteurs MAC – Attributs PIB en lecture seule donnant des informations statistiques ¹			
mac_Tx_data_packet_count	0x0101	G3-PLC MAC layer counters (class_id = 90, version = 1)	90 / Att. 2
mac_Rx_data_packet_count	0x0102		90 / Att. 3
mac_Tx_cmd_packet_count	0x0103		90 / Att. 4
mac_Rx_cmd_packet_count	0x0104		90 / Att. 5
mac_CSMA_fail_count	0x0105		90 / Att. 6
mac_CSMA_no_ACK_count	0x0106		90 / Att. 7
mac_bad_CRC_count	0x0109		90 / Att. 8
mac_Tx_data_broadcast_count	0x0108		90 / Att. 9
mac_Rx_data_broadcast_count	0x0107		90 / Att. 10
Attributs PIB de définition MAC – Variables en lecture seule, en lecture-écriture et en écriture seule ^{1 2}			
mac_short_address	0x0053	G3-PLC MAC setup (class_id = 91, version = 1)	91 / Att. 2
mac_RC_coord	0x010F		91 / Att. 3
mac_PAN_id	0x0050		91 / Att. 4
mac_key_table	0x0071		91 / Att. 5
mac_frame_counter	0x0077		91 / Att. 6
mac_tone_mask	0x0110		91 / Att. 7
mac_TMR_TTL	0x010D		91 / Att. 8
mac_max_frame_retries	0x0059		91 / Att. 9
mac_neighbour_table_entry_TTL	0x010E		91 / Att. 10
mac_neighbour_table	0x010A		91 / Att. 11
mac_high_priority_window_size	0x0100		91 / Att. 12
mac_CSMA_fairness_limit	0x010C		91 / Att. 13
mac_beacon_randomization_window_length	0x0111		91 / Att. 14
mac_A	0x0112		91 / Att. 15
mac_K	0x0113		91 / Att. 16
mac_min_CW_attempts	0x0114		91 / Att. 17
mac_cenelec_legacy_mode	0x0115		91 / Att. 18
mac_FCC_legacy_mode	0x0116		91 / Att. 19
mac_max_BE	0x0047		91 / Att. 20

Nom	Identifiant	Classe d'interfaces	class_id/attribute
mac_max_CSMA_backoffs	0x004E		91 / Att. 21
mac_min_BE	0x004F		91 / Att. 22
Attributs IB de couche d'adaptation 6LoWPAN – Variables en lecture seule et en lecture-écriture ^{3 4}			
adp_max_hops	0x0F	G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 1)	92 / Att. 2
adp_weak_LQI_value	0x1A		92 / Att. 3
adp_security_level	0x00		92 / Att. 4
adp_prefix_table	0x01		92 / Att. 5
adp_routing_configuration	0x09, 0x0A, 0x0D, 0x11-0x19, 0x1B, 0x1F		92 / Att. 6
adp_broadcast_log_table_entry_TTL	0x02		92 / Att. 7
adp_routing_table	0x0C		92 / Att. 8
adp_context_information_table	0x07		92 / Att. 9
adp_blacklist_table	0x1E		92 / Att. 10
adp_broadcast_log_table	0x0B		92 / Att. 11
adp_group_table	0x0E		92 / Att. 12
adp_max_join_wait_time	0x20		92 / Att. 13
adp_path_discovery_time	0x21		92 / Att. 14
adp_active_key_index	0x22		92 / Att. 15
adp_metric_type	0x03		92 / Att. 16
adp_coord_short_address	0x08		92 / Att. 17
adp_disable_default_routing	0xF0		92 / Att. 18
adp_device_type	0x10		92 / Att. 19
1) Voir l'ITU-T G.9903:2014, 9.3.6.2.2 et 9.3.6.2.3. 2) Les attributs suivants des attributs IB de la sous-couche MAC G3-PLC ont été exclus car il n'était pas utile de les exposer: <i>macBSN</i>, <i>macDSN</i>, <i>macAckWaitDuration</i>, <i>macFreqNotching</i>, <i>macTimeStampSupported</i>, <i>macPromiscuousMode</i>, <i>macSecurityEnabled</i>. 3) Voir l'ITU-T G.9903:2014, 9.4.1.1. 4) Les attributs suivants des attributs IB de la sous-couche Adaptation G3-PLC ont été exclus car il n'était pas utile de les exposer: <i>adpSoftVersion</i>, <i>adpSnifferMode</i>.			
NOTE Alors que l'ITU-T G.9903:2014 utilise la notation camel, les spécifications des classes d'interfaces COSEM (et le présent tableau) utilisent la notation à trait de soulignement.			

5.11.3 G3-PLC MAC layer counters (class_id = 90, version = 1)

Une instance de l'IC "G3-PLC MAC layer counters" enregistre les compteurs liés aux échanges de couche MAC. L'objectif de ces compteurs est de donner des informations statistiques pour les besoins de la gestion.

Les attributs des instances de cette IC doivent être en lecture seule. Ils peuvent être réinitialisés à l'aide de la méthode reset.

G3-PLC MAC layer counters (compteurs de couche G3-PLC MAC)		0...n	class_id = 90, version = 1			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. mac_Tx_data_packet_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x08
3. mac_Rx_data_packet_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x10
4. mac_Tx_cmd_packet_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x18
5. mac_Rx_cmd_packet_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x20
6. mac_CSMA_fail_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x28
7. mac_CSMA_no_ACK_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x30
8. mac_bad_CRC_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x38
9. mac_Tx_data_broadcast_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x40
10. mac_Rx_data_broadcast_count	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x48
Méthodes spécifiques		m/o				
1. reset (data)		O				x + 0x50

Description d'attribut

NOTE Si un compteur atteint la valeur maximale (0xFFFFFFFF), il est automatiquement arrêté.

logical_name	Identifie l'instance d'objet "G3-PLC MAC layer counters". Voir 6.2.25.
mac_Tx_data_packet_count	Attribut PIB 0x0101: Compteur statistique des paquets de données dont la transmission a abouti (MSDU).
mac_Rx_data_packet_count	Attribut PIB 0x0102: Compteur statistique des paquets de données dont la réception a abouti (MSDU).
mac_Tx_cmd_packet_count	Attribut PIB 0x0103: Compteur statistique des paquets de commande dont la transmission a abouti.
mac_Rx_cmd_packet_count	Attribut PIB 0x0104: Compteur statistique des paquets de commande dont la réception a abouti.
mac_CSMA_fail_count	Attribut PIB 0x0105: Compte le nombre de fois que des reculs CSMA atteignent macMaxCSMABackoffs.
mac_CSMA_no_ACK_count	Attribut PIB 0x0106: Compte le nombre de fois qu'un acquittement (ACK) n'est pas reçu lors de la transmission d'une trame de données monodiffusion (la perte de l'ACK est attribuée à des collisions).
mac_bad_CRC_count	Attribut PIB 0x0109: Compteur statistique du nombre de trames reçues avec un CRC incorrect.
mac_Tx_data_broadcast_count	Attribut PIB 0x0108: Compteur statistique du nombre de trames de diffusion envoyées.
mac_Rx_data_broadcast_count	Attribut PIB 0x0107: Compteur statistique des paquets de diffusion dont la réception a abouti.

Description de la méthode

reset (data)	Cette méthode force une réinitialisation de l'objet. En appelant cette méthode, la valeur 0 est attribuée à tous les compteurs. data ::= integer(0)

5.11.4 G3-PLC MAC setup (class_id = 91, version = 1)

Une instance de l'IC "G3-PLC MAC setup" contient les paramètres nécessaires à l'établissement et la gestion de la sous-couche MAC G3-PLC IEEE 802.15.4:2006.

Ces attributs influencent le comportement fonctionnel d'une mise en œuvre. Les mises en œuvre peuvent permettre de modifier les attributs pendant le fonctionnement normal, c'est-à-dire même si la séquence de démarrage du dispositif a été exécutée.

G3-PLC MAC setup (établissement de G3-PLC MAC)		0...n	class_id = 91, version = 1			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				x
2. mac_short_address	(dyn.)	long-unsigned	0x0000	0xFFFF	0xFFFF	x + 0x08
3. mac_RC_coord	(dyn.)	long-unsigned	0x0000	0xFFFF	0xFFFF	x + 0x10
4. mac_PAN_id	(dyn.)	long-unsigned	0x0000	0xFFFF	0xFFFF	x + 0x18
5. mac_key_table	(dyn.)	array				x + 0x20
6. mac_frame_counter	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x28
7. mac_tone_mask	(static)	bit-string			0x00000 0000FF FFFFFF F	x + 0x30
8. mac_TMR_TTL	(static)	unsigned	0	255	2	x + 0x38
9. mac_max_frame_retries	(static)	unsigned	0	10	5	x + 0x40
10. mac_neighbour_table_entry_TTL	(static)	unsigned	0	255	255	x + 0x48
11. mac_neighbour_table	(dyn.)	array				x + 0x50
12. mac_high_priority_window_size	(static)	unsigned	1	7	7	x + 0x58
13. mac_CSMA_fairness_limit	(static)	unsigned	Voir ci-dessous	255	25	x + 0x60
14. mac_beacon_randomization_window_length	(static)	unsigned	1	254	12	x + 0x68
15. mac_A	(static)	unsigned	3	20	8	x + 0x70
16. mac_K	(static)	unsigned	1	Voir ci-dessous	5	x + 0x78
17. mac_min_CW_attempts	(static)	unsigned	0	255	10	x + 0x80
18. mac_cenelec_legacy_mode	(static)	unsigned	0	255	1	x + 0x88
19. mac_FCC_legacy_mode	(static)	unsigned	0	255	1	x + 0x90
20. mac_max_BE	(static)	unsigned	0	20	8	x + 0x98
21. mac_max_CSMA_backoffs	(static)	unsigned	0	255	50	x + 0xA0
22. mac_min_BE	(static)	unsigned	0	20	3	x + 0xA8
Méthodes spécifiques		m/o				
1. mac_get_neighbour_table_entry(data)		o				x + 0xB0

Description d'attribut					
logical_name	Identifie l'instance d'objet "G3-PLC MAC setup". Voir 6.2.25.				
mac_short_address	Attribut PIB 0x0053: Adresse de 16 bits que le dispositif utilise pour communiquer avec le réseau PAN. Sa valeur doit être égale à 0xFFFF lorsque le dispositif ne dispose d'aucune adresse courte. Un dispositif associé dispose nécessairement d'une adresse courte, de sorte qu'il ne puisse pas être associé sans adresse courte.				
mac_RC_coord	Attribut PIB 0x010F: Coût de liaison vers le coordinateur, à utiliser dans la charge utile d'étiquette en tant que RC_COORD				
mac_PAN_id	Attribut PIB 0x0050: Il s'agit de l'identifiant à 16 bits du réseau PAN sur lequel fonctionne le dispositif. Une valeur égale à 0xFFFF indique que le dispositif n'est pas associé.				
mac_key_table	Attribut PIB 0x0071: Cet attribut contient les clés GMK exigées pour le chiffrement de la couche MAC. L'attribut peut contenir jusqu'à deux clés de 16 octets. La valeur de l'identifiant de clé doit être différente pour chaque clé. Pour des raisons de sécurité, les entrées de clé ne peuvent pas être lues, mais uniquement écrites. <pre>array mac_GMK mac_GMK ::= structure { key_id: unsigned, key: octet-string }</pre> <table> <tr> <td>key_id</td><td>Identifiant de clé utilisé pour faire référence à cette clé, peut prendre la valeur 0 ou 1.</td></tr> <tr> <td>key</td><td>Clé AES-128 key utilisée pour le chiffrement des trames échangées au niveau de la couche MAC.</td></tr> </table>	key_id	Identifiant de clé utilisé pour faire référence à cette clé, peut prendre la valeur 0 ou 1.	key	Clé AES-128 key utilisée pour le chiffrement des trames échangées au niveau de la couche MAC.
key_id	Identifiant de clé utilisé pour faire référence à cette clé, peut prendre la valeur 0 ou 1.				
key	Clé AES-128 key utilisée pour le chiffrement des trames échangées au niveau de la couche MAC.				
mac_frame_counter	Attribut PIB 0x0077: Compteur de trames sortantes pour ce dispositif, utilisé lors du chiffrement des trames au niveau de la couche MAC.				
mac_tone_mask	Attribut PIB 0x0110: Définit le masque de tonalité à utiliser lors de la formation du symbole.				
mac_TMR_TTL	Attribut PIB 0x010D: Durée de vie maximale de l'entrée des paramètres de mise en correspondance de tonalité dans la table de voisinage, en minutes.				
mac_max_frame_retries	Attribut PIB 0x0059: Nombre maximal de retransmissions				
mac_neighbour_table_entry_TTL	Attribut PIB 0x010E: Durée de vie maximale d'une entrée dans la table de voisinage, en minutes.				

mac_neighbour_table

Attribut PIB 0x010A: Voir l'ITU-T G.9903:2014, 9.3.7.2 pour CENELEC et les bandes FCC.

La table de voisinage contient des informations relatives à tous les dispositifs à l'intérieur du POS du dispositif. Un élément de la table représente un voisin direct PLC du dispositif.

```
array neighbour_table
neighbour_table::= structure
{
    short_address: long-unsigned,
    payload_modulation_scheme: boolean,
    tone_map: bit-string,
    modulation: enum,
    tx_gain: integer,
    tx_res: enum,
    tx_coeff: bit-string,
    lqi: unsigned,
    phase_differential: integer,
    TMR_valid_time: unsigned,
    neighbour_valid_time: unsigned
}
```

NOTE 1 Cette table est mise à jour à chaque fois qu'une trame est reçue d'un dispositif avoisinant et à chaque fois qu'une réponse de mise en correspondance de tonalité est reçue.

short_address	Adresse courte MAC du nœud auquel cette entrée fait référence.

payload_modulation_scheme	Schéma de modulation de charge utile à utiliser lors de la transmission vers ce voisin. FALSE: Différentiel, TRUE: Cohérent

tone_map	Le paramètre Tone Map définit la sous-bande de fréquence qui peut être utilisée pour communiquer avec le dispositif. Une valeur 1 attribuée à un bit signifie que la sous-bande de fréquence peut être utilisée, une valeur 0 signifiant que la sous-bande de fréquence ne doit pas être utilisée.

modulation	Type de modulation à utiliser pour communiquer avec le dispositif. enum: (0) Mode robuste, (1) DBPSK, (2) DQPSK, (3) D8PSK, (4) MAQ 16

NOTE 2 La modulation 16-QAM est facultative et s'applique uniquement pour la bande FCC.

tx_gain	Définit le gain de l'émetteur à utiliser pour émettre des trames vers ce dispositif.

	tx_res	Définit la résolution du gain de l'émetteur correspondant à un échelon de gain. 0: 6 dB, 1: 3 dB
mac_neighbour_table (suite)	tx_coeff	Paramètre qui spécifie le gain de l'émetteur pour chaque groupe de tonalités représenté par un bit valide de mise en correspondance de tonalité. Le récepteur mesure l'affaiblissement relatif à la fréquence du canal et peut demander à l'émetteur de le compenser en augmentant la puissance sur des parties du spectre confrontées à l'affaiblissement, afin d'égaliser le signal reçu. Chaque groupe de tonalités est mis en correspondance avec une valeur de 4 bits (CENELEC-A) ou de 2 bits (FCC). La valeur 0 du bit de poids fort indique une valeur de gain positive, une augmentation du gain de l'émetteur mis à l'échelle par TXRES étant donc demandée pour cette section. Une valeur 1 indique une valeur de gain négative, une diminution du gain de l'émetteur mis à l'échelle par TXRES étant donc demandée pour cette section. La mise en œuvre de cette fonction est facultative et est destinée aux canaux à sélection de fréquence. Si cette fonction n'est pas mise en œuvre, la valeur zéro doit être utilisée. <i>Exemple: dans les bandes CENELEC, avec des valeurs de gain égales à [1, -5, 4, -2, 0, 1], le codage de tx_coeff est égal à: '0001 1101 0100 1010 0000 0001'</i> NOTE 3 Un groupe de tonalités rassemble 6 tonalités (ou porteuses) consécutives pour les bandes CENELEC, et 3 tonalités consécutives pour les bandes FCC.
	lqi	Indicateur qualité de liaison de la liaison vers le voisin (LQI inverse) NOTE 4 Valeur LQI mesurée lors de la réception de la PPDU provenant du voisin. La mesure LQI caractérise la puissance et/ou la qualité d'un paquet reçu.
	phase_differential	Déphasage en multiples de 60 degrés entre la phase du secteur du noeud local et le noeud voisin. PhaseDifferential peut supposer six valeurs entières comprises entre 0 et 5.

TMR_valid_time	Temps restant, en minutes, pendant lequel les paramètres de réponse de mise en correspondance de tonalité dans la table de voisinage sont considérés comme étant valides. – Lors de la création de l'entrée, la valeur par défaut 0 doit être attribuée à cette valeur. – À 0, une requête de mise en correspondance de tonalité peut être formulée si des données sont envoyées à ce dispositif. Dès la réception d'une réponse de mise en correspondance de tonalité, cette valeur est définie sur mac_TMR_TTL.
neighbour_valid_time	Temps restant, en minutes, pendant lequel cette entrée dans la table de voisinage est considérée comme étant valide. A chaque fois qu'une entrée est créée ou qu'une trame (données ou ACK) est reçue de ce voisin, la valeur mac_neighbour_table_entry_TTL lui est attribuée. À zéro, cette entrée n'est plus valide dans la table et peut être supprimée.
mac_high_priority_window_size	Attribut PIB 0x0100: Taille de la fenêtre de contention à priorité élevée en nombre d'intervalles.
mac_CSMA_fairness_limit	Attribut PIB 0x010C: Limite de fidélité d'accès au canal. Spécifie le nombre de tentatives de repli qui n'ont pas abouti, l'exposant de repli étant défini sur minBE. Une valeur comprise entre $2 \times (\text{macMaxBE} - \text{macMinBE})$ et 255 peut être affectée à cet attribut.
mac_beacon_randomization_window_length	Attribut PIB 0x0111: Durée, en secondes, de randomisation de l'étiquette.
mac_A	Attribut PIB 0x0112: Ce paramètre commande la diminution linéaire CW active.
mac_K	Attribut PIB 0x0113: Facteur d'adaptation au débit binaire pour la limite de fidélité d'accès au canal. Une valeur comprise entre 1 et macCSMAFairnessLimit peut être affectée à cet attribut.
mac_min_CW_attempts	Attribut PIB 0x0114: Nombre de tentatives consécutives lors de l'utilisation de CW minimal.
mac_cenelec_legacy_mode	Attribut PIB 0x0115: Cet attribut en lecture seule indique la capacité du noeud. 0: La configuration suivante est utilisée (mode existant): – Entrelaçage élémentaire; – Les paramètres d'entrelaceur n_i et n_j ne sont pas permutés lorsque $I(i,j) = 0$. 1: La configuration suivante est utilisée (mode non existant): – entrelaçage de bloc complet; – Les paramètres d'entrelaceur n_i et n_j sont permutés lorsque $I(i,j) = 0$.

mac_FCC_legacy_mode	Attribut PIB 0x0116: Cet attribut en lecture seule indique la capacité du noeud. 0: La configuration suivante est utilisée (mode existant): – Modulation FCH différentielle; – Entrelaçage élémentaire; – Les paramètres d'entrelaceur n_i et n_j ne sont pas permutés lorsque $l(i,j) = 0$; – Bloc RS simple. 1: La configuration suivante est utilisée (mode non existant): – Modulation FCH cohérente; – entrelaçage de bloc complet; – Les paramètres d'entrelaceur n_i et n_j sont permutés lorsque $l(i,j) = 0$; – Deux blocs RS.
mac_max_BE	Attribut PIB 0x0047: Valeur maximale de l'exposant de repli. Il convient qu'il soit toujours supérieur à mac_min_BE .
mac_max_CSMA_backoffs	Attribut PIB 0x004E: Nombre maximal de tentatives de repli.
mac_min_BE	Attribut PIB 0x004F: Valeur minimale de l'exposant de repli.
Description de la méthode	
mac_get_neighbour_table_entry (data)	Cette méthode est utilisée pour récupérer la table de voisinage MAC correspondant à une adresse courte MAC. Elle peut être utilisée pour permettre au client de surveiller la topologie. Le paramètre d'appel de méthode contient une <i>mac_short_address</i>. <i>data</i>::= long-unsigned Le paramètre de réponse inclut la table de voisinage pour cette <i>mac_short_address</i>. <i>data</i>::= array neighbour_table où <i>mac_neighbour_table</i> est tel que défini dans l'attribut <i>mac_neighbour_table</i> de la présente IC.

5.11.5 G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 1)

Une instance de l'IC "G3-PLC 6LoWPAN adaptation layer setup" contient les paramètres nécessaires à l'établissement et la gestion de la couche d'adaptation G3-PLC 6LoWPAN.

Ces attributs influencent le comportement fonctionnel d'une mise en œuvre. Les mises en œuvre peuvent permettre de modifier leurs valeurs pendant le fonctionnement normal, c'est-à-dire même si la séquence de démarrage du dispositif a été exécutée.

G3-PLC 6LoWPAN adaptation layer setup (établissement de la couche d'adaptation G3-PLC 6LoWPAN)		0...n	class_id = 92, version = 1			
Attributs		Type de données	Min.	Max.	Def.	Nom court
1. logical_name	(static)	octet-string				X
2. adp_max_hops	(static)	unsigned	1	14	8	x + 0x08
3. adp_weak_LQI_value	(static)	unsigned	0	255	52	x + 0x10
4. adp_security_level	(static)	unsigned	0	7	5	x + 0x18
5. adp_prefix_table	(dyn.)	array				x + 0x20
6. adp_routing_configuration	(static)	array				x + 0x28
7. adp_broadcast_log_table_entry_TTL	(static)	long-unsigned	0	65535	2	x + 0x30
8. adp_routing_table	(dyn.)	array				x + 0x38
9. adp_context_information_table	(dyn)	array				x + 0x40
10. adp_blacklist_table	(dyn)	array				x + 0x48
11. adp_broadcast_log_table	(dyn)	array				x + 0x50
12. adp_group_table	(dyn)	array				x + 0x58
13. adp_max_join_wait_time	(static)	long-unsigned	0	1 023	20	x + 0x60
14. adp_path_discovery_time	(static)	unsigned	0	255	40	x + 0x68
15. adp_active_key_index	(static)	unsigned	0	1	0	x + 0x70
16. adp_metric_type	(static)	unsigned	0x00	0x0F	0x0F	x + 0x78
17. adp_coord_short_address	(static)	long-unsigned	0x00 00	0x7FFF	0x0000	x + 0x80
18. adp_disable_default_routing	(static)	boolean			FALSE	x + 0x88
19. adp_device type	(static)	enum	0	2	2	x + 0x90
Méthodes spécifiques		m/o				

Description d'attribut

logical_name	Identifie l'instance d'objet "G3-PLC OFDM 6LoWPAN adaptation layer setup". Voir 6.2.25.
adp_max_hops	Attribut PIB 0x0F: Définit le nombre maximal de sauts à utiliser par l'algorithme d'acheminement.
adp_weak_LQI_value	Attribut PIB 0x1A: La valeur de liaison faible définit la valeur LQI en deçà de laquelle une liaison vers un voisin est considérée comme étant une liaison faible. Une valeur de 52 représente un SNR de 3 dB.
adp_security_level	Attribut PIB 0x00: Niveau de sécurité minimal à utiliser pour les trames d'adaptation entrantes et sortantes. Seules la valeur 0 (pas de chiffrement) et la valeur 5 (chiffrement avec un code d'intégrité de 32 bits) sont prises en charge.
adp_prefix_table	Attribut PIB 0x01: Contient la liste des préfixes définis pour ce réseau PAN.

adp_routing_configuration L'élément de configuration d'acheminement spécifie tous les paramètres liés au mécanisme d'acheminement décrits dans l'ITU-T G.9903:2014. Les éléments sont spécifiés en 9.4.1.2 de cette Recommandation.

NOTE 1 Le calcul du coût de liaison est présenté dans l'ITU-T G.9903:2014, Annexe B.

array routing_configuration

routing_configuration ::= structure

```
{
    adp_net_traversal_time:      unsigned,
    adp_routing_table_entry_TTL: long-unsigned,
    adp_Kr:                      unsigned,
    adp_Km:                      unsigned,
    adp_Kc:                      unsigned,
    adp_Kq:                      unsigned,
    adp_Kh:                      unsigned,
    adp_Krt:                     unsigned,
    adp_RREQ_retries:            unsigned,
    adp_RREQ_RERR_wait:         unsigned,
    adp_blacklist_table_entry_TTL: long-unsigned,
    adp_unicast_RREQ_gen_enable: boolean,
    adp_RLC_Enabled:            boolean,
    adp_add_rev_link_cost:      unsigned
}
```

adp_net_traversal_time	Attribut PIB 0x11: Durée maximale qu'un paquet est censé mettre pour atteindre un noeud à partir d'un autre noeud, en secondes.
	Plage : 0-255
	Valeur par défaut : 20

adp_routing_table_entry_TTL	Attribut PIB 0x12: Durée de vie maximale d'une entrée de la table d'acheminement (en minutes).
	Plage : 0-65 535
	Valeur par défaut : 60

adp_Kr	Attribut PIB 0x13: Facteur de pondération du Mode robuste pour calculer le coût de liaison.
	Plage : 0-31
	Valeur par défaut : 0

adp_Km	Attribut PIB 0x14: Facteur de pondération de modulation pour calculer le coût de liaison.
	Plage : 0-31
	Valeur par défaut : 0

adp_Kc	Attribut PIB 0x15: Facteur de pondération du nombre de tonalités actives pour calculer le coût de liaison.
	Plage : 0-31
	Valeur par défaut : 0

adp_routing_configuration (suite)	adp_Kq	Attribut PIB 0x16: Facteur de pondération de LQI pour calculer le coût de liaison. Plage : 0-50 Valeur par défaut : 10 pour la bande CENELEC A/40 pour la bande FCC.
	adp_Kh	Attribut PIB 0x17: Facteur de pondération du saut pour calculer le coût de liaison. Plage : 0-31 Valeur par défaut : 4 pour la bande CENELEC A/2 pour la bande FCC.
	adp_Krt	Attribut PIB 0x1B: Facteur de pondération du nombre de chemins actifs dans la table d'acheminement pour calculer le coût de liaison. Plage : 0-31 Valeur par défaut : 0
	adp_RREQ_retries	Attribut PIB 0x18: Nombre de retransmissions RREQ en cas d'expiration du délai de réception RREP. Plage : 0-255 Valeur par défaut : 0
	adp_RREQ_RERR_wait	Attribut PIB 0x19: Nombre de secondes d'attente entre deux générations consécutives de RREQ – RERR. Plage : 0-255 Valeur par défaut : 30
	adp_blacklist_table_entry_TTL	Attribut PIB 0x1F: Durée de vie maximale d'une entrée d'un ensemble de voisins sur liste noire (en minutes). Plage : 0-65 535 Valeur par défaut : 10
	adp_unicast_RREQ_gen_enable	Attribut PIB 0x0D: Si la valeur est TRUE, le RREQ doit être généré en attribuant la valeur 1 à son fanion "unicast RREQ". Si la valeur est FALSE, le RREQ doit être généré en attribuant la valeur 0 à son fanion "unicast RREQ". Valeur par défaut : TRUE
	adp_RLC_Activé	Attribut PIB 0x09: Active l'envoi de la trame RLCREQ par le dispositif. Valeur par défaut : FALSE
	adp_add_rev_link_cost	Attribut PIB 0x0A: Représente un coût supplémentaire pour prendre en compte une éventuelle asymétrie dans la liaison. Plage : 0-255 Valeur par défaut : 0

adp_broadcast_log_table_entry_TTL	Attribut PIB 0x02: Durée de vie maximale d'une entrée adpBroadcastLogTable (en minutes).																				
adp_routing_table	Attribut PIB 0x0C: Contient la table d'acheminement. array routing_table routing_table::= structure <pre>{ destination_address: long-unsigned, next_hop_address: long-unsigned, route_cost: long-unsigned, hop_count: unsigned, weak_link_count: unsigned, valid_time: long-unsigned }</pre> NOTE 2 Cette table est actualisée à chaque fois qu'un chemin est conçu ou mis à jour (déclenché par le trafic de données) et à chaque fois que le temporisateur TTL expire. <table> <tr> <td>destination_address</td><td>Adresse de la destination.</td></tr> <tr> <td>next_hop_address</td><td>Adresse du prochain saut sur le chemin vers la destination.</td></tr> <tr> <td>route_cost</td><td>Coût de liaison cumulé sur le chemin vers la destination.</td></tr> <tr> <td>hop_count</td><td>Nombre de sauts du chemin sélectionné vers la destination.</td></tr> <tr> <td></td><td>Plage: 0-14</td></tr> <tr> <td></td><td>NOTE 3 Dans la pratique, la valeur admise maximale est limitée par adp_max_hops.</td></tr> <tr> <td>weak_link_count</td><td>Nombre de liaisons faibles vers la destination.</td></tr> <tr> <td></td><td>Plage : 0-14</td></tr> <tr> <td></td><td>NOTE 4 Dans la pratique, la valeur admise maximale est limitée par adp_max_hops.</td></tr> <tr> <td>valid_time</td><td>Temps restant, en minutes, jusqu'à ce que cette entrée de la table d'acheminement soit considérée comme étant valide.</td></tr> </table>	destination_address	Adresse de la destination.	next_hop_address	Adresse du prochain saut sur le chemin vers la destination.	route_cost	Coût de liaison cumulé sur le chemin vers la destination.	hop_count	Nombre de sauts du chemin sélectionné vers la destination.		Plage: 0-14		NOTE 3 Dans la pratique, la valeur admise maximale est limitée par adp_max_hops.	weak_link_count	Nombre de liaisons faibles vers la destination.		Plage : 0-14		NOTE 4 Dans la pratique, la valeur admise maximale est limitée par adp_max_hops.	valid_time	Temps restant, en minutes, jusqu'à ce que cette entrée de la table d'acheminement soit considérée comme étant valide.
destination_address	Adresse de la destination.																				
next_hop_address	Adresse du prochain saut sur le chemin vers la destination.																				
route_cost	Coût de liaison cumulé sur le chemin vers la destination.																				
hop_count	Nombre de sauts du chemin sélectionné vers la destination.																				
	Plage: 0-14																				
	NOTE 3 Dans la pratique, la valeur admise maximale est limitée par adp_max_hops.																				
weak_link_count	Nombre de liaisons faibles vers la destination.																				
	Plage : 0-14																				
	NOTE 4 Dans la pratique, la valeur admise maximale est limitée par adp_max_hops.																				
valid_time	Temps restant, en minutes, jusqu'à ce que cette entrée de la table d'acheminement soit considérée comme étant valide.																				
adp_context_information_table	Attribut PIB 0x07: Contient les informations de contexte associées à chaque champ d'extension CID. array context_information_table context_information_table::= structure <pre>{ CID: bit-string, context_length: unsigned, context: octet-string, C: boolean, valid_lifetime: long-unsigned }</pre> <table> <tr> <td>CID</td><td>Correspond aux informations de contexte à 4 bits utilisées pour les adresses source et de destination (SCI, DCI).</td></tr> <tr> <td></td><td>Plage: 0x00-0x0F</td></tr> </table>	CID	Correspond aux informations de contexte à 4 bits utilisées pour les adresses source et de destination (SCI, DCI).		Plage: 0x00-0x0F																
CID	Correspond aux informations de contexte à 4 bits utilisées pour les adresses source et de destination (SCI, DCI).																				
	Plage: 0x00-0x0F																				

	context_length	Indique la longueur du contexte transporté (des contextes jusqu'à 128 bits peuvent être transportés). Plage: 0-128
adp_context_information_table (suite)	context	Correspond au contexte transporté utilisé pour les besoins de la compression/décompression. Plage: 0x0000:0000:0000:0000:0000:0000:0000 – 0xFFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF
	C	Indique si le contexte est valide pour être utilisé dans la compression. FALSE: Seule la décompression est admise, TRUE: La compression et la décompression sont admises Un contexte peut être utilisé uniquement pour les besoins de la décompression. De plus, il convient de se conformer aux recommandations formulées dans le RFC 6775 afin de tenir compte de la propagation du contexte vers tous les nœuds du réseau PAN.
	valid_lifetime	Temps restant, en minutes, au cours duquel la table d'informations contextuelles est considérée comme étant valide. Il est mis à jour dès la réception du contexte conseillé. Plage: 0-65 535
adp_blacklist_table	Attribut PIB 0x1E: Contient la liste des voisins placés sur la liste noire.	
	array blacklisted_neighbour_set	
	blacklisted_neighbour_set::= structure	
	<pre> { blacklisted_neighbour_address: long-unsigned, valid_time: long-unsigned } </pre>	
	blacklisted_neighbour_address	Adresse à 16 bits du voisin placé sur la liste noire.
	valid_time	Temps restant, en minutes, jusqu'à ce que cette entrée dans la table de voisinage de liste noire soit considérée comme étant valide.

adp_broadcast_log_table	Attribut PIB 0x0B: Contient la table de journal de diffusion. NOTE 5 Cette table fournit une liste de paquets de diffusion récemment reçus par le dispositif. array broadcast_log_table broadcast_log_table::= structure <pre>{ source_address: long-unsigned, sequence_number: unsigned, valid_time: long-unsigned }</pre> <table> <tr> <td>source_address</td><td>Adresse source à 16 bits d'un paquet de diffusion. Il s'agit de l'adresse de l'initiateur de la diffusion.</td></tr> <tr> <td>sequence_number</td><td>Numéro de séquence contenu dans l'en-tête BC0.</td></tr> <tr> <td>valid_time</td><td>Temps restant, en minutes, jusqu'à ce que cette entrée de la table de journal de diffusion soit considérée comme étant valide.</td></tr> </table>	source_address	Adresse source à 16 bits d'un paquet de diffusion. Il s'agit de l'adresse de l'initiateur de la diffusion.	sequence_number	Numéro de séquence contenu dans l'en-tête BC0.	valid_time	Temps restant, en minutes, jusqu'à ce que cette entrée de la table de journal de diffusion soit considérée comme étant valide.
source_address	Adresse source à 16 bits d'un paquet de diffusion. Il s'agit de l'adresse de l'initiateur de la diffusion.						
sequence_number	Numéro de séquence contenu dans l'en-tête BC0.						
valid_time	Temps restant, en minutes, jusqu'à ce que cette entrée de la table de journal de diffusion soit considérée comme étant valide.						
adp_group_table	Attribut PIB 0x0E: Contient les adresses de groupe auxquelles le dispositif appartient. array group_address group_address::= long-unsigned group_address: Adresse de groupe à laquelle ce nœud a été abonné.						
adp_max_join_wait_time	Attribut PIB 0x20: Délai d'expiration d'accès au réseau, en secondes, pour LBD.						
adp_path_discovery_time	Attribut PIB 0x21: Délai d'expiration pour la reconnaissance de chemin d'accès, en secondes.						
adp_active_key_index	Attribut PIB 0x22: Index du GMK actif à utiliser pour la transmission de données						
adp_metric_type	Attribut PIB 0x03: Type de métrique à utiliser pour les besoins de l'acheminement						
adp_coord_short_address	Attribut PIB 0x08: Définit l'adresse courte du coordinateur.						
adp_disable_default_routing	Attribut PIB 0xF0: Si TRUE, l'acheminement par défaut (LOADng) est désactivé. Si FALSE, l'acheminement par défaut (LOADng) est activé.						
adp_device_type	Attribut PIB 0x10: Définit le type du dispositif connecté au modem: enum: (0) Dispositif PAN, (1) Coordinateur PAN, (2) Non défini						

5.12 Classes d'établissement ZigBee®

5.12.1 Vue d'ensemble

Le paragraphe 5.12 spécifie les classes d'interfaces COSEM exigées pour la configuration et la gestion externes d'un réseau ZigBee® afin d'assurer l'interfaçage avec une installation en plusieurs parties qui utilise en interne les communications ZigBee®. ZigBee® est une technologie de communication basse puissance et une norme ouverte exploitée par ZigBee® Alliance (voir www.zigbee.org).

ZigBee® est une marque déposée de ZigBee® Alliance.

NOTE 1 Une installation en plusieurs parties est une installation dans laquelle le compteur fournit des informations et/ou des services au nom de la régie d'électricité. Par exemple, le compteur interagit avec un écran domestique et/ou un interrupteur de commande de charge externe et/ou une application intelligente, afin d'informer le client de leur utilisation en temps réel, commander les dispositifs de chauffage et éventuellement déconnecter les pointes de charge lorsque l'alimentation est contrainte. Même s'il est possible pour le client de commander le réseau ZigBee®, en fonctionnement normal, la régie d'électricité commande le système radio. Il s'agit d'assurer la sécurité du réseau PAN, de sorte que les dispositifs ZigBee® (les commutateurs en charge commandés par la régie d'électricité, par exemple) fonctionnent en toute sécurité.

ZigBee® définit un réseau local de dispositifs associés par liaisons radio, acheminant et transférant les messages, et assurant le chiffrement pour la confidentialité au niveau du réseau.

NOTE 2 Ce type de réseau local est appelé réseau PAN (Personal Area Network). Le nom Zigbee® est utilisé dans la communauté ZigBee® et est étayé par la norme IEEE 802.15.4:2006, qui utilise le terme PAN. Cela est largement équivalent à un réseau HAN (Home Area Network – nom utilisé dans le contexte du comptage intelligent au Royaume-Uni) ou PAN.

Chaque réseau PAN est doté d'un dispositif appelé coordinateur ZigBee®, chargé de créer et de gérer le réseau, et qui agit en général également comme un tiers de confiance ZigBee® pour la gestion du réseau ZigBee®, des clés PCLK et des clés de liaison APS.

Il s'agit d'un processus de création de réseau PAN (et de destruction correspondante) réalisé par le coordinateur. Cela permet de déclarer l'existence du réseau sans l'aide de dispositifs, outre le coordinateur qui en fait partie intégrante. D'autres dispositifs ZigBee® peuvent rejoindre un réseau créé en coopération avec le coordinateur, et peuvent également choisir de le quitter ou être invités à le faire par le coordinateur (ce qui n'est pas actuellement exécutoire). En principe, les dispositifs sont membres du réseau de manière définitive. Ils ne le quittent pas ni ne le rejoignent de manière répétée. Pour créer un réseau PAN, le coordinateur est tenu de recevoir un déclencheur externe et détenir des informations de configuration, telles que:

- l'ID du réseau PAN étendu;
- les clés de liaison ou le code d'installation (pour la communication initiale avec les nouveaux dispositifs);
- des informations sur le canal de radiofréquences.

Pendant le processus de création du réseau PAN, le coordinateur analyse les dispositifs radio à proximité, "échange" les clés, choisit les adresses courtes et confirme l'utilisation des canaux de radiofréquences. Les détails des informations disponibles à partir des serveurs ZigBee® sur chaque dispositif sont également échangés.

NOTE 3 L'ensemble des détails du processus d'accès est présenté dans la spécification ZigBee® ZigBee® 053474. De plus amples informations relatives à la technologie ZigBee® sont disponibles à l'adresse suivante: <http://www.zigbee.org/>.

La Figure 18 présente un exemple d'architecture avec un hub Comms sur la gauche, composée d'un serveur DLMS/COSEM et du coordinateur ZigBee®. Le serveur DLMS/COSEM contient les objets d'interface nécessaires à la configuration et au contrôle du réseau ZigBee®. Il peut contenir d'autres objets COSEM permettant de prendre en charge une application de comptage. De plus, il existe deux dispositifs ZigBee® natifs et un autre serveur DLMS/COSEM (un compteur d'électricité ou un autre type de compteur) sur la droite, qui est également un dispositif de réseau ZigBee® "normal".