



13/1853/CDV

COMMITTEE DRAFT FOR VOTE (CDV)

PROJECT NUMBER:
IEC 62056-6-2 ED4

DATE OF CIRCULATION: **2021-12-03** CLOSING DATE FOR VOTING: **2022-02-25**

SUPERSEDES DOCUMENTS:
13/1826A/RR

IEC TC 13 : ELECTRICAL ENERGY MEASUREMENT AND CONTROL

SECRETARIAT: Hungary	SECRETARY: Mr Bela Bodí
OF INTEREST TO THE FOLLOWING COMMITTEES: TC 57	PROPOSED HORIZONTAL STANDARD: <input type="checkbox"/> Other TC/SCs are requested to indicate their interest, if any, in this CDV to the secretary.
FUNCTIONS CONCERNED:	
<input type="checkbox"/> EMC <input type="checkbox"/> ENVIRONMENT	
<input checked="" type="checkbox"/> SUBMITTED FOR CENELEC PARALLEL VOTING	
Attention IEC-CENELEC parallel voting The attention of IEC National Committees, members of CENELEC, is drawn to the fact that this Committee Draft for Vote (CDV) is submitted for parallel voting. The CENELEC members are invited to vote through the CENELEC online voting system.	

This document is still under study and subject to change. It should not be used for reference purposes.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

TITLE:

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

PROPOSED STABILITY DATE: 2024

NOTE FROM TC/SC OFFICERS:

Copyright © 2021 International Electrotechnical Commission, IEC. All rights reserved. It is permitted to download this electronic file, to make a copy and to print out the content for the sole purpose of preparing National Committee positions. You may not copy or "mirror" the file or printed version of the document, or any part of it, for any other purpose without permission in writing from IEC.

1	CONTENTS	2
2	FOREWORD	12
5	INTRODUCTION	14
6	1 Scope	16
7	2 Normative references	16
8	3 Terms, definitions and abbreviated terms	17
9	3.1 Terms and definitions related to the Image transfer process (see 4.4.6)	17
10	3.2 Terms and definitions related to the S-FSK PLC setup classes (see 4.10)	18
11	3.3 Terms and definitions related to the PRIME NB OFDM PLC setup ICs (see	
12	4.12)	19
13	3.5 Terms and definitions related to ZigBee® (see 4.15)	21
14	3.6 Terms and definitions related to Payment metering interface classes (see	
15	4.6)	23
16	3.7 Terms and definitions related to the Arbitrator IC (see 4.5.12)	27
17	3.8 Abbreviated terms	28
18	4 The COSEM interface classes	32
19	4.1 Basic principles	32
20	4.1.1 General	32
21	4.1.2 Referencing methods	33
22	4.1.3 Reserved base_names for special COSEM objects	34
23	4.1.4 Class description notation	34
24	4.1.5 Common data types	37
25	4.1.6 Data formats	40
26	4.1.7 The COSEM server model	45
27	4.1.8 The COSEM logical device	46
28	4.1.9 Information security	48
29	4.2 Overview of the COSEM interface classes	48
30	4.3 Interface classes for parameters and measurement data	55
31	4.3.1 Data (class_id = 1, version = 0)	55
32	4.3.2 Register (class_id = 3, version = 0)	56
33	4.3.3 Extended register (class_id = 4, version = 0)	60
34	4.3.4 Demand register (class_id = 5, version = 0)	61
35	4.3.5 Register activation (class_id = 6, version = 0)	66
36	4.3.6 Profile generic (class_id = 7, version = 1)	67
37	4.3.7 Utility tables (class_id = 26, version = 0)	73
38	4.3.8 Register table (class_id = 61, version = 0)	74
39	4.3.9 Status mapping (class_id = 63, version = 0)	76
40	4.3.10 Compact data (class_id: 62, version: 1)	78
41	4.4 Interface classes for access control and management	86
42	4.4.1 Overview	86
43	4.4.2 Client user identification	86
44	4.4.3 Association SN (class_id = 12, version = 4)	87
45	4.4.4 Association LN (class_id = 15, version = 3)	91
46	4.4.5 SAP assignment (class_id = 17, version = 0)	98
47	4.4.6 Image transfer (class_id = 18, version = 0)	99

48	4.4.7	Security setup (class_id = 64, version = 1)	107
49	4.4.8	Push interface classes and objects	114
50	4.4.9	COSEM data protection (class_id = 30, version = 0).....	127
51	4.4.10	Function control (class_id: 122, version: 0).....	143
52	4.4.11	Array manager (class_id = 123, version = 0).....	145
53	4.4.12	Communication port protection (class_id = 124, version = 0)	150
54	4.5	Interface classes for time- and event bound control	154
55	4.5.1	Clock (class_id = 8, version = 0).....	154
56	4.5.2	Script table (class_id = 9, version = 0)	158
57	4.5.3	Schedule (class_id = 10, version = 0)	159
58	4.5.4	Special days table (class_id = 11, version = 0)	163
59	4.5.5	Activity calendar (class_id = 20, version = 0)	164
60	4.5.6	Register monitor (class_id = 21, version = 0)	167
61	4.5.7	Single action schedule (class_id = 22, version = 0).....	168
62	4.5.8	Disconnect control (class_id = 70, version = 0).....	169
63	4.5.9	Limiter (class_id = 71, version = 0)	173
64	4.5.10	Parameter monitor (class_id = 65, version = 1)	175
65	4.5.11	Sensor manager (class_id = 67, version = 0)	179
66	4.5.12	Arbitrator (class_id = 68, version = 0)	184
67	4.5.13	Modelling examples: tariffication and billing	188
68	4.6	Payment metering related interface classes	190
69	4.6.1	Overview of the COSEM accounting model	190
70	4.6.2	Account (class_id = 111, version = 0)	192
71	4.6.3	Credit interface class (class_id = 112, version = 0)	202
72	4.6.4	Charge (class_id = 113, version = 0)	212
73	4.6.5	Token gateway (class_id = 115, version = 0)	218
74	4.7	Interface classes for setting up data exchange via local ports and modems	221
75	4.7.1	IEC local port setup (class_id = 19, version = 1)	221
76	4.7.2	IEC HDLC setup (class_id = 23, version = 1)	223
77	4.7.3	IEC twisted pair (1) setup (class_id = 24, version = 1)	225
78	4.7.4	Modem configuration (class_id = 27, version = 1)	228
79	4.7.5	Auto answer (class_id = 28, version = 2)	229
80	4.7.6	Auto connect (class_id = 29, version = 2)	233
81	4.7.7	GPRS modem setup (class_id = 45, version = 0)	235
82	4.7.8	GSM diagnostic (class_id: 47, version: 2)	236
83	4.7.9	LTE monitoring (class_id: 151, version: 1)	238
84	4.8	Interface classes for setting up data exchange via M-Bus	242
85	4.8.1	Overview	242
86	4.8.2	M-Bus slave port setup (class_id = 25, version = 0)	242
87	4.8.3	M-Bus client (class_id = 72, version = 1)	243
88	4.8.4	Wireless Mode Q channel (class_id = 73, version = 1).....	248
89	4.8.5	M-Bus master port setup (class_id = 74, version = 0)	248
90	4.8.6	DLMS/COSEM server M-Bus port setup (class_id = 76, version = 0)	249
91	4.8.7	M-Bus diagnostic (class_id = 77, version = 0).....	252
92	4.9	Interface classes for setting up data exchange over the Internet	256
93	4.9.1	TCP-UDP setup (class_id = 41, version = 0)	256
94	4.9.2	IPv4 setup (class_id = 42, version = 0)	257
95	4.9.3	IPv6 setup (class_id = 48, version = 0)	261
96	4.9.4	MAC address setup (class_id = 43, version = 0)	264

97	4.9.5	PPP setup (class_id = 44, version = 0)	265
98	4.9.6	SMTP setup (class_id = 46, version = 0).....	270
99	4.9.7	NTP setup (class_id = 100, version = 0)	271
100	4.10	Interface classes for setting up data exchange using S-FSK PLC	273
101	4.10.1	General	273
102	4.10.2	Overview	273
103	4.10.3	S-FSK Phy&MAC set-up (class_id = 50, version = 1)	275
104	4.10.4	S-FSK Active initiator (class_id = 51, version = 0)	281
105	4.10.5	S-FSK MAC synchronization timeouts (class_id = 52, version = 0).....	282
106	4.10.6	S-FSK MAC counters (class_id = 53, version = 0).....	284
107	4.10.7	IEC 61334-4-32 LLC setup (class_id = 55, version = 1)	288
108	4.10.8	S-FSK Reporting system list (class_id = 56, version = 0).....	289
109	4.11	Interface classes for setting up the LLC layer for ISO/IEC 8802-2	290
110	4.11.1	General	290
111	4.11.2	ISO/IEC 8802-2 LLC Type 1 setup (class_id = 57, version = 0).....	290
112	4.11.3	ISO/IEC 8802-2 LLC Type 2 setup (class_id = 58, version = 0).....	290
113	4.11.4	ISO/IEC 8802-2 LLC Type 3 setup (class_id = 59, version = 0).....	293
114	4.12	Interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for PRIME networks	295
115	4.12.1	Overview	295
116	4.12.2	Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes	296
117	4.12.3	61334-4-32 LLC SSCS setup (class_id = 80, version = 0).....	298
118	4.12.4	PRIME NB OFDM PLC Physical layer parameters	299
119	4.12.5	PRIME NB OFDM PLC Physical layer counters (class_id = 81, version = 0)	299
120	4.12.6	PRIME NB OFDM PLC MAC setup (class_id = 82, version = 0)	300
121	4.12.7	NB OFDM PLC MAC functional parameters (class_id = 83 version = 0)	301
122	4.12.8	PRIME NB OFDM PLC MAC counters (class_id = 84, version = 0)	304
123	4.12.9	PRIME NB OFDM PLC MAC network administration data (class_id = 85, version = 0)	305
124	4.12.10	PRIME NB OFDM PLC MAC address setup (class_id = 43, version = 0)	308
125	4.12.11	PRIME NB OFDM PLC Application identification (class_id = 86, version = 0)	308
126	4.13	Interface classes for setting up and managing the DLMS/COSEM narrowband OFDM PLC profile for G3-PLC networks	309
127	4.13.1	Overview	309
128	4.13.2	Mapping of G3-PLC PIB attributes to COSEM IC attributes.....	309
129	4.13.3	G3-PLC MAC layer counters (class_id = 90, version = 1).....	311
130	4.13.4	G3-PLC MAC setup (class_id = 91, version = 1)	312
131	4.13.5	G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 2)	318
132	4.14	Interface classes for setting up and managing DLMS/COSEM HS-PLC ISO/IEC 12139-1 neighbourhood networks.....	326
133	4.14.1	Overview	326
134	4.14.2	HS-PLC ISO/IEC 12139-1 MAC setup (class_id = 140, version = 0).....	326
135	4.14.3	HS-PLC ISO/IEC 12139-1 CPAS setup (class_id = 141, version = 0)	328
136	4.14.4	HS-PLC ISO/IEC 12139-1 IP SSAS setup (class_id = 142, version = 0).....	328
137	4.14.5	HS-PLC ISO/IEC 12139-1 HDLC SSAS setup (class_id = 143, version = 0)	329
138	4.15	ZigBee® setup classes	331
139	4.15.1	Overview	331

148	4.15.2	ZigBee® SAS startup (class_id = 101, version = 0).....	333
149	4.15.3	ZigBee® SAS join (class_id = 102, version = 0).....	335
150	4.15.4	ZigBee® SAS APS fragmentation (class_id = 103, version = 0)	337
151	4.15.5	ZigBee® network control (class_id = 104, version = 0).....	337
152	4.15.6	ZigBee® tunnel setup (class_id = 105, version = 0)	343
153	4.16	Interface classes for setting up and managing the DLMS/COSEM profile for LPWAN networks	345
154	4.16.1	General	345
155	4.16.2	Generic interface classes	345
157	4.17	LPWAN specific interface classes	350
158	4.17.1	General	350
159	4.17.2	LoRaWAN® interface classes	351
160	4.18	Interface classes for setting up and managing the DLMS/COSEM profile for Wi-SUN networks	357
162	4.18.1	Wi-SUN setup (class_id = 95, version 0)	357
163	4.18.2	Wi-SUN diagnostic (class_id = 96, version 0)	362
164	4.18.3	RPL diagnostic (class_id = 97, version 0)	365
165	4.18.4	MPL diagnostic (class_id = 98, version 0)	367
166	4.19	Interface classes for setting up and managing the DLMS/COSEM profile for ISO/IEC 14908 PLC networks	370
168	4.19.1	General	370
169	4.19.2	ISO/IEC 14908 identification (class_id = 130, version = 0)	370
170	4.19.3	ISO/IEC 14908 protocol setup (class_id = 131, version = 0)	371
171	4.19.4	ISO/IEC 14908 protocol status (class_id = 132, version = 0)	371
172	4.19.5	ISO/IEC 14908 diagnostic (class_id = 133, version = 0)	374
173	5	Previous versions of interface classes	377
174	5.1	General	377
175	5.2	Previous versions of interface classes for parameters and measurement data	377
177	5.2.1	Profile generic (class_id = 7, version = 0)	377
178	5.2.2	Compact data (class_id = 62, version = 0)	381
179	5.3	Previous versions of interface classes for access control and management	384
180	5.3.1	Association SN (class_id = 12, version = 0)	384
181	5.3.2	Association SN (class_id = 12, version = 1)	388
182	5.3.3	Association SN (class_id = 12, version = 2)	391
183	5.3.4	Association SN (Class_id = 12, version =3)	395
184	5.3.5	Association LN (class_id = 15, version = 0)	398
185	5.3.6	Association LN (class_id = 15, version = 1)	405
186	5.3.7	Association LN (class_id = 15, version = 2)	409
187	5.3.8	Security setup (class_id = 64, version = 0)	413
188	5.3.9	Push Setup (class_id = 40, version = 0)	416
189	5.3.10	Push Setup (class_id = 40, version = 1)	421
190	5.4	Previous versions of interface classes for time- and event-bound control	427
191	5.4.1	Parameter monitor (class_id = 65, version = 0)	427
192	5.5	Previous versions of payment metering related interface classes	429
193	5.6	Previous versions of interface classes for setting up data exchange via local ports and modems	429
195	5.6.1	IEC local port setup (class_id = 19, version = 0)	429
196	5.6.2	IEC HDLC setup, (class_id = 23, version = 0)	431
197	5.6.3	IEC twisted pair (1) setup (class_id = 24, version = 0)	433

198	5.6.4	PSTN modem configuration (class_id = 27, version = 0)	435
199	5.6.5	Auto answer (class_id = 28, version = 0)	437
200	5.6.6	PSTN auto dial (class_id = 29, version = 0)	439
201	5.6.7	Auto connect (class_id = 29, version = 1)	441
202	5.6.8	GSM diagnostic (class_id = 47, version = 0)	442
203	5.6.9	GSM diagnostic (class_id: 47, version: 1)	445
204	5.6.10	LTE monitoring (class_id: 151, version: 0)	447
205	5.7	Previous versions of interface classes for setting up data exchange via M-Bus	448
206	5.7.1	M-Bus client (class_id = 72, version = 0)	448
208	5.8	Previous versions of interface classes for setting up data exchange over the internet	453
210	5.9	Previous versions of interface classes for data exchange using S-FSK PLC	454
211	5.9.1	S-FSK Phy&MAC setup (class_id = 50, version = 0)	454
212	5.9.2	S-FSK IEC 61334-4-32 LLC setup (class_id = 55, version = 0)	458
213	5.10	Previous versions of interface classes for setting up the LLC layer for ISO/IEC 8802-2	459
214	5.11	Previous versions of interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for PRIME networks	459
215	5.12	Previous versions of interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for G3-PLC networks	459
219	5.12.1	Mapping of G3-PLC IB attributes to COSEM IC attributes (Original version)	459
221	5.12.2	G3 NB OFDM PLC MAC layer counters (class_id = 90, version = 0)	461
222	5.12.3	G3 NB OFDM PLC MAC setup (class_id = 91, version = 0)	462
223	5.12.4	G3 NB OFDM PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 0)	467
225	5.12.5	G3-PLC MAC setup (class_id = 91, version = 1)	472
226	5.12.6	G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 1)	478
227	5.13	Previous versions of interface classes for setting up and managing DLMS/COSEM HS-PLC ISO/IEC 12139-1 neighbourhood networks	484
229	5.14	Previous versions of ZigBee® setup classes	484
230	6	Relation to OBIS.....	485
231	6.1	General.....	485
232	6.2	Abstract COSEM objects.....	485
233	6.2.1	Use of value group C	485
234	6.2.2	Data of historical billing periods	486
235	6.2.3	Billing period values / reset counter entries	488
236	6.2.4	Other abstract general purpose OBIS codes	488
237	6.2.5	Clock objects (class_id = 8)	489
238	6.2.6	Modem configuration and related objects	489
239	6.2.7	Script table objects (class_id = 9)	490
240	6.2.8	Special days table objects (class_id = 11)	491
241	6.2.9	Schedule objects (class_id = 10)	491
242	6.2.10	Activity calendar objects (class_id = 20)	492
243	6.2.11	Register activation objects (class_id = 6)	492
244	6.2.12	Single action schedule objects (class_id = 22)	492
245	6.2.13	Register monitor objects (class_id = 21)	493
246	6.2.14	Parameter monitor objects (class_id = 65)	493
247	6.2.15	Limiter objects (class_id = 71)	493

248	6.2.16	Array manager objects (class_id = 123).....	493
249	6.2.17	Payment metering related objects.....	493
250	6.2.18	IEC local port setup objects (class_id = 19)	494
251	6.2.19	Standard readout profile objects (class_id = 7)	494
252	6.2.20	IEC HDLC setup objects (class_id = 23)	495
253	6.2.21	IEC twisted pair (1) setup objects (class_id = 24)	495
254	6.2.22	Objects related to data exchange over M-Bus.....	496
255	6.2.23	Objects to set up data exchange over the Internet	497
256	6.2.24	Objects to set up Push Setup (class_id = 40)	499
257	6.2.25	Objects for setting up data exchange using S-FSK PLC.....	499
258	6.2.26	Objects for setting up the ISO/IEC 8802-2 LLC layer	500
259	6.2.27	Objects for data exchange using narrowband OFDM PLC for PRIME networks	500
261	6.2.28	Objects for data exchange using narrow-band OFDM PLC for G3-PLC networks	501
263	6.2.29	ZigBee® setup objects.....	502
264	6.2.30	Objects for setting up and managing data exchange using ISO/IEC 14908 PLC networks	502
266	6.2.31	Objects for data exchange using HS-PLC ISO/IEC 12139-1 ISO/EC 12139-1 networks	502
268	6.2.32	Objects for data exchange using Wi-SUN networks	503
269	6.2.33	Association objects (class_id = 12, 15).....	503
270	6.2.34	SAP assignment object (class_id = 17).....	504
271	6.2.35	COSEM logical device name object	504
272	6.2.36	Information security related objects	504
273	6.2.37	Image transfer objects (class_id = 18)	505
274	6.2.38	Function control objects (class_id = 122).....	505
275	6.2.39	Communication port protection objects (class_id = 124)	505
276	6.2.40	Utility table objects (class_id = 26)	505
277	6.2.41	Compact data objects (class_id = 62)	506
278	6.2.42	Device ID objects	506
279	6.2.43	Metering point ID objects	507
280	6.2.44	Parameter changes and calibration objects.....	507
281	6.2.45	I/O control signal objects	507
282	6.2.46	Disconnect control objects (class_id = 70)	507
283	6.2.47	Arbitrator objects (class_id = 68)	508
284	6.2.48	Status of internal control signals objects.....	508
285	6.2.49	Internal operating status objects	508
286	6.2.50	Battery entries objects	509
287	6.2.51	Power failure monitoring objects	509
288	6.2.52	Operating time objects.....	510
289	6.2.53	Environment related parameters objects	510
290	6.2.54	Status register objects	510
291	6.2.55	Event code objects	510
292	6.2.56	Communication port log parameter objects	511
293	6.2.57	Consumer message objects.....	511
294	6.2.58	Currently active tariff objects	511
295	6.2.59	Event counter objects	511
296	6.2.60	Profile entry digital signature objects	512
297	6.2.61	Meter tamper event related objects	512

298	6.2.62	Profile entry counter objects	512
299	6.2.63	Meter tamper event related objects	513
300	6.2.64	Error register objects	513
301	6.2.65	Alarm register, Alarm filter and Alarm descriptor objects	514
302	6.2.66	General list objects	514
303	6.2.67	Event log objects (class_id 7)	515
304	6.2.68	Inactive objects	515
305	6.3	Electricity related COSEM objects	515
306	6.3.1	Value group D definitions	515
307	6.3.2	Electricity ID numbers	516
308	6.3.3	Billing period values / reset counter entries	516
309	6.3.4	Other electricity related general purpose objects	517
310	6.3.5	Measurement algorithm	518
311	6.3.6	Metering point ID (electricity related)	520
312	6.3.7	Electricity related status objects	520
313	6.3.8	List objects – Electricity (class_id = 7)	520
314	6.3.9	Threshold values	521
315	6.3.10	Register monitor objects (class_id = 21)	522
316	6.4	Coding of OBIS identifications	523
317	Annex A (informative)	Additional information on Auto answer and Auto connect ICs	524
318	Annex B (informative)	Additional information to M-Bus client (class_id = 72, version 1)	526
319	Annex C (informative)	Additional information on IPv6 setup class (class_id = 48, version = 0)	528
321	C.1	General	528
322	C.2	IPv6 addressing	528
323	C.3	IPv6 header format	529
324	C.4	IPv6 header extensions	531
325	C.4.1	Overview	531
326	C.4.2	Hop-by-Hop options	532
327	C.4.3	Destination options	532
328	C.4.4	Routing options	532
329	C.4.5	Fragment options	533
330	C.4.6	Security options	533
331	Annex D (informative)	Overview of the narrow-band OFDM PLC technology for PRIME networks	534
333	Annex E (informative)	Overview of the narrow-band OFDM PLC technology for G3-PLC networks	535
335	Annex F (informative)	Significant technical changes with respect to IEC 62056-6-2, Edition 3.0:2017	537
337	Bibliography	538	
338	Index	544	
339			
340	Figure 1 – The meaning of the definitions concerning the Image	18	
341	Figure 2 – An interface class and its instances	33	
342	Figure 3 – The COSEM server model	46	
343	Figure 4 – Combined metering device	46	
344	Figure 5 – Overview of the interface classes – Part 1	49	
345	Figure 6 – Overview of the interface classes – Part 2	50	

346	Figure 7 – Overview of the interface classes - Part 3	51
347	Figure 8 – The time attributes when measuring sliding demand	62
348	Figure 9 – The attributes in the case of block demand	62
349	Figure 10 – The attributes in the case of sliding demand (number of periods = 3)	63
350	Figure 11 – Image transfer process flow chart.....	105
351	Figure 12 – COSEM model of push operation	114
352	Figure 13 – Push windows and delays	116
353	Figure 14 – COSEM model of data protection	128
354	Figure 15 – Example: Read <i>protection_buffer</i> attribute.....	130
355	Figure 16 – Example of managing an array	146
356	Figure 17 – The generalized time concept.....	Error! Bookmark not defined.
357	Figure 17 – The generalized time concept.....	155
358	Figure 18 – State diagram of the Disconnect control IC.....	170
359	Figure 19 – Definition of upper and lower thresholds.....	183
360	Figure 20 – COSEM tariffication model (example).....	188
361	Figure 21 – COSEM billing model (example).....	189
362	Figure 22 – Outline Account model	191
363	Figure 23 – Diagram of attribute relationships.....	192
364	Figure 24 – Credit States when priority >0	203
365	Figure 25 – Operation of current_credit_status flags	205
366	Figure 26 – Interaction of current_credit_amount and available_credit with Token “Credit” and Emergency “Credit”	211
368	Figure 27 – Object model of DLMS/COSEM servers	273
369	Figure 28 – Object model of DLMS/COSEM servers	295
370	Figure 29 – Example of a ZigBee® network	Error! Bookmark not defined.
371	Figure 29 – Example of a ZigBee® network	332
372	Figure 30 – Push windows and delays	416
373	Figure 31 – Data of historical billing periods – example with module 12, VZ = 5.....	487
374	Figure A.1 – Network connectivity example for a GSM/GPRS network	524
375	Figure B.1 – Encryption key status diagram	526
376	Figure C.1 – IPv6 address formats.....	528
377		
378	Table 1 – Reserved base_names for SN referencing.....	34
379	Table 2 – Interface class overview	34
380	Table 3 – Common data types	38
381	Table 4 – List of interface classes by class_id	51
382	Table 5 – Enumerated values for physical units	57
383	Table 6 – Examples for scaler_unit	60
384	Table 7 – Parameters for selective access to the buffer attribute	71
385	Table 8 – Parameters for selective access to the buffer attribute	74
386	Table 9 – Daily billing data.....	81
387	Table 10 – Attributes of the “Compact data” object	82
388	Table 11 – A-XDR encoding of the data (SEQUENCE OF Get-Data-Result).....	82

389	Table 12 – Diagnostic and Alarm data.....	83
390	Table 13 – Attributes of the “Compact data” object	83
391	Table 14 – Encoding the data read from the buffer attribute of a “Profile generic” object.....	84
392	Table 15 – Logbook data	84
393	Table 16 – Attributes of the “Compact data” object	84
394	Table 17 – Attributes of the “Compact data” object	85
395	Table 18 – A-XDR encoding of the data read from the <i>buffer</i> attribute.....	85
396	Table 19 – Parameters for selective access to the <i>object_list</i> and <i>access_rights_list</i> attribute	89
397	Table 20 – Parameters for selective access to the <i>object_list</i> attribute.....	94
399	Table 21 – Encoding of selective access parameters with <i>data_index</i>	126
400	Table 22 – Key information required to establish data protection keys	138
401	Table 23 – Protection parameters of <i>protection_parameters_get</i> attribute	139
402	Table 24 – Protection parameters of <i>protection_parameters_set</i> attribute	140
403	Table 25 – Protection parameters of <i>get_protected_attributes</i> method	141
404	Table 26 – Protection parameters of <i>set_protected_attributes</i> method	142
405	Table 27 – Protection parameters of <i>invoke_protected_method</i> method	143
406	Table 28 – Example values for NCA and CLT	152
407	Table 29 – Schedule	159
408	Table 30 – Special days table	160
409	Table 31 – Disconnect control IC – states and state transitions.....	171
410	Table 32 – Explicit presentation of threshold value arrays.....	184
411	Table 33 – Explicit presentation of <i>action_sets</i>	184
412	Table 34 – Credit states	202
413	Table 35 – Credit state transitions	203
414	Table 36 – ADS address elements	227
415	Table 37 – Fatal error register	228
416	Table 38 – Mapping IEC 61334-4-512:2001 MIB variables to COSEM IC attributes / methods.....	274
418	Table 39 – MAC addresses in the S-FSK profile.....	280
419	Table 40 – Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes.....	296
420	Table 41 – Mapping of G3-PLC IB attributes to COSEM IC attributes.....	309
421	Table 42 – Use of ZigBee® setup COSEM interface classes	333
422	Table 43 – C/D Rule 1	349
423	Table 44 – Parameters for selective access to the <i>object_list</i> attribute.....	390
424	Table 45 – Parameters for selective access to the <i>object_list</i> and <i>access_rights_list</i> attribute	394
426	Table 46 – Parameters for selective access to the <i>object_list</i> attribute.....	404
427	Table 47 – Encoding of selective access parameters with <i>data_index</i>	420
428	Table 48 – Mapping of G3-PLC IB attributes specified in ITU-T G.9903:2013 Amd. 1 to COSEM IC attributes	460
430	Table 49 – Use of value group C for abstract objects in the COSEM context.....	485
431	Table 50 – Representation of various values by appropriate ICs	515
432	Table 51 – Measuring algorithms – enumerated values	519

433	Table 52 – Threshold objects, electricity	521
434	Table 53 – Register monitor objects, electricity	522
435	Table B.1 – Encryption key is preset in the slave and cannot be changed	527
436	Table B.2 – Encryption key is preset in the slave and new key is set after installation.....	527
437	Table B.3 – Encryption key is not preset in the slave, but can be set, case a).....	527
438	Table B.4 – Encryption key is not preset in the slave, but can be set, case b).....	527
439	Table C.1 – IPv6 header vs. IPv6 IC	531
440	Table C.2 – Optional IPv6 header extensions vs. IPv6 IC.....	532

441

442

443 INTERNATIONAL ELECTROTECHNICAL COMMISSION

444

445

446 ELECTRICITY METERING DATA EXCHANGE –
447 THE DLMS/COSEM SUITE –

448

449

Part 6-2: COSEM interface classes

450

451

FOREWORD

452

453

454

455

456

457

458

459

460

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.

461

462

463

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.

464

465

466

467

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.

468

469

470

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.

471

472

473

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.

474

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

475

476

477

478

479

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.

480

481

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.

482

483

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.

484

485

486

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.

487

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

488

489

490

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:

491

492

493

DLMS¹ User Association
Zug/Switzerland
www.dlms.com

¹ Device Language Message Specification.

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

496 This fourth edition cancels and replaces the third edition of IEC 62056-6-2 published in 2017.
497 It constitutes a technical revision.

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

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

FDIS	Report on voting
13/xxxx/FDIS	13/xxxx/RVD

501
502 Full information on the voting for the approval of this standard can be found in the report on
503 voting indicated in the above table.

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

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

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

- 510 • reconfirmed,
511 • withdrawn,
512 • replaced by a revised edition, or
513 • amended.

514
**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.**

515

516

517

INTRODUCTION

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

520 This edition is in line with the DLMS UA Blue Book Edition 14. The main new features are the
521 “Array manager” IC, version 1 of the “Compact data” IC, version 1 of the “GSM diagnostic” IC,
522 the “LTE monitoring” IC, the “NTP setup” IC, the HS-PLC setup ICs and the related new OBIS
523 codes.

524 **Object modelling and data identification**

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

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

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

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

546 COSEM models the utility meter as a *server* application – see 4.1.7 – used by *client* applications
547 that retrieve data from, provide control information to, and instigate known actions within the
548 meter via controlled access to the COSEM objects. The *clients* act as agents for third parties,
549 i.e. the business processes of energy market participants.

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

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

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

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

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

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

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

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

572 **Acknowledgement**

573 The actual document has been established by the WG Maintenance of the DLMS UA.

574 Subclauses 4.4.7 and 4.4.9 are based on parts of NIST documents. Reprinted courtesy of the
575 National Institute of Standards and Technology, Technology Administration, U.S. Department
576 of Commerce. Not copyrightable in the United States.

577

578

579 **ELECTRICITY METERING DATA EXCHANGE –**
580 **THE DLMS/COSEM SUITE –**

582 **Part 6-2: COSEM interface classes**

586 **1 Scope**

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

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

591 **2 Normative references**

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

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

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

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

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

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

608 IEC TR 62055-21:2005, *Electricity metering – Payment systems – Part 21: Framework for*
609 *standardization*

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

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

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

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

- 617 NOTE This Edition is referenced in the interface class “IEC twisted pair (1) setup” (class_id: 24, version: 1).
- 618 IEC 62056-46:2002/AMD1:2006, *Electricity metering – Data exchange for meter reading, tariff*
619 *and load control – Part 46: Data link layer using HDLC protocol*
- 620 IEC 62056-5-3:2021, *Electricity metering data exchange – The DLMS/COSEM suite –*
621 *Part 5-3: DLMS/COSEM application layer*
- 622 IEC 62056-6-1:2021, *Electricity metering data exchange – The DLMS/COSEM suite –*
623 *Part 6-1: Object identification system (OBIS)*
- 624 IEC 62056-7-3:2017, *Electricity metering data exchange – The DLMS/COSEM suite – Part 7-3:*
625 *Wired and wireless M-Bus communication profiles for local and neighbourhood networks*
- 626 IEC 62056-8-3:2013, *Electricity metering data exchange – The DLMS/COSEM suite – Part 8-3:*
627 *Communication profile for PLC S-FSK neighbourhood networks*
- 628 IEC 62056-8-6:2017, *Electricity metering data exchange – The DLMS/COSEM suite – Part 8-6:*
629 *High speed PLC ISO/IEC 12139-1 profile for neighbourhood networks*
- 630 IEC 62056-8-8:2020, *Electricity metering data exchange - The DLMS/COSEM suite - Part 8-8:*
631 *Communication profile for ISO/IEC 14908 series networks*
- 632 ISO/IEC 8802-2:1998, *Information technology – Telecommunications and information exchange*
633 *between systems – Local and metropolitan area networks – Specific requirements – Part 2:*
634 *Logical Link Control*
- 635 ISO/IEC 12139-1:2009, *Information technology – Telecommunications and information exchange*
636 *between systems – Powerline communication (PLC) – High speed PLC medium*
637 *access control (MAC) and physical layer (PHY) – Part 1: General requirements*
- 638 ISO/IEC 14908-1:2012, *Interconnection of information technology equipment – Control network*
639 *protocol Part Protocol stack*
- 640 ISO/IEC/IEEE 60559:2011, *Information technology – Microprocessor Systems – Floating-Point*
641 *arithmetic*
- 642 ISO 4217, *Codes for the representation of currencies*
- 643 **3 Terms, definitions and abbreviated terms**
- 644 For the purposes of this document, the following terms and definitions apply.
- 645 ISO and IEC maintain terminological databases for use in standardization at the following
646 addresses:
- 647 • IEC Electropedia: available at <http://www.electropedia.org/>
- 648 • ISO Online browsing platform: available at <http://www.iso.org/obp>
- 649 **3.1 Terms and definitions related to the Image transfer process (see 4.4.6)**
- 650 **3.1.1**
- 651 **Image**
- 652 binary data of a specified size

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

655 **3.1.2**

656 **ImageSize**

657 size of the whole Image to be transferred

658 Note 1 to entry: ImageSize is expressed in octets.

659 **3.1.3**

660 **ImageBlock**

661 part of the Image of size ImageBlockSize

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

663 **3.1.4**

664 **ImageBlockSize**

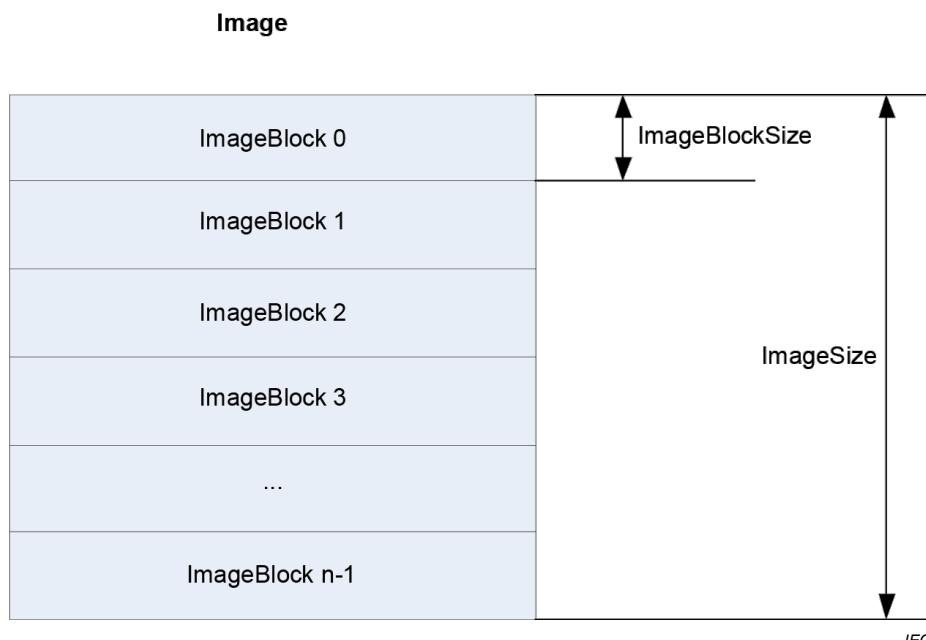
665 size of ImageBlock expressed in octets

666 **3.1.5**

667 **ImageBlockNumber**

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

669 The meaning of the definitions above is illustrated in Figure 1.



670

671 **Figure 1 – The meaning of the definitions concerning the Image**

672 **3.2 Terms and definitions related to the S-FSK PLC setup classes (see 4.10)**

673 **3.2.1**

674 **initiator**

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

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

677 [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]

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 61334-5-1:2001, 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 4.12)**Definitions related to the physical layer****3.3.1****base node**

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

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

716 **3.3.2**

717 **beacon slot**

718 the location of the beacon PDU within a frame

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

720 **3.3.3**

721 **node**

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

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

725 **3.3.4**

726 **registration**

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

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

730 **3.3.5**

731 **service node**

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

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

734 **3.3.6**

735 **subnetwork**

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

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

739 Definitions related to the MAC layer

740 **3.3.7**

741 **disconnected state <of a service node>**

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

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

746 **3.3.8**

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

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

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

751 **3.3.9**

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

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

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

757 **3.3.10**

758 **promotion**

759 the process by which a service node is qualified to switch (repeat, forward) data traffic from
760 other nodes and act as a branch point on the subnetwork tree structure. A successful promotion
761 represents the transition between Terminal and Switch state. When a service node is in the
762 Disconnected state, it cannot directly transition to Switch state

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

764 **3.3.11**

765 **demotion**

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

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

769 **3.4 Terms and definitions related to the ISO/IEC 14908 setup ICs (see 4.19)**

770 **3.4.1**

771 **domain**

772 logical network that is a unit for addressing

773 Note 1 to entry: Subnet (see below) and node addresses are assigned by the administrator responsible for the
774 domain, and they have meaning only in the context of that domain.

775 Note 2 to entry: All nodes must be in the same domain to be able to address each other.

776 **3.4.2**

777 **node**

778 abstraction for a physical node that represents the highest degree of address resolvability on a
779 network

780 Note 1 to entry: A node is identified (addressed) within a subnet by its (logical) node identifier. A physical node may
781 belong to more than one subnet; when it does, it is assigned one (logical) node number for each subnet to which it
782 belongs. A physical node may belong to at most two subnets; these subnets must be in different domains. A node
783 may also be identified (absolutely) within a network by its Unique_Node_ID.

784 **3.4.3**

785 **subnet**

786 set of nodes accessible through the same link layer protocol

787 **3.4.4**

788 **transaction**

789 sequence of messages that are correlated together

790

791 **3.5 Terms and definitions related to ZigBee® (see 4.15)**

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

793 **3.5.1**

794 **CAD**

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

3.5.2**IHD**

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

3.5.3**install code**

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

3.5.4**link key ***

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

3.5.5**MAC address/IEEE address**

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

3.5.6**ZigBee®**

ZigBee® is a 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

3.5.7**ZigBee® client**

this is similar to the role of the DLMS/COSEM Client. For a greater understanding of the interaction between the client and server the ZigBee® PRO specification should be read

3.5.8**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)

3.5.9**ZigBee® cluster**

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

3.5.10**ZigBee® mirror**

a 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

3.5.11**ZigBee® PRO**

an 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

844 (SKKE), while ZigBee® (stack profile 1) offers a smaller footprint in RAM and flash. Both offer
845 full mesh networking and work with all ZigBee® application profiles

846 **3.5.12**

847 **ZigBee® router ***

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

851 **3.5.13**

852 **ZigBee® server**

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

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

856 **3.5.14**

857 **ZigBee® Trust Center ***

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

860 **3.6 Terms and definitions related to Payment metering interface classes (see 4.6)**

861 **3.6.1**

862 **account**

863 statement of the credits and charges of an individual with reference to a contractual relationship
864 between the said individual and another party; in this case a utility service provider

865 **3.6.2**

866 **available**

867 (credit), total value that may be decremented by charges without further action

868 **3.6.3**

869 **charge**

870 representation of a financial liability on an account

871 Note 1 to entry: Within this specification charges are modelled in the form of "Charge" objects that define the amount
872 due, collection mechanism, collection periodicity, collection amount and other relevant variables.

873 Note 2 to entry: There may be one or more instalments payable and their size may be determined explicitly or in
874 terms of a rate of payment per unit of time or of consumption.

875 Note 3 to entry: Charges may also be levied as a fixed amount per vend.

876 **3.6.4**

877 **credit mode**

878 mode of operation of a meter in a payment system that does not require payment for the
879 consumption in advance

880 **3.6.5**

881 **collect**

882 take payment of an instalment of a charge, accounting for the collection amount determined by
883 the *unit_charge_active* attribute of the "Charge" object

884 **3.6.6**

885 **commodity**

886 utility product delivered to a consumer at a service point on their premises under a contract of
887 supply such as electricity, gas, water, and heat

888 **3.6.7**
889 **enabled**
890 when used in the context of “Credit” or “Charge” types; means that the “Credit” or “Charge” type
891 appears in the *credit_reference_list* or *charge_reference_list* respectively of the “Account”
892 object

893 **3.6.8**
894 **emergency credit**
895 amount of credit administered in a payment metering system working in prepayment mode,
896 representing a short term loan to the consumer

897 Note 1 to entry: This is a feature of some payment metering systems in which the consumer is able to obtain a
898 limited amount of credit as a short-term loan, often mediated locally by the prepayment unit itself. The word
899 “emergency” indicates urgent need rather than disaster.

900 **3.6.9**
901 **Enterprise Resource Planning (ERP) system**
902 **Back Office System**
903 computer system carrying out the business processing of an organisation (such as an energy
904 supplier), as distinct from the communications system. See also Head End System

905 **3.6.10**
906 **friendly credit**
907 period of time with a configurable start and end point, where the meter will not disconnect supply
908 regardless of the status of the *available_credit*. Also known as non-disconnect period

909 Note 1 to entry: This function is used in circumstances where it would be inconvenient to obtain needed credit (for
910 example, at night or in the case of a frail elderly consumer).

911 **3.6.11**
912 **Head End System**
913 **HES**
914 computer system, connected by a communications network to a population of intelligent
915 devices, whose job is the control and coordination of information flows to and from those
916 devices, typically on behalf of a separate ERP (“back office”) system

917 **3.6.12**
918 **Home Area Network**
919 **HAN**
920 communications network constructed with the principal aim of connecting devices in one
921 premises

922 **3.6.13**
923 **in use**
924 state of a “Credit” object that, at the point of query, has a positive *current_credit_amount* and
925 that Credit is being consumed by some active Charges represented by “Charge” objects

926 Note 1 to entry: When the *current_credit_amount* reaches zero, the credit status becomes exhausted.

927 **3.6.14**
928 **load limiting**
929 mode of operation of some payment metering systems (not necessarily in prepayment mode)
930 in which the consumer is able to draw on a supply provided they do not exceed a configured
931 level of demand

932 Note 1 to entry: The implied purpose is for management of the consumer's finances: where demand is subject to
933 limitation for the benefit of the generation or distribution system the term “load management” is more often used.

3.6.15**local communications**

mechanism of communicating with the meter over some media, within the vicinity of that meter such as over a HAN or optical port

3.6.16**manual entry**

entering of a token to the payment metering installation via means of a manual process

3.6.17**managed payment mode**

specialisation of credit mode that allows operation of an Account, Credit and possibly Charges in a meter where the payment for the service is received by the utility after the service has been consumed

Note 1 to entry: When in managed payment mode tokens are not normally used, however the credit is adjusted using the methods in the "Credit" object.

Note 2 to entry: The meter is allowed to go into an allowable amount of debt before being credited from the client in line with a received cash payment by the utility.

Note 3 to entry: In this example cash is used as a generic term for a real life payment of currency to the utility which could be executed as legal tender, automated electronic transfer, etc.

3.6.18**payment metering installation**

set of payment metering equipment installed and ready for use at a consumer's premise

Note 1 to entry: This includes mounting the equipment as appropriate, and where a multi-device installation is involved, the connection of each unit of equipment as appropriate. It also includes the connection of utility supply network to each supply interface, the connection of the consumer's load interface, and the commissioning of the equipment into an operational state as a payment metering installation.

3.6.19**prepayment mode**

mode of operation of a meter in a payment system, whereby the consumer pays for service in advance of consumption

3.6.20**post-payment**

method of operation of a payment system whereby a consumer may consume service before paying for it

Note 1 to entry: This term can be used interchangeably with the term Credit mode when used in the context of operational modes.

Note 2 to entry: This term is usually used in conjunction with a system description whereas Credit mode is used when referring to the operational mode of a meter or account.

3.6.21**remote communications**

transportation of a token or other message from a client to a server running a payment metering application process via some form of WAN and access network. This could be point to point, mesh radio, fibre optic connection, etc., and may travel through multiple devices and over multiple protocols before reaching the meter

3.6.22**repayable**

credit_types such as emergency credit where an amount added to *current_credit_amount* of a "Credit" object has to be repaid before the Credit is selectable again

3.6.23**reserved credit**

amount of credit that is held in reserve in the account of a payment meter, for use at a later time, at the discretion of the consumer

Note 1 to entry: The mechanism for reserving this credit may be subject to agreement between the utility supplier and the consumer. For example a proportion of every token may be added to the reserve Credit or the supplier might give the consumer an allowance every month, but these arrangements will be project specific.

3.6.24**selectable**

specific state of a “Credit” object where the consumer’s immediate confirmation is needed before it can be brought into use

Note 1 to entry: For example, Emergency Credit has the nature of a short-term loan and should therefore only be deployed with the consumer’s agreement. The term refers respectively to the need to get agreement and to the fact of having received agreement. Only a “Credit” made (1) Selectable can be (2) Selected / Invoked. Not all Credits need to be selected by an external trigger, as in most cases the meter application automatically performs this action.

3.6.25**selected/invoked**

specific state of a “Credit” object where the value of *current_credit_amount* is included in the calculation of *available_credit* in the related “Account”

Note 1 to entry: This is the state of a Credit before becoming In use and is considered in the *available_credit* attribute of the “Account”, but is not yet being consumed by any Charge (due to a higher priority Credit being In use).

3.6.26**service**

provision of a commodity (such as water, electricity gas or heat)

3.6.27**social credit**

credit that is given free of payment for reasons such as the relief of poverty

Note 1 to entry: Typically such a credit is given at fixed times (e.g. monthly) in limited amounts. This particular type of credit could also be consumption based, such that the consumer must keep consumption below a limiting threshold in order to use the social credit. This could be controlled by the consumer being disconnected if the limit is breached.

Note 2 to entry: Social credits are modelled of “Credit” objects of type *emergency_credit*, *time_based_credit*, *consumption_based_credit*.

3.6.28**temporary debt**

transient liability to the meter that accrues when *Charges* are collected at a time when all credits are exhausted

Note 1 to entry: This temporary debt amount is accumulated in *amount_to_clear* in the “Account” object.

3.6.29**token**

self-contained package of data related to the purchase of credit or to other system functions, embodied in a token carrier (q.v.). The token forms a link between source and destination of the transaction. The token contents may reflect money, energy, time, etc., in harmony with the currency declared in the meter

Note 1 to entry: Defined in IEC TR 62055-21:2005 as “<Equipment-related definition>[sic] information content including an instruction issued on a token carrier by a vending or management system that is capable of subsequent transfer to and acceptance by a specific payment meter, or one of a group of meters, with appropriate security”.

1028 **3.6.30**

1029 **token carrier**

1030 means of transferring a token from one system element to another, typically in material
1031 "physical" or electronic "virtual" form

1032 Note 1 to entry: In a general sense, the token refers to the instruction and information being transferred, while the
1033 token carrier refers to the physical device being used to carry the instruction and information, or to the
1034 communications medium in the case of a virtual token carrier.

1035 **3.6.31**

1036 **token carrier interface**

1037 interface between the token carrier and the payment metering installation

1038 Note 1 to entry: For example, it may be a keypad for numeric tokens, or a physical token carrier acceptor, or a
1039 communications connection to a local or remote machine for a virtual token carrier interface.

1040 Note 2 to entry: The token carrier interface may also be used to pass additional information to or from the payment
1041 meter, such as for the purposes of payment system management.

1042 **3.6.32**

1043 **top-up**

1044 **credit token**

1045 credit purchased by the consumer and capable of being delivered in the form of a token (as well
1046 as by other means) in a physical or virtual token carrier

1047 **3.6.33**

1048 **transient device communications**

1049 transportation of a token within a payment metering installation through some electronic
1050 communication mechanism involving a transient device. This could be done by local radio,
1051 galvanic connection, optical connection, etc., from various devices, e.g. HHT, mobile phone

1052 Note 1 to entry: In Home Displays are not classed as transient devices despite the fact that they may operate in a
1053 transient manner as far as the network is concerned. Transient devices shall be considered as devices that join the
1054 network for a short time and only very rarely. In home displays are considered to be on the network for long periods
1055 and only absent for very short periods in comparison to the life of the network.

1056 **3.6.34**

1057 **vend**

1058 operation or transaction resulting in the available credit held on a payment meter to be
1059 increased by use of a credit token

1060 Note 1 to entry: Vend would normally relate to a transaction in conjunction with a vending system at a point of sale,
1061 resulting in the creation of a token that can be transported by means of a physical or virtual token carrier.

1062 **3.7 Terms and definitions related to the Arbitrator IC (see 4.5.12)**

1063 **3.7.1**

1064 **action**

1065 operation that can be requested locally or remotely from the server

1066 **3.7.2**

1067 **actor**

1068 entity requesting an action

1069 Note 1 to entry: It can be the local application process or a client.

1070 **3.7.3**

1071 **arbitrator**

1072 function modelled in COSEM that can determine, based on pre-configured rules, which action
1073 is carried out when multiple actors request potentially conflicting actions to control the same
1074 resource

1075

3.8 Abbreviated terms

Abbreviation	Explanation
3GPP	3 rd Generation Partnership Project
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)
ARFCN	Absolute radio-frequency channel number
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)
CDMA	Code Division Multiple Access
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
CPAS	Common Part Adaptation Sublayer
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 Configuration 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

Abbreviation	Explanation
DSCP	Differentiated Services Code Point
DSSID	Direct Switch ID
EAP	Extensible Authentication Protocol
eARFCN	Enhanced Absolute radio-frequency channel number
EDGE	Enhanced Data rates for GSM Evolution
EMC	Emergency Credit (in relation to payment metering)
ERP	Enterprise Resource Planning
EUI-48	48-bit Extended Unique Identifier
EUI-64	64-bit Extended Unique Identifier
E-UTRA	Evolved UMTS Terrestrial Radio Access
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
HS-PLC	High-Speed Power Line Carrier
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

Abbreviation	Explanation
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
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
M2M	Machine to Machine
MAC	Medium Access Control
M-Bus	Meter Bus
MCC	Mobile Country Code
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)
MNC	Mobile Network Code
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

Abbreviation	Explanation
PIN	Personal Identity Number
PLC	Power Line Carrier
PLMN	Public Land Mobile Network
PNPDU	Promotion Needed PDU
POS	Point Of Sale (Payment metering)
POS	Personal Operating Space (ZigBee ®)
PPDU	Physical Protocol Data Unit
PPP	Point-to-Point Protocol
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RB	Radio Band
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
RREP	Route Reply
RREQ	Route Request
RRER	Route Error
RSRQ	Reference Signal Received Quality
RSRP	Reference Signal Received Power
RSSI	Received 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
SHA	Secure Hash Algorithm
SID	Switch identifier
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SN	Short Name
SNA	Subnetwork Address
SSAS	Service Specific Adaptation Sublayer
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

Abbreviation	Explanation
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.	

1076 **4 The COSEM interface classes**

1077 **4.1 Basic principles**

1078 **4.1.1 General**

1079 This Clause 4.1 describes the basic principles on which the COSEM interface classes (ICs) are
 1080 built. It also gives a short overview on how interface objects – instantiations of the ICs – are
 1081 used for communication purposes. Data collection systems and metering equipment from
 1082 different vendors, following these specifications, can exchange data in an interoperable way.

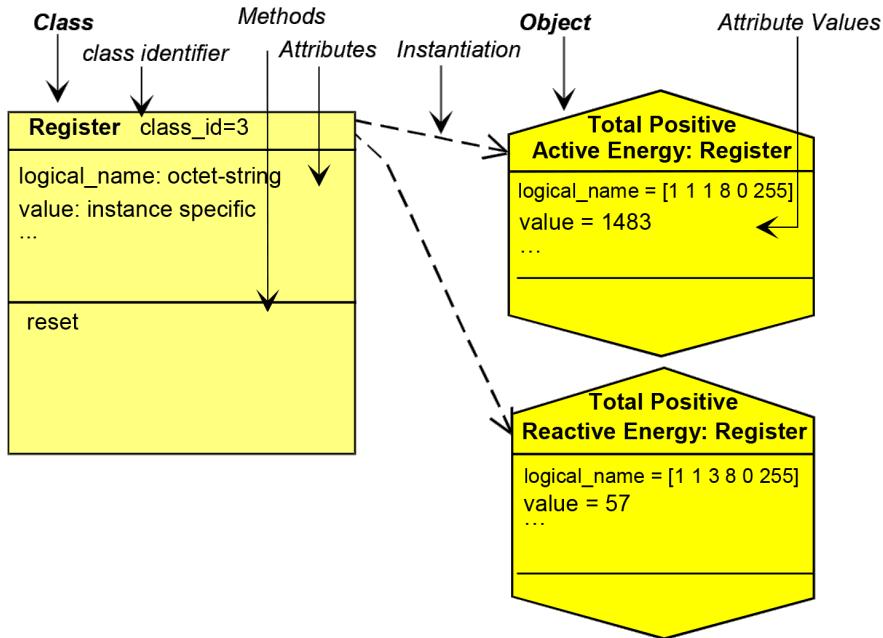
1083 For specification purposes, this standard uses the technique of object modelling.

1084 An object is a collection of attributes and methods. Attributes represent the characteristics of
 1085 an object. The value of an attribute may affect the behaviour of an object. The first attribute of
 1086 any object is the *logical_name*. It is one part of the identification of the object. An object may
 1087 offer a number of methods to either examine or modify the values of the attributes.

1088 Objects that share common characteristics are generalized as an interface class, identified with
 1089 a *class_id*. Within a specific IC, the common characteristics (attributes and methods) are
 1090 described once for all objects. Instantiations of ICs are called COSEM interface objects.

1091 Manufacturers may add proprietary methods and attributes to any object; see 4.1.2.

1092 Figure 2 illustrates these terms by means of an example:



1093

IFC

1094

Figure 2 – An interface class and its instances

1095 The IC “Register” is formed by combining the features necessary to model the behaviour of a
 1096 generic register (containing measured or static information) as seen from the client (data
 1097 collection system, hand held terminal). The contents of the register are identified by the attribute
 1098 *logical_name*. The *logical_name* contains an OBIS identifier (see IEC 62056-6-
 1099 1:2021 IEC 62056-6-1:2021). The actual (dynamic) content of the register is carried by its *value*
 1100 attribute.

1101 Defining a specific meter means defining several specific objects. In the example of Figure 2,
 1102 the meter contains two registers; i.e. two specific instances of the IC “Register” are instantiated.
 1103 Through the instantiation, one COSEM object becomes a “total, positive, active energy register”
 1104 whereas the other becomes a “total, positive, reactive energy register”.

1105 NOTE The COSEM interface objects (instances of COSEM ICs) represent the behaviour of the meter as seen from
 1106 the “outside”. Therefore, modifying the value of an attribute – for example resetting the *value* attribute of a register
 1107 – is always initiated from the outside. Internally initiated changes of the attributes – for example updating the *value*
 1108 attribute of a register – are not described in this model.

1109 **4.1.2 Referencing methods**

1110 Attributes and methods of COSEM objects can be referenced in two different ways:

1111 **Using logical names (LN referencing):** In this case, the attributes and methods are referenced
 1112 via the identifier of the COSEM object instance to which they belong.

1113 The reference for an attribute is: class_id, value of the *logical_name* attribute,
 1114 attribute_index.

1115 The reference for a method is: class_id, value of the *logical_name* attribute, method_index,
 1116 where:

- 1117 • attribute_index is used as the identifier of the attribute required. Attribute indexes are
 1118 specified in the definition of each IC. They are positive numbers starting with 1.
 1119 Proprietary attributes may be added: these shall be identified with negative numbers;

- 1120 • method_index is used as the identifier of the method required. Method indexes are
 1121 specified in the definition of each IC. They are positive numbers starting with 1.
 1122 Proprietary methods may be added: these shall be identified with negative numbers.

1123 **Using short names (SN referencing):** This kind of referencing is intended for use in simple
 1124 devices. In this case, each attribute and method of a COSEM object is identified with a 13-bit
 1125 integer. The syntax for the short name is the same as the syntax of the name of a DLMS named
 1126 variable. See IEC 61334-4-41:1996 and IEC 62056-5-3:2021 Clause 8.

1127 4.1.3 Reserved base_names for special COSEM objects

1128 In order to facilitate access to devices using SN referencing, some short_names are reserved
 1129 as base_names for special COSEM objects. The range for reserved base_names is from
 1130 0xFA00 to 0xFFFF. The following specific base_names are defined, see Table 1.

1131 **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"

1132

1133 4.1.4 Class description notation

1134 This subclause describes the notation used to define the ICs.

1135 A short text describes the functionality and application of the IC. Table 2 gives an overview of
 1136 the IC including the class name, the attributes, and the methods. Each attribute and method
 1137 shall be described in detail. The template is shown below.

1138 **Table 2 – Interface class overview**

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.	...				x + 0x...
2.	...				x + 0x...
3.	...				x + 0x...

1139

1140 4.1.4.1 Class name

1141 Describes the interface class (e.g. "Register", "Clock", "Profile generic"...).

1142 NOTE 1 Interface classes names are mentioned in quotation marks.

1143 4.1.4.2 Cardinality

1144 Specifies the number of instances of the IC within a logical device (see 4.1.8).

1145 **value** The IC shall be instantiated exactly “value” times.

1146 *min...max*. The IC shall be instantiated at least “min.” times and at most “max.” times. If min.
 1147 is zero (0) then the IC is optional, otherwise (min. > 0) “min.” instantiations of the IC are
 1148 mandatory.

1149 **4.1.4.3 class_id**

1150 Identification code of the IC (range 0 to 65 535). The class_id of each object is retrieved together
 1151 with the logical name by reading the object_list attribute of an “Association LN” / “Association
 1152 SN” object.

- 1153 – class_id-s from 0 to 8 191 are reserved to be specified by the DLMS UA.
- 1154 – class_id-s from 8 192 to 32 767 are reserved for manufacturer specific ICs.
- 1155 – class_id-s from 32 768 to 65 535 are reserved for user group specific ICs.

1156 The DLMS UA reserves the right to assign ranges to individual manufacturers or user groups.

1157 class_id-s 32768 to 32778 are allocated to the STS Association

1158 **4.1.4.4 version**

1159 Identification code of the version of the IC. The version of each object is retrieved together with
 1160 the class_id and the logical name by reading the object_list attribute of an “Association LN” /
 1161 “Association SN” object.

1162 **Within one logical device, all instances of a certain IC shall be of the same version.**

1163 **4.1.4.5 Attributes**

1164 Specifies the attributes that belong to the IC.

1165 (*dyn.*) Classifies an attribute that carries a process value, which is updated by the meter
 1166 itself.

1167 (*static*) Classifies an attribute, which is not updated by the meter itself (for example
 1168 configuration data).

1169 NOTE 2 There are some attributes which may be either static or dynamic depending on the application. In these
 1170 cases this property is not indicated.

1171 NOTE 3 Attribute names use the underscore notation. When mentioned in the text they are in *italic*. Example:
 1172 *logical_name*

1173 **4.1.4.6 logical_name**

1174 octet-string It is always the first attribute of an IC. It identifies the instantiation (COSEM
 1175 object) of this IC. The value of the *logical_name* conforms to OBIS; see
 1176 Clauses 6 and IEC 62056-6-1:**2021**.

1177 **4.1.4.7 Data type**

1178 Defines the data type of an attribute; see 4.1.5.

1179

1180 **4.1.4.8 Min.**

1181 Specifies if the attribute has a minimum value.

1182 x The attribute has a minimum value.

1183 <empty> The attribute has no minimum value.

1184 **4.1.4.9 Max.**

1185 Defines if the attribute has a maximum value.

1186 x The attribute has a maximum value.

1187 <empty> The attribute has no maximum value.

1188 **4.1.4.10 Def.**

1189 Specifies if the attribute has a default value. This is the value of the attribute after reset.

1190 x The attribute has a default value.

1191 <empty> The default value is not defined by the IC specification.

1192 **4.1.4.11 Short name**

1193 When Short Name (SN) referencing is used, each attribute and method of object instances has
1194 to be mapped to short names.

1195 The base_name x of each object instance is the DLMS named variable the logical name attribute
1196 is mapped to. It is selected in the implementation phase. The IC definition specifies the offsets
1197 for the other attributes and for the methods.

1198 **4.1.4.12 Specific methods**

1199 Provides a list of the specific methods that belong to the object.

1200 Method Name () The method has to be described in the subsection "Method description".

1201 NOTE 4 Method names use the underscore notation. When mentioned in the text they are in *italic*. Example:
1202 *add_object*.

1203 **4.1.4.13 m/o**

1204 Defines if the method is mandatory or optional.

1205 *m (mandatory)* The method is mandatory.

1206 *o (optional)* The method is optional.

1207

1208

1209 **4.1.4.14 Attribute description**

1210 Describes each attribute with its data type (if the data type is not simple), its data format and
1211 its properties (minimum, maximum and default values).

1212 **4.1.4.15 Method description**

1213 Describes each method and the invoked behaviour of the COSEM object(s) instantiated.

1214 NOTE 5 Services for accessing attributes or methods by the protocol are specified in IEC 62056-5-3:2021 6.6 to
1215 6.17.

1216 **4.1.4.16 Selective access**

1217 The xDLMS attribute-related services typically reference the entire attribute. However, for
1218 certain attributes selective access to just a part of the attribute may be provided. The part of
1219 the attribute is identified by specific selective access parameters. These are defined as part of
1220 the attribute specification.

1221 Selective access is available with the following interface class attributes and methods:

- 1222 • “Profile generic” objects, **buffer** attribute;
- 1223 • “Association SN” objects, **object_list** and **access_rights_list** attribute;
- 1224 • “Association LN” objects, **object_list** attribute;
- 1225 • “Compact data”, objects, **compact_buffer** attribute;
- 1226 • “Push” objects, **push_object_list** attribute;
- 1227 • “Data protection” objects, **protection_object_list** attribute **get_protected_attributes**
1228 method and **set_protected_attributes** method.

1229 **4.1.5 Common data types**

1230 Table 3 contains the list of data types usable for attributes of COSEM objects.

1231

Table 3 – 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 4.1.6.2.
float64	[24]	OCTET STRING (SIZE(8))	
date-time	[25]	OCTET STRING SIZE(12))	For formatting, see 4.1.6.1.
date	[26]	OCTET STRING (SIZE(5))	
time	[27]	OCTET STRING (SIZE(4))	
delta-integer	[28]	Integer8	-128...127
delta-long	[29]	Integer16	-32 768...32 767
delta-double-long	[30]	Integer32	-2 147 483 648... 2 147 483 647
delta-unsigned	[31]	Unsigned8	0...255
delta-long-unsigned	[32]	Unsigned16	0...65 535
delta-double-long-unsigned	[33]	Unsigned32	0...4 294 967 295
-- 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.	

Type description	Tag ^a	Definition	Value range
-- 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 62056-5-3:2021 Clause 8.

1232

1233 **4.1.6 Data formats**1234 **4.1.6.1 Date and time formats**1235 Date and time information may be represented using the data type *octet-string*.1236 NOTE 1 In this case the encoding includes the tag of the data type *octet-string*, the length of the octet-string and
1237 the elements of *date*, *time* and /or *date-time* as applicable.1238 Date and time information may be also represented using the data types *date*, *time* and *date-time*.
12391240 NOTE 2 In these cases, the encoding includes only the tag of the data types *date*, *time* or *date-time* as applicable
1241 and the elements of date, time or date-time.1242 NOTE 3 The (SIZE ()) specifications are applicable only when date, time or date time are represented by the data
1243 types *date*, *time* or *date-time*.

1244

1245 **date** OCTET STRING (SIZE(5))

```

1246   {
1247     year highbyte,
1248     year lowbyte,
1249     month,
1250     day of month,
1251     day of week
1252   }
```

1253 Where:

1254 **year:** interpreted as long-unsigned
1255 range 0...big
1256 0xFFFF = not specified

1257 year highbyte and year lowbyte represent the 2 bytes of the long-unsigned

1259
1260 **month:** interpreted as unsigned
1261 range 1...12, 0xFD, 0xFE, 0xFF
1262 1 is January
1263 0xFD = daylight_savings_end
1264 0xFE = daylight_savings_begin
1265 0xFF = not specified

1266
1267 **dayOfMonth:** interpreted as unsigned
1268 range 1...31, 0xFD, 0xFE, 0xFF
1269 0xFD = 2nd last day of month
1270 0xFE = last day of month
1271 0xE0 to 0xFC = reserved
1272 0xFF = not specified

1273

1274 **dayOfWeek:** interpreted as unsigned
 1275 range 1...7, 0xFF
 1276 1 is Monday
 1277 0xFF = not specified
 1278

1279 For repetitive dates, the unused parts shall be set to “not specified”.

1280 For countries not using the Gregorian calendar, Month 1 is the starting month of the calendar
 1281 and the range of dayOfMonth may be different.

1282 The elements dayOfMonth and dayOfWeek shall be interpreted together:

- 1283 – if last dayOfMonth is specified (0xFE) and dayOfWeek is wildcard, this specifies the last
 1284 calendar day of the month;
- 1285 – if last dayOfMonth is specified (0xFE) and an explicit dayOfWeek is specified (for example
 1286 7, Sunday) then it is the last occurrence of the weekday specified in the month, i.e. the last
 1287 Sunday;
- 1288 – if the year is not specified (0xFFFF), and dayOfMonth and dayOfWeek are both explicitly
 1289 specified, this shall be interpreted as the dayOfWeek on, or following dayOfMonth;
- 1290 – if the year and month are specified, and both the dayOfMonth and dayOfWeek are explicitly
 1291 specified but the values are not consistent it shall be considered as an error.

1292 Examples:

- 1293 1) year = 0xFFFF, month = 0x FF, dayOfMonth = 0xFE, dayofWeek = 0xFF: last day of the month in every year
 1294 and month;
- 1295 2) year = 0xFFFF, month = 0x FF, dayOfMonth = 0xFE, dayofWeek = 0x07: last Sunday in every year and month;
- 1296 3) year = 0xFFFF, month = 0x03, dayOfMonth = 0xFE, dayofWeek = 0x07: last Sunday in March in every year;
- 1297 4) year = 0xFFFF, month = 0x03, dayOfMonth = 0x01, dayofWeek = 0x07: first Sunday in March in every year;
- 1298 5) year = 0xFFFF, month = 0x03, dayOfMonth = 0x16, dayofWeek = 0x05: fourth Friday in March in every year;
- 1299 6) year = 0xFFFF, month = 0x0A, dayOfMonth = 0x16, dayofWeek = 0x07: fourth Sunday in October in every
 1300 year;
- 1301 7) year = 0x07DE, month = 0x08, dayOfMonth = 0x13, (2014.08.13, Wednesday) dayofWeek = 0x02 (Tuesday):
 1302 error, as the dayOfMonth and dayOfWeek in the given year and month do not match.

1303

1304 **time** OCTET STRING (SIZE(4))

1305 {
 1306 hour,
 1307 minute,
 1308 second,
 1309 hundredths
 1310 }

1311 Where:

1313

1314 hour:	interpreted as unsigned
	range 0...23, 0xFF,
1316 minute:	interpreted as unsigned
	range 0...59, 0xFF,
1319 second:	interpreted as unsigned

1321 range 0...59, 0xFF,
1322
1323 hundredths: interpreted as unsigned
1324 range 0...99, 0xFF
1325
1326 For hour, minute, second and hundredths: 0xFF = not specified.
1327
1328 For repetitive times the unused parts shall be set to “not specified”.
1329
1330
1331 *date-time* OCTET STRING (SIZE(12))
1332 {
1333 year highbyte,
1334 year lowbyte,
1335 month,
1336 day of month,
1337 day of week,
1338 hour,
1339 minute,
1340 second,
1341 hundredths of second,
1342 deviation highbyte,
1343 deviation lowbyte,
1344 clock status
1345 }
1346
1347 The elements of *date* and *time* are encoded as defined above. Some may be set to “not specified”
1348 as defined above.
1349
1350 In addition:
1351
1352 deviation: interpreted as long
1353 range -720...+720 in minutes of local time to UTC
1354 0x8000 = not specified
1355
1356 Deviation highbyte and deviation lowbyte represent the 2 bytes of the long.
1357
1358 *clock_status* interpreted as unsigned. The bits are defined as follows:
1359
1360 bit 0 (LSB): invalid ^a value,
1361 bit 1: doubtful ^b value,
1362 bit 2: different clock base ^c,
1363 bit 3: invalid clock status ^d,
1364 bit 4: reserved,
1365 bit 5: reserved,
1366 bit 6: reserved,
1367 bit 7 (MSB): daylight saving active ^e
1368
1369 0xFF = not specified
1370
1371 NOTE a Time could not be recovered after an incident. Detailed conditions are manufacturer specific (for
1372 example after the power to the clock has been interrupted). For a valid status, bit 0 shall not be set if
1373 bit 1 is set.
1374 NOTE b Time could be recovered after an incident but the value cannot be guaranteed. Detailed conditions are
1375 manufacturer specific. For a valid status, bit 1 shall not be set if bit 0 is set.
1376 NOTE c Bit is set if the basic timing information for the clock at the actual moment is taken from a timing source
1377 different from the source specified in *clock_base*.
1378 NOTE d This bit indicates that at least one bit of the clock status is invalid. Some bits may be correct. The
1379 exact meaning shall be explained in the manufacturer’s documentation.

1380 e Flag set to true: the transmitted time contains the daylight saving deviation (summer time).

1381 Flag set to false: the transmitted time does not contain daylight saving deviation (normal time).

1382

1383 4.1.6.2 Floating point number formats

1384 Floating point number formats are defined in ISO/IEC 14908-1:2012, Interconnection **of**
1385 *information technology equipment – Control network protocol Part Protocol stack*

1386 ISO/IEC/IEEE 60559:2011.

1387 The single format is:

1	8	23	...widths
s	e	F	

msb
lsb msb
 lsb

...order

1388

1389 where:

1390 s is the sign bit;

1391 e is the exponent; it is 8 bits wide and the exponent bias is +127;

1392 f is the fraction, it is 23 bits.

1393 With this, the value is (if $0 < e < 255$):

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

1395 The double format is:

1	11	52	...widths
s	e	F	

msb
lsb msb
 lsb

...order

1396

1397 where:

1398 s is the sign bit;

1399 e is the exponent; it is 11 bits wide and the exponent bias is +1 023;

1400 f is the fraction, it is 52 bits.

1401 With this, the value is (if $0 < e < 2 047$):

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

1403 For details, see ISO/IEC 14908-1:2012, Interconnection **of information technology equipment –**
1404 *Control network protocol Part Protocol stack*

1405 ISO/IEC/IEEE 60559:2011.

1406 Floating-point numbers shall be represented as a fixed length octet-string, containing the 4
1407 bytes (float32) of the single format or the 8 bytes (float64) of the double format floating-point
1408 number as specified above, most significant byte first.

EXAMPLE 1 The decimal value "1" represented in single floating-point format is:

Bit 31	Bits 30-23	Bits 22-0
Sign bit	Exponent field: 01111111	Significand
0	Decimal value of exponent field	1.0000000000000000000000000000
0: +	and exponent: 127-127 = 0	Decimal value of the significand: 1.0000000
1: -		

1410

1411 NOTE The significand is the binary number 1 followed by the radix point followed by the binary bits of the fraction.

1412 The encoding, including the tag of the data type is (all values are hexadecimal): 17 3F 80 00 00.

1413 EXAMPLE 2 The decimal value “1” represented in double floating-point format is:

1414

1415 The encoding, including the tag of the data type is (all values are hexadecimal):
1416 18 3F F0 00 00 00 00 00 00 00.

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

Bit 31	Bits 30-23	Bits 22-0
Sign bit	Exponent field: 10001110	Significand
0	Decimal value of exponent field	1.111001001101000000000000000
0: +	and exponent: 142-127 = 15	Decimal value of the significand: 1.8937988
1: -		

1418

1419 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:

1421

1422 The encoding, including the tag of the data type is (all values are hexadecimal): 18 40 EE 4D 00 00 00 00 00 00.

4.1.6.3 Null- data and delta-value encoding

1424 Null-data encoding and delta-value encoding are two methods to improve efficiency when array
1425 or arrays of structures have to be transferred, for example Profile generic IC buffer attribute or
1426 Register table IC table cell values attribute.

With null-data encoding, the value 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).

With delta-value encoding the original value may be replaced by a delta-value as follows:

- the first value shall be encoded using an appropriate integer or unsigned data type;
 - subsequent values shall be encoded the same way, or they may be replaced by a delta value encoded using an appropriate delta data type.

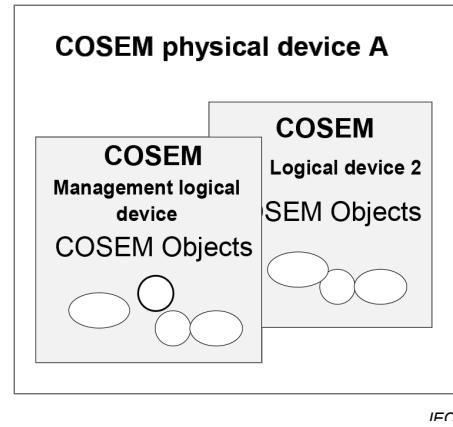
1434 The two methods can be used individually or combined and they can be used with compression
1435 to further improve efficiency.

14,36

1437 4.1.7 The COSEM server model

1438 The COSEM server is structured into three hierarchical levels as shown in Figure 3:

- Level 1: Physical device
 Level 2: Logical device
 Level 3: Accessible COSEM objects



1439

Figure 3 – The COSEM server model

1440 The example in Figure 4 shows how a combined metering device can be structured using the
 1441 COSEM server model.

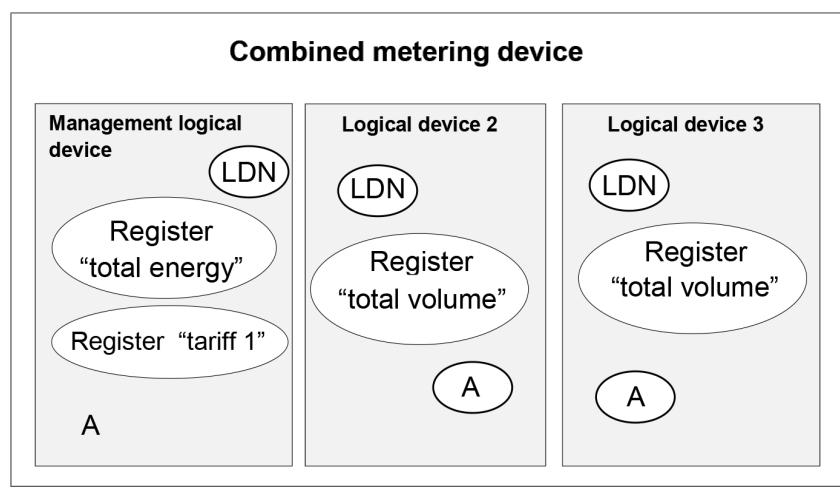
Physical device

Logical device

Objects

LDN: COSEM logical device name object

A: Association object



1442

Figure 4 – Combined metering device

1444 4.1.8 The COSEM logical device

1445 4.1.8.1 General

1446 The COSEM logical device contains a set of COSEM objects. Each physical device shall contain
 1447 a “Management logical device”.

1448 The addressing of COSEM logical devices shall be provided by the addressing scheme of the
 1449 lower layers of the protocol stack used.

1450 4.1.8.2 COSEM logical device name (LDN)

1451 The COSEM logical device can be identified by its unique COSEM LDN. This name can be
 1452 retrieved from an instance of IC “SAP assignment” (see 4.4.5), or from a COSEM object
 1453 “COSEM logical device name” (see 6.2.35).

1454 The LDN is defined as an octet-string of up to 16 octets. The first three octets shall carry the
 1455 manufacturer identifier². The manufacturer shall ensure that the LDN, starting with the three
 1456 octets identifying the manufacturer and followed by up to 13 octets, is unique.

1457 **4.1.8.3 The “association view” of the logical device**

1458 In order to access COSEM objects in the server, an application association (AA) shall first be
 1459 established with a client. AAs identify the partners and characterize the context within which
 1460 the associated applications will communicate. The major parts of this context are:

- 1461 • the application context;
 1462 • the authentication mechanism;
 1463 • the xDLMS context.

1464 AAs are modelled by special COSEM objects:

- 1465 • instances of the IC “Association SN” – see 4.4.3 – are used with short name referencing;
 1466 • instances of the IC “Association LN” – see 4.4.4 – are used with logical name referencing.

1467 Depending on the AA established between the client and the server, different access rights may
 1468 be granted by the server. Access rights concern a set of COSEM objects – the visible objects
 1469 – that can be accessed (‘seen’) within the given AA. In addition, access to attributes and
 1470 methods of these COSEM objects may also be restricted within the AA (for example a certain
 1471 type of client can only read a particular attribute of a COSEM object, but cannot write it). Access
 1472 right may also stipulate required cryptographic protection.

1473 The list of the visible COSEM objects – the “association view” – can be obtained by the client
 1474 by reading the *object_list* attribute of the appropriate association object.

1475 **4.1.8.4 Mandatory contents of a COSEM logical device**

1476 The following objects shall be present in each COSEM logical device. They shall be accessible
 1477 for GET/Read in all AAs with this logical device:

- 1478 • COSEM logical device name object;
 1479 • current “Association” (LN or SN) object.

1480 If the “SAP Assignment” object is present, then the COSEM logical device name object does
 1481 not have to be present.

1482 For identifying the firmware the following objects are mandatory:

- 1483 • an active firmware identifier object that holds the identifier of the currently active
 firmware;
- 1485 • an active firmware signature object that holds the digital signature of the currently
 active firmware.

1487 Note : The digital signature algorithm is not specified here.

1488 If a Logical Device has multiple firmwares then an active firmware identifier object and an
 1489 active firmware signature object shall be present for each firmware.

² Administered by the DLMS User Association, in cooperation with the FLAG Association.

1490 The following objects may be optionally present:

- 1491 • one or more active firmware version object(s) that hold(s) the version of the currently
1492 active firmware.

1493

1494 **4.1.8.5 Management logical device**

1495 As specified in 4.1.8.1, the management logical device is a mandatory element of any physical
1496 device. It has a reserved address. It shall support an AA to a public client with the lowest level
1497 security (no security) authentication. Its role is to support revealing the internal structure of the
1498 physical device and to support notification of events in the server.

1499 In addition to the “Association” object modelling the AA with the public client, the management
1500 logical device shall contain a “SAP assignment” object, giving its SAP and the SAP of all other
1501 logical devices within the physical device. The SAP assignment object shall be readable at least
1502 by the public client.

1503 If there is only one logical device within the physical device, the “SAP assignment” object may
1504 be omitted.

1505 **4.1.9 Information security**

1506 DLMS/COSEM provides several information security features for accessing and transporting
1507 data:

- 1508 • data access security controls access to the data held by a DLMS/COSEM server;
- 1509 • data transport security allows the sending party to apply cryptographic protection to the
1510 xDLMS APDUs sent. This requires ciphered APDUs. The receiving party can remove or
1511 check this protection;
- 1512 • COSEM data security allows protecting COSEM attribute values, as well as method
1513 invocation and return parameters.

1514 For a description of these security mechanisms, see IEC 62056-5-3:2021 Clause 5.

1515 Information security is provided on the DLMS/COSEM Application layer level and on the
1516 COSEM object level and it is supported / managed by the following objects:

- 1517 • “Association SN”, see 4.4.3;
- 1518 • “Association LN”, see 4.4.4; and
- 1519 • “Security setup”, see 4.4.7;
- 1520 • “Data protection”, see 4.4.9.

1521

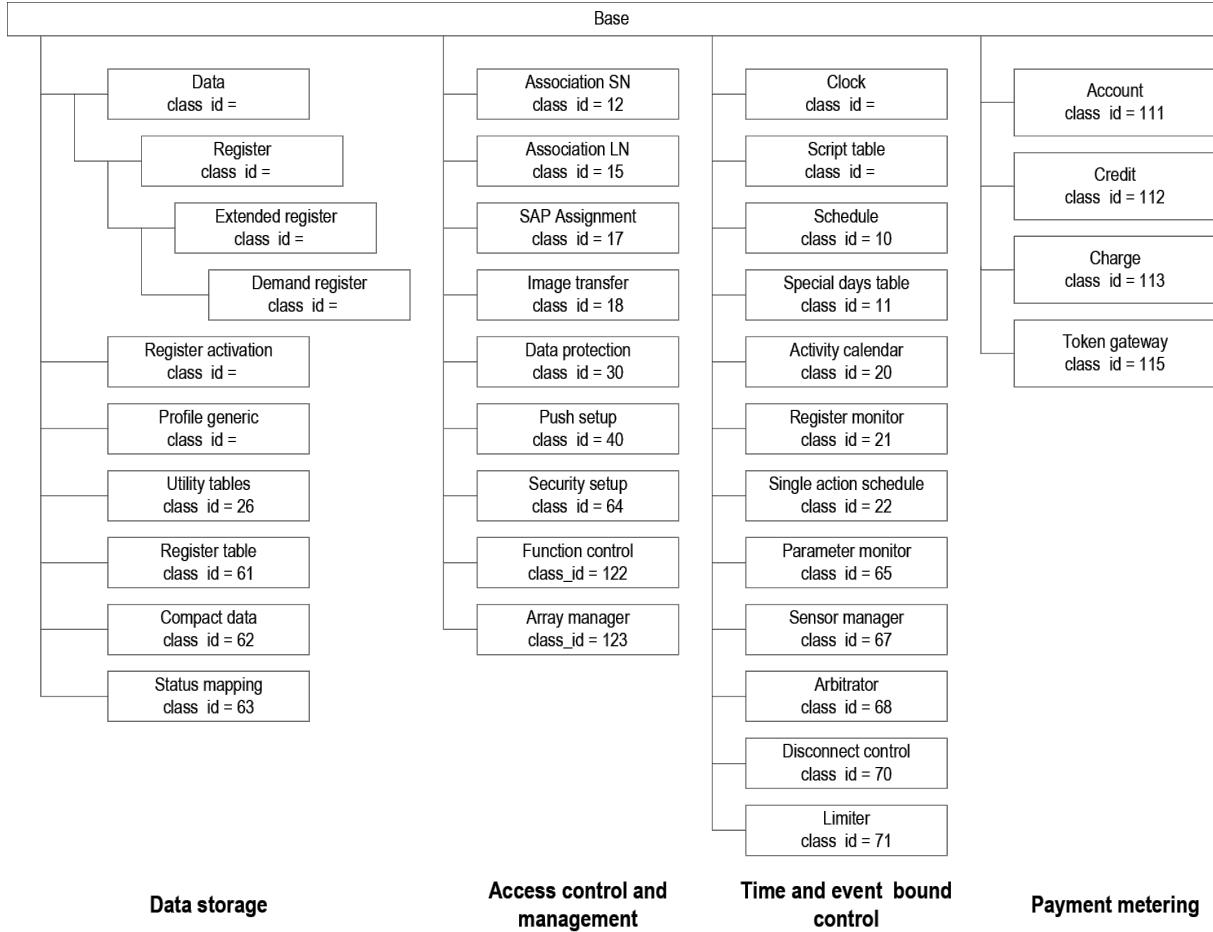
1522 **4.2 Overview of the COSEM interface classes**

1523 The ICs defined currently and the relations between them are shown in Figure 5 and Figure 6.

1524 NOTE 1 The IC “base” itself is not specified explicitly. It contains only one attribute *logical_name*.

1525 NOTE 2 In the description of the “Demand register”, “Clock” and “Profile generic” ICs, the 2nd attributes are labelled
1526 differently from that of the 2nd attribute of the “Data” IC, namely *current_average_value*, *time* and *buffer* vs. *value*.
1527 This is to emphasize the specific nature of the *value*.

1528 NOTE 3 On these Figures the interface classes are presented in each group by increasing class_id. In the clauses
1529 specifying the various groups of interface classes, the new interface classes are put at the end of the relevant clause.



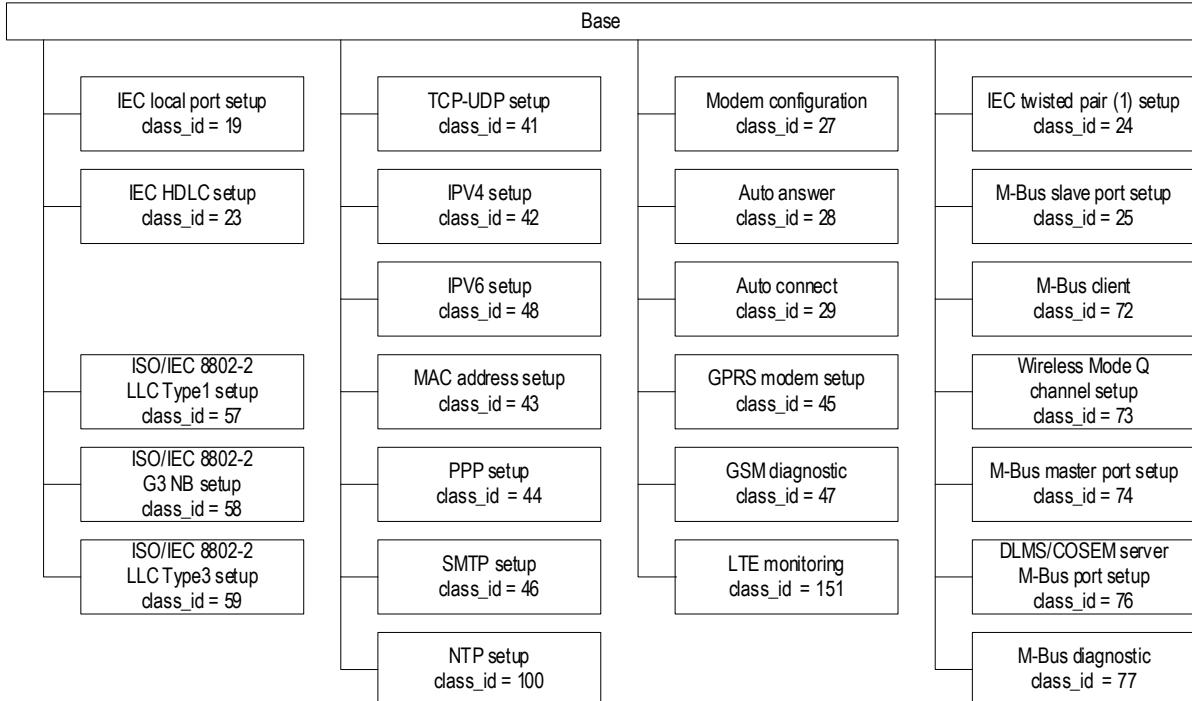
1530

IFC

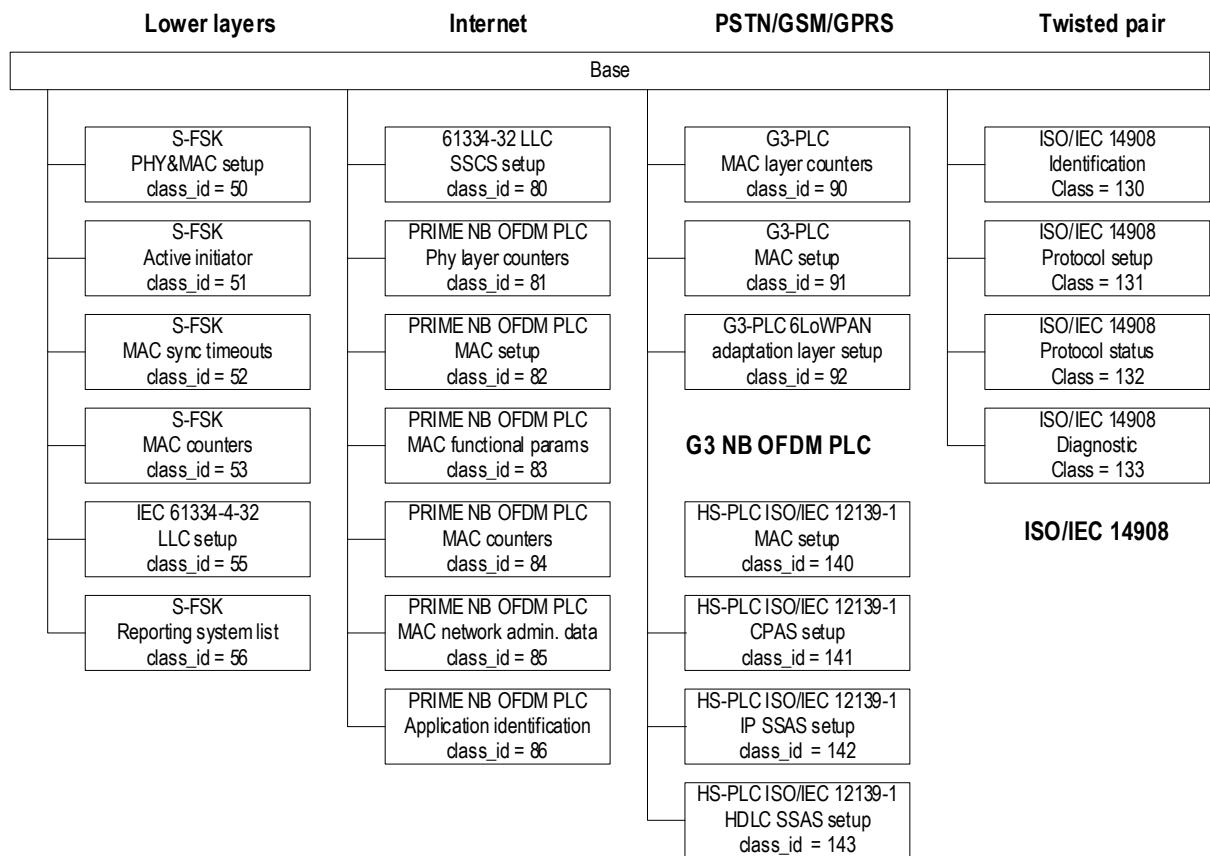
1531

Figure 5 – Overview of the interface classes – Part 1

1532



1533



1534

S-FSK PLC

PRIME NB OFDM PLC

HS-PLC ISO/IEC 12139-1

1535

Figure 6 – Overview of the interface classes – Part 2

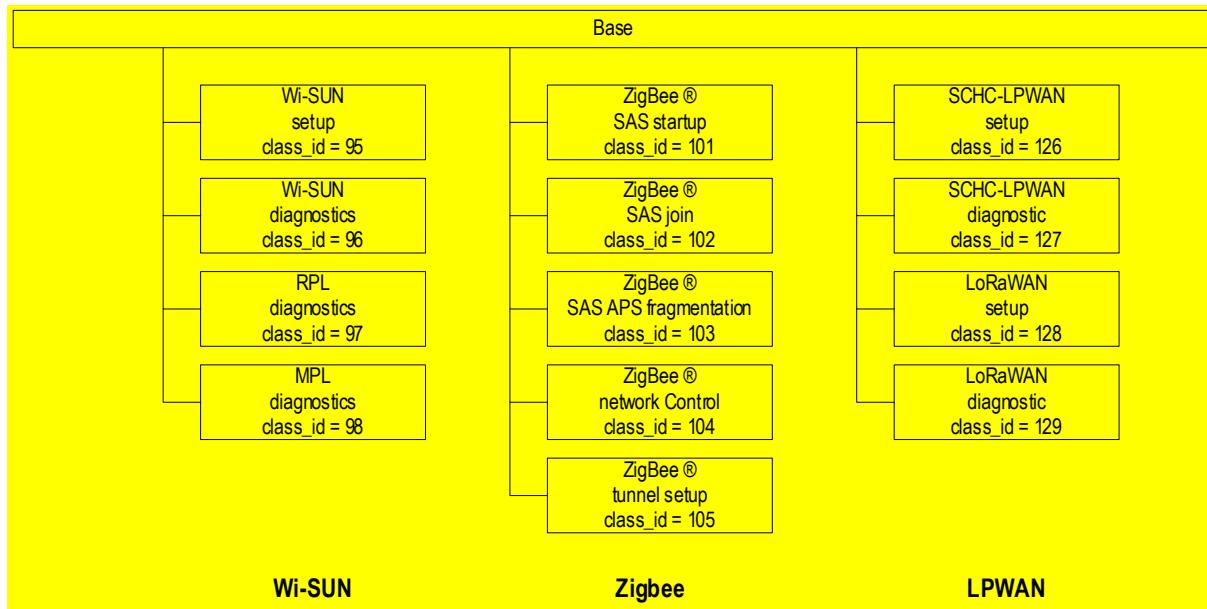


Figure 7 – Overview of the interface classes - Part 3

1538

1539

1539 Table 4 lists the interface classes by class_id.

Table 4 – List of interface classes by class_id

Interface class name	class_id	version(s)	Clause
Data	1	0	4.3.1
Register	3	0	4.3.2
Extended register	4	0	4.3.3
Demand register	5	0	4.3.4
Register activation	6	0	4.3.5
Profile generic	7	1 0	4.3.6 5.2.1
Clock	8	0	4.5.1
Script table	9	0	4.5.2
Schedule	10	0	4.5.3
Special days table	11	0	4.5.4
Association SN	12	4	4.4.3
		3	5.3.4
		2	5.3.3
		1	5.3.2
		0	5.3.1
Association LN	15	3	4.4.4
		2	5.3.7
		1	5.3.6
		0	5.3.5
SAP Assignment	17	0	4.4.5
Image transfer	18	0	4.4.6

Interface class name	class_id	version(s)	Clause
IEC local port setup	19	1 0	4.7.1 5.6.1
Activity calendar	20	0	4.5.5
Register monitor	21	0	4.5.6
Single action schedule	22	0	4.5.7
IEC HDLC setup	23	1 0	4.7.2 5.6.2
IEC twisted pair (1) setup	24	1 0	4.7.3 5.6.3
M-BUS slave port setup	25	0	4.8.2
Utility tables	26	0	4.3.7
Modem configuration	27	1	4.7.4
PSTN modem configuration		0	5.6.4
Auto answer	28	2 1 0	4.7.5 – 5.6.5
Auto connect	29	2 1 0	4.7.6 5.6.7 5.6.6
PSTN Auto dial		0	5.6.6
Data protection	30	0	4.4.9
Push setup	40	2 1 0	4.4.8.2 5.3.10 5.3.9
TCP-UDP setup	41	0	4.9.1
IPv4 setup	42	0	4.9.2
MAC address setup (Ethernet setup)	43	0	4.9.4 4.12.10
PPP setup	44	0	4.9.5
GPRS modem setup	45	0	4.7.7
SMTP setup	46	0	4.9.6
GSM diagnostic	47	2 1 0	4.7.8 5.6.9 5.6.8
IPv6 setup	48	0	4.9.3
S-FSK Phy&MAC setup	50	1 0	4.10.3 5.9.1
S-FSK Active initiator	51	0	4.10.4
S-FSK MAC synchronization timeouts	52	0	4.10.5
S-FSK MAC counters	53	0	4.10.6
IEC 61334-4-32 LLC setup		1	4.10.7
S-FSK IEC 61334-4-32 LLC setup	55	0	5.9.2
S-FSK Reporting system list	56	0	4.10.8
ISO/IEC 8802-2 LLC Type 1 setup	57	0	4.11.2
ISO/IEC 8802-2 LLC Type 2 setup	58	0	4.11.3

Interface class name	class_id	version(s)	Clause
ISO/IEC 8802-2 LLC Type 3 setup	59	0	4.11.4
Register table	61	0	4.3.8
Compact data	62	1 0	4.3.10 5.2.2
Status mapping	63	0	4.3.9
Security setup	64	1 0	4.4.7 5.3.8
Parameter monitor	65	1 0	4.5.10 5.4.1
Sensor manager	67	0	4.5.11
Arbitrator	68	0	4.5.12
Disconnect control	70	0	4.5.8
Limiter	71	0	4.5.9
M-Bus client	72	1 0	4.8.3 5.7.1
Wireless Mode Q channel	73	0	4.8.4
M-Bus master port setup	74	0	4.8.5
DLMS/COSEM server M-Bus port setup	76	0	4.8.6
M-Bus diagnostic	77	0	4.8.7
61334-4-32 LLC SSCS setup	80	0	4.12.3
PRIME NB OFDM PLC Physical layer counters	81	0	4.12.5
PRIME NB OFDM PLC MAC setup	82	0	4.12.6
PRIME NB OFDM PLC MAC functional parameters	83	0	4.12.7
PRIME NB OFDM PLC MAC counters	84	0	4.12.8
PRIME NB OFDM PLC MAC network administration data	85	0	4.12.9
PRIME NB OFDM PLC Application identification	86	0	4.12.11
G3-PLC MAC layer counters	90	1 0	4.13.3 5.12.3
G3 NB OFDM PLC MAC layer counters			
G3-PLC MAC setup		2	4.13.4
G3-PLC MAC setup	91	1 0	5.12.5 5.12.3
G3 NB OFDM PLC MAC setup			
G3-PLC 6LoWPAN adaptation layer setup		2	4.13.5
G3-PLC 6LoWPAN adaptation layer setup	92	1 0	5.12.6 5.12.4
Wi-SUN setup	95	0	4.18.1
Wi-SUN diagnostic	96	0	4.18.2
RPL diagnostic	97	0	4.18.3
MPL diagnostic	98	0	4.18.4
NTP Setup	100	0	4.9.7
ZigBee® SAS startup	101	0	4.15.2
ZigBee® SAS join	102	0	4.15.3
ZigBee® SAS APS fragmentation	103	0	4.15.4
ZigBee® network control	104	0	4.15.5
ZigBee® tunnel setup	105	0	4.15.6

Interface class name	class_id	version(s)	Clause
Account	111	0	4.6.2
Credit	112	0	4.6.3
Charge	113	0	4.6.4
Token gateway	115	0	4.6.5
Function control	122	0	4.4.10
Array manager	123	0	4.4.11
Communication port protection	124	0	4.4.12
SCHC-LPWAN setup	126	0	4.16.2.1
SCHC-LPWAN diagnostic	127	0	4.16.2.2
LoRaWAN setup	128	0	4.17.2.2
LoRaWAN diagnostic	129	0	4.17.2.3
ISO/IEC14908 Identification	130	0	4.19.2
ISO/IEC 14908 Protocol setup	131	0	4.19.3
ISO/IEC 14908 protocol status	132	1	4.19.4
ISO/IEC 14908 diagnostic	133	1	4.19.5
HS-PLC ISO/IEC 12139-1 MAC setup	140	0	4.14.2
HS-PLC ISO/IEC 12139-1 CPAS setup	141	0	4.14.3
HS-PLC ISO/IEC 12139-1 IP SSAS setup	142	0	4.14.4
HS-PLC ISO/IEC 12139-1 HDLC SSAS setup	143	0	4.14.5
LTE monitoring	151	1	4.7.9
		0	5.6.10

1541

1542

1543 **4.3 Interface classes for parameters and measurement data**1544 **4.3.1 Data (class_id = 1, version = 0)**1545 **4.3.1.1 Overview**

1546 This IC allows modelling various data, such as configuration data and parameters. The data are
 1547 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
<i>Specific methods</i>	m/o				

1548

1549 **4.3.1.2 Attribute description**1550 **4.3.1.2.1 logical_name**

1551 Identifies the “Data” object instance. See Clause 6 and IEC 62056-6-1:2021.

1552 **4.3.1.2.2 value**

1553 Contains the data.

1554 The data type depends on the instantiation defined by the *logical_name* and possibly from the
 1555 manufacturer. It has to be chosen so, that together with the *logical_name*, an unambiguous
 1556 interpretation is possible. Any simple and complex data types listed in 4.1.5 can be used, unless
 1557 the choice is restricted in Clause 6.

1558

1559 CHOICE

```

1560 {
  -- simple data types

 1562   null-data          [0],
 1563   boolean            [3],
 1564   bit-string         [4],
 1565   double-long        [5],
 1566   double-long-unsigned [6],
 1567   octet-string       [9],
 1568   visible-string     [10],
 1569   utf8-string        [12],
 1570   bcd                [13],
 1571   integer             [15],
 1572   long                [16],
 1573   unsigned            [17],
 1574   long-unsigned       [18],
 1575   long64              [20],
 1576   long64-unsigned     [21],
 1577   enum                [22],
 1578   float32             [23],
 1579   float64             [24],
 1580   date-time           [25],
 1581   date                [26],
 1582   time                [27],

```

```

1583     delta-integer      [28],
1584     delta-long         [29],
1585     delta-double-long   [30],
1586     delta-unsigned      [31],
1587     delta-long-unsigned [32],
1588     delta-double-long-unsigned [33],
1589
1590     -- complex data types
1591     array               [1],
1592     structure            [2],
1593     compact-array         [19]
1594   }

```

1595

1596

1597 **4.3.2 Register (class_id = 3, version = 0)**1598 **4.3.2.1 Overview**

1599 This IC allows modelling a process or a status value with its associated scaler and unit.
1600 “Register” objects know the nature of the process or status value. It is identified by the attribute
1601 *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				

1602

1603 **4.3.2.2 Attribute description**1604 **4.3.2.2.1 logical_name**

1605 Identifies the “Register” object instance. See Clause 6 and
1606 IEC 62056-6-1:**2021**.

1607 **4.3.2.2.2 value**

1608 Contains the current process or status value.

1609 The data type of the value depends on the instantiation defined by *logical_name* and possibly
1610 on the choice of the manufacturer. It has to be chosen so that, together with the *logical_name*,
1611 an unambiguous interpretation of the value is possible.

1612 When a “Register” object is used instead of a “Data” object, (with the *scaler_unit* attribute not
1613 used or with *scaler* = 0, *unit* = 255) then the data types allowed for the *value* attribute of the
1614 “Data” IC are allowed.

```

1615           CHOICE
1616           {

```

```

1617          -- simple data types
1618          null-data      [0],
1619          bit-string     [4],
1620          double-long    [5],
1621          double-long-unsigned [6],
1622          octet-string   [9],
1623          visible-string [10],
1624          utf8-string    [12],
1625          integer        [15],
1626          long           [16],
1627          unsigned       [17],
1628          long-unsigned   [18],
1629          long64          [20],
1630          long64-unsigned [21],
1631          enum           [22],
1632          float32         [23],
1633          float64         [24],
1634      }

```

1635 4.3.2.2.3 **scaler_unit**

1636 Provides information on the unit and the scaler of the value.

```

1637
1638          scal_unit_type ::= structure
1639          {
1640          scaler,
1641          unit
1642          }
1643
1644          scaler:    integer
1645

```

1646 This is the exponent (to the base of 10) of the multiplication factor.

1647 REMARK If the value is not numerical, then the scaler shall be set to 0.

1648 See Table 6 for examples for **scaler_unit**.

```

1649
1650          unit:    enum
1651

```

1652 See Table 5 for values of unit.

1654 4.3.2.3 **Method description**

1655 4.3.2.3.1 **reset(data)**

1656 Forces a reset of the object. By invoking this method, the value is set to the default value. The
1657 default value is an instance specific constant.

```

1658          data ::= integer (0)
1659

```

1660

1661

1662 **Table 5 – Enumerated values for physical units**

unit::= enum	Unit	Quantity	Unit name	SI definition (comment)
(1)	a	time	year	

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

unit::=enum	Unit	Quantity	Unit name	SI definition (comment)
(40)	Wb	magnetic flux (ϕ)	weber	$\text{Wb} = \text{Vs}$
(41)	T	magnetic flux density (B)	tesla	Wb/m^2
(42)	A/m	magnetic field strength (H)	ampere per metre	A/m
(43)	H	inductance (L)	henry	$\text{H} = \text{Wb/A}$
(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	$\text{V}^2(60*60\text{s})$
(49)	A ² h	ampere-squared hour, r_{I2h} , ampere-squared hour meter constant or pulse value	ampere-squared-hours	$\text{A}^2(60*60\text{s})$
(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	$3,6*10^3 \text{ J/m}^3$	
(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	$\text{m}^2 \cdot \text{kg} \cdot \text{s}^{-2} / \text{kg}$ $= \text{m}^2 \cdot \text{s}^{-2}$
....				
(70)	dBm	signal strength, dB milliwatt (e.g. of GSM radio systems)		
(71)	db μ V	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		

unit::= enum	Unit	Quantity	Unit name	SI definition (comment)
^a Usage of these units (14, 16 & 18) is deprecated as OBIS codes specify a corrected volume (flux). This is to avoid contradiction of units associated with selected OBIS codes.				

1663

1664

Table 6 – Examples for scalar_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	3467 V
3467	1	V	34 670 V

1665

4.3.3 Extended register (class_id = 4, version = 0)**4.3.3.1 Overview**

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					

1671

4.3.3.2 Attribute description**4.3.3.2.1 logical_name**

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

4.3.3.2.2 value

See the specification of the IC "Register" in 4.3.3.2.2.

4.3.3.2.3 scaler_unit

See the specification of the IC "Register" in 4.3.3.2.3.

4.3.3.2.4 status

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

1682 The data type and the encoding depend on the instantiation and possibly on the choice of the
 1683 manufacturer. For the interpretation, extra information from the manufacturer may be necessary.

```

1684     CHOICE
1685     {
1686         -- simple data types
1687         null-data      [0],
1688         bit-string     [4],
1689         double-long-unsigned [6],
1690         octet-string   [9],
1691         visible-string [10],
1692         utf8-string    [12],
1693         unsigned       [17],
1694         long-unsigned   [18],
1695         long64-unsigned [21]
1696     }

```

1697 **Def.** Depending on the status type definition.

1698 **4.3.3.2.5 capture_time**

1699 Provides an “Extended register” specific date and time information showing when the value of
 1700 the attribute *value* has been captured.

1701 octet-string, formatted as specified in 4.1.6.1 for *date-time*.

1702 **4.3.3.3 Method description**

1703 **4.3.3.3.1 reset (data)**

1704 This method forces a reset of the object. By invoking this method, the attribute *value* is set to
 1705 the default value. The default value is an instance specific constant.

1706 The attribute *capture_time* is set to the time of the reset execution.

1707 data::= integer (0)

1708

1709

1710

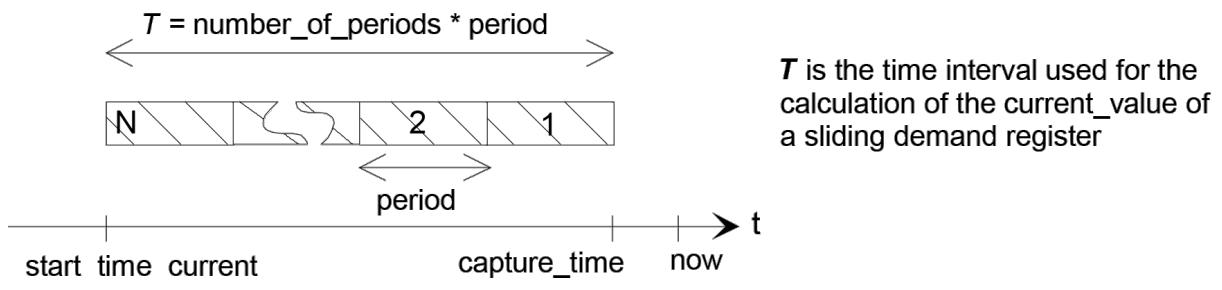
1711

1712

1713 **4.3.4 Demand register (class_id = 5, version = 0)**

1714 **4.3.4.1 Overview**

1715 This IC allows modelling a demand value with its associated scaler, unit, status and time
 1716 information. A “Demand register” object measures and computes a *current_average_value*
 1717 periodically and it stores a *last_average_value*. The time interval *T* over which the demand is
 1718 calculated is defined by specifying *number_of_periods* and *period*. See Figure 8.



1719

IEC

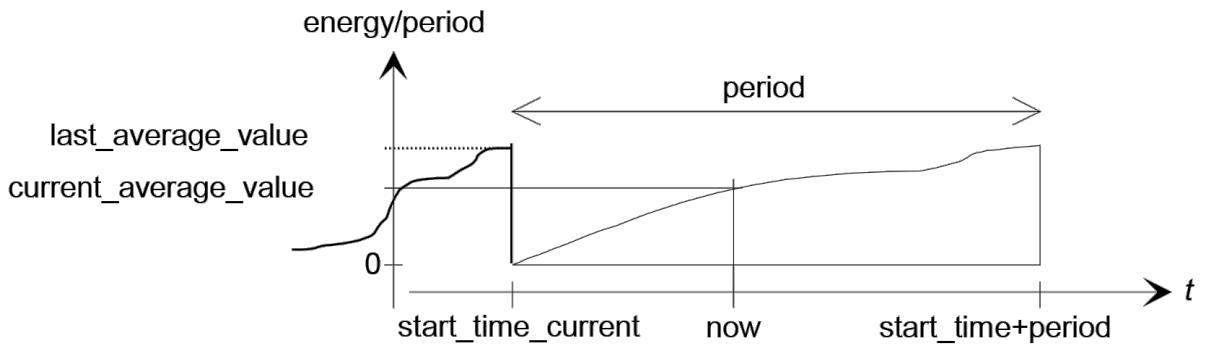
1720

Figure 8 – The time attributes when measuring sliding demand1721
1722

The demand register delivers two types of demand: *current_average_value* and *last_average_value* (see Figure 9 and Figure 10).

1723
1724

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

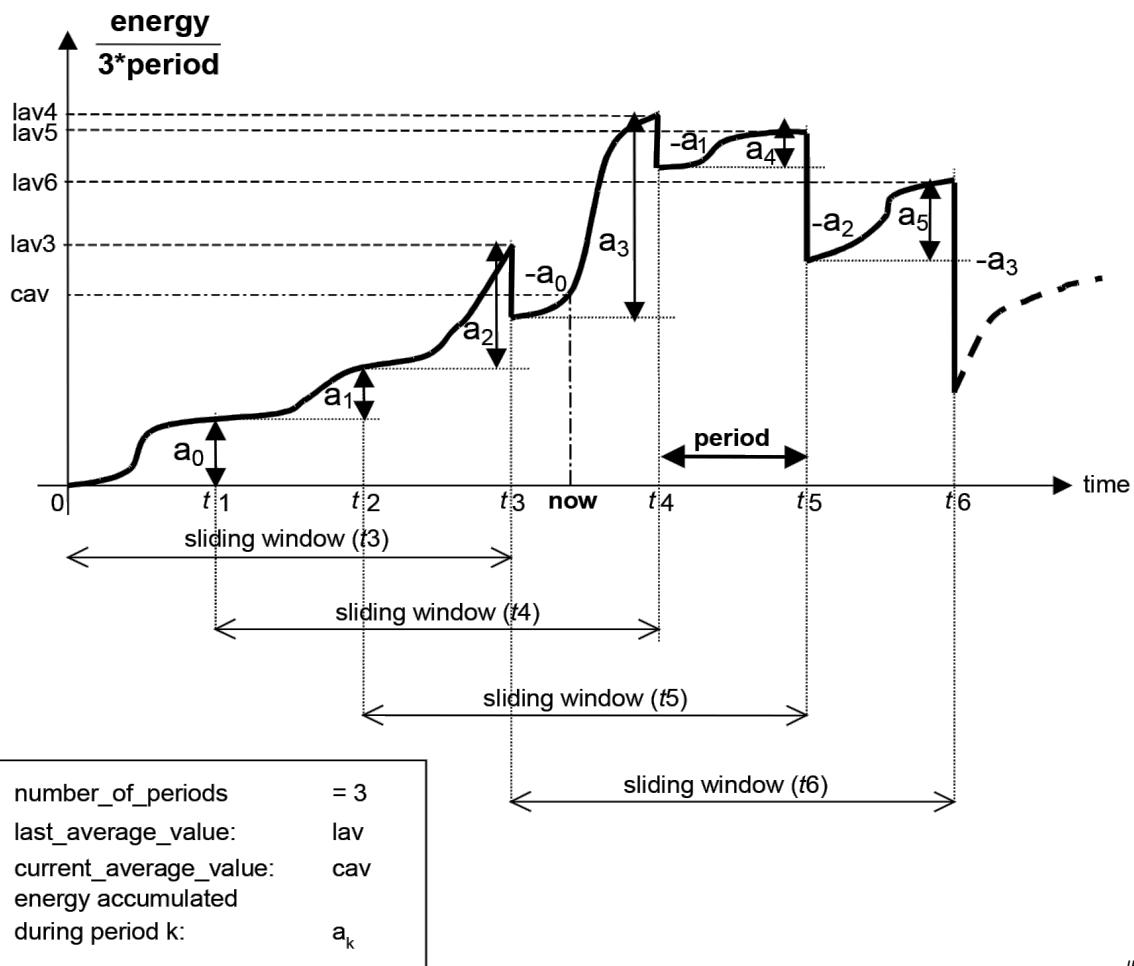


1725

IEC

1726

Figure 9 – The attributes in the case of block demand



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					
2. next_period (data)	o					

1731 **4.3.4.2 Attribute description**1732 **4.3.4.2.1 logical_name**

1733 Identifies the “Demand register” object instance. See 6.3.1 and IEC 62056-6-1:2021.

1734 **4.3.4.2.2 current_average_value**

1735 Provides the current value (running demand) of the energy accumulated since *start_time*,
1736 divided by *number_of_periods*period*.

1737 The data type of the value depends on the instantiation defined by *logical_name* and possibly
1738 on the choice of the manufacturer. The type has to be chosen so that – together with the
1739 *logical_name* – unambiguous interpretation of the value is possible.

1740 If a quantity other than energy is measured, other calculation methods may apply (for example
1741 for calculating average values of voltage or current).

1742 CHOICE For data types, see “Register” value attribute (4.3.2.2.2).

1743 **4.3.4.2.3 last_average_value**

1744 Provides the value of the energy accumulated (over the last *number_of_periods*period*) divided
1745 by *number_of_periods*period*. The energy of the current (not terminated) period is not
1746 considered by the calculation.

1747 If a quantity other than energy is measured, other calculation methods may apply (for example
1748 for calculating average values of voltage or current).

1749 CHOICE For data types, see “Register” value attribute (4.3.2.2.2).

1750 **4.3.4.2.4 scaler_unit**

1751 See the specification of IC “Register” in 4.3.2.2.3.

1752 **4.3.4.2.5 status**

1753 Provides “Demand register” specific status information. The data type and the encoding depend
1754 on the instantiation and possibly on the choice of the manufacturer. For the interpretation, extra
1755 information from the manufacturer may be necessary.

1756 CHOICE For data types, see “Extended register” IC status attribute (4.3.3.2.4).

1757 **Def.** Depending on the status type definition.

1758 **4.3.4.2.6 capture_time**

1759 Provides the date and time when the *last_average_value* has been calculated.

1760 octet-string, formatted as specified in 4.1.6.1 for date-time.

1761

1762

1763 **4.3.4.2.7 start_time_current_**

1764 Provides the date and time when the measurement of the current_average_value has been
1765 started.

1766 octet-string, formatted as specified in 4.1.6.1 for date-time.

1767 **4.3.4.2.8 period**

1768 Period is the interval between two successive updates of the last_average_value.
1769 (number_of_periods*period is the denominator for the calculation of the demand).

1770 double-long-unsigned Measuring period in seconds

1771 The behaviour of the meter after writing a new value to this attribute shall be specified by the
1772 manufacturer.

1773 **4.3.4.2.9 number_of_periods**

1774 The number of periods used to calculate the last_average_value. number_of_periods >= 1

- 1775 – number_of_periods > 1 indicates that the last_average_value represents "sliding demand",
- 1776 – number_of_periods = 1 indicates that the last_average_value represents "block demand".

1777 The behaviour of the meter after writing a new value to this attribute shall be specified by the
1778 manufacturer.

1779 **4.3.4.3 Method description**1780 **4.3.4.3.1 reset (data)**

1781 This method forces a reset of the object. Activating this method provokes the following actions:

- 1782 – the current period is terminated;
- 1783 – the current_average_value and the last_average_value are set to their default values;
- 1784 – the capture_time and the start_time_current are set to the time of the execution of reset
1785 (data).

1786 data::= integer (0)

1787 **4.3.4.3.2 next_period (data)**

1788 This method is used to trigger the regular termination (and restart) of a period. Closes
1789 (terminates) the current measuring period. Updates capture_time and start_time and copies
1790 current_average_value to last_average_value, sets current_average_value to its default value.
1791 Starts the next measuring period.

1792 REMARK The old last_average_value (and capture_time) can be read during the time "period".
1793 The old current_average_value is not available any more at the interface.

1794 data::= integer (0)

1795

1796

1797 **4.3.5 Register activation (class_id = 6, version = 0)**1798 **4.3.5.1 Overview**

1799 This IC allows modelling the handling of different tariffication structures. To each “Register
 1800 activation” object, groups of “Register”, “Extended register” or “Demand register” objects,
 1801 modelling different kind of quantities (for example active energy, active demand, reactive
 1802 energy, etc.) are assigned. Subgroups of these registers, defined by the *activation masks* define
 1803 different tariff structures (for example day tariff, night tariff). One of these activation masks, the
 1804 *active_mask*, defines which subset of the registers, assigned to the “Register activation” object
 1805 instance is active. Registers not included in the *register_assignment* attribute of any “Register
 1806 activation” object are always enabled by default.

1807

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				
2. add_mask (data)	o				
3. delete_mask (data)	o				

1808

1809 **4.3.5.2 Attribute description**1810 **4.3.5.2.1 logical_name**

1811 Identifies the “Register activation” object instance. See 6.2.11.

1812 **4.3.5.2.2 register_assignment**

1813 Specifies an ordered list of COSEM objects assigned to the “Register activation” object. The
 1814 list may contain different kinds of COSEM objects, for example instances of “Register”,
 1815 “Extended register” or “Demand register”.

1816 array object_definition

1817 object_definition::= structure

1818 {
 1819 class_id: long-unsigned,
 1820 logical_name: octet-string
 1821 }

1823 **4.3.5.2.3 mask_list**

1824 Specifies a list of register activation masks. Each entry (mask) is identified by its *mask_name*
 1825 and contains an array of indices referring to the registers assigned to the mask (the first object
 1826 in *register_assignment* is referenced by index 1, the second object by index 2,...).

1827 array register_act_mask

1828

```

1829          register_act_mask ::= structure
1830
1831          {
1832              mask_name:          octet-string,
1833              index_list:         index_array
1834          }
1835
1836          mask_name has to be uniquely defined within the object
1837          index_array ::= array    unsigned
1838

```

1839 **4.3.5.2.4 active_mask**

1840 Defines the currently active mask. The mask is defined by its mask_name (see mask_list,
1841 4.3.5.2.3).

1842 The active_mask defines the registers currently enabled; all other registers listed in the
1843 register_assignment are disabled.

1844 **4.3.5.3 Method description**

1845 **4.3.5.3.1 add_register (data)**

1846 Adds one more register to the attribute register_assignment. The new register is added at the
1847 end of the array; i.e. the newly added register has the highest index. The indices of the existing
1848 registers are not modified.

```

1849          data ::= structure
1850          {
1851              class_id:        long-unsigned,
1852              logical_name:   octet-string
1853          }

```

1854 **4.3.5.3.2 add_mask (data)**

1855 Adds another mask to the attribute mask_list. If there exists already a mask with the same name,
1856 the existing mask will be overwritten by the new mask.

1857 data ::= register_act_mask (see 4.3.5.3.1)

1858 **4.3.5.3.3 delete_mask (data)**

1859 Deletes a mask from the attribute mask_list. The mask is defined by its mask name.

1860 data ::= octet-string (mask_name)

1861

1862 **4.3.6 Profile generic (class_id = 7, version = 1)**

1863 **4.3.6.1 Overview**

1864 This IC provides a generalized concept allowing to store, sort and access data groups or data
1865 series, called *capture objects*. Capture objects are appropriate attributes or elements of (an)
1866 attribute(s) of COSEM objects. The capture objects are collected periodically or occasionally.

1867 A profile has a *buffer* to store the captured data. To retrieve only a part of the buffer, either a
1868 value range or an entry range may be specified, asking to retrieve all entries that fall within the
1869 range specified.

1870 The list of *capture objects* defines the values to be stored in the *buffer* (using auto capture or
 1871 the method *capture*). The list is defined statically to ensure homogenous buffer entries (all
 1872 entries have the same size and structure). If the list of capture objects is modified, the *buffer* is
 1873 cleared. If the buffer is captured by other “Profile generic” objects, their *buffer* is cleared as
 1874 well, to guarantee the homogeneity of their *buffer* entries.

1875 The *buffer* may be defined as sorted by one of the *capture objects*, e.g. the clock, or the entries
 1876 are stacked in a “last in first out” order. For example, it is very easy to build a “maximum demand
 1877 register” with a one entry deep sorted profile capturing and sorted by a “Demand register”
 1878 *last_average_value* attribute. It is just as simple to define a profile retaining the three largest
 1879 values of some period.

1880 The size of profile data is determined by three parameters:

- 1881 a) the number of entries filled (*entries_in_use*). This will be zero after clearing the profile;
- 1882 b) the maximum number of entries to retain (*profile_entries*). If all entries are filled and a
 1883 capture () request occurs, the least important entry (according to the requested sorting
 1884 method) will get lost. This maximum number of entries may be specified. Upon changing it,
 1885 the buffer will be adjusted;
- 1886 c) the physical limit for the buffer. This limit typically depends on the objects to capture. The
 1887 object will reject an attempt of setting the maximum number of entries that is larger than
 1888 physically possible.

1889

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			1	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				

1890

1891 **4.3.6.2 Attribute description**

1892 **4.3.6.2.1 logical_name**

1893 Identifies the “Profile generic” object instance. For examples, see 6.2.17, 6.2.19, 6.2.21, 6.2.22,
 1894 6.2.43, 6.2.46, 6.2.49, 6.2.50, 6.2.61, 6.2.62, 6.2.64, 6.2.65, 6.2.66, 6.2.67, 6.3.2, 6.3.6, etc.

1895 **4.3.6.2.2 buffer**

1896 Contains a sequence of entries. Each entry contains values of the captured objects.

```

1897           compact-array or array      entry
1898
1899           entry ::= structure
1900
1901           {
1902             CHOICE
1903           {
1904             -- simple data types
1905             null-data          [0],
1906             boolean            [3],
1907             bit-string         [4],
1908             double-long        [5],
1909             double-long-unsigned [6],
1910             octet-string       [9],
1911             visible-string    [10],
1912             utf8-string        [12],
1913             bcd                [13],
1914             integer            [15],
1915             long               [16],
1916             unsigned           [17],
1917             long-unsigned      [18],
1918             long64              [20],
1919             long64-unsigned    [21],
1920             enum               [22],
1921             float32            [23],
1922             float64            [24],
1923             date-time          [25],
1924             date               [26],
1925             time               [27],
1926             delta-integer      [28],
1927             delta-long          [29],
1928             delta-double-long   [30],
1929             delta-unsigned     [31],
1930             delta-long-unsigned [32],
1931             delta-double-long-unsigned [33],
1932
1933             -- complex data types
1934             array              [1],
1935             structure          [2],
1936             compact-array       [19]
1937           }
1938         }
1939

```

1940 The number and the order of the elements of the structure holding the entries is the same as in
1941 the definition of the *capture_objects*. The *buffer* is filled by auto captures or by subsequent calls
1942 of the method *capture*. The sequence of the entries within the array is ordered according to the
1943 *sort_method* specified.

1944 Default: The *buffer* is empty after reset.

1945 REMARK 1 Reading the entire *buffer* delivers only those entries, which are “in use”.

1946 REMARK 2 The value of a captured object may be replaced by “null-data” if it can be unambiguously recovered
1947 from the previous value (for example for time: if it can be calculated from the previous value and *capture_period*; or
1948 for a value: if it is equal to the previous value).

1949 *selective access* (see 4.1.4.16) to the attribute buffer may be available (optional). The selective
1950 access parameters are as defined in 4.3.6.2.9.

1951 4.3.6.2.3 capture_objects

1952 Specifies the list of capture objects that are assigned to the object instance.

Upon a call of the *capture* (data) method or automatically in defined intervals, the selected attributes are copied into the *buffer* of the profile.

```

array      capture_object_definition
capture_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 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.

```

4.3.6.2.4 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.

4.3.6.2.5 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 *capture* () 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 *capture* () 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 (first in first out),
        (2) lifo (last in first out),
        (3) largest,
        (4) smallest,
        (5) nearest_to_zero,
        (6) farthest_from_zero

```

Def. fifo

4.3.6.2.6 sort_object

If the profile is sorted, this attribute specifies the register or clock that the ordering is based upon.

capture_object_definition See 4.3.6.2.3

Def. no object to sort by (only possible with sort_method fifo or lifo)

2000 NOTE 1 If the sort_method is FIFO or LIFO, then all elements of the capture_object_definition specifying the sort
2001 object can be zero.

2002 4.3.6.2.7 entries_in_use

Counts the number of entries stored in the buffer. After a call of the *reset ()* method, the buffer does not contain any entries, and this value is zero. Upon each subsequent call of the *capture ()* method, this value will be incremented up to the maximum number of entries that will get stored (see *profile_entries*, 4.3.6.2.8).

2007

2008 double-long-unsigned 0...profile_entries

2009

224

2012 4.3.6.2.8 profile entries

2013 Specifies how many entries shall be retained in the buffer.

2014 double-long-unsigned 1...(limited by physical size)

2015 **Def.** 1

2016 4.3.6.2.9 Parameters for selective access to the buffer attribute

2017 Table 7 defines the parameters required for selective access to the buufer.

Table 7 – 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_object: capture_object_definition

(Defines the capture_object restricting the range of entries to be
retrieved. Only simple data types are allowed.)

from_value: (Oldest or smallest entry to retrieve)

2027

```
2029
2030 { -- simple data types
2031   double-long           [5],
2032   double-long-unsigned   [6],
2033   octet-string          [9],
2034   visible-string         [10],
2035   utf8-string            [12],
2036   integer                [15],
2037   long                   [16],
2038   unsigned               [17],
2039   long-unsigned          [18],
2040   long64                 [20],
2041   long64-unsigned        [21],
2042   float32                [23],
2043   float64                [24],
2044   date-time              [25]
```

```

2045           date          [26],  

2046           time          [27]  

2047           }  

2048  

2049           to_value:     (Newest or largest entry to retrieve)  

2050  

2051           CHOICE  

2052             {as for 'from_value' above}  

2053  

2054           selected_values: array capture_object_definition  

2055  

2056             (List of columns to retrieve. If the array is empty (has no entries), all  

2057             captured data are returned. Otherwise, only the columns specified in  

2058             the array are returned. The type capture_object_definition is specified  

2059             above (capture_objects)).  

2060           }  

2061  

2062           entry_descriptor ::= structure  

2063             {  

2064               from_entry:      double-long-unsigned first entry to retrieve,  

2065               to_entry:        double-long-unsigned last entry to retrieve  

2066               (if to_entry == 0: highest possible entry),  

2067  

2068               from_selected_value: long-unsigned index of first value to retrieve,  

2069               to_selected_value:  long-unsigned index of last value to retrieve  

2070               (if to_selected_value == 0: highest possible selected_value)  

2071             }

```

2073 NOTE 2 from_entry and to_entry identify the lines, from_selected_value to_selected_value identify the columns of
2074 the buffer to be retrieved.

2075 NOTE 3 Numbering of entries and selected values starts from 1.

2076

2077 4.3.6.3 Method description

2078 4.3.6.3.1 reset (data)

2079 Clears the *buffer*. It has no valid entries afterwards; *entries_in_use* is zero after this call. This
2080 call does not trigger any additional operations on the capture objects. Specifically, it does not
2081 reset any attributes captured.

2082 data ::= integer (0)

2083 4.3.6.3.2 capture(data)

2084 Copies the values of the objects to capture into the *buffer* by reading each capture object.
2085 Depending on the *sort_method* and the actual state of the *buffer* this produces a new entry or
2086 a replacement for the less significant entry. As long as not all entries are already used, the
2087 *entries_in_use* attribute will be incremented.

2088 This call does not trigger any additional operations within the capture objects such as *capture*
2089 () or *reset* ().

2090 Note, that if more than one attribute of an object need to be captured, they have to be defined
2091 one by one on the list of *capture_objects*. If the attribute_index = 0, all attributes are captured.

2092 data ::= integer (0)

2093

2094

2095 **4.3.6.4 Behaviour of the object after modification of certain attributes**

2096 Any modification of one of the *capture_objects* describing the static structure of the *buffer* will
 2097 automatically call a *reset ()* and this call will propagate to all other profiles capturing this profile.

2098 If writing to *profile_entries* is attempted with a value too large for the buffer, it will be rejected.

2099 **4.3.6.5 Restrictions**

2100 When defining the *capture objects*, circular reference to the profile shall be avoided.

2101 **4.3.6.6 Profile used to define a subset of preferred readout values**

2102 By setting *profile_entries* to 1, a “Profile generic” object can be used to define a set of preferred
 2103 readout values. See also 6.2.19. Setting *capture_period* to 1 ensures that the values are
 2104 updated every second.

2105 **4.3.7 Utility tables (class_id = 26, version = 0)**2106 **4.3.7.1 Overview**

2107 This IC allows encapsulating ANSI C12.19 table data. Each “table” is represented by an
 2108 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				

2109 **4.3.7.2 Attribute description**2110 **4.3.7.3 logical_name**

2111 Identifies the “Utility tables” object instance. See 6.2.40.

2112 **4.3.7.4 table_ID**

2113 Table number. This table number is as specified in the ANSI standard and may be either a
 2114 standard table or a manufacturer’s table.

2115 **4.3.7.5 length**

2116 Number of octets in table buffer.

2117 **4.3.7.6 buffer**

2118 Contents of the table.

2119 Selective access (see 4.1.4.16) to the attribute *buffer* may be available (optional). The selective
 2120 access parameters are defined in Table 8.

2121

2122

2123

2124

Table 8 – 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.

2125

```

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-unsigned   Number of elements requested or transferred. Values of count
                        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.
}

```

2126

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

4.3.8.1 Overview

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:**2021** 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 Table 17 or the “UNIPEDE voltage dip quantities” table, see Table 19.

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

2138

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

2139

2140 4.3.8.2 Attribute description

2141 4.3.8.2.1 logical_name

2142 Identifies the “Register table” object instance.

When the format of the *logical_name* 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 *value* attribute).

When the format of the *logical_name* 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 62056-6-1:2021, 6.4.

2149 4.3.8.2.2 table_cell_values

2150 Holds the value of the attributes captured, as they would be returned to a GET or Read .request
2151 to the individual attributes.

2152 compact array or array table_cell_entry

2153 table_cell_entry ::= CHOICE

2154

```
2155          -- simple data types
2156          null-data           [0],
2157          bit-string          [4],
2158          double-long         [5],
2159          double-long-unsigned [6],
2160          octet-string        [9],
2161          visible-string      [10],
2162          utf8-string         [12],
2163          bcd                 [13],
2164          integer              [15],
2165          long                [16],
2166          unsigned             [17],
2167          long-unsigned        [18],
2168          long64               [20],
2169          long64-unsigned      [21],
2170          float32              [23],
2171          float64              [24]
```

```
2172  
2173          -- complex data types  
2174          structure  
2175      }
```

If the captured attribute is attribute_0, redundant values may be replaced by “null-data” if their value can be unambiguously recovered (for example `scalar`, `unit`).

2178 4 3 8 2 3 table cell definition

Specifies the list of attributes captured in the register table

2180 structure

```
2181 { class id: long-unsigned,
```

```
2183         logical_name:          octet-string,  
2184         group_E_values:       array unsigned,  
2185         attribute_index:      integer  
2186     }
```

2187 where:

- 2188 – class_id defines the common class_id of the objects the attributes of which are captured;
2189 – logical_name contains the common logical name of the objects, with E = 255 (wildcard);
2190 – group_E_values contain the list of cell identifiers, of type *unsigned*, as defined in the
2191 respective table of IEC 62056-6-1:2021;
2192 – attribute_index is a pointer to the attribute within the object. attribute_index 0 refers to all
2193 public attributes.

2194 If the *logical_name* of the “Register table” object is in the format A.B.C.D.255.F and the defined
2195 attribute of all objects identified in the respective table in IEC 62056-6-1:2021 are captured,
2196 then attribute 3 may not be accessible. In this case:

- 2197 – the class_id shall be 1 “Data”, 3 “Register” or 4 “Extended register”;
2198 – the *logical_name* of the objects to be captured is defined by the *logical_name* of the
2199 “Register table” object and the respective table INIEC 62056-6-1:2021.

2200 the attribute index shall be 2 (value).

2201 4.3.8.2.4 scaler_unit

2202 See the description of IC “Register”, 4.3.2.2.3.

2203 In the case when “value” attributes of “Register” or “Extended register” objects are captured,
2204 the *scaler_unit* shall be common for all objects and this attribute shall hold a copy.

2205 If other attributes or ICs are captured, the *scalar_unit* attribute has no meaning and shall be
2206 inaccessible.

2207 4.3.8.3 Method description

2208 4.3.8.3.1 reset (data)

2209 Clears the *table_cell_values*. It has no effect on the attributes captured.

2210 data::= integer (0)

2211 4.3.8.3.2 capture (data)

2212 Copies the values of the attributes into the *table_cell_values*. If the attribute_index = 0, all
2213 attributes are captured.

4.3.8.4 Behaviour of the object after modification of the table_cell_definition attribute

2216 Any modification to this attribute will automatically call the *reset* (data) method and this will
2217 propagate to all profiles capturing this object.

2218 If writing to `table_cell_definition` is attempted with a value too large the buffer holding the
2219 `table_cell_values` attribute, it will be rejected.

2220 4.3.9 Status mapping (class_id = 63, version = 0)

2221 4.3.9.1 Overview

2222 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				

2223

2224 **4.3.9.2 Attribute description**2225 **4.3.9.2.1 logical_name**

2226 Identifies the “Status mapping” object instances. See 6.2.46, 6.2.49, 6.2.50, 6.2.55 and 6.3.7.

2227 **4.3.9.2.2 status_word**

2228 Contains the current value of the status word.

2229 CHOICE

```

2230     {
2231         bit-string          [4],
2232         double-long-unsigned [6],
2233         octet-string        [9],
2234         visible-string      [10],
2235         utf8-string         [12],
2236         unsigned            [17],
2237         long-unsigned        [18],
2238         long64-unsigned      [21]
2239     }

```

2240 The size of the *status_word* is n*8 bits, the maximum size is 65 536 bits.2241 Manufacturers may choose any of the types listed above. However, the status word is always
2242 interpreted as a bit-string.2243 **4.3.9.2.3 mapping_table**2244 Contains the mapping of the *status_word* to the positions in the reference table.

2245

```

2246     structure
2247     {
2248         ref_table_id:       unsigned,
2249         ref_table_mapping: CHOICE
2250         {
2251             long-unsigned      [18],
2252             array long-unsigned [1]
2253         }
2254     }

```

2255 Where:

- 2256 – *ref_table_id* identifies the reference status table;
- 2257 – if the “long-unsigned” choice is taken, the value points to an entry in the reference table.
2258 This entry is mapped to the leading bit of the status word. The next entry is mapped to the
2259 next bit and so on. The last entry that is mapped to the trailing bit is determined by the
2260 length of the status word;
- 2261 – if the “array” choice is taken, the elements of the array point to entries in the reference
2262 status table. The order of the elements in the array corresponds to the position in the status

2263 word. The first element in the array maps the table entry referenced to the leading bit and
 2264 the last element to the trailing bit

2265 **4.3.10 Compact data (class_id: 62, version: 1)**

2266 **4.3.10.1 Overview**

2267 NOTE This version 1 supports both relative and absolute selective access.

2268 Instances of the “Compact data” IC allow capturing the values of COSEM object attributes as
 2269 determined by the *capture_objects* attribute. Capturing can take place:

- 2270 • on an external trigger (explicit capturing); or
 - 2271 • upon reading the *compact_buffer* attribute (implicit capturing)
- 2272 as determined by the *capture_method* attribute.

2273 The values are stored in the *compact_buffer* attribute as an octet-string.

2274 The set of data types is identified by the *template_id* attribute. The data type of each attribute
 2275 captured is held by the *template_description* attribute.

2276 The client can reconstruct the data in the uncompacted form – i.e. including the COSEM
 2277 attribute descriptor, the data type and the data values – using the *capture_objects*, *template_id*
 2278 and *template_description* attributes.

2279

Compact data	0...n	class_id = 62, version = 1			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. compact_buffer (dyn.)	octet-string				x + 0x08
3. capture_objects (static)	array				x + 0x10
4. template_id (static)	unsigned				x + 0x18
5. template_description (dyn.)	octet-string				x + 0x20
6. capture_method (static)	enum				x + 0x28
Specific methods	m/o				
1. reset (data)	o				
2. capture (data)	o				

2280

2281 **4.3.10.2 Attribute description**

2282 **4.3.10.2.1 logical_name**

2283 Identifies the “Compact data” object instance. See 6.2.41.

2284 **4.3.10.2.2 compact_buffer**

2285 Contains the values of the attributes captured as an *octet-string*.

2286 When the data captured is of type *octet-string*, *bit-string*, *visible-string*, *utf8-string* or *array* the
 2287 length is also included here.

2288 **4.3.10.2.3 capture_objects**

2289 Specifies the list of COSEM object attributes that are assigned to the “Compact data” object
 2290 instance.

2291 When defining the *capture_object* attribute, circular references shall be avoided.

2292 The *template_id* attribute shall be the first element in the *capture_objects* array.

2293 Upon an explicit or implicit invocation of the *capture* (data) method the values of the selected
 2294 attributes are captured into the *compact_buffer*.

2295 Two, mutually exclusive selective access mechanisms are available:

- 2296 – relative selective access, i.e. entries defined relative to current date or entry are returned:
 2297 this mechanism is controlled by the *data_index* element; or
- 2298 – absolute selective access, i.e. entries in an explicitly defined date range or entry range are
 2299 returned: this mechanism is controlled by the *restriction* element.

```
2300     array capture_object_definition
2301         capture_object_definition ::= structure
2302             {
2303                 class_id:          long-unsigned,
2304                 logical_name:      octet-string,
2305                 attribute_index:   integer,
2306                 data_index:        long-unsigned,
2307                 restriction:      restriction_element
2308             }
```

2309 Where:

- 2310 – *attribute_index* is a pointer to the attribute within the object, identified by *class_id* and
 2311 *logical_name*: *attribute_index* 1 refers to the 1st attribute (i.e. the *logical_name*),
 2312 *attribute_index* 2 to the 2nd attribute etc.; *attribute_index* 0 refers to all public attributes;
- 2313 – *data_index* is a pointer selecting one or several specific elements of an attribute with a
 2314 complex data type (structure or array):
 - 2315 • if the data type of the attribute is simple, then *data_index* has no meaning;
 - 2316 • if the data type of the attribute is a structure or an array, then *data_index* points to one
 2317 or several specific elements in the structure or array;
 - 2318 • when the attribute is the *buffer* of a “Profile generic” object, the *data_index* carries
 2319 selective access parameters relative to current date or entry.

2320

<i>data_index</i> :	MS-Byte		LS-Byte
	Upper nibble	Lower nibble	

2321

- 2322 • 0x0000 = identifies the whole attribute;
- 2323 • 0x0001 to 0x0FFF = identifies one element in the complex attribute. The first element
 2324 in the complex attribute is identified by *data_index* 1;
- 2325 • 0x1000 to 0xFFFF = selective access to the array holding the *buffer* of a “Profile
 2326 generic” object. The data-index selects entries within a number of last (recent) time
 2327 periods, or a number of last (recent) entries, as well as the columns in the array.

2328 The encoding is specified in Table 21.

2329 When the attribute is the buffer of a “Profile generic” object, then *restriction_element* specifies
 2330 selective access parameters in an explicitly defined date range or entry range.

2331

```

2332             restriction_element ::= structure
2333             {
2334                 restriction_type: enum:
2335                     (0) none,
2336                     (1) restriction_by_date,
2337                     (2) restriction_by_entry
2338                 restriction_value: CHOICE
2339                 {
2340                     null-data,           // no restrictions apply
2341                     restriction_by_date,
2342                     restriction_by_entry
2343                 }
2344             }
2345
2346             restriction_by_date ::= structure
2347             {
2348                 from_date: octet-string,
2349                 to_date: octet-string
2350             }
2351
2352             restriction_by_entry ::= structure
2353             {
2354                 from_entry: double-long-unsigned,
2355                 to_entry: double-long-unsigned
2356             }

```

- 2357 – restriction_element defines absolute selective access to a “Profile generic” *buffer* by date range (from_date to to_date) or by entries (from_entry to to_entry). To use this absolute selective access mechanism, data_index shall be set to 0x0000;

2360 restriction_element is composed of restriction_type and restriction_value:

- 2361 • for restriction by date range the restriction_type element holds (1) restriction by date and the restriction_value element holds restriction_by_date structure;
- 2362 • for restriction by entries the restriction_type element holds (2) restriction by entry and the restriction_value element holds restriction_by_entry structure;

2365 otherwise, the restriction_type element holds (0) none and the restriction_value element holds null-data. This choice shall be taken also if relative selective access is to be used.

2367 4.3.10.2.4 template_id

2368 Contains the identifier of the template.

2369 It shall uniquely identify the instance of the “Compact data” IC and the *template_description*.

2370 4.3.10.2.5 template_description

2371 Provides the data type of each attribute captured. It is an *octet-string* generated automatically by the server upon the programming of the *capture_objects* and it has the following structure:

- 2373 – the first octet is 0x02 (the tag of a structure);
- 2374 – this is followed by the number of elements in the structure – the same as the number of elements in the *capture_objects* array – encoded as a variable length integer;
- 2375 – this is followed by the data type of each attribute, in the same order as in the *capture_object* array:
 - 2378 • in the case of attributes with simple data type, the data type is represented by a single octet, carrying the tag of the data type. In the case of *bit-string* [4], *octet-string* [9], *visible-string* [10], *utf8-string* [12] the length of the string is part of the data held in the *compact_buffer*;

- in the case of an *array* [1], the data type is represented by a single octet 0x01. This is followed by the type description of the elements in the array. The number of elements in the array is part of the data held in the *compact_buffer*;

2385 in the case of a *structure* [2], the data type is represented by a single octet 0x02, followed by
2386 the number of elements inside the structure followed by the tag of each element of the structure.

2387 4.3.10.2.6 capture_method

2388 Defines the way the *compact_buffer* is updated.

2389 enum

(0) Capture upon invoking the *capture* (data) method. This may occur remotely or locally (explicit capturing),

2392 (1) Capture upon reading the *compact_buffer* attribute (implicit capturing).

4.3.10.3 Method description

4.3.10.3.1 reset (data)

2395 Clears the *compact_buffer*. After invoking this method the buffer holds an octet-string of 0 length until
2396 a new capture takes place.

This call does not trigger any additional operations of the capture objects. Specifically, it does not reset any captured attributes.

2399 data::= integer(0)

2400 4.3.10.3.2 capture (data)

2401 Copies the values of the attributes into the *compact_buffer* by reading each capture object.

2402 This call does not trigger any additional operations within the capture objects such as capture () or
2403 reset ().

2404 data::= integer(0)

2405 4.3.10.4 Behaviour of the object after modification of certain attributes

2406 Any modification of the *capture_objects* shall reset the *compact_buffer* and automatically
2407 update the *template_description*.

2408 4.3.10.5 **Restrictions**

2409 When defining the *capture_object* attribute, circular references shall be avoided.

2410 4.3.10.6 Examples for using compact data

2411 4.3.10.6.1 Daily billing data

Table 9 shows the daily billing data that are captured – together with the mandatory *template_id* – to the *compact buffer* attribute of a “Compact data” object.

Table 9 – Daily billing data

Data	class_id	Logical name	attribute_id	data_index	Size (bytes)	Type	Value
1	2	3	4	5	6	7	8
Template Id	62	0-0:66.0.0.255	4	0	1	unsigned	0

Unix time	1	0-0:1.1.0.255	2	0	4	double-long-unsigned	1374573317
Operating status	1	0-0:96.5.0.255	2	0	1	unsigned	0x29
Error register	1	0-0:97.97.0.255	2	0	1	unsigned	0x18
Total index	3	7-0:13.83.1.255	2	0	4	double-long-unsigned	6422483
Index F1	3	7-0:13.83.1.255	2	0	4	double-long-unsigned	865234
Index F2	3	7-0:13.83.1.255	2	0	4	double-long-unsigned	1234567
Index F3	3	7-0:13.83.1.255	2	0	4	double-long-unsigned	2345678
Activity calendar name	20	0-0:13.0.0.255	2	0	6	octet-string	"ABCDEF"
Event counter	1	0-0:96.15.1.255	2	0	2	long-unsigned	7890

2415

2416 Table 10 shows the attributes of the “Compact data” object.

2417

Table 10 – Attributes of the “Compact data” object

capture_objects (array)	For the elements of the array, see columns 2, 3, 4 and 5 of Table 9.
template_id (unsigned)	0
template_description (octet-string)	-- For the data types, see column 7 of Table 9. 02 0A 11 06 11 11 06 06 06 06 09 12
compact_buffer (octet-string) 32 bytes	-- For the values see column 8 of Table 9. 00 51EE5305 29 18 0061FFD3 000D33D2 0012D687 0023CACE 06414243444546 1ED2

2418

2419 For comparison, the A-XDR encoding of the same data as if they were accessed using a GET-
2420 WITH-LIST service is shown in Table 11. Only the encoding of the result (SEQUENCE OF Get-
2421 Data-Result) is shown.

2422

Table 11 – A-XDR encoding of the data (SEQUENCE OF Get-Data-Result)

Encoding	Explanation	Length
09	SEQUENCE of 9 elements	1
00 06 51EE5305	double-long-unsigned	6
00 11 29	unsigned	3
00 11 18	unsigned	3
00 06 0061FFD3	double-long-unsigned	6
00 06 000D33D2	double-long-unsigned	6
00 06 0012D687	double-long-unsigned	6
00 06 0023CACE	double-long-unsigned	6
00 09 06 414243444546	octet-string of length 6	9
00 12 1ED2	long-unsigned	4
Total		50 bytes
NOTE The leading 00-s in each element are there to indicate the CHOICE “Data” in Get-Data-Result.		

2423

2424 **4.3.10.6.2 Diagnostic and Alarm data**

2425 Table 12 shows the diagnostic and alarm data that are captured – together with the mandatory
 2426 *template_id* – to the *compact_buffer* attribute of a “Compact data” object.

2427 **Table 12 – Diagnostic and Alarm data**

Data	class_id	Logical name	attribute_id	data_index	Size (bytes)	Type	Value
1	2	3	4	5	6	7	8
Template Id	62	0-0:66.0.0.255	4	0	1	unsigned	1
Current Diagnostic	3	7-0:96.5.1.255	2	0	2	long-unsigned	0x4200
Daily Diagnostic	3	7-1:96.5.1.255	2	0	2	long-unsigned	0x4108
Billing Period Diagnostic	1	7-2:96.5.1.255	2	0	2	long-unsigned	0x4308
Synchronization event counter	1	0-0:96.15.2.255	2	0	2	long-unsigned	763
Metrological firmware version	1	7-0:0.2.1.255	2	0	8	octet-string	“ABCDEFGH”
Metrological event counter	1	0-0:96.15.1.255	2	0	2	long-unsigned	1532
Non-metrological firmware version	1	7-1:0.2.1.255	2	0	8	octet-string	“DEFGHIJK”

2428

2429 Table 13 shows the attributes of the “Compact data” object.

2430 **Table 13 – Attributes of the “Compact data” object**

capture_objects (array)	For the elements of the array, see columns 2, 3, 4 and 5 of Table 12.
template_id (unsigned)	1
template_description (octet-string)	-- For the data types, see column 7 of Table 12. 02 08 11 12 12 12 12 09 12 09
compact_buffer (octet-string) 29 bytes	-- For the values, see column 8 of Table 12. 01 4200 4108 4308 02FB 084142434445464748 05FC 084445464748494A4B

2431

2432 For comparison, the A-XDR encoding of the data as if they were read from the buffer attribute
 2433 of a “Profile generic” object is shown in Table 14 (only the Data is shown).

2434 **Table 14 – Encoding the data read from the buffer attribute of a “Profile generic” object**

Encoding	Explanation	Length
01 01	array of one element	2
02 07	structure of 7 elements	2
12 4200	long-unsigned	3
12 4108	long-unsigned	3
12 4308	long-unsigned	3
12 02FB	long-unsigned	3
09 08 4142434445464748	octet-string of length 8	10
12 05FC	long-unsigned	3
09 08 4445464748494A4B	octet-string of length 8	10
	Total	39 bytes

2435

2436 **4.3.10.6.3 Logbook reading**2437 In this example, the data to be compacted is the *buffer* attribute of a Logbook held by a “Profile
2438 generic” object capturing 2 elements, as shown in Table 15.2439 **Table 15 – Logbook data**

Data	class_id	Logical name	Attribute_id	data_index	Size (bytes)	Type	Value	
1	2	3	4	5	6	7	8	
Unix time	1	0-0:1.1.0.255	2	0	4	double-long-unsigned	See Note	
Event code	1	0-0:96.11.2.255	2	0	1	unsigned		
NOTE For this example, the following values are assumed:								
<ul style="list-style-type: none"> – UNIX timestamp: 1374573317D, – status: 0x29, – for simplicity of the example, the values are the same for all entries, – there are 50 entries in the buffer. 								

2440

2441 Table 16 shows the data to be captured by the “Compact data” object.

2442 **Table 16 – Attributes of the “Compact data” object**

Data	class_id	Logical name	attribute_id	Data index	Size (bytes)	Type	Value
1	2	3	4	5	6	7	8
Template Id	62	0-0:66.0.0.255	4	0	1	unsigned	2
Logbook buffer ¹	7	0-0:99.98.0.255	2	0	dyn. ²	array of structure	2
¹ See Table 15.							
² The size is dynamic and depends on the number of entries captured.							

2443

2444 Table 17 shows the attributes of the “Compact data” object.

2445

Table 17 – Attributes of the “Compact data” object

capture_objects (array)	For the elements of the array, see columns 2, 3, 4 and 5 of Table 16. The capture object has only 2 elements: the Template_id and the buffer attribute of the Logbook.
template_id (unsigned)	2
template-description (octet-string)	<p>-- For the data types, see column 7 of Table 16.</p> <p>02 02 11 01 02 02 06 11</p> <p>Meaning:</p> <p>02 02 - a structure of 2 elements</p> <p>11 - first element is an unsigned</p> <p>01 02 02 - second element is an array of structure with two elements in the structure</p> <p>06 first one is a double-long-unsigned</p> <p>11 second one is an unsigned</p>
compact_buffer (octet-string)	<p>-- For the values, see column 8 of Table 16.</p> <p>02 -- value of the template-id</p> <p>32 -- number of the elements in the array</p> <p>and $50 \times 5 = 250$ bytes (for 50 elements in the log book)</p> <p>252 bytes in total</p>

2446

2447 For comparison, the A-XDR encoding of the same data when read from the *buffer* attribute of a
 2448 “Profile generic” object is shown in Table 18.

2449

Table 18 – A-XDR encoding of the data read from the *buffer* attribute

Encoding	Explanation	Length
01 32	array of 50 elements	2
02 02	structure of 2 elements	2
06 51EE5305	double-long-unsigned	5
11 29	unsigned	2
02 02	structure of 2 elements	2
06 51EE5305	double-long-unsigned	5
11 29	unsigned	2
02 02	structure of 2 elements	2
06 51EE5305	double-long-unsigned	5
11 29	unsigned	2
....
	Total	452 bytes

2450

2451

2452

2453 **4.4 Interface classes for access control and management**2454 **4.4.1 Overview**

2455 Interface classes in this category model the logical structure of the DLMS/COSEM server, allow
2456 configuring and managing access to its resources, updating the firmware and managing security:

- 2457 • the “Association SN” class – see 4.4.3 – and the “Association LN” class – see 4.4.4 –
2458 model AAs. Their instances, the Association objects provide the list of objects accessible
2459 in each AA, manage and control access rights to their attributes and methods. They also
2460 manage the authentication of the communicating partners;
- 2461 • the “SAP Assignment” class – see 4.4.5 – models the logical structure of the server;
- 2462 • the “Image transfer” class – see 4.4.6 – models the firmware update process;
- 2463 • the “Security setup” class – see 4.4.7 – models the elements of the security context.
2464 “Security setup” objects are referenced from the “Association” objects and allow
2465 configuring security suites and security policies and managing security material;
- 2466 • the “Push setup” class – see 4.4.8 – models the push operation of the server;
- 2467 • the “Data protection” class – see 4.4.9 – specifies the necessary elements to apply
2468 cryptographic protection to COSEM object attribute values as well as to method invocation
2469 and return parameters;
- 2470 • the “Function control” class – see 4.4.10 – allows enabling and disabling functions in the
2471 server;
- 2472 • the “Array manager” class – see 4.4.11 – allows managing attributes of type array of other
2473 interface objects.

2474 **4.4.2 Client user identification**

2475 This new feature enables the server to distinguish between different users from the client side
2476 and to log their activities accessing the meter.

2477 Each AA established between a client and a server can be used by several users on the client
2478 side. The properties of the AA are configured in the server, using the “Association” and the
2479 “Security setup” objects. All users of an AA on the client side use these same properties.

2480 The security keys are known by the client and the server but they need not be known by the
2481 users of the client.

2482 The list of users – identified by their *user_id* and *user_name* – is known both by the client and
2483 the server. In the server it is held by the *user_list* attribute of the “Association” objects.

2484 NOTE The way a client authenticates a user to log into a client system is outside the scope of this specification.

2485 During AA establishment, the *user_id* – belonging to the *user_name* – is carried by the calling-
2486 AE-invocation-id field of the AARQ APDU. If the *user_id* provided is on the *user_list*, the AA
2487 can be established – provided that all other conditions are met – and the *current_user* attribute
2488 is updated. The value of this attribute can be logged.

2489 If the server does not “know” the user, the AA shall not be established. The server may silently
2490 discard the request to establish the AA or it may send back an appropriate error message.

2491 The user identification process is optional: if the *user_list* is empty – i.e. it is an array of 0
2492 elements – the function is disabled.

2493 **4.4.3 Association SN (class_id = 12, version = 4)**2494 **4.4.3.1 Overview**

2495 COSEM logical devices able to establish AAs within a COSEM context using SN referencing,
 2496 model the AAs using instances of the “Association SN” IC. A COSEM logical device may have
 2497 one instance of this IC for each AA the device is able to support.

2498 The **short_name** of the “Association SN” object itself is fixed within the COSEM context.
 2499 See 4.1.3.

Association SN	0...n	class_id = 12, version = 4			
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

2500

2501 **4.4.3.2 Attribute description**2502 **4.4.3.2.1 logical_name**

2503 Identifies the “Association SN” object instance. See 6.2.33.

2504 **4.4.3.2.2 object_list**

2505 Contains the list of all objects with their base_name (short_name), class_id, version and
 2506 logical_name. The base_name is the DLMS objectName of the first attribute (*logical_name*).

```

 2507     objlist_type ::= array          objlist_element
 2508
 2509             objlist_element ::= structure
 2510             {
 2511                 base_name:    long,
 2512                 class_id:     long-unsigned,
 2513                 version:      unsigned,
 2514                 logical_name: octet-string
 2515             }
  
```

2516 selective access (see 4.1.4.16) to the attribute *object_list* may be available. The access selector
 2517 values and their parameters are as defined in Table 19.

2518 **4.4.3.2.3 access_rights_list**

2519 Contains the access rights to attributes and methods.

2520 The link between the *object_list* and the *access_rights_list* is the *base_name*, present in both
 2521 the *objlist_element* structure and the *access_right_element* structure. Therefore, the
 2522 *base_names* on the two lists shall be the same. The number – and preferably, the order – of
 2523 the elements in the array of *objlist_element* and the array of *access_right_element* shall also
 2524 be the same.

```

2525           access_rights_type ::= array      access_rights_element
2526
2527           access_rights_element ::= structure
2528           {
2529             base_name:          long,
2530             attribute_access:   attribute_access_descriptor,
2531             method_access:      method_access_descriptor
2532           }
2533
2534           attribute_access_descriptor ::= array      attribute_access_item
2535
2536           attribute_access_item ::= structure
2537           {
2538             attribute_id:       integer,
2539             access_mode:       enum,
```

2541 When the enum value is interpreted as an unsigned8, the meaning of each bit is as shown below:

```

2542           {
2543             Bit    attribute access_mode
2544
2545             (0)  read-access,
2546             (1)  write-access,
2547             (2)  authenticated request,
2548             (3)  encrypted request,
2549             (4)  digitally signed request,
2550             (5)  authenticated response,
2551             (6)  encrypted response,
2552             (7)  digitally signed response
2553           }
2554
2555           access_selectors: CHOICE
2556           {
2557             null-data      [0],
2558             array integer  [1]
2559           }
2560
2561
2562           method_access_descriptor ::= array      method_access_item
2563
2564           method_access_item ::= structure
2565           {
2566             method_id:       integer,
2567             access_mode:     enum
```

2568
 2569 When the enum value is interpreted as an unsigned8, the meaning of each bit is as shown
 2570 below:
 2571

```
2572     {
2573         Bit    method access_mode
2574
2575         (0)  access,
2576         (1)  not-used,
2577         (2)  authenticated request,
2578         (3)  encrypted request,
2579         (4)  digitally signed request,
2580         (5)  authenticated response,
2581         (6)  encrypted response,
2582         (7)  digitally signed response
2583     }
2584 }
2585
```

2586 selective access (see 4.1.4.16) to the attribute access_rights_list may be available (optional).
 2587 The access selector values and their parameters are as defined in Table 19.

2588 **Table 19 – Parameters for selective access to the object_list and access_rights_list
 2589 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 logical_name. 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

2590

2591 **4.4.3.2.4 security_setup_reference**

2592 References the “Security setup” object by its *logical_name*. The referenced object manages
 2593 security for a given “Association SN” object instance.

2594 **4.4.3.2.5 user_list**

2595 Contains the list of users allowed to use the AA managed by the given instance of the
 2596 “Association SN” IC.

```
2597             array      user_list_entry
2598
2599             user_list_entry::= structure
2600             {
2601                 user_id:      unsigned,
```

2602 user_name visible-string
 2603 }

2604 Where:

2605 – user_id is the identifier of the user (this value is carried by the calling-AE-invocation-id field
 2606 of the AARQ);

2607 – user_name is the name of the user.

2608 If the user_list attribute is empty – i.e. it is an array of 0 elements – any user can use the AA,
 2609 i.e. the calling-AE-invocation-id field of the AARQ is ignored.

2610 If the user_list attribute is not empty then only the users in the list can establish the AA, i.e. the
 2611 calling-AE-invocation-id field of the AARQ shall be present and its value shall match one of
 2612 user_ids in the user_list or else, the AA is not established.

2613 **4.4.3.2.6 current_user**

2614 Holds the identifier of the current user.

2615 current_user ::= user_list_entry (see 4.4.3.2.5)

2616 If the user_list is empty, then current_user shall be a structure {user_id: unsigned 0, user_name:
 2617 visible string of 0 elements}

2618 **4.4.3.3 Method description**

2619 **4.4.3.3.1 read_by_logicalname (data)**

2620 Reads attributes for selected objects. The objects are specified by their class_id and their
 2621 logical_name. With this method, the parameterized access feature can also be used.

```
2622                    data ::= array        attribute_identification  

  2623                    attribute_identification ::= structure  

  2624                    {  

  2625                    class_id:            long-unsigned,  

  2626                    logical_name:      octet-string,  

  2627                    attribute_index:    integer  

  2628                    }
```

2630 where attribute_index is a pointer (i.e. offset) to the attribute within the object.

2631 attribute_index 0 delivers all attributes; attribute_index 1 delivers the first attribute (i.e.
 2632 logical_name), etc.).

2633 For the response: data is according to the type of the attribute.

2634 NOTE 1 If at least one attribute has no read access right under the current association, then a read_by_logicalname
 2635 () to attribute index 0 reveals the error message “scope-of-access-violated”, see IEC 62056-5-3:2021, Clause 8.

2636 **4.4.3.3.2 change_secret (data)**

2637 Changes the LLS or HLS secret (for example password).

```
2638                    data ::= octet-string    new secret
```

2639 NOTE 2 The structure of the “new secret” depends on the security mechanism implemented. The “new secret” may
 2640 contain additional check bits and it may be encrypted.

2641 NOTE 3 In the case of HLS with GMAC, the (HLS_) secret is held by the “Security setup” object referenced in
 2642 attribute.

2643 **4.4.3.3.3 reply_to_HLS_authentication (data)**

2644 The remote invocation of this method delivers to the server the result of the secret processing
 2645 by the client of the server's challenge to the client, f(StoC), as the data service parameter of
 2646 the Read.request primitive invoked with parameterised access.

2647 data::= octet-string client's response to the challenge

2648 If the authentication is accepted, then the response (Read.confirm primitive) contains Result
 2649 == OK and the result of the secret processing by the server of the client's challenge to the
 2650 server, f(CtoS) in the data service parameter of the Read.response service.

2651 data::= octet-string server's response to the challenge

2652 If the authentication is not accepted, then the result parameter in the response shall contain a
 2653 non-OK value, and no data shall be sent back.

2654 **4.4.3.3.4 add_user (data)**

2655 Adds a user to the user_list.

2656 data::= user_list_entry (see 4.4.3.2.5)

2657 **4.4.3.3.5 remove_user (data)**

2658 Removes a user from the user_list.

2659 data::= user_list_entry (see 4.4.3.2.5)

2660

2661

2662 **4.4.4 Association LN (class_id = 15, version = 3)**2663 **4.4.4.1 Overview**

2664 COSEM logical devices able to establish AAs within a COSEM context using LN referencing,
 2665 model the AAs through instances of the "Association LN" IC. A COSEM logical device has one
 2666 instance of this IC for each AA the device is able to support.

Association LN	0...MaxNbofAss.	class_id = 15, version = 3			
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
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

2667

2668 **4.4.4.2 Attribute description**2669 **4.4.4.2.1 logical_name**

2670 Identifies the “Association LN” object instance. See 6.2.33.

2671 **4.4.4.2.2 object_list**2672 Contains the list of visible COSEM objects with their class_id, version, *logical_name* and the
2673 access rights to their attributes and methods within the given AA.

```

2674          object_list_type ::= array      object_list_element
2675          object_list_element ::= structure
2676
2677          {
2678          class_id:           long-unsigned,
2679          version:            unsigned,
2680          logical_name:       octet-string,
2681          access_rights:     access_right
2682          }
2683
2684          access_right ::= structure
2685          {
2686          attribute_access:   attribute_access_descriptor,
2687          method_access:      method_access_descriptor
2688          }
2689
2690          attribute_access_descriptor ::= array      attribute_access_item
2691
2692          attribute_access_item ::= structure
2693          {
2694          attribute_id:        integer,
2695          access_mode:         enum
2696

```

2697 When the enum value is interpreted as an unsigned8, the meaning of each bit is as shown
2698 below:

```

2699          {
2700          Bit    attribute_access_mode
2701
2702          (0)   read-access,
2703          (1)   write-access,
2704          (2)   authenticated request,
2705          (3)   encrypted request,
2706          (4)   digitally signed request,

```

```

2707          (5)  authenticated response,
2708          (6)  encrypted response,
2709          (7)  digitally signed response
2710      }
2711
2712      access_selectors: CHOICE
2713      {
2714          null-data      [0],
2715          array integer  [1]
2716      }
2717  }
2718
2719  method_access_descriptor ::= arraymethod_access_item
2720
2721  method_access_item ::= structure
2722  {
2723      method_id:    integer,
2724      access_mode:  enum
2725
2726      When the enum value is interpreted as an unsigned8, the meaning of each bit
2727      is as shown below:
2728  {
2729      Bit   method access_mode
2730
2731      (0)  access,
2732      (1)  not-used,
2733      (2)  authenticated request,
2734      (3)  encrypted request,
2735      (4)  digitally signed request,
2736      (5)  authenticated response,
2737      (6)  encrypted response,
2738      (7)  digitally signed response
2739  }
2740  }

```

2741 Where:

- 2742 – the attribute_access_descriptor and the method_access_descriptor elements always
 2743 contain all implemented attributes or methods;
 - 2744 – access_selectors contain a list of the supported selector values.
- 2745 *selective access* (see 4.1.4.16) to the attribute *object_list* may be available
 2746 (optional). The selective access parameters are as defined in 4.4.4.2.2.1

2748 4.4.4.2.2.1 Parameters for selective access to the *object_list* attribute

- 2749 • If no selective access is requested, (no Access_Selection_Parameters parameter is
 2750 present in the GET.request (.indication) service primitive for the object_list attribute) the
 2751 corresponding .response (.confirmation) service shall contain all object_list_elements of
 2752 the object_list attribute.
- 2753 • When selective access is requested to the object_list attribute (the
 2754 Access_Selection_Parameter parameter is present), the response shall contain a 'filtered'
 2755 list of object_list_elements, as shown in Table 20:

2756

2757

2758

Table 20 – Parameters for selective access to the object_list attribute

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.

2759

2760

2761 **4.4.4.2.3 associated_partners_id**

2762 Contains the identifiers of the COSEM client and server (logical device) APs within the physical
2763 devices hosting these APs, which belong to the AA modelled by the “Association LN” object.

2764 associated_partners_type::= structure
2765 {
2766 client_SAP: integer,
2767 server_SAP: long-unsigned
2768 }
2769

2770 The range for the client_SAP is 0...0x7F.

2771 The range for the server_SAP is 0x0000...0x3FFF.

2772 The SAPs shall be in the range allowed by the data type and the media.

2773 **4.4.4.2.4 application_context_name**

2774 In the COSEM environment, it is intended that an application context pre-exists and is
2775 referenced by its name during the establishment of an AA. This attribute contains the name of
2776 the application context for that AA.

2777 context_name_type::= CHOICE
2778 {
2779 context_name_structure [2],
2780 octet-string [9]
2781 }

2782 The application context name is specified as OBJECT IDENTIFIER in IEC 62056-5-3:**2021**,
2783 7.2.2.2.

2784 When the context_name_type is encoded as a structure, it includes the arc labels of the
2785 OBJECT IDENTIFIER.

2786 context_name_structure::= structure

```

2787     {
2788         joint_iso_ctt_element:           unsigned,
2789         country_element:               unsigned,
2790         country_name_element:         long-unsigned,
2791         identified_organization_element: unsigned,
2792         DLMS_UA_element:              unsigned,
2793         application_context_element:   unsigned,
2794         context_id_element:            unsigned
2795     }

```

2796 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
2797 01 (all values are hexadecimal).

2798 When the context_name_type is encoded as an octet-string, it holds the value of the OBJECT
2799 IDENTIFIER. See IEC 62056-5-3:**2021**, Clause D.4.

2800 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
2801 hexadecimal).

2802 4.4.4.2.5 xDLMS_context_info

2803 Contains all the necessary information on the xDLMS context for the given AA.

```

2804         xDLMS_context_type ::= structure
2805         {
2806             conformance:          bit-string,
2807             max_receive_pdu_size: long-unsigned,
2808             max_send_pdu_size:   long-unsigned,
2809             dlms_version_number: unsigned,
2810             quality_of_service:  integer,
2811             cyphering_info:       octet-string
2812         }
2813

```

2814 Where:

- 2815 – the conformance element contains the xDLMS conformance block supported by the server.
2816 The length of the bit-string is 24 bits;
- 2817 – the max_receive_pdu_size element contains the maximum length for an xDLMS APDU,
2818 expressed in bytes that the client may send negotiated during the application association
2819 process and limited by the server-max-receive-pdu-size parameter of the xDLMS
2820 initiateResponse APDU;
- 2821 – the max_send_pdu_size element, in an active AA contains the maximum length for an
2822 xDLMS APDU, expressed in bytes that the server may send. It is limited by the client-max-
2823 receive-pdu-size parameter of the xDLMS initiateRequest APDU;
- 2824 – the dlms_version_number element contains the DLMS version number supported by the
2825 server;
- 2826 – the quality_of_service element is not used;
- 2827 – the cyphering_info element – in an active AA – contains the dedicated key parameter of the
2828 xDLMS initiateRequest APDU. See IEC 62056-5-3:**2021**, Clause 8.

2829 4.4.4.2.6 authentication_mechanism_name

2830 Contains the name of the authentication mechanism for the AA.

```

2831         mechanism_name_type ::= CHOICE
2832         {
2833             mechanism_name_structure [2],
2834             octet-string [9]
2835         }

```

2836 The authentication mechanism name is specified as an OBJECT IDENTIFIER in IEC 62056-5-
 2837 3:2021, 7.2.2.3.

2838 When the mechanism_name_type is encoded as a structure, it includes the arc labels of the
 2839 OBJECT IDENTIFIER.

```
2840
2841     mechanism_name_structure ::= structure
2842     {
2843         joint_iso_ctt_element:           unsigned,
2844         country_element:              unsigned,
2845         country_name_element:        long-unsigned,
2846         identified_organization_element: unsigned,
2847         DLMS_UA_element:             unsigned,
2848         authentication_mechanism_name_element: unsigned,
2849         mechanism_id_element:        unsigned
2850     }
```

2852 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
 2853 11 01 (all values are hexadecimal):

2854 When the mechanism_name_type is encoded as an octet-string, it holds the value of the
 2855 OBJECT IDENTIFIER. See IEC 62056-5-3:2021, Clause D.4.

2856 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
 2857 hexadecimal).

2858 No mechanism_name is required when no authentication is used

2859 **4.4.4.2.7 secret**

2860 Contains the secret for the LLS or HLS authentication process.

2861 NOTE In the case of HLS with GMAC, the (HLS) secret is held by the “Security setup” object referenced in attribute
 2862 9, *security_setup_reference*, 4.4.4.2.9.

2863 **4.4.4.2.8 association_status**

2864 Indicates the current status of the association, which is modelled by the object.

```
2865     enum:      (0)    non-associated,association-pending,
2866                  (1)    associated
```

2868 **4.4.4.2.9 security_setup_reference**

2869 References a “Security setup” object by its logical name. The referenced object manages
 2870 security for a given “Association LN” object instance.

2871 **4.4.4.2.10 user_list**

2872 Contains the list of users allowed to use the AA managed by the given instance of the
 2873 “Association LN” IC.

```
2874     array      user_list_entry
2875
2876     user_list_entry ::= structure
2877     {
2878         user_id:          unsigned,
2879         user_name:        visible-string
2880     }
```

2881 Where:

2882 user_id is the identifier of the user (this value is carried by the calling-AE-invocation-id field of
 2883 the AARQ);

2884 user_name is the name of the user.

2885 If the *user_list* attribute is empty – i.e. it is an array of 0 elements – any user can use the AA,
 2886 i.e. the calling-AE-invocation-id field of the AARQ is ignored.

2887 If the *user_list* attribute is not empty then only the users in the list can establish the AA, i.e. the
 2888 calling-AE-invocation-id field of the AARQ shall be present and its value shall match one of
 2889 user_ids in the *user_list* or else the AA is not established.

2890 **4.4.4.2.11 current_user**

2891 Holds the identifier of the current user.

2892 current_user::= user_list_entry (see 4.4.4.2.10)

2893 If the *user_list* is empty, then current_user shall be a structure {user_id: unsigned 0, user_name:
 2894 visible string of 0 elements}

2895 **4.4.4.3 Method description**

2896 **4.4.4.3.1 reply_to_HLS_authentication (data)**

2897 The remote invocation of this method delivers to the server the result of the secret processing
 2898 by the client of the server's challenge to the client, f(StoC), as the *data* service parameter of
 2899 the ACTION.request primitive invoked.

2900 data::= octet-string client's response to the challenge

2901 If the authentication is accepted, then the response (ACTION.confirm primitive) contains Result
 2902 == OK and the result of the secret processing by the server of the client's challenge to the
 2903 server, f(CtoS) in the *data* service parameter of the response service.

2904 data::= octet-string server's response to the challenge

2905 If the authentication is not accepted, then the result parameter in the response shall contain a
 2906 non-OK value, and no data shall be sent back.

2907 **4.4.4.3.2 change_HLS_secret (data)**

2908 Changes the HLS secret (for example encryption key).

2909 data::= octet-string new HLS secret

2910 The structure of the “new secret” depends on the security mechanism implemented. The “new
 2911 secret” may contain additional check bits and it may be encrypted.

2912 **4.4.4.3.3 add_object (data)**

2913 Adds the referenced object to the *object_list*.

2914 data::= object_list_element (see 4.4.4.2.2)

2915 **4.4.4.3.4 remove_object (data)**

2916 Removes the referenced object from the *object_list*.

2917 data::= object_list_element (see 4.4.4.2.2)

2918 **4.4.4.3.5 add_user (data)**

2919 Adds a user to the *user_list*.

2920 data::= user_list_entry (see 4.4.4.2.10)

2921 **4.4.4.3.6 remove_user (data)**

2922 Removes a user from the *user_list*.

2923 data::= user_list_entry (see 4.4.4.2.10)

2924

2925

2926 **4.4.5 SAP assignment (class_id = 17, version = 0)**2927 **4.4.5.1 Overview**

2928 This IC allows modelling the logical structure of physical devices, by providing information on
2929 the assignment of the logical devices to their SAPs. See IEC 62056-5-3:2021, 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				

2930

2931 **4.4.5.2 Attribute description**2932 **4.4.5.2.1 logical_name**

2933 Identifies the “SAP assignment” objects instance. See 6.2.34.

2934 **4.4.5.2.2 SAP_assignment_list**

2935 Contains the list of all logical devices and their SAP addresses within the physical device.

```
2936         asslist_type ::= array      asslist_element
2937         asslist_element ::= structure
2938         {
2939             SAP:           long-unsigned,
2940             logical_device_name: CHOICE
2941             {
2942                 octet-string    [9],
2943                 visible-string [10],
2944                 utf8-string     [12]
2945             }
2946         }
```

2947 REMARK: The actual addressing is performed by the supporting communication layers

2948 **4.4.5.3 Method**

2949 **4.4.5.3.1 connect_logical_device (data)**

2950 Connects a logical device to a SAP. Connecting to SAP 0 will disconnect the device. More than
2951 one device cannot be connected to one SAP (exception SAP 0).

2952 data::= asslist_element

2953

2954 **4.4.6 Image transfer (class_id = 18, version = 0)**

2955 **4.4.6.1 General**

2956 Instances of the Image transfer IC model the process of transferring binary files, called Images
2957 to COSEM servers.

2958 NOTE This specification includes some improvements and precisions to the text. The main changes are:

- 2959 • The description of the Image transfer process is given as an example only. The text and the flow chart are
2960 updated;
- 2961 • Data exchange between the client and a conceptual Image server is out of Scope;
- 2962 • A clear distinction is made between the Image transferred and the Images to activate;
- 2963 • Steps 1 and 6 are optional now;
- 2964 • Size of the *image_transferred_blocks_status* bit-string may be dynamic;
- 2965 • It is specified now that setting the value of the *image_transfer_enabled* attribute to FALSE disables the image
2966 transfer process;
- 2967 • It is specified now that re-initiating the image transfer process resets the whole process;
- 2968 • Some precisions have been added to the effect of invoking the methods;

2969 **4.4.6.2 The steps of the image transfer process**

2970 The Image transfer usually takes place in several steps:

- 2971 • Step 1: (Optional): Get ImageBlockSize;
- 2972 • Step 2: Client initiates Image transfer;
- 2973 • Step 3: Client transfers ImageBlocks;
- 2974 • Step 4: Client checks completeness of the Image;
- 2975 • Step 5: Server verifies the Image (Initiated by the client or on its own);
- 2976 • Step 6 (Optional): Client checks the information on the images to activate;
- 2977 • Step 7: Server activates the Image(s) (Initiated by the client or on its own).

2978 For an example with more detailed explanations, see 4.4.6.5.

2979

2980

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 (data)	m				
2. image_block_transfer (data)	m				
3. image_verify (data)	m				
4. image_activate (data)	m				

2981

2982 **4.4.6.3 Attribute description**2983 **4.4.6.3.1 logical_name**

2984 Identifies the “Image transfer” object instance. See 6.2.37.

2985 **4.4.6.3.2 image_block_size**2986 Holds the ImageBlockSize, expressed in octets, which can be handled by the server.
2987 ImageBlockSize shall not exceed the ServerMaxReceivePduSize negotiated.2988 NOTE 1 *image_block_size* is a property of the server.2989 **4.4.6.3.3 image_transferred_blocks_status**2990 Provides information about the transfer status of each ImageBlock. Each bit in the bit-string
2991 provides information about one individual ImageBlock:

2992 0 = Not transferred,

2993 1 = Transferred

2994 NOTE 2 The size of the attribute may be dynamic, i.e. it may grow upon reception of new ImageBlocks.

2995 **4.4.6.3.4 image_first_not_transferred_block_number**2996 Provides the ImageBlockNumber of the first ImageBlock not transferred. Once the Image is
2997 complete, the value returned should be equal to or above the number of blocks calculated from
2998 the Image size and the ImageBlockSize.2999 **4.4.6.3.5 image_transfer_enabled**

3000 Controls the enabling of the Image transfer process.

3001 boolean: FALSE = Disabled,

3002 TRUE = Enabled

3003 The image transfer methods can be invoked successfully only if the value of this attribute is
3004 TRUE. Setting the value of this attribute to FALSE disables all methods (invoking them
3005 fails).

3006 4.4.6.3.6 image_transfer_status

3007 Holds the status of the Image transfer process.

3008	enum:	(0)	Image transfer not initiated,
3009		(1)	Image transfer initiated,
3010		(2)	Image verification initiated,
3011		(3)	Image verification successful,
3012		(4)	Image verification failed,
3013		(5)	Image activation initiated,
3014		(6)	Image activation successful,
3015		(7)	Image activation failed

3017 4.4.6.3.7 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.

```
3020     array      image_to_activate_info_element  
3021  
3022     image_to_activate_info_element::= structure  
3023     {  
3024         image_to_activate_size:           double-long-unsigned,  
3025         image_to_activate_identification: octet-string,  
3026         image_to_activate_signature:      octet-string  
3027     }
```

3028 Where:

3029 image to activate size is the size of the Image to be activated, expressed in octets;

3030 image_to_activate_identification is the identification of the Image to be activated, and may
3031 contain information like manufacturer, device type, version information, etc.;

3032 image_to_activate signature is the signature of the Image to be activated.

3033 NOTE 3 The algorithm to generate the signature is out of the Scope of this specification.

3035 4.4.6.4 Method description

4.4.6.4.1 image_transfer_initiate (data)

3037 Initializes the Image transfer process.

```
3038     data ::= structure  
3039     {  
3040         image_identifier: octet-string,  
3041         image_size: double-long-unsigned  
3042     }
```

3043 Where:

3044 image identifier identifies the Image to be transferred;

3045 image size holds the ImageSize, expressed in octets.

3046 NOTE 4 The *image_identifier* identifies the Image to be transferred (container) but it is not necessarily linked to its
3047 content, i.e. the Images which will be activated. That information can be retrieved from the *image_to_activate*
3048 attribute after verification of the Image transferred.

3049 After a successful invocation of the method the *image_transfer_status* attribute is set to (1)
3050 and the *image_first_not_transferred_block_number* is set to 0. Any subsequent invocation of
3051 the method resets the whole Image transfer process and all ImageBlocks need to be
3052 transferred again.

3053 **4.4.6.4.2 image_block_transfer (data)**

3054 Transfers one block of the Image to the server.

3055 data ::= structure
3056 {
3057 *image_block_number*: double-long-unsigned,
3058 *image_block_value*: octet-string
3059 }
3060 After a successful invocation of the method the corresponding bit in the
3061 *image_transferred_blocks_status* attribute is set to 1 and the
3062 *image_first_not_transferred_block_number* attribute is updated.

3063 **4.4.6.4.3 image_verify (data)**

3064 Verifies the integrity of the Image before activation.

3065 data ::= integer (0)

3066 The result of the invocation of this method may be success, temporary_failure or other_reason.
3067 If it is not success, then the result of the verification can be found by retrieving the value of the
3068 *image_transfer_status* attribute.

3069 In the case of success, the *image_to_activate_info* attribute holds the information about the
3070 images to activate.

3071 NOTE 5 xDLMS services Action-Result / Data-Access-Result codes are specified in IEC 62056-5-3:2021, Clause
3072 8.

3073 **4.4.6.4.4 image_activate (data)**

3074 Activates the Image.

3075 data ::= integer (0)

3076 If the Image transferred has not been verified before, then this is done as part of the Image
3077 activation.

3078 The result of the invocation of this method may be success, temporary-failure or other-reason.
3079 If it is not success, then the result of the activation can be learned by retrieving the value of the
3080 *image_transfer_status* attribute.

3081 NOTE 6 xDLMS services Action-Result / Data-Access-Result codes are specified in IEC 62056-5-3:2021, Clause
3082 8.

3083

3084

3085

3086 **4.4.6.5 Example for a standard image transfer process**

3087 In this subclause an example of a usual Image transfer process from a client prospective is
3088 given including a flow chart shown in Figure 11. Please note that it is not mandatory to follow
3089 the process; depending on the use case some steps may be omitted or others may be necessary.

3090 Precondition: The Image transfer has to be enabled: *image_transfer_enabled* = TRUE.

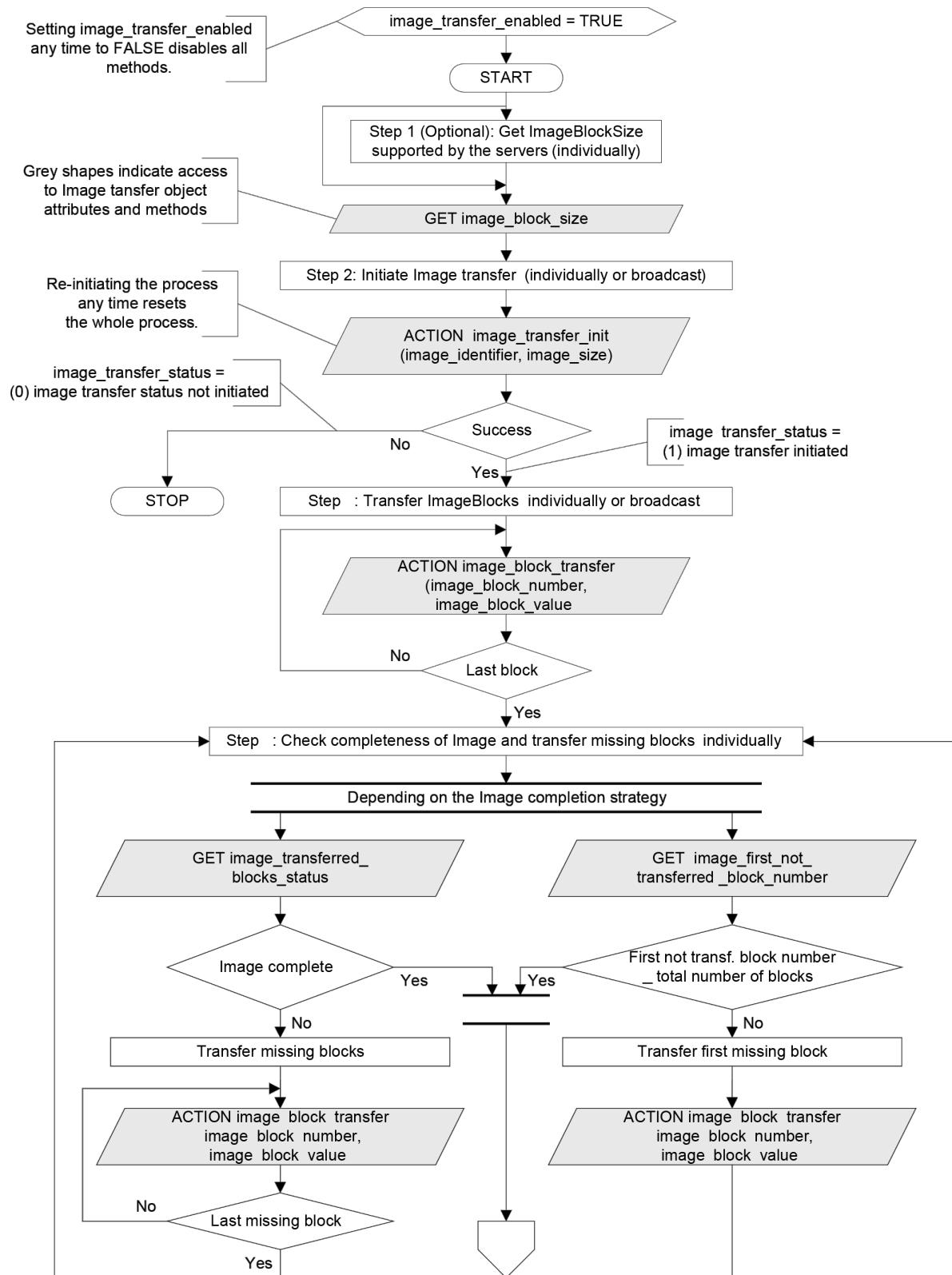
3091 Setting *image_transfer_enabled* any time to FALSE disables all methods (invoking those
3092 methods fails). The value of the status attributes is not defined.

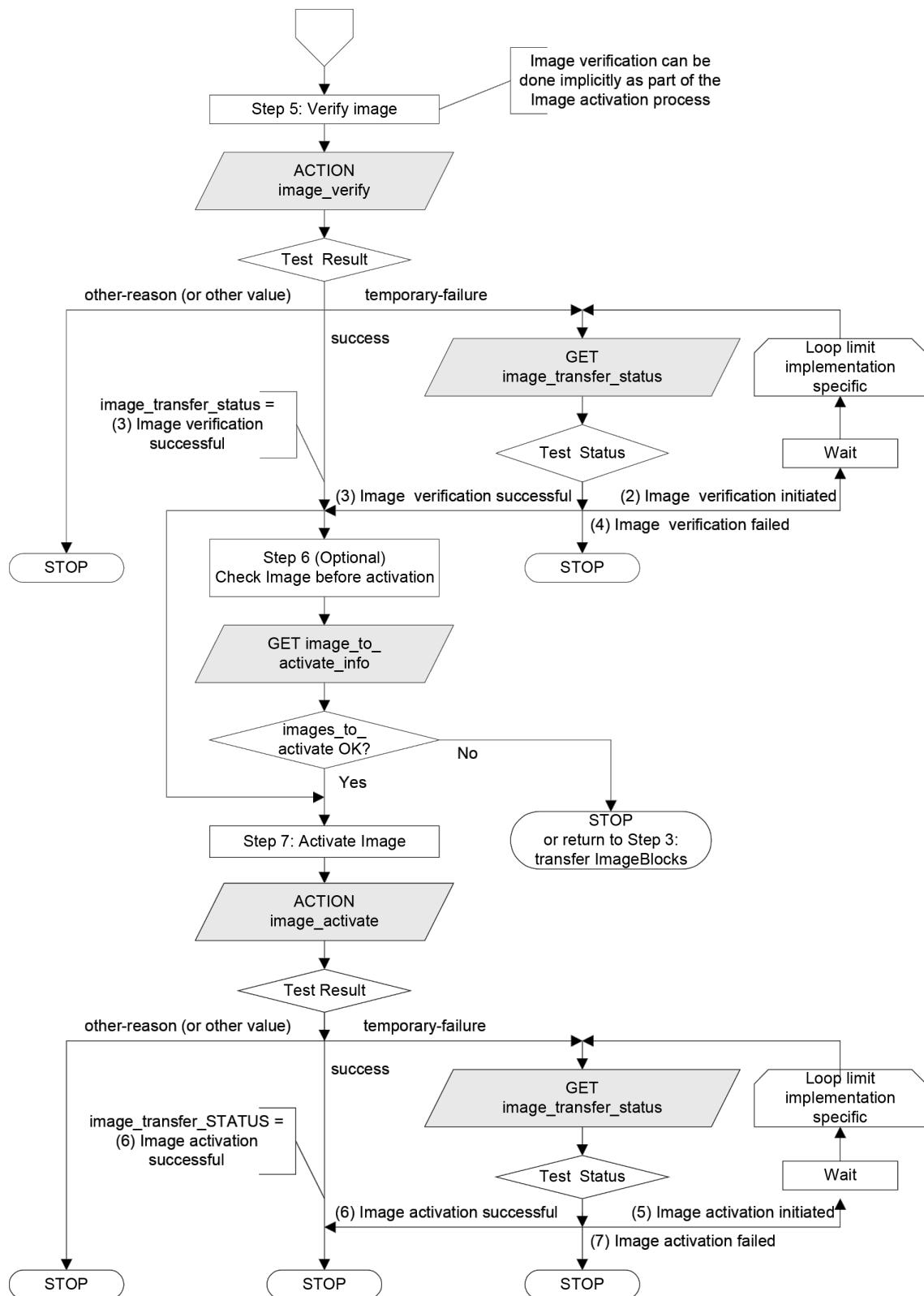
3093 Step 1 (Optional): Get ImageBlockSize

3094 If the client does not know the size of the image blocks the image transfer target server can
3095 handle, it shall read the *image_block_size* attribute of the relevant “Image transfer” object of
3096 each server the image has to be transferred to, before starting the process. The client can
3097 transfer then ImageBlocks of the right size.

3098 If ImageBlocks are sent using broadcast to a group of COSEM servers the ImageBlockSize
3099 shall be the same in each member of the group.

3100





3102

IFC

3103

Figure 11 – Image transfer process flow chart

3104 Step 2: Client initiates Image transfer

3105 The client initiates the Image transfer process individually or using broadcast in all servers by
3106 invoking the *image_transfer_initiate* method. The method invocation parameter holds the
3107 identifier and the size of the Image to be transferred. The server shall make the memory space
3108 – necessary to accommodate the Image – available.

3109 After a successful initiation, the value of the *image_transfer_status* attribute is (1). The
3110 *image_transferred_blocks_status* attribute shall be reset, the value of the
3111 *image_first_not_transferred_block_number* attribute shall be set to 0 and the value of the
3112 *image_to_activate_info* attribute should be reset. The image transfer process is initiated and
3113 the COSEM server is prepared to accept ImageBlocks.

3114 Step 3: Client transfers ImageBlocks

3115 The client transfers ImageBlocks to (a group of) server(s) by invoking the *image_block_transfer*
3116 method individually or using broadcast. The method invocation parameters include the
3117 ImageBlockNumber and one ImageBlock. ImageBlocks are accepted only by those COSEM
3118 servers, in which the Image transfer process has been successfully initiated. Other servers
3119 silently discard any ImageBlocks received.

3120 Step 4: Client checks completeness of the Image

3121 The client checks – with each server individually – the completeness of the Image transferred.
3122 If the Image is not complete, it transfers the ImageBlocks not (yet) transferred. This is an
3123 iterative process, continued until the whole Image is successfully transferred.

3124 To identify and transfer the ImageBlocks not transferred, two mechanisms are available.

- 3125 • the client may retrieve the status of each ImageBlock: either not transferred or transferred.
3126 This is performed by retrieving the value of the *image_transferred_blocks_status* attribute.
3127 The client transfers then the ImageBlocks not (yet) transferred;
- 3128 • alternatively, the client may retrieve the ImageBlockNumber of the first block not
3129 transferred. This is performed by retrieving the value of the
3130 *image_first_not_transferred_block_number* attribute. The client transfers then this
3131 ImageBlock not (yet) transferred;
- 3132 • after this, the client checks again the completeness of the Image.

3133 NOTE 1 The two mechanisms can be freely combined.

3134 Step 5: Server verifies the Image

3135 The Image is verified by the server. This can be initiated by invoking the *image_verify* method
3136 by the client or it may be initiated also by the server. The result can be:

- 3137 • success, if the verification could be completed;
- 3138 • temporary-failure, if the verification has not been completed;
- 3139 • other-reason, if the verification failed.

3140 NOTE 2 The conditions of verifying the Image are out of the scope of this document.

3141 Step 6 (Optional): Client checks the information on the images to activate

3142 The result of the Image verification may be checked by the client by retrieving the value of the
3143 *image_transfer_status* attribute. The value of this attribute is updated as a result of the Image
3144 verification.

3145 An Image transferred may hold one or more Images to be activated. For each Image to be
3146 activated, the `image_to_activate_info` attribute holds the parameters: {`image_to_activate_size`,
3147 `image_to_activate_identification`, `image_to_activate_signature`}.

3148 If this information is not what is expected, the client can restart transferring the image.

3149 Otherwise it goes to the next step, activation of the Image

3150 **Step 7: Server activates the Image(s)**

3151 The Image is activated by the server. This can be initiated by invoking the `image_activate`
3152 method by the client or it may be initiated also by the server. If the activation is done without a
3153 previous verification, then verification is done implicitly as part of the activation. The result of
3154 the invocation of the `Image_activate` method can be:

- 3155 • success, if the Image activation has been successfully started;
- 3156 • temporary-failure, if the verification/activation has not been completed;
- 3157 • other-reason, if the activation failed.

3158 In the case of success, the server performs the activation of the new Image(s). During this
3159 process, it is not accessible. After the Image(s) has (have) been activated, the result of may be
3160 checked by the client by retrieving the value of the `image_transfer_status` attribute or by reading
3161 the contents of the appropriate COSEM objects holding the identifier, version and digital
3162 signature of the active firmware.

3163 **4.4.7 Security setup (class_id = 64, version = 1)**

3164 **4.4.7.1 Overview**

3165 Instances of the “Security setup” IC contain the necessary information on the security suite in
3166 use and the security policy applicable between a client and a server identified by their respective
3167 system title. They also provide methods to increase the level of security and to manage
3168 symmetric keys, asymmetric key pairs and certificates.

Security setup	0...n	class_id = 64, version = 1			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. security_policy (static)	enum				x + 0x08
3. security_suite (static)	enum				x + 0x10
4. client_system_title (dyn.)	octet-string				x + 0x18
5. server_system_title (static)	octet-string				x + 0x20
6. certificates (dyn.)	array				x + 0x28
Specific methods	m/o				
1. security_activate (data)	o				x + 0x30
2. key_transfer (data)	o				x + 0x38
3. key_agreement (data)	o				x + 0x48
4. generate_key_pair (data)	o				x + 0x50
5. generate_certificate_request (data)	o				x + 0x58
6. import_certificate (data)	o				x + 0x60
7. export_certificate (data)	o				x + 0x68
8. remove_certificate (data)	o				x + 0x70

3169

3170 **4.4.7.2 Attribute description**3171 **4.4.7.2.1 logical_name**

3172 Identifies the “Security setup” object instance. See 6.2.36.

3173 **4.4.7.2.2 security_policy**3174 Enforces authentication and/or encryption and/or digital signature using the security algorithms
3175 available within the security suite.

3176 It applies independently for requests and responses.

3177 enum: When the enum value is interpreted as an unsigned, the meaning
3178 of each bit is as shown below:

3179	Bit	Security policy
3180	0	unused, shall be set to 0,
3181	1	unused, shall be set to 0,
3182	2	authenticated request,
3183	3	encrypted request,
3184	4	digitally signed request,
3185	5	authenticated response,
3186	6	encrypted response,
3187	7	digitally signed response

3190 NOTE 1 With this, the value (0) means that no cryptographic protection is required, as in version 0 of the “Security
3191 setup” IC.

3192 The access rights may require stronger protection than what is required by the security policy.

3193 **4.4.7.2.3 security_suite**

3194 Specifies the security algorithms available.

3195 enum:
 3196 (0) AES-GCM-128 authenticated encryption and AES-128 key wrap,
 3197 (1) AES-GCM-128 authenticated encryption, ECDSA P-256 digital signature, ECDH
 3198 P-256 key agreement, SHA-256 hash, V.44 compression and AES-128 key wrap,
 3199 (2) AES-GCM-256 authenticated encryption, ECDSA P-384 digital signature, ECDH P-
 3200 384 key agreement, SHA-384 hash, V.44 compression and AES-256 key wrap,
 3201 (3) ... (15) reserved
 3202

3203 **4.4.7.2.4 client_system_title**

3204 Carries the (current) client system title:

- 3205 – in the S-FSK PLC environment, the active initiator sends its system title using the CIASE
 3206 protocol;

3207 NOTE 2 It is also held by the active_initiator attribute of the S-FSK Active initiator object; see 4.10.4.

- 3208 – during confirmed or unconfirmed AA establishment, it is carried by the calling-AP-title field
 3209 of the AARQ APDU;

3210 If a client system title has already been sent during a registration process, like in the case
 3211 of the S-FSK PLC profile, the client system title carried the AARQ APDU should be the same.
 3212 Otherwise, the AA shall be rejected and appropriate diagnostic information should be sent.

- 3213 – in a pre-established AA, it can be written by the client using an unsecured service.

3214 **4.4.7.2.5 server_system_title**

3215 Carries the server system title.

- 3216 – in the S-FSK PLC environment, the server sends its system title during the discover process,
 3217 using the CIASE protocol;
- 3218 – during confirmed AA establishment, it is carried by the responding-AP-title field of the AARE
 3219 APDU.

3220 This attribute shall be read only.

3221 **4.4.7.2.6 certificates**

3222 Carries information on X.509 v3 certificates available and stored in the server.

```
3223              array             certificate_info
3224
3225              certificate_info::= structure
3226              {
3227                 certificate_entity: enum:
3228                                     (0) server,
3229                                     (1) client,
3230                                     (2) certification authority,
3231                                     (3) other
3232                 certificate_type: enum:
3233                                     (0) digital signature,
3234                                     (1) key agreement,
3235                                     (2) TLS,
3236                                     (3) other
3237                 serial_number: octet-string,
3238                 issuer:            octet-string,
3239                 subject:          octet-string,
```

```
3240                     subject_alt_name: octet-string  
3241                     }
```

3242 For the elements serial_number, issuer, subject, subject_alt_name see IEC 62056-5-3:2021,
3243 5.6.4.

3244 A server that uses public key cryptography stores the following public key certificates:

3245 For the server:

- 3246 – Digital Signature Certificate,
 - 3247 – Static Key Agreement Certificate,
 - 3248 – TLS Certificate.

3249 For each client and third party:

- 3250 – Digital Signature Certificate,
 - 3251 – Static Key Agreement Certificate,
 - 3252 – TLS Certificate.

3253 For Certification Authorities:

- 3254 – Certification Authority Certificate.

3255 4.4.7.3 Method description

4.4.7.3.1 security_activate (data)

3257 Activates and strengthens the security policy:

3258 **data::=** enum, as specified at the *security_policy* attribute.

3259 Strengthening the security policy means that another kind of protection is added. Once a kind
3260 of protection is added, it cannot be removed.

3261 If the method is invoked with a value indicating a security policy that is weaker than the value
3262 of the security_policy attribute, the method invocation fails.

3263 The new security policy applies as soon as the method invocation has been confirmed with
3264 success.

3265 4.4.7.3.2 key_transfer (data)

3266 Used to transfer one or more symmetric keys.

3267 The *data* parameter includes identification of the key(s) transported and the wrapped key(s)
3268 themselves.

3269 data::= array key_transfer_data

3271 key transfer data::= structure

3272

3273

3274 (0) global unicast encryption key,

3275 (1) global broadcast encryption key,
(2) the first secret

3276 (2) authentication key,
3277 (2) master key (KEK)

3277 (3) master
3278

3278

3281

3282 NOTE 3 It is possible to transfer multiple keys by invoking the method with an array of n *key_transfer_data*, or to
 3283 invoke the method n times with an array of a single *key_transfer_data*.

3284 The key wrap algorithm is as specified by the security suite. In suites 0, 1, and 2 the AES key
 3285 wrap algorithm is used.

3286 The KEK is the master key.

3287 If the unwrapping of the key(s) is successful, the result of the method invocation is success.

3288 If the unwrapping of the key(s) is not successful, the method invocation fails and an appropriate
 3289 reason for the failure shall be sent back. The keys are not changed.

3290 The new keys are activated immediately after result of the method invocation is sent back with
 3291 result = success. Notice that this rule equally applies to all keys, including the master key.

3292 NOTE 4 The APDUs carrying the method invocation service primitives are protected as stipulated by the
 3293 *security_policy* and the access rights to the methods. The method invocation parameters can be additionally
 3294 protected by invoking the method through a “Data protection” object. The required protection can be specified in
 3295 project specific companion specifications.

3296 This note equally applies to the *key_agreement (data)* method.

3297 NOTE 5 If using the new keys fails, the client may try to recover from this situation by reverting to using the previous
 3298 keys or repeating the key transfer process.

3299 4.4.7.3.3 **key_agreement (data)**

3300 Used to agree on one or more symmetric keys using the key agreement algorithm as specified
 3301 by the security suite. In the case of suites 1 and 2 the ECDH key agreement algorithm is used
 3302 with the Ephemeral Unified Model C(2e, 0s, ECC CDH) scheme.

3303 The *data* parameter includes the identification of the key(s) and data used in the key agreement
 3304 transaction.

```
3305           data ::= array key_agreement_data
3306
3307           key_agreement_data ::= structure
3308           {
3309               key_id: enum:
3310                   (0) global unicast encryption key,
3311                   (1) global broadcast encryption key,
3312                   (2) authentication key,
3313                   (3) master key (KEK)
3314               key_data: octet-string
3315           }
```

3317 NOTE 6 It is possible to agree on multiple keys by invoking the method with an array of n *key_agreement_data*, or
 3318 to invoke the method n times with an array of a single *key_agreement_data*.

3319 The element *key_id* identifies the key(s) to be agreed on.

3320 The element *key_data* is the public key of ephemeral key pair right concatenated with digital
 3321 signature, which is calculated over the *key_id* value and public key of ephemeral key pair value
 3322 using the digital signature private key: $Q_{e, U} \parallel \text{ECDSA}(\text{key_id} \parallel Q_{e, U})$.

3323 If the server could verify the digital signature of *key_data*, compute the shared secret and derive
3324 the key, then the method invocation is successful and the server sends its own *key_data* to the
3325 client. Otherwise, the method invocation fails and a reason for the failure is sent back.

3326 The new keys are activated immediately after result of the method invocation is sent back with
3327 result = success.

3328 NOTE 7 If using the new keys fails, the client may try to recover from this situation by reverting to using the previous
3329 keys or repeating the key transfer process.

3330 **4.4.7.3.4 generate_key_pair (data)**

3331 Generates an asymmetric key pair as required by the security suite. The data parameter
3332 identifies the usage of the key pair to be generated.

3333

3334 data ::= enum:

3335
3336 (0) digital signature key pair,
3337 (1) key agreement key pair,
3338 (2) TLS key pair

3339 NOTE 8 There is maximum one key pair for digital signature, one key pair for key agreement and one key pair for
3340 TLS.

3341 NOTE 9 Key agreement key pair is the static key used with the One-Pass Diffie-Hellman C(1e, 1s, ECC CDH)
3342 scheme when the server takes the role of Party V or with the Static Unified Model C(0e, 2s, ECC CDH) scheme.

3343 **4.4.7.3.5 generate_certificate_request (data)**

3344 When this method is invoked, the server sends the Certificate Signing Request (CSR) data that
3345 is necessary for a CA to generate a certificate for a server public key. The data parameter
3346 identifies the key pair for which the certificate will be requested.

3347

 data ::= enum:

3348 (0) digital signature key pair,
3349 (1) key agreement key pair,
3350 (2) TLS key pair

3351

3352 NOTE 10 There is maximum one key pair for digital signature, one key pair for key agreement and one key pair for
3353 TLS.

3354 The method invocation response parameters include the data necessary to generate the
3355 certificate.

3356 NOTE 11 It is the responsibility of the client to forward it to the Certification Authority.

3357

3358 response data::= octet-string,

3359 The octet-string contains the DER encoding of the CSR as specified in PKCS #10 (RFC 2986).

3360 The CSR shall be signed by the private key belonging to the public key in the request. This acts
3361 as the “proof of possession” of the private key.

3362 **4.4.7.3.6 import_certificate (data)**

3363 Imports an X.509 v3 certificate of a public key.

3364 data ::= octet-string

3365 The octet-string contains the DER encoding of the CSR as specified in PKCS #10 (RFC 2986).

3366 **4.4.7.3.7 export_certificate (data)**

3367 Exports an X.509 v3 certificate in the server.

3368 Certificate is identified with entity identification or the serial number of the certificate.

```

3369         data ::= certificate_identification
3370
3371         certificate_identification ::= structure
3372         {
3373             certificate_identification_type: enum:
3374                 (0) certificate_identification_entity,
3375                 (1) certificate_identification_serial
3376
3377             certification_identification_options: CHOICE
3378             {
3379                 certificate_identification_by_entity,
3380                 certificate_identification_by_serial
3381             }
3382         }
3383
3384         certificate_identification_by_entity ::= structure
3385         {
3386             certificate_entity: enum:
3387                 (0) server,
3388                 (1) client,
3389                 (2) certification authority,
3390                 (3) other
3391             certificate_type: enum:
3392                 (0) digital signature,
3393                 (1) key agreement,
3394                 (2) TL
3395                 (3) other
3396
3397             system_title: octet-string
3398         }
3399
3400         certificate_identification_by_serial ::= structure
3401         {
3402             serial_number: octet-string,
3403             issuer: octet-string
3404         }
3405
3406         response_data ::= octet-string
3407         response data octet-string is formatted as X.509 v3 DER format.
3408

```

3409 If no matching certificate is found, the response shall contain an appropriate error code.

3410 **4.4.7.3.8 remove_certificate (data)**

3411 Removes X.509 v3 certificate in the server.

3412 data ::= certificate_identification

3413 certificate_identification see 4.4.7.3.7.

3414 NOTE 12 The certificates of the server for digital signature, key agreement and TLS cannot be removed from the
 3415 server. When update of these certificates is needed the new key pair is generated, new certificate signing request is
 3416 generated and new certificate is imported

3417

3418

3419 **4.4.8 Push interface classes and objects**

3420 **4.4.8.1 Overview**

3421 There are several occasions on which DLMS messages can be ‘pushed’ to a destination without
 3422 being explicitly requested, e.g.:

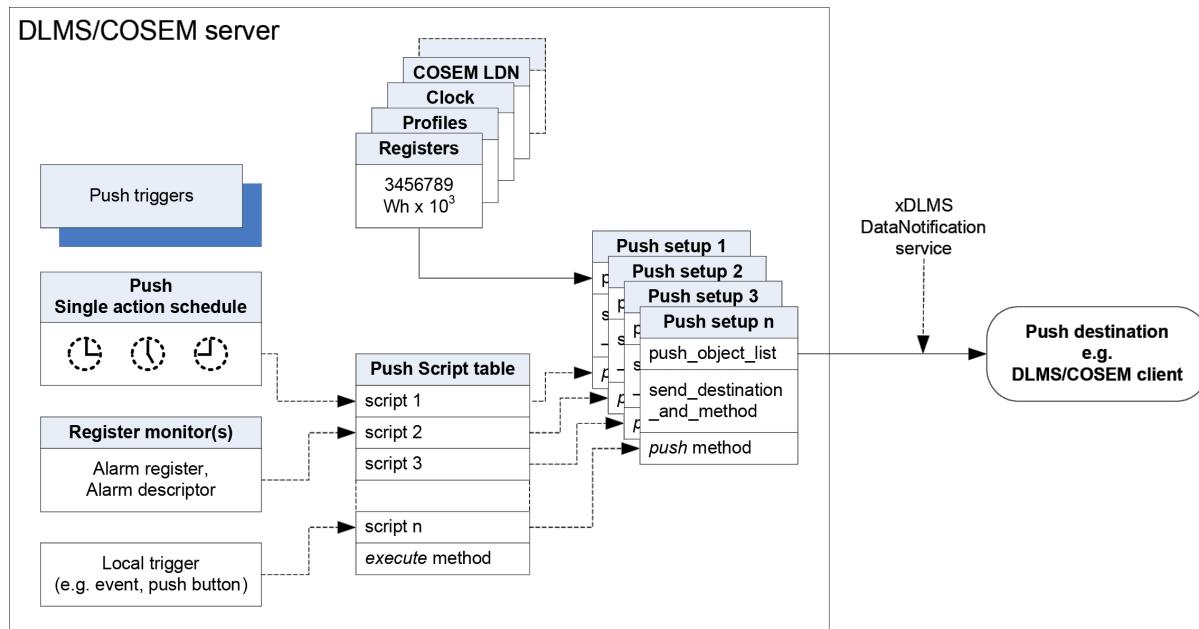
- 3423 • if a scheduled time is reached;
- 3424 • if a value being locally monitored exceeds a threshold;
- 3425 • on triggering by a local event (e.g. power-up/down, push button pressed, meter cover
 3426 opened).

3427 The DLMS/COSEM push mechanism follows the publish/subscribe pattern:

3428 *Publish/subscribe is a messaging pattern where senders (publishers) of messages do not
 3429 program the messages to be sent directly to specific receivers (subscribers). Rather,
 3430 published messages are characterized into classes, without knowledge of what, if any,
 3431 subscribers there may be. Subscribers express interest in one or more classes, and only
 3432 receive messages that are of interest, without knowledge of what, if any, publishers there
 3433 are. [Wikipedia]*

3434 In DLMS/COSEM, publishing is modelled by the *object_list* attribute of “Association” objects
 3435 providing the list of COSEM objects and their attributes accessible in a given AA. Subscription
 3436 is modelled by writing the appropriate attributes of “Push setup” objects. The required data are
 3437 sent – upon the trigger specified – using the xDLMS DataNotification service.

3438 The COSEM model of the Push operation is shown in Figure 12.



3439

IEC

3440

Figure 12 – COSEM model of push operation

3441 The core element of modelling the push operation is the “Push setup” IC. The *push_object_list*
 3442 attribute contains a list of references to COSEM object attributes to be pushed.

3443 When push uses a gateway, then the version of the gateway protocol for end devices without
 3444 WAN/NN knowledge is to be applied. This is described in the IEC 62056-5-3:**2021**  10.9.4.3.

3445 The various triggers (e.g. schedulers, monitors, local triggers etc.) call a script entry in a Push
 3446 “Script table” object (new instances of the “Script table” IC) which invokes then the *push* method
 3447 of the related “Push setup” object. The destination, the communication media, the protocol, the
 3448 encoding, the timing as well as any retries of the push operation are determined by the other
 3449 attributes of the “Push setup” object.

3450 Each trigger can cause the data to be sent to a dedicated destination. Therefore, for every
 3451 trigger or a group of triggers an individual “Push setup” object is available defining the content
 3452 and the destination of the push message as well as the communication medium used.

3453 For the purposes of pushing data when an alarm occurs, Alarm monitor objects (new instances
 3454 of the “Register monitor” IC) are available.

3455 The alarms are held by *Alarm register* or *Alarm descriptor* objects, see 6.2.65.

3456 The structure of the *Alarm descriptor* as well as the alarm conditions needs to be defined in a
 3457 project specific companion specification.

3458 *Alarm descriptor objects* are monitored by Alarm “Register monitor” objects, see 6.2.13. The
 3459 *actions* attribute provides the link to the *action_up* and maybe the *action_down* scripts in the
 3460 Push “Script table” object – see 6.2.7 – which invoke then the *push* method of the desired “Push
 3461 setup” object. When an alarm occurs, the predefined set of data – that may include among other
 3462 data the *Alarm register* and *Alarm descriptor* objects – are pushed.

3463 The push data is sent – when the conditions for pushing are met – using the unsolicited, non-
 3464 client/server type xDLMS service, the DataNotification service. When the data pushed is long,
 3465 it can be sent in blocks.

3466 The push process takes place within the application context of the AA in which the “Push setup”
 3467 object is visible. The security context is determined by the “Security setup” object referenced
 3468 from the “Association” object.

3469 NOTE Details are subject to project specific companion specifications.

3470 All information necessary to process the data received by the client shall be either:

- 3471 • retrieved by the client e.g. by reading the *push_object_list attribute*; or
- 3472 • shall be part of the data pushed; or
- 3473 • shall be predefined in the client application.

3474

3475 **4.4.8.2 Push setup (class_id = 40, version = 2)**

3476 **4.4.8.2.1 Overview**

3477 The Push setup IC contains a list of references to COSEM object attributes to be pushed. It
 3478 also contains the push destination and method as well as the communication time windows and
 3479 the handling of retries.

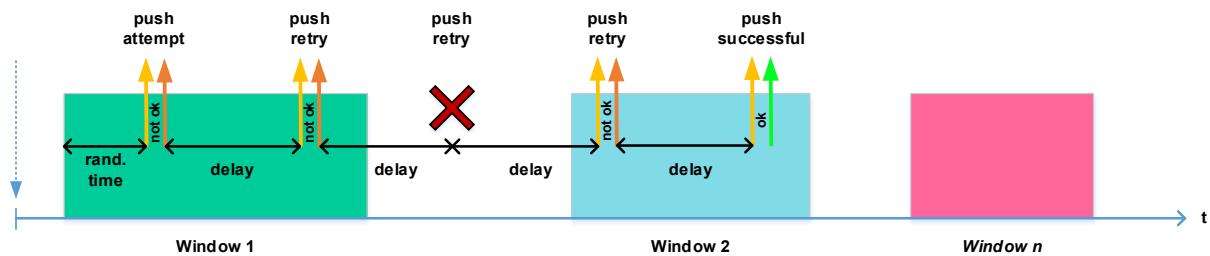
3480 In version 1 the possibility of data protection was added offering the same options as defined
 3481 in the Data protection IC.

3482 In version 2 six new possibilities are specified:

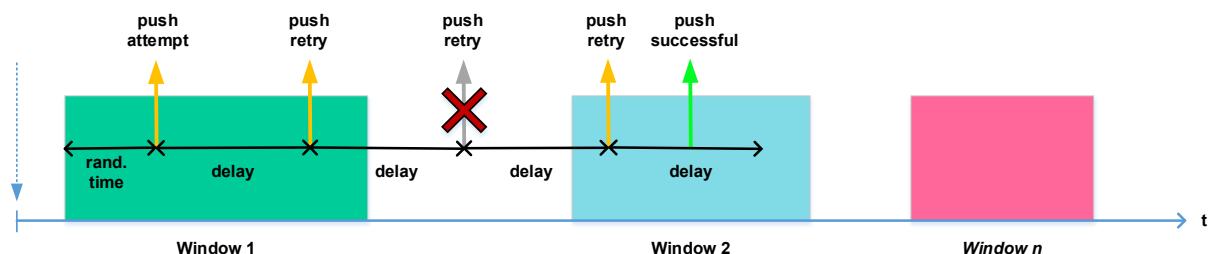
- 3483 • a new entry selection mechanism allows selecting entries related to the last confirmed entry;
- 3485 • a new column selection mechanism allows explicitly selecting columns of Profile generic object buffer attributes;
- 3487 • the **repetition_delay** attribute now provides an exponential formula that allows increasing the repetition delay with each retry attempt;
- 3489 • a new **push_operation_method** attribute allows specifying if the DataNotification.request service primitive is invoked with Service_Class == Unconfirmed or confirmed and how the push retry operation works;
- 3492 • a new **confirmation_parameters** attribute allows limiting the time interval in which entries related to last confirmed entry can be selected;
- 3494 • a new **last_confirmation_date_time** attribute holds the date_time when the DataNotification service was last confirmed.

3496 This version of the interface class is intended to facilitate the use of relative and absolute data selection. It would be common practice to use an instance of the Push setup IC with relative data selection for routine data push, and an instance of the Push setup IC with absolute data for special cases.

3500 The push is started when the *push* method is invoked, 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 retries may be made when supporting protocol layer failure is indicated or when confirmation is missing. Push operation with windows, delays and retries is shown in Figure 13.



3509 Case a) Retry on supporting layer failure



3511 Case b) Retry on missing confirmation

3512 Figure 13 – Push windows and delays

Push setup	0...n	class_id = 40, version = 2			
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)	structure				x + 0x30
8. port_reference (static)	octet-string				x + 0x38
9. push_client_SAP (static)	integer				x + 0x40
10. push_protection_parameters (static)	array				x + 0x48
11. push_operation_method (static)	enum				x + 0x50
12. confirmation_parameters (static)	structure				x + 0x58
13. last_confirmation_date_time (dyn.)	date-time				x + 0x60
Specific methods	m/o				
1. push (data)	m				x + 0x38
2. reset (data)	o				x + 0x70

3513

3514 **4.4.8.2.2 Attribute description**3515 **4.4.8.2.2.1 logical_name**

3516 Identifies the “Push setup” object instance. See 6.2.24.

3517 **4.4.8.2.2.2 push_object_list**

3518 Defines the list of attributes to be pushed.

3519 Upon invocation of the push (data) method the items are sent to the destination defined in the
3520 send_destination_and_method attribute.

```

3521           array      push_object_definition
3522           push_object_definition ::=   structure
3523           {
3524               class_id:      long-unsigned,
3525               logical_name: octet-string,
3526               attribute_index: integer,
3527               data_index:      long-unsigned,
3528               restriction:    restriction_element,
3529               columns:        column_element
3530           }

```

```

3531
3532     restriction_element ::= structure
3533     {
3534         restriction_type: enum:
3535             (0) none,
3536             (1) restriction_by_date,
3537             (2) restriction_by_entry
3538
3539         restriction_value: CHOICE
3540         {
3541             null-data, // no restrictions apply
3542             restriction_by_date,
3543             restriction_by_entry
3544         }
3545     }
3546     restriction_by_date ::= structure
3547     {
3548         from_date: octet-string,
3549         to_date: octet-string
3550     }
3551     restriction_by_entry ::= structure
3552     {
3553         from_entry: double-long-unsigned,
3554         to_entry: double-long-unsigned
3555     }
3556     column_element ::= array capture_object_definition
3557

```

3558 Where:

3559 – class_id, logical_name and attribute_index reference a COSEM object attribute. For
 3560 referencing all attributes of an object, see IEC 62056-5-3:2021, 9.1.4.3.7, attribute_0
 3561 referencing.

3562 If the attribute is simple or if attribute_0 is referenced, then:

- 3563 – data_index has no meaning and shall be set to 0x0000;
- 3564 – restriction_element restriction_type shall be 0 (none) and restriction_element
 3565 restriction_value shall be null-data;
- 3566 – column_element shall be an array of 0 elements.

3567 If the attribute is structure or an array – other than Profile generic object buffer then:

- 3568 – data_index points to one element in the structure or array and can take any value between
 3569 0x0001 and 0xFFFF;
- 3570 – restriction_element restriction_type shall be 0 (none) and restriction_element
 3571 restriction_value shall be null-data;
- 3572 – column_element shall be an array of 0 elements.

3573 If the attribute is a Profile generic object buffer, then data_index, restriction and column define
 3574 selective access parameters to select entries and columns.

3575 There are three, mutually exclusive entry selection mechanisms available:

- 3576 1) Relative selective access related to current date and time. Entries in last time intervals
 3577 relative to current date and time or last entries as identified by data_index are selected;
- 3578 2) Relative selective access related to last confirmed entry. Entries in next time intervals
 3579 relative to last confirmed entry or next entries as identified by data_index are selected;

3580 3) Absolute selective access. Entries in an explicitly defined date range or entry range as
 3581 identified by restriction are selected.

3582 There are two, mutually exclusive column selection mechanisms available:

3583 a) a defined number of columns starting from column 1 are selected;

3584 b) an explicitly defined list of columns are selected.

3585 In the case of entry selection mechanism (1) and (2) the selective access parameters are
 3586 defined by `data_index` interpreted as three different fields as shown below. The encoding is
 3587 shown in Table 21.

<code>data_index</code>	MS-Byte		LS-Byte
	Upper nibble	Lower nibble	

3588 Where:

3589 – `data_index` MS-Byte Upper nibble defines how the entries are selected;

3590 – `data_index` MS-Byte Lower nibble is used to specify the number of columns when applicable;

3591 – `data_index` LS-Byte defines the number of entries or time intervals.

3592 In the case of entry selection mechanism (1) and (2), `restriction_element restriction_type` shall
 3593 be 0 and `restriction_element restriction_value` shall be `null_data`.

3594 In the case of entry selection mechanism (3) the selective access parameters are defined by
 3595 `restriction_element`. In this case `Data_index MS_byte Upper_nibble` shall be set to 0x0, and
 3596 `Data_index LS_byte` shall be set to 0x00.

3597 – for restriction by date range `restriction_type` holds (1) range by date and `restriction_value`
 3598 holds `restriction_by_date` structure;

3599 – for restriction by entries `restriction_type` holds (2) range by entry and `restriction_value` holds
 3600 `restriction_by_entry` structure;

3601 if selective access is not required, `restriction_type` holds (0) none and `restriction_value`
 3602 holds null-data. This choice shall be taken also if relative selective access is used.

3603 In the case of column selection mechanism (a), `Data_index MS_byte Lower_nibble` defines the
 3604 number of columns to be selected, starting from column 1. In this case, `column_element` shall
 3605 be an empty array.

3606 In the case of column selection mechanism (b), the columns to be selected are identified by
 3607 `column_element`. In this case, `Data_index MS_byte Lower_nibble` shall be set to 0.

3608 If relative selective access related to last confirmed entry is used, the last confirmed entry is
 3609 updated when the `DataNotification.confirm` service primitive is invoked with `Result ==`
 3610 `CONFIRMED`.

3611 The `DataNotification` APDU carrying the push data is protected as stipulated by the security
 3612 suite, security policy and the security material held by the “Security setup” object referenced
 3613 from the Association SN / Association LN object within the application context of the AA which
 3614 is linked to the `push_client_SAP` attribute; as well as by the `push_potection_parameters`
 3615 attribute. For attributes to which no access right is granted within this AA, or which cannot be
 3616 accessed for any other reason, null-data should be returned.

3617 NOTE 1 If the `push_object_list` array is empty, the push operation is disabled.

3618 NOTE 2 The `push_object_list` attribute itself can be also pushed to identify the data pushed.

3619 NOTE 3 Last confirmed entry is an internal variable maintained by the server AP

3620

3621 **4.4.8.2.2.3 send_destination_and_method**

3622 Contains the destination address (e.g. phone number, email address, IP address) where the
 3623 data specified by the *push_object_list* has to be sent, as well as the sending method.

3624 `send_destination_and_method ::= structure`

```
3625     {
3626         transport_service:    transport_service_type,
3627         destination:        octet-string,
3628         message:           message_type
3629     }
```

3630 Where:

3631 – the `transport_service` element defines the type of service used to push the data:

3632 `transport_service_type ::= enum:`

```
3633
3634     (0)      TCP,
3635     (1)      UDP,
3636     (2)      reserved for FTP,
3637     (3)      reserved for SMTP,
3638     (4)      SMS,
3639     (5)      HDLC,
3640     (6)      reserved for M-Bus,
3641     (7)      reserved for ZigBee®
3642     (200...255) manufacturer specific
3643
```

3644 – the `destination` element contains the target address where the data has to be sent. The
 3645 elements of the target address depend on the transport service used.

3646 Each “Push setup” object instance specifies a single destination. If it is required to push
 3647 data to several destinations, several “Push setup” objects have to be instantiated,

3648 – the `message_type` element identifies the encoding of the xDLMS APDU used.

```
3649     message_type ::= enum:
3650         (0)      A-XDR encoded xDLMS APDU,
3651         (1)      XML encoded xDLMS APDU,
3652         (128...255)   manufacturer specific
```

3653 – All other `transport_service_type` and `message_type` values are reserved for future use.

3654 **4.4.8.2.2.4 communication_window**

3655 Defines the time points when the communication window(s) for the push become(s) active
 3656 (`start_time`) and inactive (`end_time`). See Figure 13.

```
3657     array      window_element
3658         window_element ::= structure
3659         {
3660             start_time: octet-string,
3661             end_time:  octet-string
3662         }
3663
```

3664 start_time and end_time are formatted as specified in 4.1.6.1 for *date-time* including wildcards.

3665 If the end of a communication window is reached when a push has already started the operation
 3666 will be completed.

3667 If no communication windows are defined (array [0]) the push operation is always possible.

3668 4.4.8.2.2.5 randomisation_start_interval

3669 Defines a maximum delay, in seconds, of sending the first DataNotification APDU after invoking
 3670 the push method. This is to avoid a situation where many servers may push simultaneously. It
 3671 is used as follows:

- 3672 • the delay is random from 0 up to the maximum value. A value of 0 deactivates the
 3673 mechanism;
- 3674 • if the push is invoked within an active communication window, the random delay is applied.
 3675 It is not applied for retries;
- 3676 • if the push is invoked outside a communication window, then the random delay is applied
 3677 when the communication windows opens;
- 3678 • if the random delay extends beyond the current communication window, then the
 3679 DataNotification APDU will be sent in the next window if available;
- 3680 • if no communication windows are defined, then the random delay is applied whenever the
 3681 push method is invoked including all retries.

3682 4.4.8.2.2.6 number_of_retries

3683 Defines the maximum number of retries in the case of unsuccessful or skipped push attempts.
 3684 After a successful push operation no further push attempts are made until the push operation
 3685 is triggered again. For further details on the retransmission operation see the
 3686 *push_operation_method* attribute (4.4.8.2.2.11).

3687 4.4.8.2.2.7 repetition_delay

3688 Defines the elements for calculating the time delay until the next attempt after an unsuccessful
 3689 push. See Figure 13.

```
3690
3691     repetition_delay ::= structure
3692     {
3693         repetition_delay_min: long-unsigned,
3694         repetition_delay_exponent: long-unsigned,
3695         repetition_delay_max: long-unsigned
3696     }
```

3697 Where:

- 3698 • repetition_delay_min is the minimum delay, in seconds, until a next push attempt is started;
- 3699 • repetition_delay_exponent is used for calculating the next delay;
- 3700 • repetition_delay_max is the maximum delay, in seconds. Any calculated time delay will be capped
 3701 by this value.

3703 The repetition delay is calculated using the following formula:

$$3704 \quad \text{repetition_delay} = \text{repetition_delay_min} \times (\text{repetition_delay_exponent} \times 0.01)^{n-1}$$

3705 where n refers to the ordinal number of the push retries with n = 1 for the first retry etc.

3706 NOTE 4 The repetition delay itself is not influenced by the communication window. But a push retry only can
 3707 be made if a communication window is active at that time. Otherwise it is handled like an unsuccessful push attempt.

3708 NOTE 5 The push data is not stored in an intermediate buffer. In the case of push retries, the current values
 3709 of the attributes may change with every push retry attempt.

3710 **4.4.8.2.2.8 port_reference**

3711 Contains the logical name of a communication port setup object allowing the selection of a
 3712 specific communication channel for the push based on the transport_service_type. This mainly
 3713 applies in cases where several channels of the same type are supported.

3714 If this information is not available or not needed, the attribute may be left empty (octet-string
 3715 [0]).

3716 **4.4.8.2.2.9 push_client_SAP**

3717 Defines the client SAP where the push is directed to in the supporting layer of the
 3718 DataNotification service. The push process takes place within the application context of the AA
 3719 which is linked to the push_client_SAP attribute. The security context is determined by the
 3720 Security setup object referenced in the related Association SN /LN object.

3721 **4.4.8.2.2.10 push_protection_parameters**

3722 Specifies all protection parameters to be applied to the DataNotification APDU when the push
 3723 data is sent. It offers the same options as the Data Protection IC but it is bound to the sending
 3724 of the data defined in the push_object_list attribute.

3725 array protection_parameters_element

```

3726           protection_parameters_element ::= structure
3727           {
3728             protection_type: enum:
3729               (0) authentication,
3730               (1) encryption,
3731               (2) authentication and encryption,
3732               (3) digital signature
3733
3734             protection_options: structure
3735             {
3736               transaction_id:          octet-string,
3737               originator_system_title: octet-string,
3738               recipient_system_title:  octet-string,
3739               other_information:       octet-string,
3740
3741               key_info:   key_info_element
3742             }
3743           }
```

3744 Where:

- 3745 – – transaction_id holds the identifier of the transaction;
- 3746 – – originator_system_title holds the system title of the originator that applies the protection;
- 3747 – – recipient_system_title holds the system title of the recipient which will check and remove
 3748 the given protection;
- 3749 – – other_information carries other information. Its contents may be specified in project
 3750 specific companion specifications. An octet-string of length 0 indicates that this field is not
 3751 used;

3752 – – key_info holds the information necessary for the recipient to obtain the right key for
 3753 checking and removing authentication and encryption. In the case of digital signature,
 3754 key_info is not necessary and it shall be a structure of 0 elements.

3755 The fields transaction-id....other-information are A-XDR encoded OCTET STRINGS. The length
 3756 and the value of each field are included in the AAD when applicable.

```

3757     key_info_element ::= structure
3758     {
3759         key_info_type: enum:
3760             (0) identified_key,
3761                 -- used with identified_key_info_options
3762
3763             (1) wrapped_key,
3764                 -- used with wrapped_key_info_options
3765
3766             (2) agreed_key
3767                 -- used with agreed_key_info_options
3768
3769         key_info_options: CHOICE
3770         {
3771             identified_key_info_options,
3772             wrapped_key_info_options,
3773             agreed_key_info_options
3774         }
3775     }
3776     identified_key_info_options ::= enum:
3777         (0) global_unicast_encryption_key,
3778             (1) global_broadcast_encryption_key
3779
3780     wrapped_key_info_options ::= structure
3781     {
3782         kek_id: enum:
3783             (0) master_key
3784             key_ciphered_data: octet-string
3785         }
3786         agreed_key_info_options ::= structure
3787         {
3788             key_parameters: octet-string,
3789             key_ciphered_data: octet-string
3790         }

```

3791 This attribute is first written by the client. The server may need to fill in some additional elements.

3792 The use of the various elements is the same as specified in Table 22 and Table 23 of the Data
 3793 protection IC (class_id = 30, version = 0).

3794 4.4.8.2.2.11 push_operation_method

3795 Defines if the DataNotification.request service primitive is invoked with Service_Class ==
 3796 Unconfirmed or Confirmed and how the push retry operates.

```

3797     enum:
3798         (0) unconfirmed, retry on supporting protocol layer failure,
3799         (1) unconfirmed, retry on missing supporting protocol layer confirmation,
3800         (2) confirmed, retry on missing confirmation.

```

3801 In case (0), the repetition delay for the next retry attempt is started upon supporting layer failure
 3802 reported through the DataNotification.confirm service primitive.

3803

3804 In cases (1) and (2), the repetition delay for the next retry attempt is started when the
 3805 DataNotification.request service primitive is invoked.

3806 In cases (0) and (1), the push operation is deemed as successful when supporting layer
 3807 confirmation is reported by invoking DataNotification.confirm service primitive with Result ==
 3808 CONFIRMED.

3809 In case (2), the push operation is deemed as successful when the server AL receives the data-
 3810 notification-confirm APDU and invokes DataNotification.confirm service primitive with
 3811 Service_Class == Confirmed and Result == CONFIRMED.

3812 **4.4.8.2.2.12 confirmation_parameters**

3813 Defines the selection of entries defined by data_index to avoid pushing data from too far in the
 3814 past. If entries have not yet been confirmed then all entries are selected. This attribute is
 3815 applicable only for relative selective access related to last confirmation.

```
3816     confirmation_parameters ::= structure
3817     {
3818         confirmation_start_date: date-time,
3819         confirmation_interval: double-long-unsigned
3820     }
```

3821 Where:

- 3823 – confirmation_start_date is the starting date and time for selecting entries. Fields of date-
 3824 time not specified are not used. If all fields are not specified the use of
 3825 confirmation_start_date is disabled;
- 3826 – confirmation_interval is a time interval in seconds backwards from the current date and
 3827 time. If confirmation_interval is set to zero, the use of confirmation_interval is disabled.

3828 **4.4.8.2.2.13 last_confirmation_date_time**

3829 Holds the date and time when the AL most recently invoked the DataNotification.confirm service
 3830 primitive with Result == CONFIRMED.

3831 *date-time* is formatted as specified in 4.1.6.1.



3832 **4.4.8.2.3 Method description**

3833 **4.4.8.2.3.1 push (data)**

3834 Activates the push process leading to an attempt to send a DataNotification APDU carrying the
 3835 push data.

3836 *data* ::= integer(0)

3837 **4.4.8.2.3.2 reset (data)**

3838 Resets the push process to initial state.

3839 *data* ::= integer(0)

3840

3841

3842

Table 21 – Encoding of selective access parameters with data_index

	MS_Bit upper nibble: Selects the time intervals 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	Values 0x6 to 0x2 apply only for the Push interface class version 2 (see 4.4.8.2).
0x6	Complete number of months after last confirmation: This is equal to a selective access to the profile <i>buffer</i> resulting in all entries of the complete number of months after the last confirmed entry.
0x5	Complete number of days after last confirmation: This is equal to a selective access to the profile <i>buffer</i> resulting in all entries of the complete number of days after the last confirmed entry.
0x4	Complete number of hours after last confirmation: This is equal to a selective access to the profile <i>buffer</i> resulting in all entries of the complete number of hours after the last confirmed entry.
0x3	Complete number of minutes after last confirmation: This is equal to a selective access to the profile <i>buffer</i> resulting in all entries of the complete number of minutes after the last confirmed entry.
0x2	Number of entries after last confirmation: This is equal to a selective access to the profile <i>buffer</i> resulting in number of entries after the last confirmed entry.
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 !=0: 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: <ul style="list-style-type: none"> – when entries in time intervals are selected, specifies the number of time intervals, value 0x00 defines all time periods ; – when entries are selected, specifies the number of entries, value 0x00 defines all 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.
Example 5)	0x4608: The entries for the last complete 8 hours after the last confirmed entry are selected. The first 6 columns are included.

3843

3844 **4.4.9 COSEM data protection (class_id = 30, version = 0)**3845 **4.4.9.1 Overview**

3846 Instances of this IC allow applying cryptographic protection on COSEM data i.e. on attribute
3847 values and method invocation and return parameters. This is achieved by accessing attributes
3848 and/or methods of other COSEM objects indirectly through instances of the “Data protection”
3849 interface class that provide the necessary mechanisms and parameters to apply / verify /
3850 remove protection on COSEM data.

3851 NOTE 1 “Accessing” includes reading / writing / capturing / pushing COSEM object attributes or invoking methods.

3852 NOTE 2 When attributes and methods of COSEM objects are accessed directly, protection can be provided by
3853 protecting the xDLMS APDUs as stipulated by the relevant security policy and the access rights.

3854 NOTE 3 For definitions and abbreviations related to cryptographic security see IEC 62056-5-3:**2021**, Clause 3.

3855 Protection on COSEM data is aligned with and complements protection on xDLMS APDUs as
3856 defined in IEC 62056-5-3:**2021**, 5.7.

3857 The use cases for COSEM data protection include, but are not limited to:

- 3858 • retrieving a pre-defined set of protected attribute values;
- 3859 • storing a pre-defined set of protected attribute values in “Profile generic” objects for later
3860 retrieval;
- 3861 • pushing a pre-defined set of protected attribute values;
- 3862 • reading or writing selected attributes of other COSEM objects with protection;
- 3863 • invoking a method of another COSEM object with protected method invocation and return
3864 parameters.

3865 Protection may comprise any combination of authentication, encryption and digital signature
3866 and can be applied in a layered manner. The parties applying and removing the protection are
3867 the DLMS/COSEM server and another identified party, which may be a DLMS/COSEM client or
3868 a third party.

3869 Applying data protection between a DLMS/COSEM server and a third party allows keeping
3870 critical / sensitive data confidential towards the client through which the third party accesses
3871 the server. Signing COSEM data by a third party supports non-repudiation.

3872 For end-to-end protection between third parties and servers, see also IEC 62056-5-3:**2021**,
3873 4.1.7 and 5.2.5.

3874 The protection parameters are always controlled by the client with some elements filled in by
3875 the server as appropriate.

3876 The security suite is determined by the “Security setup” object referenced from the current
3877 “Association SN” / “Association LN” object.

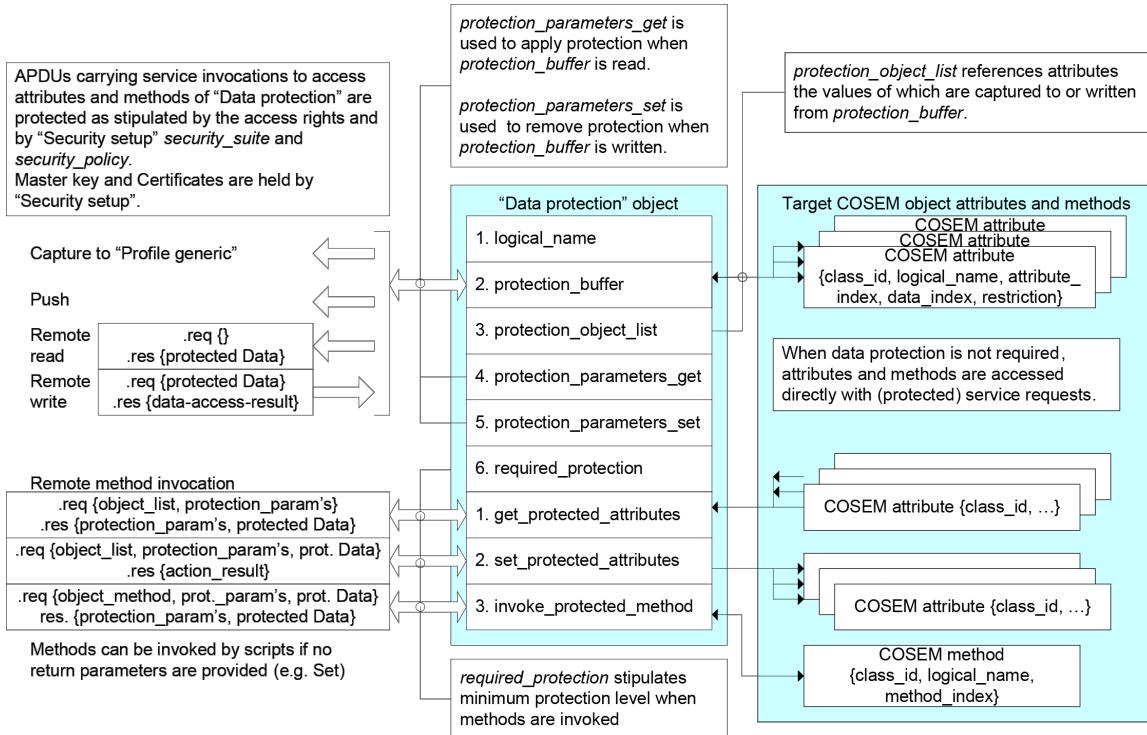
3878 Figure 14 shows the COSEM model of data protection and the relationship of a “Data protection”
3879 object with other COSEM objects.

3880 For accessing attributes of other COSEM objects with protected data, there are two
3881 mechanisms available:

- 3882 • reading or writing the *protection_buffer* attribute. The *protection_buffer* can be also
3883 captured in “Profile generic” objects or pushed using “Push setup” objects;
- 3884 • invoking the *get_protected_attributes* / *set_protected_attributes* method.

3885 For accessing a method of another COSEM object with protected data, the
 3886 *invoke_protected_method* method is available.

3887 APDUs carrying service invocations to access attributes and methods of “Data protection”
 3888 objects are protected as stipulated by access rights to these attributes and methods, and by
 3889 “Security setup” *security_suite* and *security_policy*.



3890

IEC

3891 The master key and Certificates – as required by the security suite – are held by “Security setup”.

3892

Figure 14 – COSEM model of data protection

3893 Protection on COSEM data is applied and removed in the various cases as follows:

- 3894 a) When the *protection_buffer* attribute is read / captured in a “Profile generic” object / pushed:
 - 3895 • attributes determined by *protection_object_list* are captured;
 - 3896 • protection according to *protection_parameters_get* is applied on the set of attributes and the result is put to *protection_buffer*;
 - 3897 • the value of *protection_buffer* is returned / captured in the “Profile generic” *buffer* / pushed using “Push setup” objects.
- 3900 b) When the *protection_buffer* is written:
 - 3901 • protected Data are written to *protection_buffer*; and
 - 3902 • protection according to *protection_parameters_set* is removed and the resulting attribute values are written to the attributes specified by *protection_object_list*.
- 3904 c) When the *get_protected_attributes* method is invoked:
 - 3905 • attributes determined by the *object_list* element of *get_protected_attributes_request* are captured;
 - 3906 • protection according to the *required_protection* attribute and response *protection_parameters* is applied. If *protection_parameters* do not satisfy *required_protection* then the method invocation fails;

- 3910 • the protected attribute values are returned.
- 3911 d) When the *set_protected_attributes* method is invoked:
- 3912 • protection on protected_attributes is verified and removed using the
3913 protection_parameters that must meet *required_protection*;
- 3914 • the resulting attribute values are put in the attributes specified by the object_list
3915 element;
- 3916 e) When the *invoke_protected_method* method is invoked:
- 3917 • protection from protected method invocation parameters is removed using the
3918 protection parameters in the request that must meet *required_protection*;
- 3919 • the method specified by the object_method element of
3920 *invoke_protected_method_request* is invoked with this method invocation parameter;
- 3921 • on the return parameters, the protection using the response protection parameters that
3922 must meet *required_protection* is applied. If protection_parameters do not satisfy
3923 *required_protection* then the method invocation fails;
- 3924 • the protected method return parameters are returned.

3925 Figure 15 shows, as an example, how protected Data in *protection_buffer* is constructed from
3926 the attributes determined by *protection_object_list* according to the *protection_parameters_get*.
3927 See also IEC 62056-5-3:2021, Figure 29.

3928 When the *protection_buffer* attribute is read the following steps are performed:

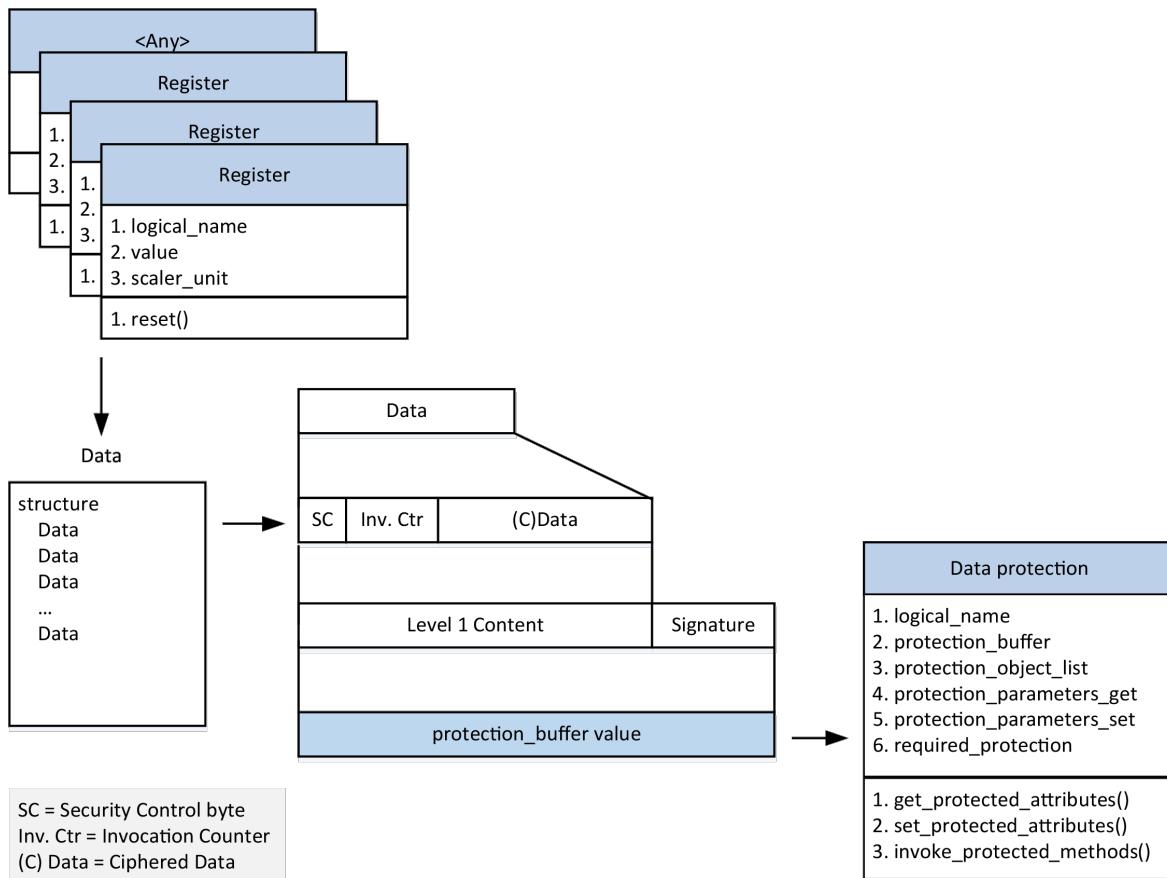
- 3929 f) prerequisites: *protection_object_list*, *protection_parameters_get*, master key, key
3930 agreement and digital signature certificates as needed;
- 3931 g) capture COSEM object attributes determined by *protection_object_list* and create Data, a
3932 structure containing the individual Data of the attributes captured;
- 3933 h) protect Data according to *protection_parameters_get*.

3934 NOTE 4 In the example shown in Figure 15 two layers of protection are applied:

- 3935 – the first layer is a combination of compression / encryption / authentication as determined by the Security
3936 control byte SC, resulting (C)Data,
- 3937 – the second layer is digital signature applied to (C)Data.
- 3938 i) put the protected data, of data type octet-string, into *protection_buffer*;
- 3939 j) return the value of *protection_buffer*.

3940 It may be necessary to read also *protection_parameters_get* to obtain the protection parameters
3941 to verify / remove protection by the recipient.

3942 The invocation counter used when protection is applied / removed is related to the key used.
3943 When the protection is applied the corresponding invocation counter is incremented. When the
3944 key is changed the invocation counter shall be reset to 0.



3945

IEC

3946

Figure 15 – Example: Read *protection_buffer* attribute

3947

Data protection		0...n	class_id = 30, version = 0			
Attribute (s)		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. protection_buffer	(dyn.)	octet-string				x + 0x08
3. protection_object_list	(static)	array				x + 0x10
4. protection_parameters_get	(static)	array				x + 0x18
5. protection_parameters_set	(static)	array				x + 0x20
6. required_protection	(static)	enum				x + 0x28
Specific methods		m/o				
1. get_protected_attributes (data)		m				
2. set_protected_attributes (data)		m				
3. invoke_protected_method (data)		m				

3948

3949 **4.4.9.2 Attribute description**3950 **4.4.9.2.1 logical_name**

3951 Identifies the “Data protection” object instance. See 6.2.36.

3952 **4.4.9.2.2 protection_buffer**

3953 Contains the protected Data.

3954 When read, the attributes determined by *protection_object_list* are captured then protection is
3955 applied according to *protection_parameters_get*.

3956 When written, the protected Data is put into the *protection_buffer*, then the protection is verified
3957 / removed according to *protection_parameters_set* and the attributes determined by
3958 *protection_object_list* are set.

3959 **4.4.9.2.3 protection_object_list**

3960 Defines the list of attributes to be captured to *protection_buffer* when it is read or the list of
3961 attributes to be set when *protection_buffer* is written.

3962 Two, mutually exclusive selective access mechanisms are available:

- 3963 – relative selective access, i.e. entries defined relative to current date or entry are returned:
3964 this mechanism is controlled by the *data_index* element; or
- 3965 – absolute selective access, i.e. entries in an explicitly defined date range or entry range are
3966 returned: this mechanism is controlled by the *restriction* element.

3967 array *object_definition*

3968 *object_definition*::= structure
3969 {
3970 class_id: long-unsigned,
3971 logical_name: octet-string,
3972 attribute_index: integer,
3973 data_index: long-unsigned,
3974 restriction: restriction_element
3975 }
3976 }

3977 Where:

- 3978 – *attribute_index* is a pointer to the attribute within the object, identified by *class_id* and
3979 *logical_name*: *attribute_index* 1 refers to the 1st attribute (i.e. the *logical_name*),
3980 *attribute_index* 2 to the 2nd attribute etc.; *attribute_index* 0 refers to all public attributes;
- 3981 – *data_index* is a pointer selecting one or several specific elements of an attribute with a
3982 complex data type (structure or array):
 - 3983 • if the data type of the attribute is simple, then *data_index* has no meaning;
 - 3984 • if the data type of the attribute is a structure or an array, then *data_index* points to one
3985 or several specific elements in the structure or array;
 - 3986 • when the attribute is the *buffer* of a “Profile generic” object, the *data_index* carries
3987 selective access parameters relative to current date or entry.

3988

<i>data_index</i> :	MS-Byte		LS-Byte
	Upper nibble	Lower nibble	

- 3990 – 0x0000 = identifies the whole attribute;
- 3991 – 0x0001 to 0xFFFF = identifies one element in the complex attribute. The first element in the
3992 complex attribute is identified by data_index 1;
- 3993 – 0x1000 to 0xFFFF = selective access to the array holding the buffer of a “Profile generic”
3994 object. The data-index selects entries within a number of last (recent) time periods, or a
3995 number of last (recent) entries, as well as the columns in the array.

3996 The encoding is specified in Table 25.

3997 When the attribute is the buffer of a “Profile generic” object, then restriction_element specifies
3998 selective access parameters in an explicitly defined date range or entry range.

```

3999             restriction_element ::= structure
4000             {
4001                 restriction_type: enum:
4002                     (0) none,
4003                     (1) restriction_by_date,
4004                     (2) restriction_by_entry
4005                 restriction_value: CHOICE
4006                 {
4007                     null-data,           // no restrictions apply
4008                     restriction_by_date,
4009                     restriction_by_entry
4010                 }
4011             }
4012
4013             restriction_by_date ::= structure
4014             {
4015                 from_date: octet-string,
4016                 to_date: octet-string
4017             }
4018
4019             restriction_by_entry ::= structure
4020             {
4021                 from_entry: double-long-unsigned,
4022                 to_entry: double-long-unsigned
4023             }
4024

```

- 4025 – restriction_element defines absolute selective access to a “Profile generic” buffer by date
4026 range (from_date to to_date) or by entries (from_entry to to_entry). To use this absolute
4027 selective access mechanism, data_index shall be set to 0x0000;
- 4028 – restriction_element is composed of restriction_type and restriction_value:
- 4029 • for restriction by date range the restriction_type element holds (1) restriction by date
4030 and the restriction_value element holds restriction_by_date structure;
 - 4031 • for restriction by entries the restriction_type element holds (2) restriction by entry and
4032 the restriction_value element holds restriction_by_entry structure;
 - 4033 • otherwise, the restriction_type element holds (0) none and the restriction_value
4034 element holds null-data. This choice shall be taken also if relative selective access is
4035 to be used.

4036

4037 4.4.9.2.4 protection_parameters_get

4038 Contains all necessary parameters to specify the protection applied when the *protection_buffer*
4039 is read.

```

4040           array protection_parameters_element
4041
4042           protection_parameters_element ::= structure
4043           {
4044             protection_type: enum:
4045               authentication,
4046               encryption,
4047               authentication and encryption,
4048               digital signature
4049             protection_options: structure
4050             {
4051               transaction_id:          octet-string,
4052               originator_system_title: octet-string,
4053               recipient_system_title: octet-string,
4054               other_information:      octet-string,
4055               key_info:                key_info_element
4056             }
4057           }
4058

```

4059 Where:

- 4060 – transaction_id holds the identifier of the transaction;
- 4061 – originator_system_title holds the system title of the originator that applies the protection;
- 4062 – recipient_system_title holds the system title of the recipient which will check and remove the given protection;
- 4063 – other_information carries other information. Its contents may be specified in project specific companion specifications. An octet-string of length 0 indicates that this field is not used;
- 4064 – key_info holds the information necessary for the recipient to obtain the right key for checking and removing authentication and encryption. In the case of digital signature, key_info is not necessary and it shall be a structure of 0 elements.

4069 The fields transaction-idother-information are A-XDR encoded OCTET STRINGs. The length
4070 and the value of each field are included in the AAD when applicable.

```

4071           key_info_element ::= structure
4072           {
4073             key_info_type: enum:
4074               (0) identified_key,
4075                 -- used with identified_key_info_options
4076               (1) wrapped_key,
4077                 -- used with wrapped_key_info_options
4078               (2) agreed_key
4079                 -- used with agreed_key_info_options
4080             key_info_options: CHOICE
4081             {
4082               identified_key_info_options,
4083               wrapped_key_info_options,
4084               agreed_key_info_options
4085             }
4086           }
4087           identified_key_info_options ::= enum:
4088             global_unicast_encryption_key,
4089             global_broadcast_encryption_key
4090
4091           wrapped_key_info_options ::= structure
4092           {

```

```

4093     kek_id:          enum:
4094         (0) master_key,
4095
4096         key_ciphered_data: octet-string
4097     }
4098     agreed_key_info_options ::= structure
4099     {
4100         key_parameters:    octet-string,
4101         key_ciphered_data: octet-string
4102     }
4103

```

4104 This attribute is first written by the client. The server may need to fill in some additional elements,
 4105 in which case this attribute has to be read back by the client – and if needed, forwarded to the
 4106 third party – so that they can use these parameters to verify / remove protection.

4107

4108 For the use of the various elements see Table 22 and Table 23

4109 **4.4.9.2.5 protection_parameters_set**

4110 Contains all necessary parameters to verify / remove the protection applied when the
 4111 *protection_buffer* is written.

4112 array protection_parameters_element

4113 protection_parameters_element: see the *protection_parameters_get* attribute (4.4.9.2.4).

4114 This attribute is written by the client and it is used by the server to verify and remove protection.

4115 For the use of the various elements see Table 22 and Table 24

4116 **4.4.9.2.6 required_protection**

4117 Stipulates the required protection on the attribute values / invocation and return parameters of
 4118 methods accessed through the “Data protection” object.

4119 enum:

4120 When the enum value is interpreted as an unsigned, the meaning of each bit is as shown below:

4121	Bit	Required protection
4122	0	unused, shall be set to 0,
4123	1	unused, shall be set to 0,
4124	2	authenticated request,
4125	3	encrypted request,
4126	4	digitally signed request,
4127	5	authenticated response,
4128	6	encrypted response,
4129	7	digitally signed response

4130 **4.4.9.3 Method description**

4131 **4.4.9.3.1 get_protected_attributes (data)**

4132 Gets the value of the attributes specified by the *object_list* element, with the protection
 4133 according to the *protection_parameters* in *get_protected_attributes_response* applied.

4134
 4135 Method invocation parameters:
 4136
 4137 data ::= get_protected_attributes_request
 4138
 4139 get_protected_attributes_request ::= structure
 4140 {
 4141 object_list: array object_definition,
 4142 protection_parameters: array protection_parameters_element,
 4143 }
 4144

4145 Return parameters:

4146 data ::= get_protected_attributes_response
 4147
 4148 get_protected_attributes_response ::= structure
 4149 {
 4150 protection_parameters: array protection_parameters_element,
 4151 protected_attributes: octet-string
 4152 }

4153 Where:

- 4154 – object_list: see the *protection_object_list* attribute (4.4.9.2.3);
 4155 – protection_parameters ::= array protection_parameters_element.
 4156 protection_parameters_element: see the *protection_parameters_get* attribute (4.4.9.2.4).

4157 The protection_parameters in get_protected_attributes_response are elaborated by copying the
 4158 protection_parameters from get_protected_attributes_request except that the server may need
 4159 to fill in some elements. The remote party uses the protection_parameters in
 4160 get_protected_attributes_response to verify / remove protection on the protected_attributes
 4161 returned.

4162 For the use of the various elements see Table 22 and Table 25.

4163 4.4.9.3.2 set_protected_attributes (data)

4164 Sets the value of the attributes specified by the object_list element, after verifying and removing
 4165 the protection on the protected_attributes element according to the protection_parameters
 4166 element.

4167 Method invocation parameters:

4168 data ::= set_protected_attributes_request
 4169 set_protected_attributes_request ::= structure
 4170 {
 4171 object_list: array object_definition,
 4172 protection_parameters: array protection_parameters_element,
 4173 protected_attributes: octet-string
 4174 }

4175 Where:

- 4176 – object_list: see the *protection_object_list* attribute (4.4.9.2.3);
 4177 – protection_parameters ::= array protection_parameters_element.
 4178
 4179 protection_parameters_element: see the *protection_parameters_get* attribute (4.4.9.2.4).
 4180

4181 For the use of the various elements see Table 22 and Table 26.

4182 **4.4.9.3.3 invoke_protected_method (data)**

4183 Invokes the method specified by the object_method element, after verifying and removing the
 4184 protection on protected_method_invocation_parameters according to the
 4185 protection_parameters in invoke_protected_method_request.

4186 After invoking the method specified by the object_method element, the protection according to
 4187 the protection_parameters in invoke_protected_method_response – that must meet
 4188 *required_protection* – is applied on the return parameters, and
 4189 invoke_protected_method_response composed of protection_parameters and
 4190 protected_method_return_parameters is returned.

4191 Method invocation parameters:

```

4192     data ::= invoke_protected_method_request
4193
4194         invoke_protected_method_request ::= structure
4195         {
4196             object_method:          object_method_definition,
4197
4198             protection_parameters: array      protection_parameters_element,
4199             protected_method_invocation_parameters: octet-string
4200         }
4201
4202         object_method_definition ::= structure
4203         {
4204             class_id:        long-unsigned,
4205             logical_name:    octet-string,
4206             method_index:   integer,
4207         }
4208
4209
4210     Return parameters:
4211     data ::= invoke_protected_method_response
4212
4213         invoke_protected_method_response ::= structure
4214         {
4215             protection_parameters: array      protection_parameters_element,
4216             protected_method_return_parameters: octet-string
4217         }
4218
4219         protection_parameters_element: see the specification of the
4220         protection_parameters_get attribute (4.4.9.2.4).
4221

```

4222 If the method invoked does not provide return parameters then
 4223 invoke_protected_method_response shall be a structure, with protection_parameters an array
 4224 of zero elements and protected_method_return_parameters an octet-string of zero length.

4225 To verify and remove protection from protected_method_invocation_parameters, the server
 4226 uses the protection_parameters in invoke_protected_method_request that must meet
 4227 *required_protection*.

4228 The protection_parameters in the invoke_protected_method_response are elaborated by
 4229 copying the protection_parameters from invoke_protected_method_request to
 4230 invoke_protected_method_response except that the server may need to fill in some elements.

4231 The protection_parameters in invoke_protected_method_response – that must meet
4232 *required_protection* – are then applied to protect data in the
4233 protected_method_return_parameters.

4234 For the use of the various elements see Table 22 and Table 27.

4235

Table 22 – Key information required to establish data protection keys

Key information choices		Comment
key_info_options		
(0) identified_key_info_options	S	The EK is identified
(0) global_unicast_encryption_key	S	GUEK
(1) global_broadcast_encryption_key	S	GBEK
(1) wrapped_key_info_options	S	The EK is transported using key wrap
kek_id	M	
(0) master_key	M	Identifies the key used for wrapping the key_ciphered_data. 0 = Master Key (KEK)
key_ciphered_data	M	Randomly generated key wrapped with KEK.
(2) agreed_key_info_options	S	The key is agreed by the parties using either: <ul style="list-style-type: none">– the One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme or– the Static Unified Model C(0e, 2s, ECC CDH) scheme
key_parameters	M	Identifier of the key agreement scheme: 0x01: C(1e, 1s, ECC CDH) 0x02: C(0e, 2s, ECC CDH) All other reserved.
key_ciphered_data	M	<ul style="list-style-type: none">– In the case of the C(1e, 1s, ECC CDH) scheme: the public key Q_u of the ephemeral key agreement key pair of party U, signed with the private digital signature key of party U.– In the case of the C(0e, 2s, ECC CDH) scheme: an octet-string of length zero. <p>NOTE In the second case party U has to provide a nonce, NonceU. See IEC 62056-5-3:2021, 5.3.4.6.4 and 5.3.4.6.5.</p>
<p>M: Mandatory (part of a structure)</p> <p>S: Selectable (part of a CHOICE)</p>		

4236

4237

Table 23 – Protection parameters of *protection_parameters_get* attribute

protection_parameters_get		Written by the client	Filled in by the server
array protection_parameters_element	M	x	
protection_type	M	x	
(0) authentication	S	x	
(1) encryption	S	x	
(2) authentication and encryption	S	x	
(3) digital signature	S	x	
protection_options	M	x	
transaction_id	M	x	
originator_system_title (Shall carry the server system title)	M	x	
recipient_system_title (Shall carry the remote party system title)	M	x	
other_information	M	x	
key_info	M	x	
key_info_type	M	x	
(0) identified_key	S	x	
(1) wrapped_key	S	x	
(2) agreed_key	S	x	
key_info_options	M	x	
identified_key_options	S	x	
global_unicast_encryption_key	S	x	
global_broadcast_encryption_key	S	x	
wrapped_key_options	S	x	
kek_id	M	x	
(0) master_key	M	x	
key_ciphered_data	M	x	
agreed_key_options	S	x	
key_parameters	M	x	
key_ciphered_data	M		x
The <i>protection_parameters_get</i> attribute is written by the client, but when <i>key_info_type</i> is (2) agreed key, the <i>key_ciphered_data</i> of <i>agreed_key_options</i> is empty.			
When the One-pass Diffie-Hellman c(1e, 1s, ECC CDH) scheme is used, this element is filled by the server and the remote party has to read back <i>protection_parameters_get</i> in order to be able to verify and remove protection.			
M: Mandatory (part of a structure)			
S: Selectable (part of a CHOICE)			

4238

4239

Table 24 – Protection parameters of *protection_parameters_set* attribute

protection_parameters_set		Written by the client	Filled in by the server
array protection_parameters_element	M	x	
protection_type	M	x	
(0) authentication	S	x	
(1) encryption	S	x	
(2) authentication and encryption	S	x	
(3) digital signature	S	x	
protection_options	M	x	
transaction_id	M	x	
originator_system_title (Shall carry the remote party system title)	M	x	
recipient_system_title (Shall carry the server systems title)	(M	x	
other_information	M	x	
key_info	M	x	
key_info_type	M	x	
(0) identified_key	S	x	
(1) wrapped_key	S	x	
(2) agreed_key	S	x	
key_info_options	M	x	
identified_key_options	S	x	
(0) global_unicast_encryption_key	S	x	
(1) global_broadcast_encryption_key	S	x	
wrapped_key_options	S	x	
kek_id	M	x	
(0) master_key	M	x	
key_ciphered_data	M	x	
agreed_key_options	S	x	
key_parameters	M	x	
key_ciphered_data	M	x	

M: Mandatory (part of a structure)

S: Selectable (part of a CHOICE)

4240

4241

Table 25 – Protection parameters of *get_protected_attributes* method

Protection parameters	request	response
array protection_parameters_element	M	M (=)
protection_type	M	M (=)
(0) authentication	S	S (=)
(1) encryption	S	S (=)
(2) authentication and encryption	S	S (=)
(3) digital signature	S	S (=)
protection_options	M	M (=)
transaction_id	M	M (=)
originator_system_title	M	M ¹
recipient_system_title	M	M ²
other_information	M	M (=)
key_info	M	M (=)
key_info_type	M	M (=)
(0) identified_key	S	S (=)
(1) wrapped_key	S	S (=)
(2) agreed_key	S	S (=)
key_info_options	M	M (=)
identified_key_options	S	S (=)
(0) global_unicast_encryption_key	S	S (=)
(1) global_broadcast_encryption_key	S	S (=)
wrapped_key_options	S	S (=)
kek_id	M	M (=)
(0) master_key	M	M (=)
key_ciphered_data	–	M ³
agreed_key_options	S	S (=)
key_parameters	M	M (=)
key_ciphered_data	–	M ⁴
Notice that protection parameters are taken from request and put into response. Protection parameters are only applied for protection of response.		
1 originator_system_title from request is put into recipient_system_title of response.		
2 recipient_system_title from request is put into originator_system_title of response.		
3 key_ciphered_data is filled by the server.		
4 When the One-pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme is used, this element is filled by the server.		
M: Mandatory (part of a structure)		
S: Selectable (part of a CHOICE)		

4242

4243

Table 26 – Protection parameters of *set_protected_attributes* method

Protection parameters	request	response
array protection_parameters_element	M	
protection_type	M	
(0) authentication	S	
(1) encryption	S	
(2) authentication and encryption	S	
(3) digital signature	S	
protection_options	M	
transaction_id	M	
originator_system_title	M	
recipient_system_title	M	
other_information	M	
key_info	M	
key_info_type	M	
(0) identified_key	S	
(1) wrapped_key	S	
(2) agreed_key	S	
key_info_options	M	
identified_key_options	S	
(0) global_unicast_encryption_key	S	
(1) global_broadcast_encryption_key	S	
wrapped_key_options	S	
kek_id	M	
(0) master_key	M	
key_ciphered_data	M	
agreed_key_options	S	
key_parameters	M	
key_ciphered_data	M	
Notice that protection parameters are taken from request and are not present in response. Protection parameters are only used for removing protection on the request.		
M: Mandatory (part of a structure)		
S: Selectable (part of a CHOICE)		

4244

4245

Table 27 – Protection parameters of *invoke_protected_method* method

Protection parameters	request	response
array protection_parameters_element	M	M(=)
protection_type	M	M(=)
(0) authentication	S	S(=)
(1) encryption	S	S(=)
(2) authentication and encryption	S	S(=)
(3) digital signature	S	S(=)
protection_options	M	M(=)
transaction_id	M	M(=)
originator_system_title	M	M ¹
recipient_system_title	M	M ²
other_information	M	M(=)
key_info	M	M(=)
key_info_type	M	M(=)
(0) identified_key	S	S(=)
(1) wrapped_key	S	S(=)
(2) agreed_key	S	S(=)
key_info_options	M	M(=)
identified_key_options	S	S(=)
(0) global_unicast_encryption_key	S	S(=)
(1) global_broadcast_encryption_key	S	S(=)
wrapped_key_options	S	S(=)
kek_id	M	M(=)
(0) master_key	M	M(=)
key_ciphered_data	M	M ³
agreed_key_options	S	S(=)
key_parameters	M	M(=)
key_ciphered_data	M	M ⁴
Notice that protection parameters are taken from request and put into response. Protection parameters in the request are used to remove protection from request and protection parameters in the response are used to apply protection on response.		
1 originator_system_title from request is put into recipient_system_title of response.		
2 recipient_system_title from request is put into originator_system_title of response.		
3 key_ciphered_data is sent by the originator in the request and filled by the server for the response.		
4 When the One-pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme is used, this element is filled by the server.		
M: Mandatory (part of a structure)		
S: Selectable (part of a CHOICE)		

4246

4247 **4.4.10 Function control (class_id: 122, version: 0)**4248 **4.4.10.1 Overview**4249 Instances of the IC “Function control” allow enabling and disabling functions in the server. Each
4250 function that can be enabled / disabled is identified by a name and is defined by a particular set
4251 of object identifiers referenced.

4252 To allow enabling and disabling of functions controlled by time, “Single action schedule” and
 4253 “Script table” objects are also specified.

Function control	0...n	class_id = 122, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. activation_status (dyn.)	array				x + 0x08
3. function_list (static)	array				x + 0x10
Specific methods	m/o				
1. set_function_status (data)	m				
2. add_function (data)	o				
3. remove_function (data)	o				

4254

4.4.10.2 Attribute description

4.4.10.2.1 logical_name

4257 Identifies the “Function control” object instance. See 6.2.38.

4.4.10.2.2 activation_status

4259 Shows the current status of each functional block defined in the *function_list* attribute.

```
4260     activation_status ::= array function_status_type
4261     function_status_type ::= structure
4262     {
4263       function_name   octet-string,
4264       function_status boolean
4265     }
```

4266

4267 TRUE =function enabled,

4268 FALSE = function disabled.

4.4.10.3 function_list

4270 Defines a list of functions which can be enabled or disabled by invoking the *set_function_status* method.

```
4272     function_list ::= array    functional_block
4273
4274     functional_block ::=    structure
4275     {
4276       function_name:          octet-string,
4277       function_specification: array function_definition
4278     }
4279
4280     function_definition ::= structure
4281     {
4282       class_id:              long-unsigned,
4283       logical_name:          octet-string
4284     }
```

4286 The function_specification element contains a list of objects involved in this function. The
 4287 objects are referenced by their class ids and logical names.

4288 In case of functions that cannot be linked to specific objects, class_id = 0 is used and the
4289 second element holds a descriptive name instead of a logical_name.

4290 The names of the functions and the behaviour of the server in case of enabling or disabling a
4291 function have to be defined in project specific companion specifications.

4292 Please also note that this attribute controls only the functions that are available in the server
4293 and that can be enabled or disabled. It is not possible to download new functions to the server
4294 by modifying this attribute or by invoking the *add_function* method.

4295 **4.4.10.4 Method description**

4296 **4.4.10.4.1 set_function_status (data)**

4297 Enables or disables one or more functions defined in attribute *function_list*.

4298 data ::= array function_status_type

4299 as defined in attribute *activation_status* (4.4.10.2.2).

4300 The status of functions that are not listed is not modified.

4301 **4.4.10.4.2 add_function (data)**

4302 Adds a new function to the attribute *function_list*.

4303 If a function with the same name already exists, the existing function will be replaced by the
4304 new function.

4305 data ::= functional_block

4306 functional_block see 4.4.10.3.

4307 **4.4.10.4.3 remove_function (data)**

4308 Removes a function from the attribute *function_list*.

4309 data ::= octet-string, holding the function_name.

4310

4311 **4.4.11 Array manager (class_id = 123, version = 0)**

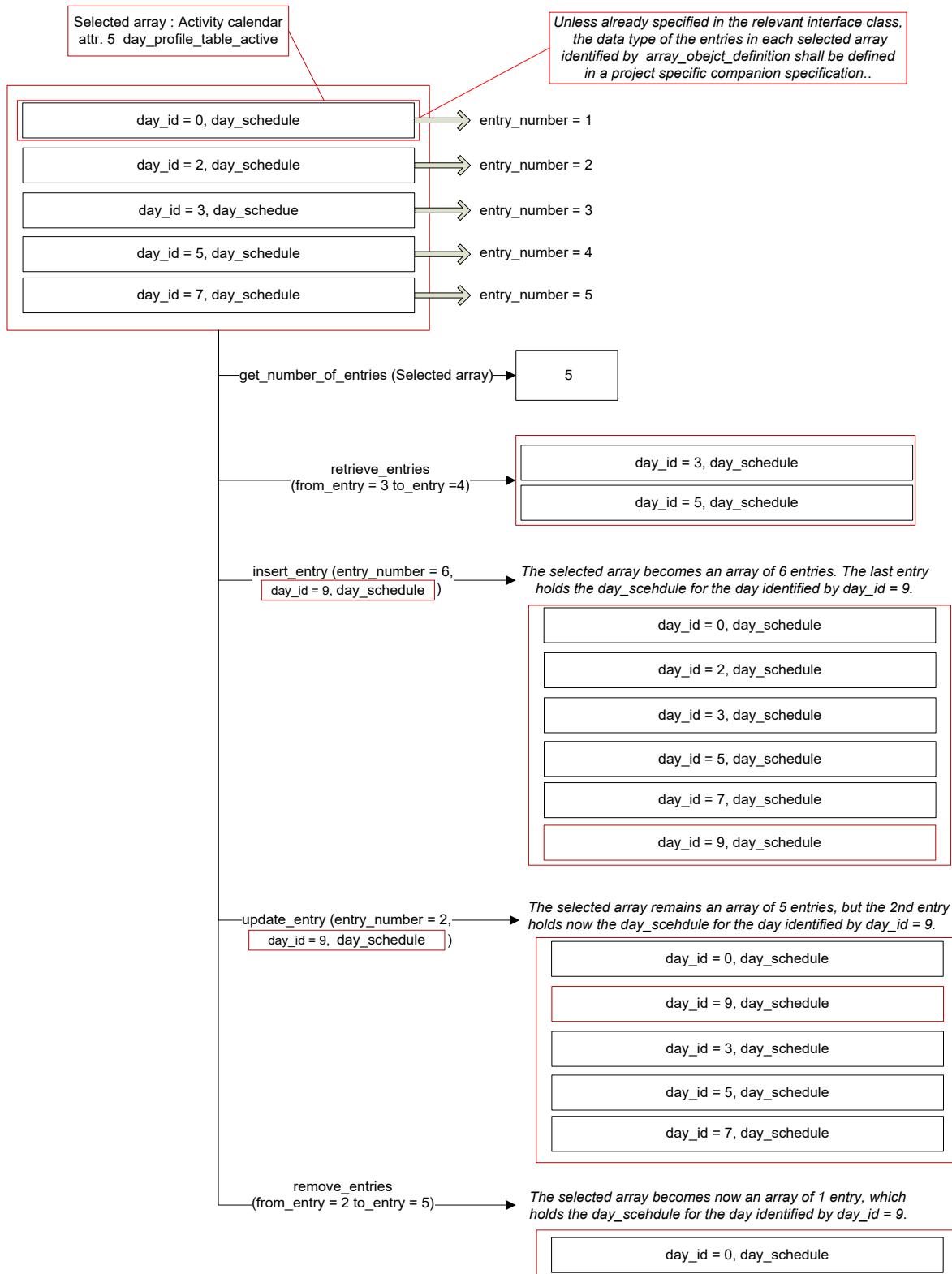
4312 **4.4.11.1 Overview**

4313 Instances of the “Array manager” IC allow managing attributes of type *array* of other interface
4314 objects, i.e.:

- 4315 • retrieving the number of entries;
- 4316 • selectively reading a range of entries;
- 4317 • inserting a new entry or updating an existing entry;
- 4318 • removing a range of entries.

4319 Each instance allows managing several attributes of type *array* assigned to it.

4320 An example of the application is shown in Figure 16.



4321

4322

4323

Figure 16 – Example of managing an array

Array manager	0...n	class_id = 123, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. array_object_list (static)	array				x + 0x08
Specific methods	m/o				
1. retrieve_number_of_entries(data)	m				
2. retrieve_entries (data)	m				
3. insert_entry (data)	o				
4. update_entry (data)	o				
5. remove_entries (data)	o				

4324

4325 **4.4.11.2 Attribute description**4326 **4.4.11.2.1 logical_name**

4327 Identifies the “Array manager” object instance. See 6.2.16.

4328 **4.4.11.2.2 array_object_list**4329 Defines the list of attributes of type array that can be managed by an instance of this IC. Each
4330 “Array manager” object can manage 1 to n arrays.

```

4331           array      array_object_list_definition
4332
4333           array_object_list_definition ::=   structure
4334           {
4335             array_object_id:          unsigned,
4336             array_object_list_element: array_object_definition
4337           }
4338           array_object_definition ::=   structure
4339           {
4340             class_id:        long-unsigned,
4341             logical_name:    octet-string,
4342             attribute_index: integer
4343           }
4344

```

4345 Where attribute_index is a pointer to the attribute within the object identified by its class_id and
4346 logical_name. Attribute_index 1 refers to the 1st attribute (i.e. the logical_name),
4347 attribute_index 2 to the 2nd, etc.

4348 Each attribute identified by the array_object_definition shall be of type array.

4349 An attribute of type array holds a number of entries that can be referenced by their entry number.
4350 The number 1 identifies the first entry and the number m identifies the last entry, where m is
4351 the number of entries of a selected array.4352 Unless already specified in the relevant interface class, the data type of the entries in each
4353 array identified by array_object_definition shall be specified in a project specific companion
4354 specification.4355 When the target array is accessed through an “Array manager” object, its access rights are
4356 observed.

4357 **4.4.11.3 Method description**4358 **4.4.11.3.1 retrieve_number_of_entries (data)**

4359 Returns the number of entries in the array identified.

4360 The method invocation parameter identifies an array from the array_object_list by its
4361 array_object_id:

4362 data:= unsigned

4363 The return parameter holds the number of entries in the array identified.

4364 **4.4.11.3.2 retrieve_entries (data)**

4365 Returns a range of entries in one of the arrays of the array_object_list.

4366 The method invocation parameters identify an array from the array_object_list by its
4367 array_object_id and the entries to be retrieved:

4368 data::= entries_to_retrieve
4369
4370 entries_to_retrieve::= structure
4371 {
4372 array_object_id: unsigned,
4373 list_of_entries: structure
4374 {
4375 from_entry: long-unsigned,
4376 to_entry: long-unsigned
4377 }
4378 }
4379

4380 Where:

- 4381 – array_object_id identifies one of the arrays managed by an “Array manager” object;
4382 – list_of_entries identifies the entries to be retrieved.

4383 If the number of entries in the selected array is m:

- 4384 – if from_entry < last_entry < m, then the entries in this range are retrieved;
4385 – if from_entry < last_entry but last_entry > m, then the entries identified by from_entry up to
4386 the last entry are retrieved;
4387 – if from_entry < last_entry, but from_entry > m, then the method returns an array of 0
4388 elements;
4389 – if from_entry > last_entry the method invocation fails.

4390 The return parameters hold an array of the entries selected:

4391 data::= retrieved_entries:
4392 retrieved_entries: array entry
4393 entry::= attribute specific

4394

4395 The data type depends on the attribute of type *array* as specified in the IC specification or in
 4396 the project specific companion specification.

4397 **4.4.11.3.3 insert_entry (data)**

4398 Inserts a new entry in a selected array.

```
4399           data ::=      structure
4400           {
4401             array_object_id:      unsigned,
4402             entry_to_insert:     structure
4403             {
4404               entry_number:    long-unsigned,
4405               entry:          attribute specific
4406
4407             The data type of the entry depends on the attribute of type array as specified
4408             in the IC specification or in the project specific companion specification.
4409           }
4410         }
```

4411 Where:

- 4412 – *array_object_id* identifies one of the arrays managed by an “Array manager” object;
- 4413 – *entry_to_insert* identifies the *entry_number* and holds the entry itself to be inserted.

4414

4415 If *entry_number* = 0:

- 4416 – if the selected array is already full, then the method invocation fails;
- 4417 – else, the new entry is inserted at the beginning of the array: it becomes the first entry and
 4418 all the other entries of the selected array are shifted up by one;

4419

4420 If *entry_number* > m (where m is the number of entries in the array):

- 4421 – if the selected array is already full, then the method invocation fails;
- 4422 – else, the new entry is inserted at the end of the selected array: it becomes the last entry.
 4423 The other entries are not shifted.

4424 If an *entry_number* identifies an existing entry, it is inserted after the existing entry.

4425 If the entries are shifted due to inserting a new entry then all the references to them have to be
 4426 updated.

4427 **4.4.11.3.4 update_entry (data)**

4428 Updates an existing entry in a selected array.

```
4429           data ::=      structure
4430           {
4431             array_object_id:      unsigned,
4432             entry_to_update:     structure
4433             {
4434               entry_number:    long-unsigned,
4435               entry:          attribute specific
4436
4437             The data type of the entry depends on the attribute of type array as specified
4438             in the IC specification or in the project specific companion specification.
4439           }
4440         }
```

The data type of the entry depends on the attribute of type *array* as specified
 in the IC specification or in the project specific companion specification.

4441

4442 Where:

- 4443 – array_object_id identifies one of the arrays managed by an “Array manager” object;
 4444 – entry_to_update identifies the entry_number and holds the entry itself to be updated.

4445 When an entry_number identifies an existing entry, it is overwritten.

4446 In any other case, the method invocation fails.

4447 **4.4.11.3.5 remove_entries (data)**

4448 Removes a range of entries in one of the arrays of the array_object_list.

4449 The method invocation parameters identify an array from the array_object_list by its
 4450 array_object_id and the entries to be removed:

```

4451           data ::= entries_to_remove
4452           entries_to_remove ::= structure
4453             {
4454               array_object_id: unsigned;
4455               list_of_entries: structure
4456                 {
4457                   from_entry: long-unsigned,
4458                   to_entry: long-unsigned
4459                 }
  
```

4463 Where:

- 4464 – array_object_id identifies one of the arrays managed by an “Array manager” object;
 4465 – list_of_entries identifies the entries to be removed.

4466

4467 If the number of entries in the selected array is m:

- 4468 – if from_entry < last_entry < m, then the entries in this range are removed;
 4469 – if from_entry < last_entry but last_entry > m, then the entries identified by from_entry up to
 4470 the last entry are removed;
 4471 – if from_entry < last_entry, but from_entry > m, then no entries are removed;
 4472 – if from_entry > last_entry the method invocation fails.

4473 If the entries are shifted due to removing entries, then all the references to them have to be
 4474 updated.

4475 **4.4.12 Communication port protection (class_id = 124, version = 0)**4476 **4.4.12.1 Overview**

4477 Instances of the “Communication port protection” IC can be used to protect communication
 4478 ports of DLMS/COSEM servers against possibly malicious communication attempts, in
 4479 particular to prevent brute force attacks by reducing the possible number of attempts.

4480 Each instance references a single communication port by its logical name (OBIS code). If an
 4481 acceptable number of failed attempts is exceeded then the communication port is temporarily

4482 locked. The lockout time may increase with each failed attempt, until a maximum lockout time
 4483 is reached.

4484 A failed attempt is one that leads to discarding the APDU carrying a service request. The criteria
 4485 for detecting a failed attempt are out of the Scope of this document.

4486 The objects count both the number of failed attempts between two resets and the cumulative
 4487 number of failed attempts.

4488 It is possible to configure the communication ports such that they are locked to all attempts or
 4489 unlocked to all attempts

Communication port protection		0...n	class_id = 124, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1. logical_name	(static)	octet-string				x
2. protection_mode	(static)	enum			1	x + 0x08
3. allowed_failed_attempts	(static)	long-unsigned				x + 0x10
4. initial_lockout_time	(static)	double-long-unsigned				x + 0x18
5. steepness_factor	(static)	unsigned			1	x + 0x20
6. max_lockout_time	(static)	double-long-unsigned				x + 0x28
7. port_reference	(static)	octet-string				x + 0x30
8. protection_status	(dyn.)	enum				x + 0x38
9. failed_attempts	(dyn.)	double-long-unsigned				x + 0x40
10. cumulative_failed_attempts	(dyn.)	double-long-unsigned				x + 0x48
Specific methods	m/o					
1. reset (data)	o	o				x + 0x50

4490

4.4.12.2 Attribute description

4.4.12.2.1 logical_name

4493 Identifies the “Communication port protection” object instance. See 6.2.39.

4.4.12.2.2 protection_mode

4495 Controls the protection mode.

4496 enum
 4497 (0) locked,
 4498 (1) locked_on_failed_attempts,
 4499 (2) unlocked

4500 When set to locked, no communication can take place via the port referenced by attribute 7.

4501 When set to “locked_on_failed_attempts”, the port becomes temporarily locked when the
 4502 number of failed communication attempts exceeds an allowed number. The length of the lockout
 4503 period may increase with each failed attempt, as determined by attributes 5 and 6.

4504 When set to “unlocked” the lockout mechanism is disabled; communication through the port
 4505 referenced by attribute 7 is always possible.

4506 When the port is locked no communication can take place, even valid APDUs are discarded.

4507 **4.4.12.2.3 allowed_failed_attempts**

4508 Holds the number of allowed failed communication attempts before the lockout mechanism is
 4509 triggered.

4510 **4.4.12.2.4 initial_lockout_time**

4511 Holds the initial value of the lockout time, in seconds, after the first failed communication
 4512 attempt when the value of allowed_failed_attempts is reached.

4513 **4.4.12.2.5 steepness_factor**

4514 Holds a factor that controls how the lockout time is increased with each failed attempt when
 4515 allowed_failed_attempts is reached, until the max_lockout_time is reached.

4516 The current lockout time is calculated using the following formulae:

4517 $NCA = \text{failed_attempts} - \text{allowed_failed_attempts}$

4518 $CLT = \text{initial_lockout_time} \times (\text{steepness_factor}^{(NCA-1)})$

4519 Where:

- 4520 • NCA is number of counted failed attempts: that is: the number of attempts that count
 4521 towards extending the lockout time,
- 4522 • CLT is the current lockout time.

4523 Table 30 shows example values with an initial lockout time of 60s.

4524 **Table 28 – Example values for NCA and CLT**

NCA	steepness_factor		
	1	2	3
	CLT (in seconds)		
1	60	60	60
2	60	120	180
3	60	240	540
4	60	480	1620
5	60	960	4860

4525

4526 Once lockout occurs, even valid attempts are discarded until the current lockout time has
 4527 expired.

4528 **4.4.12.2.6 max_lockout_time**

4529 Holds the maximum time, in seconds, for which the communication port can be locked, even if
 4530 the number of failed attempts keeps increasing. The purpose of this attribute is to avoid a denial
 4531 of service attack.

4532 **4.4.12.2.7 port_reference**

4533 Contains the logical name of an object that identifies the communication port being protected.
 4534 That object may be a communication port setup object or any other suitably chosen object.

4535 If this information is not available or not needed, the attribute may be left empty (octet-string
4536 [0]).

4537 **4.4.12.2.8 protection_status**

4538 Holds the current protection status of the communication port.

4539 enum

4540 (0) unlocked,

4541 (1) temporarily_locked,

4542 (2) locked

4543 The status is unlocked if the protection_mode is unlocked or the number of failed attempts does
4544 not exceed the number of allowed_failed_attempts.

4545 NOTE 1 The status becomes unlocked again when the current_lockout_time has elapsed.

4546 The status is temporarily_locked if the number of failed communication attempts exceeds
4547 allowed_failed_attempts and the current_lockout_time has not yet expired.

4548 The status is locked if the protection_mode is locked.

4549 NOTE 2 When a port is locked, the server still may be accessed on another port.

4550 failed_attempts Holds the total number of failed attempts since the last reset, independent of
4551 the protection_mode and the triggering of the protection mechanism.

4552 **4.4.12.2.9 cumulative_failed_attempts**

4553 Holds the cumulative number of failed attempts, independent of the protection_mode and the
4554 triggering of the protection mechanism.

4555 This attribute is never reset.

4556 **4.4.12.3 Method description**

4557 **4.4.12.3.1 reset (data)**

4558 Resets the variables of the lockout mechanism:

4559 – the value of failed_attempts is reset to 0;

4560 – the current lockout time is reset to 0;

4561 – the protection_status is set to "(0) unlocked" if the protection_mode is (1)
4562 locked_on_failed_attempts, and is not affected otherwise.

4563 data ::= integer (0)

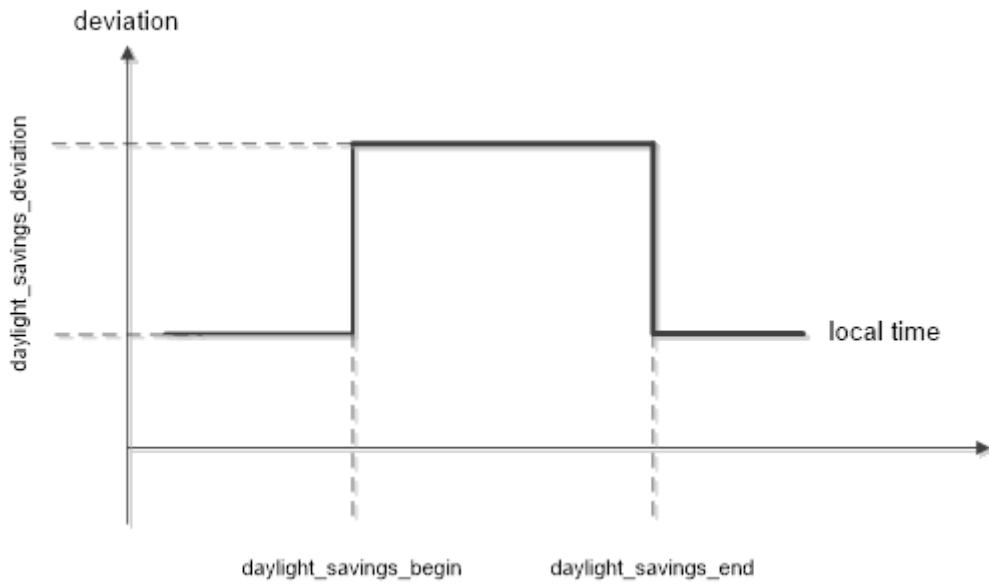
4564

4565

4566 4.5 Interface classes for time- and event bound control**4567 4.5.1 Clock (class_id = 8, version = 0)****4568 4.5.1.1 Overview**

4569 This IC models the device clock, managing all information related to date and time including
4570 deviations of the local time to a generalized time reference (UTC) due to time zones and daylight
4571 saving time schemes. The IC also offers various methods to adjust the clock.

4572 The *date* information includes the elements year, month, day of month and day of week. The
4573 *time* information includes the elements hour, minutes, seconds, hundredths of seconds, and the
4574 deviation of the local time from UTC. The daylight saving time function modifies the deviation
4575 of local time to UTC depending on the attributes; see Figure 17.

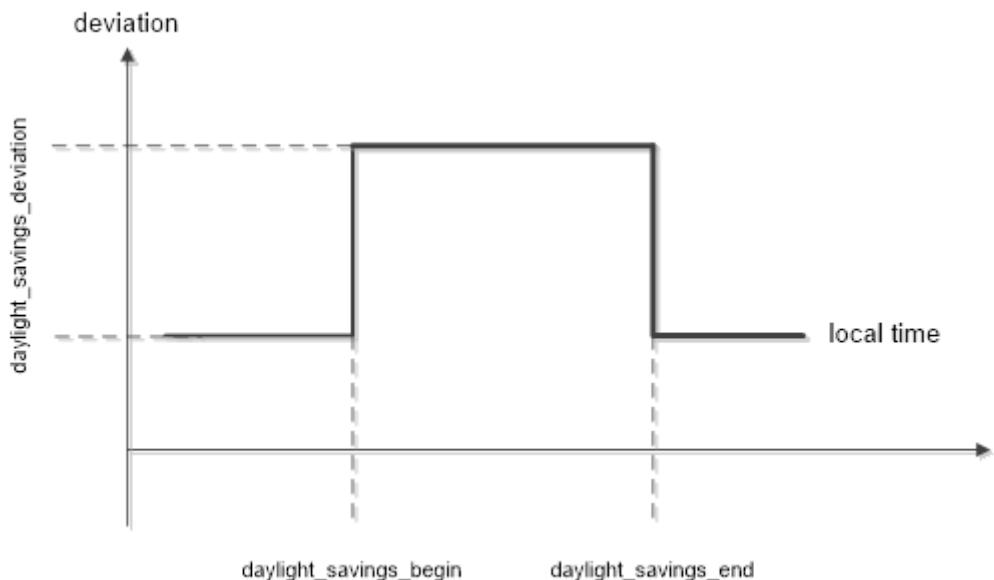


4576

4577

Figure 17 – The generalized time concept

4579 The start and end point of that function is normally set once. An internal algorithm calculates
4580 the real switch point depending on these settings.



4581

4582

4583

Figure 17 – The generalized time concept

4584

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	+840		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	+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

4585

4.5.1.2 Attribute description

4.5.1.2.1 logical_name

Identifies the “Clock” object instance. See 6.2.5.

4589 **4.5.1.2.2 time**

4590 Contains the meter's local date and time, its deviation to UTC and the status.

4591 octet-string, formatted as specified in 4.1.6.1 for *date-time*.

4592 When this attribute is set, all the fields in the octet-string representing *date-time* shall be
4593 evaluated and the local date and time in the meter shall be set according to the rules defined
4594 in 4.1.6.1.

4595 Only specified fields of the *date-time* are changed.

4596 EXAMPLE For setting the *date* without changing the *time*, all time-relevant octets in the octet string representing
4597 *date-time* shall be set to "not specified".

4598 **4.5.1.2.3 time_zone**

4599 The deviation of local, normal time to UTC in minutes. The value depends on the geographical
4600 location of the meter.

4601 **4.5.1.2.4 status**

4602 The *clock_status* maintained by the meter. The *clock_status* element indicated in the *time*
4603 attribute is equal to the value of this attribute.

4604 unsigned, formatted as specified in 4.1.6.1 for *clock_status*

4605 **4.5.1.2.5 daylight_savings_begin**

4606 Defines the local switch date and time when the local time starts to deviate from the normal
4607 time.

4608 For generic definitions, wildcards are allowed.

4609 octet-string, formatted as specified in 4.1.6.1 for *date-time*.

4610 **4.5.1.2.6 daylight_savings_end**

4611 Defines the local switch date and time when the local time ends to deviate from the local normal
4612 time.

4613 octet-string, formatted as specified in 4.1.6.1 for *date-time*.

4614 **4.5.1.2.7 daylight_savings_deviation**

4615 Contains the number of minutes by which the deviation in generalized time shall be corrected
4616 at daylight savings begin.

4617 integer: Deviation in the range of ± 120 min

4618 **4.5.1.2.8 daylight_savings_enabled**

4619 boolean: TRUE = DST enabled,

4620 FALSE = DST disabled

4621 NOTE This is a parameter to enable and disable the daylight savings time feature. The current status is indicated
4622 in the status attribute.

4623 **4.5.1.2.9 clock_base**

4624 Defines where the basic timing information comes from.

4625 enum: (0) not defined,
 4626 (1) internal crystal,
 4627 (2) mains frequency 50 Hz,
 4628 (3) mains frequency 60 Hz,
 4629 (4) GPS (global positioning system),
 4630 (5) radio controlled.
 4631

4632 **4.5.1.3 Method description**4633 **4.5.1.3.1 adjust_to_quarter (data)**

4634 Sets the meter's time to the nearest (+/-) quarter of an hour value
 4635 (*:00, *:15, *:30, *:45).

4636 data::= integer (0)

4637 **4.5.1.3.2 adjust_to_measuring_period (data)**

4638 Sets the meter's time to the nearest (+/-) starting point of a measuring period.

4639 data::= integer (0)

4640 **4.5.1.3.3 adjust_to_minute (data)**

4641 Sets the meter's time to the nearest minute.

4642 If second_counter < 30 s, second_counter is set to 0.

4643 If second_counter ≥ 30 s, second_counter is set to 0, and minute_counter and all depending
 4644 clock values are incremented if necessary.

4645 data::= integer (0)

4646 **4.5.1.3.4 adjust_to_preset_time (data)**

4647 This method is used in conjunction with the preset_adjusting_time method. If the meter's time
 4648 lies between validity_interval_start and validity_interval_end, then time is set to preset_time.

4649 data::= integer (0)

4650 **4.5.1.3.5 preset_adjusting_time (data)**

4651 Presets the time to a new value (preset_time) and defines a validity_interval within which the
 4652 new time can be activated.

4653 data::= structure
 4654 {
 4655 preset_time: octet-string,
 4656 validity_interval_start: octet-string,
 4657 validity_interval_end: octet-string
 4658 }
 4659 all octet-strings formatted as specified in 4.6.1 for *date-time*

4660 **4.5.1.3.6 shift_time (data)**

4661 Shifts the time by n ($-900 \leq n \leq 900$) s.

4662 $\text{data} ::= \text{long}$

4663

4664 **4.5.2 Script table (class_id = 9, version = 0)**4665 **4.5.2.1 Overview**

4666 This IC allows modelling the triggering of a series of actions by executing scripts using the
4667 *execute* (data) method.

4668 “Script table” objects contain a table of script entries. Each entry consists of a script identifier
4669 and a series of action specifications. An action specification activates a method or modifies an
4670 attribute of a COSEM object within the logical device.

4671 A certain script may be activated by other COSEM objects within the same logical device or
4672 from the outside.

4673 If two scripts have to be executed at the same time instance, then the one with the smaller
4674 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				

4675

4676 **4.5.2.2 Attribute description**4677 **4.5.2.2.1 logical_name**

4678 Identifies the “Script table” object instance. See 6.2.7.

4679 **4.5.2.2.2 scripts**

4680 Specifies the different scripts, i.e. the lists of actions.

4681 array script

4682
4683 $\text{script} ::= \text{structure}$
4684 {
4685 script_identifier: long-unsigned,
4686 actions: array action_specification
4687 }

4688
4689 The script_identifier 0 is reserved. If specified with an *execute* method, it
4690 results in a null script (no actions to perform).

4691
4692 $\text{action_specification} ::= \text{structure}$
4693 {

```

4694         service_id:          enum,
4695         class_id:           long-unsigned,
4696         logical_name:       octet-string,
4697         index:              integer,
4698         parameter:         service specific
4699     }

```

4700 Where:

- 4701 – the service_id element defines which action to be applied to the referenced object:
 - 4702 (1) write attribute,
 - 4703 (2) execute specific method
- 4704 – the index element defines (with service_id 1) which attribute of the selected object is
 4705 affected; or (with service_id 2) which specific method is to be executed. The first attribute
 4706 (logical_name) has index 1, the first specific method has index 1 as well.

4707 NOTE 1 The action_specification is limited to activate methods that do not produce any response (from the server
 4708 to the client).

4709 NOTE 2 A “dummy” action specification with all elements 0 means that the action is not configured.

4.5.2.3 Method description

4.5.2.3.1 execute (data)

4712 Executes the script specified in parameter data.

4713 data::= long-unsigned

4714 If data matches one of the script_identifiers in the script table, then the corresponding
 4715 action_specification is executed.

4716

4.5.3 Schedule (class_id = 10, version = 0)

4.5.3.1 Overview

4719 This IC, together with the IC “Special days”, allows modelling time- and date-driven activities
 4720 within a device. Table 29 and Table 30 provide an overview and show the interactions between
 4721 the two ICs.

4722

Table 29 – 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

4723

4724

Table 30 – Special days table

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

4725

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				
2. insert (data)	o				
3. delete (data)	o				

4726

4.5.3.2 Attribute description

4.5.3.2.1 logical_name

Identifies the “Schedule” object instance. See 6.2.9

4.5.3.2.2 entries

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

```

4733           array      schedule_table_entry
4734           schedule_table_entry::= structure
4735           {
4736             index:          long-unsigned,
4737             enable:         boolean,
4738             script_logical_name: octet-string,
4739             script_selector:   long-unsigned,
4740             switch_time:     octet-string,
4741             validity_window: long-unsigned,
4742             exec_weekdays:   bit-string,
4743             exec_specdays:   bit-string,
4744             begin_date:      octet-string,
4745             end_date:        octet-string
4746           }

```

Where:

- 4748 – script_logical_name: defines the logical name of the “Script table” object;
- 4749 – script_selector: defines the script_identifier of the script to be executed;
- 4750 – 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;
- 4752 – 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;
- 4755 – exec_weekdays defines the days of the week on which the entry is valid;

4756 – exec_specdays perform the link to the IC “Special days table”, day_id;
 4757 begin_date and end_date define the date period in which the entry is valid (wildcards are
 4758 allowed). The format follows the rules set in 4.1.6.1 for *date*.

4759 **4.5.3.3 Method description**

4760 **4.5.3.3.1 enable/disable (data)**

4761 Sets the disabled bit of range A entries to true and then enables the entries of range B.

4762 data ::= structure
 4763 {
 4764 firstIndexA: long-unsigned,
 4765 lastIndexA: long-unsigned,
 4766 firstIndexB: long-unsigned,
 4767 lastIndexB: long-unsigned
 4768 }
 4769

4770 Where:

4771 – firstIndexA first index of the range that is disabled,
 4772 – lastIndexA last index of the range that is disabled,
 4773 – firstIndexB first index of the range that is enabled,
 4774 – lastIndexB last index of the range that is enabled,
 4775 – firstIndexA/B < lastIndexA/B: all entries of the range A/B are disabled / enabled,
 4776 – firstIndexA/B = lastIndexA/B: one entry is disabled/enabled,
 4777 – firstIndexA/B > lastIndexA/B: nothing disabled/enabled,
 4778 – firstIndexA/B and lastIndexA/B > 9999: no entry is disabled/enabled

4780

4781 **4.5.3.3.2 insert (data)**

4782 Inserts a new entry in the table. If the index of the entry exists already, the existing entry is
 4783 overwritten by the new entry.

4784 entry: schedule_table_entry
 4785 data ::= corresponding to entry

4786 **4.5.3.3.3 delete (data)**

4787 Deletes a range of entries in the table.

4788 data ::= structure
 4789 {
 4790 firstIndex: long-unsigned,
 4791 lastIndex: long-unsigned
 4792 }
 4793

4794 Where:

4795 firstIndex: first index of the range that is deleted,
 4796 lastIndex: last index of the range that is deleted,

4797 firstIndex < lastIndex: all entries of the range A/B are deleted,

4798 firstIndex = lastIndex: one entry is deleted,

4799 firstIndex > lastIndex: nothing deleted

4800 Remarks concerning “inconsistencies” in the table entries:

4801 If the same script should be executed several times at a specific time instance, then it is executed only once.

4802 If different scripts should be executed at the same time instance, then the execution order is according to the “index”.
4803 The script with the lowest “index” is executed first.

4804

4805 **4.5.3.4 Recovery after power failure**

4806 After a power failure, the whole schedule is processed to execute all the necessary scripts that
4807 would get lost during a power failure. For this, the entries that were not executed during the
4808 power failure have to be detected. Depending on the validity window they are executed in the
4809 correct order (as they would have been executed in normal operation).

4810 **4.5.3.5 Handling of time changes**

4811 There are four different “actions” of time changes:

- 4812 a) time setting forward (by writing to the attribute *time* of the “Clock” object);
- 4813 b) time setting backward (by writing to the attribute *time* of the “Clock” object);
- 4814 c) time synchronization (using the method *adjust_to_quarter* of the “Clock” object);
- 4815 d) daylight saving action.

4816 All these four actions need a different handling executed by the schedule in interaction with the
4817 time setting activity.

4818 **4.5.3.6 Time setting forward**

4819 This is handled the same way as a power failure. All entries missed are executed depending on
4820 the validity_window. A (manufacturer specific defined) short time setting can be handled like
4821 time synchronization.

4822 **4.5.3.7 Time setting backward**

4823 This results in a repetition of those entries that are activated during the repeated time. A
4824 (manufacturer specific defined) short time setting can be handled like time synchronization.

4825 **4.5.3.8 Time synchronization**

4826 Time synchronization is used to correct small deviations between a master clock and the local
4827 clock. The algorithm is manufacturer specific. It shall guarantee that no entry of the schedule
4828 gets lost, or is executed twice. The validity_window attribute has no effect, because all entries
4829 have to be executed in normal operation.

4830 **4.5.3.9 Daylight saving**

4831 If the clock is put forward, then all scripts, which fall into the forwarding interval (and would
4832 therefore get lost) are executed. If the clock is put back, re-execution of the scripts, which fall
4833 into the backwarding interval, is suppressed.

4834 **4.5.4 Special days table (class_id = 11, version = 0)**4835 **4.5.4.1 Overview**

4836 This IC allows defining special dates. On such dates, a special switching behaviour overrides
 4837 the normal one. The IC works in conjunction with the class "Schedule" or "Activity calendar".
 4838 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				
2. delete (data)	o				

4839

4840 **4.5.4.2 Attribute description**4841 **4.5.4.2.1 logical_name**

4842 Identifies the "Special days table" object instance. See 6.2.8.

4843 **4.5.4.2.2 entries**

4844 Specifies a special day identifier for a given date. The date may have wildcards for repeating
 4845 special days like Christmas.

```
4846           array      spec_day_entry
4847           spec_day_entry ::= structure
4848           {
4849             index:          long-unsigned,
4850             specialday_date: octet-string,
4851             day_id:         unsigned
4852           }
```

4853 Where:

- 4854 – specialday_date formatting follows the rules set in 4.6.1 for *date*;
- 4855 – the range of the day_id shall match the length of the bit-string exec_specdays in the related
 4856 object of IC "Schedule".

4857 **4.5.4.3 Method description**4858 **4.5.4.3.1 insert (data)**

4859 Inserts a new entry in the table.

```
4860           entry ::= spec_day_entry
```

4861 If a special day with the same index or with the same date as an already defined day is inserted,
 4862 the old entry will be overwritten.

4863 **4.5.4.3.2 delete (data)**

4864 Deletes an entry in the table.

4865 index Index of the entry that shall be deleted.

4866 data::= long-unsigned

4867

4868

4869

4870 **4.5.5 Activity calendar (class_id = 20, version = 0)**

4871 **4.5.5.1 Overview**

4872 This IC allows modelling the handling of various tariff structures in the meter. The IC provides
 4873 a list of scheduled actions, following the classical way of calendar based schedules by defining
 4874 seasons, weeks, etc.

4875 An “Activity calendar” object may coexist with the more general “Schedule” object and it can
 4876 even overlap with it. If actions in a “Schedule” object are scheduled for the same activation time
 4877 as in an “Activity calendar” object, the actions triggered by the “Schedule” object are executed
 4878 first.

4879 After a power failure, only the “last action” missed from the object “Activity calendar” is executed
 4880 (delayed). This is to ensure proper tariffication after power up. If a “Schedule” object is present,
 4881 then the missed “last action” of the “Activity calendar” shall be executed at the correct time
 4882 within the sequence of actions requested by the “Schedule” object.

4883 The “Activity calendar” object defines the activation of certain scripts, which can perform
 4884 different activities inside the logical device. The interface to the IC “Script table” is the same as
 4885 for the IC “Schedule” (see 4.5.3).

4886 If an instance of the IC “Special days table” (see 4.5.4) is available, relevant entries there take
 4887 precedence over the “Activity calendar” object driven selection of a day profile. The day profile
 4888 referenced in the “Special days table” activates the day_schedule of the *day_profile_table* in
 4889 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				

4890

4891 **4.5.5.2 Attribute description**

4892 NOTE: Attributes called ..._active are currently active. Attributes called ..._passive will be activated by the specific
 4893 method activate_passive_calendar.

4894 **4.5.5.2.1 logical_name**

4895 Identifies the “Activity calendar” object instance. See 6.2.10.

4896 **4.5.5.2.2 calendar_name**

4897 Typically contains an identifier, which is descriptive to the set of scripts activated by the object.

4898 **4.5.5.2.3 season_profile**

4899 Contains a list of seasons defined by their starting date and a specific week_profile to be
 4900 executed. The list is sorted according to season_start (in increasing order);

```
4901                  array         season
4902
4903                  season ::= structure
4904                  {
4905                  season_profile_name:     octet-string,
4906                  season_start:            octet-string,
4907                  week_name:              octet-string
4908                  }
```

4909 Where:

- 4910 – season_profile_name is a user defined name identifying the current season_profile;
- 4911 – season_start defines the starting time of the season, formatted as specified in 4.6.1 for date-time. 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;

4915 NOTE The current season is terminated by the season_start of the next season.

- 4916 – week_name defines the week_profile active in this season.

4917

4918 **4.5.5.2.4 week_profile_table**

4919 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.

```
4921                  array         week_profile
4922
4923                  week_profile ::= structure
4924                  {
4925                  week_profile_name:     octet-string,
4926                  monday:                day_id,
4927                  tuesday:              day_id,
4928                  wednesday:            day_id,
4929                  thursday:             day_id,
4930                  friday:               day_id,
4931                  saturday:            day_id,
4932                  sunday:              day_id
4933                  }
4934
4935                  day_id: unsigned
```

4936 Where:

- 4937 – week_profile_name is a user defined name identifying the current week_profile;
- 4938 – Monday defines the day_profile valid on Monday;
- 4939 – ...

4940 Sunday defines the day_profile valid on Sunday.

4941 **4.5.5.2.5 day_profile_table**

4942 Contains an array of day_profiles, identified by their day_id. For each day_profile, a list of
 4943 scheduled actions is defined by a script to be executed and the corresponding activation time
 4944 (start_time). The list is sorted according to start_time.

```

4945           array      day_profile
4946
4947           day_profile ::= structure
4948           {
4949             day_id:          unsigned,
4950             day_schedule:    array day_profile_action
4951           }
4952
4953           day_profile_action ::= structure
4954           {
4955             start_time:       octet-string,
4956             script_logical_name: octet-string,
4957             script_selector:   long-unsigned
4958           }
4959

```

4960 Where:

- 4961 – day_id is a user defined identifier, identifying the current day_profile;
- 4962 – start_time: defines the time when the script is to be executed (no wildcards); the format
 4963 follows the rules set in 4.1.6.1 for *time*;
- 4964 – script_logical_name: defines the *logical_name* of the “Script table” object;
- 4965 – script_selector: defines the script_identifier of the script to be executed.

4966

4967 **4.5.5.2.6 activate_passive_calendar_time**

4968 Defines the time when the object itself calls the specific method *activate_passive_calendar*. A
 4969 definition with "not specified" notation in all fields of the attribute will deactivate this automatism.
 4970 Partial "not specified" notation in just some fields of date and time are not allowed.

4971 octet-string, formatted as specified in 4.6.1 for *date-time*.

4972 **4.5.5.3 Method**

4973 **4.5.5.3.1 activate_passive_calendar (data)**

4974 This method copies all attributes called ..._passive to the corresponding attributes
 4975 called ..._active.

4976 data ::= integer (0)

4977

4978 4.5.6 Register monitor (class_id = 21, version = 0)

4979 4.5.6.1 Overview

4980 This IC allows modelling the function of monitoring of values modelled by “Data”, “Register”,
4981 “Extended register” or “Demand register” objects. It allows specifying thresholds, the value
4982 monitored, and a set of scripts (see 4.5.2) that are executed when the value monitored crosses
4983 a threshold.

4984 The IC "Register monitor" requires an instantiation of the IC "Script table" in the same logical
4985 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				

4986

4.5.6.2 Attribute description

4.5.6.2.1 logical_name

4989 Identifies the “Register monitor” object instance. See 6.2.13 and 6.3.10.

4.5.6.2.2 thresholds

4991 Provides the threshold values to which the attribute of the referenced register is compared.

4992 array threshold

4993 threshold: The threshold is of the same type as the monitored attribute of the
4994 referenced object.

4.5.6.2.3 monitored_value

4996 Defines which attribute of an object is to be monitored. Only values with simple data types are
4997 allowed.

```
4998           value_definition ::= structure
4999           {
5000             class_id:          long-unsigned,
5001             logical_name:      octet-string,
5002             attribute_index:   integer
5003           }
```

5005 4.5.6.2.4 actions

5006 Defines the scripts to be executed when the monitored attribute of the referenced object crosses
5007 the corresponding threshold. The attribute actions has exactly the same number of elements as
5008 the attribute thresholds. The ordering of the action_items correspond to the ordering of the
5009 thresholds (see 4.5.6.2.2).

5010 array action set

```

5011      action_set ::= structure
5012      {
5013          action_up:    action_item,
5014          action_down:   action_item
5015      }

```

5016 Where:

- 5017 – action_up defines the action when the attribute value of the monitored register crosses the threshold in the upwards direction;
- 5019 – action_down defines the action when the attribute value of the monitored register crosses the threshold in the downwards direction.

```

5021      action_item ::= structure
5022      {
5023          script_logical_name: octet-string,
5024          script_selector:    long-unsigned
5025      }

```

5026

5027 **4.5.7 Single action schedule (class_id = 22, version = 0)**

5028 **4.5.7.1 Overview**

5029 This IC allows modelling the execution of periodic actions within a meter. Such actions are not necessarily linked to tariffication (see “Activity calendar” (4.5.5) or “Schedule” (4.5.3)).

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				

5031

5032 **4.5.7.2 Attribute description**

5033 **4.5.7.2.1 logical_name**

5034 Identifies the “Single action schedule” object instance. See 6.2.12.

5035 **4.5.7.2.2 executed_script**

5036 Contains the logical name of the “Script table” object and the script selector of the script to be executed.

```

5038      script ::= structure
5039      {
5040          script_logical_name: octet-string,
5041          script_selector:    long-unsigned
5042      }

```

5043 Script_logical_name and script_selector define the script to be executed.

5044 **4.5.7.2.3 type**

5045 enum:

5046
5047 (1) size of execution_time = 1; wildcard in date allowed,
5048 (2) size of execution_time = n; all time values are the same, wildcards in date not allowed,
5049 (3) size of execution_time = n; all time values are the same, wildcards in date are allowed,
5050 (4) size of execution_time = n; time values may be different, wildcards in date not allowed,
5051 (5) size of execution_time = n; time values may be different, wildcards in date are allowed
5052

5053 **4.5.7.2.4 execution_time**

5054 Specifies the time and the date when the script is executed.

5055 array execution_time_date
5056
5057 execution_time_date::= structure
5058 {
5059 time: octet-string,
5060 date: octet-string
5061 }

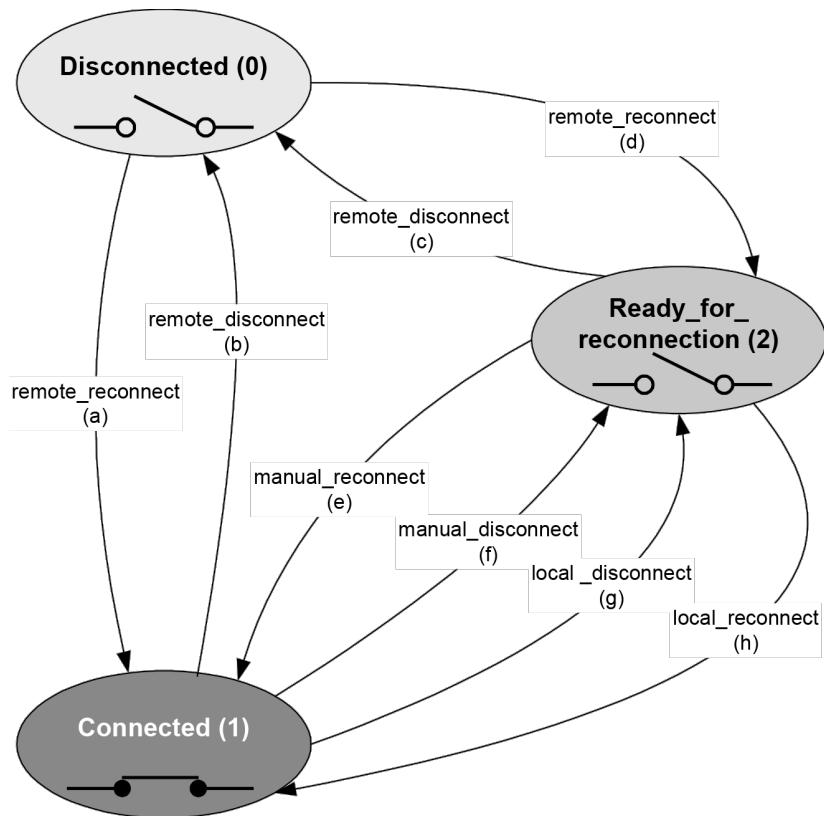
5062 The two octet-strings contain *time* and *date*, in this order; *time* and *date* are formatted as
5063 specified in 4.1.6.1. Hundredths of seconds shall be zero.

5064 **4.5.8 Disconnect control (class_id = 70, version = 0)**

5065 **4.5.8.1 Overview**

5066 Instances of the “Disconnect control” IC manage an internal or external disconnect unit of the
5067 meter (e.g. electricity breaker, gas valve) in order to connect or disconnect – partly or entirely
5068 – the premises of the consumer to / from the supply. The state diagram and the possible state
5069 transitions are shown in Figure 18.

5070



5071

IEC

5072

Figure 18 – State diagram of the Disconnect control IC

5073 Disconnect and reconnect can be requested:

- 5074 • Remotely, via a communication channel: `remote_disconnect`, `remote_reconnect`;
- 5075 • Manually, using e.g. a push button: `manual_disconnect`, `manual_reconnect`;
- 5076 • Locally, by a function of the meter, e.g. limiter, prepayment: `local_disconnect`,
- 5077 `local_reconnect`.

5078 The states and state transitions of the Disconnect control IC are shown in Table 31. The
 5079 possible state transitions depend on the control mode. The Disconnect control object doesn't
 5080 feature a memory, i.e. any commands are executed immediately.

5081 To define the behaviour of the disconnect control object for each trigger, the control mode shall be set.

5082

Table 31 – Disconnect control IC – states and state transitions

States		
State number	State name	State description
0	Disconnected	The <i>output_state</i> is set to FALSE and the consumer is disconnected.
1	Connected	The <i>output_state</i> is set to TRUE and the consumer is connected.
2	Ready_for_reconnection	The <i>output_state</i> 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.

5083

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 (data)	m				
2. remote_reconnect (data)	m				

5084

5085 **4.5.8.2 Attribute description**5086 **4.5.8.2.1 logical_name**

5087 Identifies the “Disconnect control” object instance. See 6.2.22 and 6.2.46.

5088 **4.5.8.2.2 output_state**

5089 Shows the actual physical state of the device connection the supply.

5090 boolean: TRUE = (BOOLEAN 1) Connected,
 5091 FALSE = (BOOLEAN 0) Disconnected

5092 NOTE 1 In electricity metering, the supply is connected when the disconnector device is closed.

5093 NOTE 2 In gas and water metering, the supply is connected when the valve is open.

5094 **4.5.8.2.3 control_state**

5095 Shows the internal state of the disconnect control object.

5096 enum: (0) Disconnected,
 5097 (1) Connected,
 5098 (2) Ready_for_reconnection
 5099

5100 **4.5.8.2.4 control_mode**

5101 Configures the behaviour of the disconnect control object for all triggers, i.e. the possible state
 5102 **transitions**.

control_mode	Disconnection			Reconnection		
	Remote	Manual	Local	Remote	Manual	Local
enum:	(b)	(c)	(f)	(g)	(a)	(d)
(0)	–	–	–	–	–	–
(1)	x	x	x	x	–	x
(2)	x	x	x	x	–	x
(3)	x	x	–	x	–	x
(4)	x	x	–	x	x	–
(5)	x	x	x	x	–	x
(6)	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.

5103

5104 **4.5.8.3 Method description**5105 **4.5.8.3.1 remote_disconnect (data)**

5106 Forces the “Disconnect control” object into ‘disconnected’ state if remote disconnection is
 5107 enabled (control mode > 0).

5108 data::= integer (0)

5109 **4.5.8.3.2 remote_reconnect (data)**

5110 Forces the “Disconnect control” object into the ‘ready_for_reconnection’ state if a direct remote
 5111 reconnection is disabled (control_mode = 1, 3, 5, 6).

5112 Forces the “Disconnect control” object into the ‘connected’ state if a direct remote reconnection
 5113 is enabled (control_mode = 2, 4).

5114 data::= integer (0)

5115

5116 **4.5.9 Limiter (class_id = 71, version = 0)**5117 **4.5.9.1 Overview**

5118 Instances of the “Limiter” IC allow defining a set of actions that are executed when the value of
 5119 a *value* attribute of a monitored object “Data”, “Register”, “Extended Register”, “Demand
 5120 Register”, etc. crosses the threshold value for at least minimal duration time.

5121 The threshold value can be normal or emergency threshold. The *emergency threshold* is
 5122 activated via the *emergency_profile* defined by *emergency profile id*, *emergency activation time*,
 5123 and *emergency duration*. The *emergency profile id* element is matched to an *emergency profile*
 5124 *group id*: this mechanism enables the activation of the emergency threshold only for a specific
 5125 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				

5126

5127 **4.5.9.2 Attribute description**5128 **4.5.9.2.1 logical_name**

5129 Identifies the “Limiter” object instance. See 6.2.15.

5130 **4.5.9.2.2 monitored_value**

5131 Defines an attribute of an object to be monitored. Only attributes with simple data types are
 5132 allowed.

```

5133           value_definition ::= structure
5134           {
5135             class_id:      long-unsigned,
5136             logical_name: octet-string,
5137             attribute_index: integer
5138           }
5139

```

5140 **4.5.9.2.3 threshold_active**

5141 Provides the active threshold value to which the attribute monitored is compared.

5142 threshold: The threshold is of the same type as the attribute monitored

5143 **4.5.9.2.4 threshold_normal**

5144 Provides the threshold value to which the attribute monitored is compared when in normal
5145 operation.

5146 threshold: see 4.5.9.2.3

5147 **4.5.9.2.5 threshold_emergency**

5148 Provides the threshold value to which the attribute monitored is compared when an emergency
5149 profile is active.

5150 threshold: see 4.5.9.2.3

5151 **4.5.9.2.6 min_over_threshold_duration**

5152 Defines minimal over threshold duration in seconds required to execute the over threshold
5153 action.

5154 **4.5.9.2.7 min_under_threshold_duration**

5155 Defines minimal under threshold duration in seconds required to execute the under threshold
5156 action.

5157 **4.5.9.2.8 emergency_profile**

5158 An *emergency_profile* is defined by three elements: *emergency_profile_id*,
5159 *emergency_activation_time*, *emergency_duration*.

5160 An emergency profile is activated if the *emergency_profile_id* element matches one of the
5161 elements on the *emergency_profile_group_id_list*, and time matches the
5162 *emergency_activation_time* and *emergency_duration* element:

```

5163           emergency_profile ::= structure
5164           {
5165             emergency_profile_id:      long-unsigned,
5166             emergency_activation_time: octet-string,
5167             emergency_duration:       double-long-unsigned
5168           }

```

5169 Where:

5170 – the *emergency_activation_time* element defines the date and time when the
5171 *emergency_profile* activated. The octet-string is encoded as specified in 4.1.6.1 for *date-time*,
5172

5173 – the `emergency_duration` element defines the duration in seconds, for which the
 5174 `emergency_profile` is activated.

5175 When an emergency profile is active, the `emergency_profile_active` attribute is set to TRUE.
 5176
 5177

5178 **4.5.9.2.9 `emergency_profile_group_id_list`**

5179 Defines a list of group id-s of the emergency profile.

5180 The emergency profile can be activated only if `emergency_profile_id` element of the
 5181 `emergency_profile` attribute matches one of the elements on the
 5182 `emergency_profile_group_id_list`:

```
5183           array      emergency_profile_group_id  

  5184           emergency_profile_group_id::= long-unsigned  

  5185  

  5186
```

5187 **4.5.9.2.10 `emergency_profile_active`**

5188 Indicates that the `emergency_profile` is active.

5189 **4.5.9.2.11 `actions`**

5190 Defines the scripts to be executed when the monitored value crosses the threshold for minimal
 5191 duration time.

```
5192           action ::= structure  

  5193           {  

  5194             action_over_threshold:    action_item,  

  5195             action_under_threshold:   action_item  

  5196           }
```

5197 Where:

- 5198 – `action_over_threshold` defines the action when the value of the attribute monitored crosses
 5199 the threshold in upwards direction and remains over threshold for minimal over threshold
 5200 duration time;
- 5201 – `action_under_threshold` defines the action when the value of the attribute monitored crosses
 5202 the threshold in the downwards direction and remains under threshold for minimal under
 5203 threshold duration time.

```
5204  

  5205           action_item ::= structure  

  5206           {  

  5207             script_logical_name: octet-string,  

  5208             script_selector:     long-unsigned  

  5209           }
```

5210
 5211

5212
 5213 **4.5.10 Parameter monitor (class_id = 65, version = 1)**

5214 Instances of the “Parameter monitor” IC are used to monitor a list of COSEM object attributes
 5215 that hold parameters.

5216 If the value of an attribute in the `parameter_list` changes, the identifier and the value of that
 5217 attribute is automatically captured to the `changed_parameter` attribute. The time when the
 5218 change of the parameter occurred is captured in the `capture_time` attribute. These attributes

5219 may be captured by a “Profile generic” object. In this way, a log of all parameter changes can
5220 be built. For the OBIS code of the Parameter monitor log objects, see IEC 62056-6-1:2021, 6.5.

5221 NOTE 1 In the case of simultaneous or quasi simultaneous parameter changes the order of capturing and
5222 logging the changed parameters has to be managed by the application.

5223 Several Parameter monitor objects and corresponding Profile generic objects can be
5224 instantiated to manage a number of parameter groups. The link between the Parameter monitor
5225 object and the corresponding Profile generic object is via the *capture_object* attribute of the
5226 Profile generic object.

5227 NOTE 2 As the various parameters may be of different type and length, the entries in the profile column holding
5228 the parameters will be also of different type and length. This can be managed by capturing different kind of
5229 parameters into different Parameter list Profile generic objects and parameter logs.

5230 NOTE 3 The Profile generic object holding the parameter change log may capture other suitable object
5231 attributes, such as the *time* attribute of the Clock object and any other relevant values.

5232

5233 Use of the Parameter monitor IC for detecting configuration parameter changes

5234 Each client that has access to the server has to be aware of the current parametrisation in order
5235 to be able to correctly exchange data with servers. The parameters may be grouped and each
5236 group may have a name.

5237 Although the parametrisation may be known initially, it may change during the lifetime of the
5238 meter, for example it may be changed by another client such as a field service device.

5239 The current configuration may always be retrieved by reading the configuration parameters at
5240 the beginning of the exchange. This is not efficient and in the case of push operation it is not
5241 practical.

5242 A solution is needed that allows clients to verify if the configuration of the server is as expected
5243 or to detect if any change has occurred.

5244 To achieve this, version 1 of the Parameter monitor IC specifies new attributes to hold:

- 5245 • the *parameter_list_name*;
- 5246 • the *hash_algorithm_id*;
- 5247 • the *parameter_value_digest*; and
- 5248 • the *parameter values*.

5249 When a client configures the server, it can retrieve these attributes and share them with other
5250 clients.

5251 When a client starts a data exchange with the server, it can read the *parameter_value_digest*
5252 attribute and compare it to the value it has stored. If they match, the configuration is as expected,
5253 if not, the client can read the *parameter_values* attribute to identify which one(s) have changed.

5254 The attributes can also be captured into Profile generic object buffer attributes to assure the
5255 validity of the data captured. The resulting overhead can be minimized by the use of null-data
5256 encoding.

Parameter monitor	0...n	class_id = 65, version = 1			
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
5. parameter_list_name	octet-string				x + 0x20
6. hash_algorithm_id	enum				x + 0x28
7. parameter_value_digest	octet-string				x + 0x30
8. parameter_values	structure				x + 0x38
Specific methods	m/o				x + 0x40
1. add_parameter (data)	o				x + 0x48
2. delete_parameter (data)	o				x + 0x50

5257

5258 **4.5.10.1 Attribute description**5259 **4.5.10.1.1 logical_name**

5260 Identifies the “Parameter monitor” object instance. See 6.2.14.

5261 **4.5.10.1.2 changed_parameter**

5262 Holds the identifier and the value of the most recently changed parameter.

```

5263           structure
5264           {
5265             class_id:     long-unsigned,
5266             logical_name: octet-string,
5267             attribute_index: integer,
5268             attribute_value: CHOICE
5269             -- CHOICE as specified in the “Data” interface class
5270           }
5271

```

5272 **4.5.10.1.3 capture_time**5273 Provides data and time information showing when the value of the *changed_parameter* attribute
5274 has been captured.5275 *date-time* is formatted as specified in 4.1.6.1.5276 **4.5.10.1.4 parameter_list**

5277 Holds the list of parameters managed by a given instance of the “Parameter monitor” IC.

```

5278           parameter_list::= array parameter_list_element
5279
5280           parameter_list_element::= structure
5281           {
5282             class_id:     long-unsigned,
5283             logical_name: octet-string,
5284             attribute_index: integer
5285           }

```

5286 NOTE 4 The list of parameters monitored may be changed by using the *add_parameter* or *delete_parameter*
5287 methods or writing this attribute.

5288 NOTE 5 The list of parameters monitored may be changed by using the *add_parameter* or *delete_parameter* methods
5289 or writing this attribute.

5290 **4.5.10.1.5 parameter_list_name**

5291 Holds the name given to the list of parameter as defined by the *parameter_list* attribute.

5292 **4.5.10.1.6 hash_algorithm_id**

5293 enum

- 5295 (0) SHA-256,
5296 (1) SHA-384,
5297 (2) Last 16 bytes of SHA-256,
5298 (3) Last 8 bytes of SHA-256,
5299 (4) Last 4 bytes of SHA-256

5300

5301 **4.5.10.1.7 parameter_value_digest**

5302 Result of the digest calculation according to the algorithm specified by the *hash_algorithm_id*
5303 attribute. The digest is calculated on all the parameter list values as defined by attribute
5304 *parameter_list* in the order they are placed in the array. To calculate the digest, the whole set
5305 of *parameter_list* attribute values is first transformed into an octet-string. Rules for this
5306 transformation are defined IEC 62056-5-3:2021, 5.3.4.3. Data conversion, for all the data types
5307 which are not an octet-string.

5308 **4.5.10.1.8 parameter_values**

5309 Holds a copy of the values of the attributes referenced in *parameter_list* attribute.

5310 It is a structure that holds the A-XDR encoded value of each attribute in the order of the
5311 *parameter_list* array.

5312 **4.5.10.2 Method description**

5313 **4.5.10.2.1 add_parameter (data)**

5314 Adds one parameter to the *parameter_list*.

5315 data::= parameter_list_element

5316 NOTE 6 A parameter can be logged as soon as it is added to the list. Adding an element to the list does not affect
5317 the *buffer* of the “Profile generic” object capturing the *changed_parameter* attribute.

5318 **4.5.10.2.2 delete_parameter (data)**

5319 Deletes one parameter from the *parameter_list*.

5320 data::= parameter_list_element

5321 NOTE 7 When a parameter is deleted from the parameter list, its changes will not be logged any more. Removing
5322 an element from the list does not affect the *buffer* of the “Profile generic” object capturing the *changed_parameter*
5323 attribute.

5324

5325

5326 **4.5.11 Sensor manager (class_id = 67, version = 0)**5327 **4.5.11.1 General**

5328 NOTE This interface class is used mainly in metering energy types other than electricity.

5329 Most measuring instruments under the scope of the MID operate with dedicated sensors
5330 (transducers and transmitters) connected to the processing unit. These sensors have to be
5331 permanently supervised concerning their functioning and limits to fulfil the metrological
5332 requirements for subsequent calculation of monetary values.

5333 In addition, the measured values have to be monitored. These values may be related to a
5334 physical quantity – raw values of voltage, current, resistance, frequency, digital output –
5335 provided by the sensor, and the measured quantities resulting from the processing of the
5336 information provided by the sensor.

5337 It is necessary to monitor and often to log the relevant values in order to obtain diagnostic
5338 information that allows:

- 5339 • the identification of the sensor device;
- 5340 • the connection and the sealing status of the sensor;
- 5341 • the configuration of the sensors;
- 5342 • the monitoring of the operation of the sensors;
- 5343 • the monitoring of the result of the processing.

5344 The “Sensor manager” interface class allows managing detailed information related to a sensor
5345 by a single object.

5346 For simpler sensors / devices, already existing COSEM objects – identifying the sensors,
5347 holding measurement values and monitoring those measurement values – can be used.

5348 **4.5.11.2 Overview**

5349 Instances of the “Sensor manager” IC manage complex information related to sensors. They
5350 also allow monitoring the raw data and the processed value, derived by processing the raw-
5351 data using appropriate algorithms as required by the particular application. This IC includes a
5352 number of functions:

- 5353 • nameplate data of the sensor and site information (attributes 2 to 6);
- 5354 • an “Extended register” function for the *raw-value* (attributes 7 to 10);
- 5355 • a “Register monitor” function for the *raw-value* (attributes 11-12);

5356 NOTE 1 Not every raw data (e.g. the voltage output of a pressure sensor) has its own OBIS code / object. This
5357 is the reason to include raw data in the Sensor manager class.

- 5358 • a “Register monitor” function for the *processed_value* (attributes 13 to 15).

5359 NOTE 2 Not all “modules” are necessarily present. The attributes not used are possibly not implemented or
5360 not accessible.

5361

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

5362

5363 4.5.11.3 Attribute description

5364 4.5.11.3.1 logical_name

5365 Identifies the “Sensor manager” object instance.

5366 NOTE 3 Sensor manager objects are used in relation to non-electricity metering. Therefore no OBIS codes are defined in this document.

5368 4.5.11.3.2 serial_number

5369 Identifies the sensor (->site information).

5370 NOTE 4 For simple sensors, the serial number may be held by “Data” objects with appropriate OBIS codes if this is the only information required.

5372 4.5.11.3.3 metrological_identification

5373 Describes metrologically relevant information of the sensor, e.g. metrological identifier, calibration date.

5375 4.5.11.3.4 output_type

5376 describes the physical output of the sensor (->site information)

5377	enum:	(0)	not specified,
5378		(1)	4–20 mA,
5379		(2)	0–20 mA
5380		(3)	0–5 V,
5381		(4)	0–10 V,
5382		(5)	Pt100,
5383		(6)	Pt500,
5384		(7)	Pt1000,
5385		(8)	HART
5386		(128)	manufacturer specific

5387 All other values are reserved.

5388 4.5.11.3.5 adjustment_method

5389 Describes the sensor adjustment method, e.g. by 3-Measuring-point-equation.

5390 4.5.11.3.6 sealing_method

5391 Type of seals applied to the sensor.

5392 enum: (0) none,
5393 (1) mechanical (e.g. wire or protective sticker);
5394 (2) electronic (e.g. contact);
5395 (3) software (e.g. password protection)

5396 4.5.11.3.7 raw_value

5397 Physical value from the sensor (e.g. voltage from a pressure to voltage converter).

5398 For the possible data type choices, see the “Register” IC in 4.3.3.2.2.

5399 4.5.11.3.8 scaler_unit

5400 The scaler and the unit of the *raw_value*.

5401 For the definition of *scaler_unit*, see the “Register” IC in 4.3.3.2.3.

4.5.11.3.9 status

5403 Status of last *raw_value* captured (for the definition of status, see the “Extended register” IC,
5404 4.3.3.2.4). The status keeps information such as:

- 5405 – sensor failure,
 - 5406 – sensor activated / inactivated.

5407 The meaning of the elements of the *status* shall be provided for each “Sensor manager” object
5408 instance.

4.5.11.3.10 capture_time

5410 Provides a "Sensor manager" object specific date and time information showing when the value
5411 of the attribute *raw_value* has been captured.

5412 *date_time* is formatted as specified in 4.1.6.1.

5413 For the possible data type choices, see the “Extended register” interface class in 4.3.3.2.5.

5414 4.5.11.3.11 raw_value_thresholds

5415 Provides the threshold values to which the *raw_value* attribute is compared.

5416 array threshold

threshold: The threshold is of the same type as the raw-value (4.5.11.3.7).

5418 4.5.11.3.12 raw_value_actions

5419 Defines the scripts to be executed when the `raw_value` crosses the corresponding threshold.
5420 The attribute `raw_value_actions` has exactly the same number of elements as the attribute

5421 raw_value_thresholds. The ordering of the *action_items* corresponds to the ordering of the
5422 thresholds. For the specification of actions, see the *Register monitor* IC in 4.5.6.2.4.

5423 **4.5.11.3.13 processed_value**

5424 References the attribute holding the processed value of the raw data provided by the sensor.

```
5425                processed_value_definition ::= structure
5426                {
5427                class_id: long-unsigned,
5428                logical_name: octet-string,
5429                attribute_index: integer
5430                }
```

5431 Note that more than one “Sensor manager” objects and processed value objects may belong to
5432 the same sensor.

5433 **4.5.11.3.14 processed_value_thresholds**

5434 Provides the threshold values to which the processed value is compared.

```
5435                array      threshold
```

5436 threshold: The threshold is of the same type as the value processed.
5437 (referenced by the *processed-value* attribute 4.5.11.3.13).

5438 **4.5.11.3.15 processed_value_actions**

5439 Defines the scripts to be executed when the processed value crosses the corresponding
5440 threshold.

5441 The attribute *processed_value_actions* has exactly the same number of elements as the
5442 attribute *processed_value_thresholds*. The ordering of the *action_items* corresponds to the
5443 ordering of the thresholds.

5444 For the specification of actions, see the *Register monitor* IC in 4.5.6.2.4.

5445 **4.5.11.4 Method**

5446 **4.5.11.4.1 reset (data)**

5447 This method resets the *raw_value* to the default value. The default value is an instance specific
5448 constant.

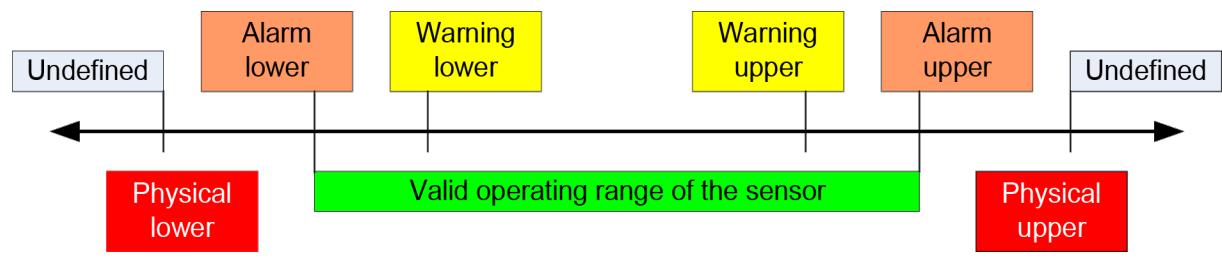
5449 The attribute *capture_time* is set to the time of the reset execution.

```
5450                data ::= integer (0)
```

5451

5452 **4.5.11.5 Example for absolute pressure sensor**

5453 Figure 19 illustrates the definition of relevant upper and lower thresholds.



5454

IEC

5455

Figure 19 – Definition of upper and lower thresholds

5456 Table 32 and Table 33 show examples of the various thresholds and the actions performed
5457 when the thresholds are crossed.

5458

Table 32 – 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

5459

5460

Table 33 – 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

5461

4.5.12 Arbitrator (class_id = 68, version = 0)

4.5.12.1 Overview

Instances of the “Arbitrator” IC allow determining, based on pre-configured rules comprising permissions and weightings, which action is carried out when multiple actors may request potentially conflicting actions to control the same resource. Generally, there is one “Arbitrator” object instantiated for each resource for which competing action requests need to be handled.

The “Arbitrator” IC allows:

- configuring the possible, potentially conflicting actions that can be requested;
- configuring the permissions for each actor to request the possible actions;
- configuring the weighting for each actor for each possible request.

NOTE 1 Examples for a resource are the supply control switch or a gas valve of the meter. Examples for possible actions are disconnect supply, enable reconnection, reconnect supply, prevent disconnection, prevent reconnection.

The actions that can be requested are held in the *actions* attribute as an array of script identifiers. The scripts – held by separate “Script table” objects – allow performing a wide range of functions. There may be actions designed to inhibit the execution of other actions: these are modelled as null-scripts, i.e. pointing to a script_identifier (0) of a “Script table” object.

NOTE 2 The “Arbitrator” objects do not contain the names of the actions, but their purpose and effect can be deduced by looking at the relevant “Script table” objects.

The permissions are held in the *permissions_table* attribute as an array of bit-strings: each element in the array represents one actor and each bit in the bit-string represents one action from the *actions* array.

The weightings are held in the *weightings_table* attribute as a two-dimensional array: each line represents one actor and there is one weight allocated to each possible action for that actor. For actions designed to inhibit other actions a very high weight may be allocated compared to the weight of the action that is to be inhibited.

Actions are requested by invoking the *request_action* method. The method invocation parameters contain:

- the identifier of the actor. This element, an unsigned number, points to a line in the *permissions_table*, *weightings_table* and *most_recent_requests_table* attributes;

- 5491 NOTE 3 Names of the actors may be specified in project specific companion specifications.
- 5492 • the list of actions requested, in the form of a bit-string. Each bit corresponds to one
 5493 element of the *actions* array: for the actions requested the bit is set to 1, for the actions
 5494 not requested (inactions) the bit is set to 0. An actor may request none, one, several or all
 5495 actions in a single request in one invocation. The reason to allow requesting multiple
 5496 actions in a single request is to allow the actor to request an action, and at the same time
 5497 to prevent another actor reversing that action.

5498 NOTE 4 For example, an actor may request disconnecting the supply and preventing another actor to reconnect
 5499 it.

5500 NOTE 5 An earlier action request by an actor can be cleared by not requesting the same action (i.e. by
 5501 requesting an inaction) in another invocation of the *request_action* method by the same actor. With this, a request
 5502 for inhibiting an action is lifted.

5503 The *most_recent_requests_table* attribute holds the list of the most recent request of each actor,
 5504 in the form of an array of bit-strings: each element in the array represents the last request of
 5505 an actor, and each bit in the bit-string represents one action / inaction requested.

5506 When the *request_action* method is invoked by an actor the AP carries out the following
 5507 activities:

- 5508 • it checks the *permissions_table* attribute entry for the given actor to ascertain if the
 5509 actions requested are permitted or not;
- 5510 • it updates the *most_recent_requests_table* attribute by setting or clearing the bit in the bit-
 5511 string for that actor for each action requested that is also permitted (bit is set); or not
 5512 requested / not permitted (bit is cleared);
- 5513 • it applies the *weightings_table* for the *most_recent_requests_table*: for each bit set in the
 5514 *most_recent_requests_table* the corresponding weight of each actor is applied;
- 5515 • for each action, the weights are summed; and then
- 5516 • if there is a unique highest total weight for an action, this value is written to the
 5517 *last_outcome* attribute and the corresponding script is executed. If there is no highest
 5518 unique total weight, nothing happens.

5519

Arbitrator	0...n	class_id = 68, version = 0				
Attribute description	Data type	Min.	Max.	Def.	Short name	
1. logical_name	(static)	octet-string			x	
2. actions	(static)	array			all empty	x + 0x08
3. permissions_table	(static)	array			all bits cleared	x + 0x10
4. weightings_table	(static)	array			all zero	x + 0x18
5. most_recent_requests_table	(dyn.)	array			all bits cleared	x + 0x20
6. last_outcome	(dyn.)	unsigned	0	n	0	x + 0x28
Specific methods	m/o					
1. request_action (data)	m					x + 0x30
2. reset (data)	o					x + 0x38

5520

5521 4.5.12.2 Attribute description

5522 4.5.12.2.1 logical_name

5523 Identifies the “Arbitrator” object instance. See 6.2.47.

5524 **4.5.12.2.2 actions**

5525 Defines the actions that can be requested.

```
5526     actions ::= array action_item
5527         action_item ::= structure
5528             {
5529                 script_logical_name: octet-string,
5530                 script_selector: long-unsigned
5531             }
```

5534 Where:

- 5535 – script_logical_name: defines the *logical_name* of the “Script table” object;
- 5536 – script_selector: defines the script_identifier of the script to be executed.

5537 Entries that are intended to inhibit other actions are represented by null-scripts, i.e. pointing to
5538 script_identifier (0) of a “Script table” object.

5539 **4.5.12.2.3 permissions_table**

5540 Contains the permissions for each actor to request actions.

```
5541     permissions_table ::= array actor_permissions
5542         actor_permissions ::= bit-string
5543
5544
```

5545 Each entry represents the permissions for one actor to request each action.

5546 Each bit in the bit-string corresponds to one action in the *actions* array. The length of the bit-
5547 string shall be the same as the number of elements in the *actions* array.

5548 The leading bit corresponds to the first element and the trailing bit corresponds to the last
5549 element in the *actions* array.

5550 The bits shall be set for actions allowed and cleared for actions not allowed.

5551 NOTE 1 These *actor_permissions* model business rules rather than acting as a form of access control.

5552 **4.5.12.2.4 weightings_table**

5553 Holds the weight allocated for each actor and to each possible action of that actor.

```
5554     weightings_table ::= array actor_weighting_list
5555         actor_weighting_list ::= array actor_action_weight
5556             actor_action_weight ::= long-unsigned
5557
5558
```

5559 The number and order of elements in the *weightings_table* array shall be the same as in the
5560 *permissions_table* array.

5561 The number of elements in the *actor_weighting_list* array shall be the same as in the *actions*
5562 array. The first element shows the weight for the first action and the last element shows the
5563 weight of the last action.

5564 NOTE 2 It is preferred using powers of 2 as weights. Using different weights for different actors and actions helps
 5565 to avoid situations when there is no unique outcome after evaluating the action request (in which case no action is
 5566 performed).

5567 **4.5.12.2.5 most_recent_requests_table**

5568 Holds the most recent requests of each actor.

5569 most_recent_requests_table ::= array most_recent_request
 5570 most_recent_request ::= bit-string
 5571

5572 Each entry represents the most recent request of an actor.

5573 The number and order of elements in the *most_recent_requests_table* array shall be the same
 5574 as in the *permissions_table* array.

5575 The length of the *most_recent_request* bit-string shall be the same as the number of elements
 5576 in the *actions* array. The leading bit corresponds to the first element and the trailing bit
 5577 corresponds to the last element in the *actions* array.

5578 For each action that has been requested and which is also permitted the bit is set. For each
 5579 action that is not requested (inaction) or not allowed the bit is cleared.

5580 **4.5.12.2.6 last_outcome**

5581 Contains the outcome of the most recent request that has resulted a unique highest total weight
 5582 for an action requested and therefore performed.

5583 The number identifies a bit in the *request_action_list* bit-string: the number 1 corresponds to
 5584 the leading bit and consequently to the first element in the *actions* array.

5585 **4.5.12.3 Method description**

5586 **4.5.12.3.1 request_action (data)**

5587 Defines the actions that are requested by an actor.

5588 data ::= structure
 5589 {
 5590 request_actor: unsigned,
 5591 request_action_list: bit-string
 5592 }

5593 Where:

- 5594 – *request_actor* is an index into the corresponding arrays. The number 1 identifies the first
 5595 entry in the *permissions_table*, the first entry of the *weightings_table* and the first entry in
 5596 the *most_recent_requests_table* arrays. The number *n* identifies the highest entry in these
 5597 arrays;
- 5598 – *request_action_list* identifies the action(s) to be performed. For each action requested the
 5599 bit is set. For each action not requested (inaction) the bit is not set. The length of the bit-
 5600 string shall be the same as the number of elements in the *actions* array. The leading bit
 5601 corresponds to the first element and the trailing bit corresponds to the last element.

5602 Although several actions can be requested, only one action (modelled by a script) will be
 5603 actually performed, provided that it is permitted for that actor and there is a single highest total
 5604 outcome. As specified above, an action may be a null-script.

5605 **4.5.12.3.2 reset (data)**

5606 Clears the configurable fields of the object instance, that is:

- 5607 – clears all bits in the *permissions_table* attribute;
- 5608 – sets all values in the *weightings_table* attribute to zero;
- 5609 – clears all bits in the *most_recent_requests_table* attribute;
- 5610 – sets the *last_outcome* attribute to zero.

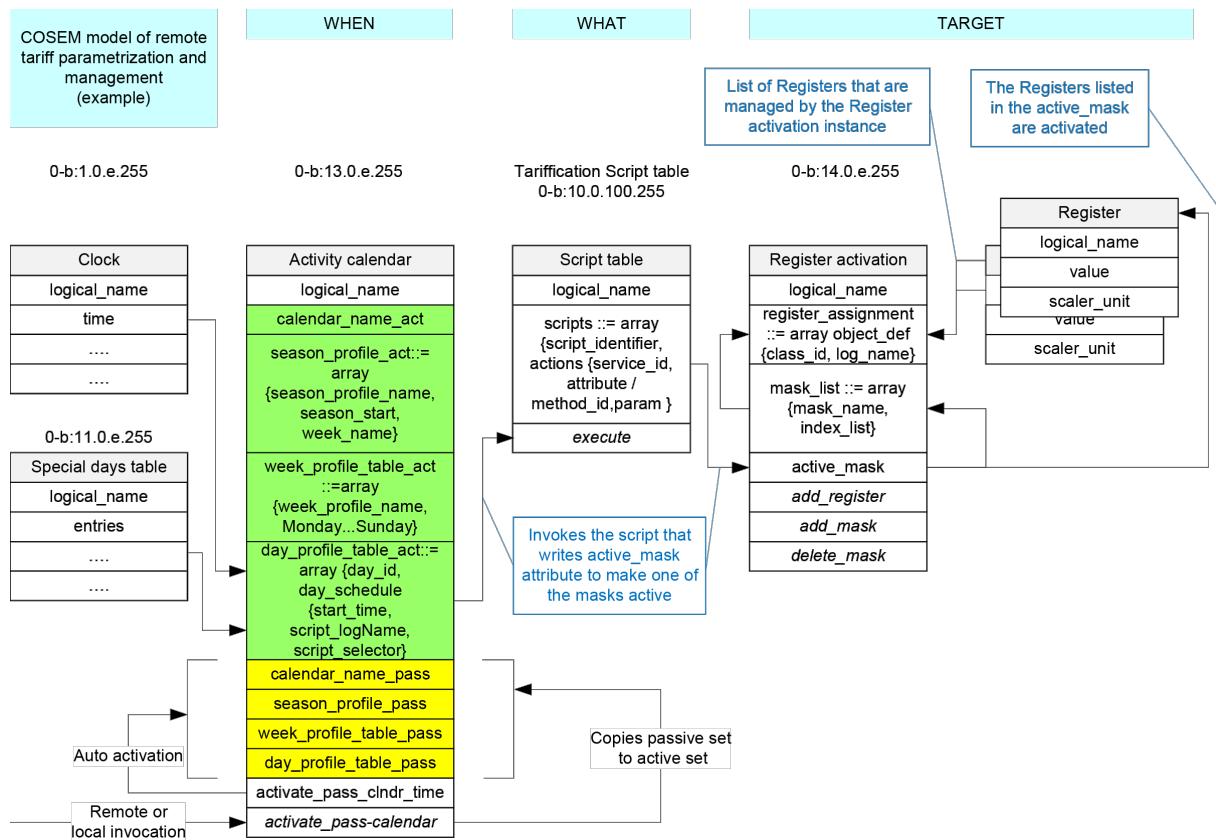
5611 NOTE 3 Following a reset it can be expected that every invocation of *request_action* will leave the *last_outcome*
 5612 attribute equal to zero, and no action will be performed since the elements in the *permissions_table* attribute are all
 5613 cleared.

5614 data ::= integer (0)

5615

5616 **4.5.13 Modelling examples: tariffication and billing**

5617 Figure 20 shows an example of modelling tariff parametrization and management using COSEM
 5618 objects.



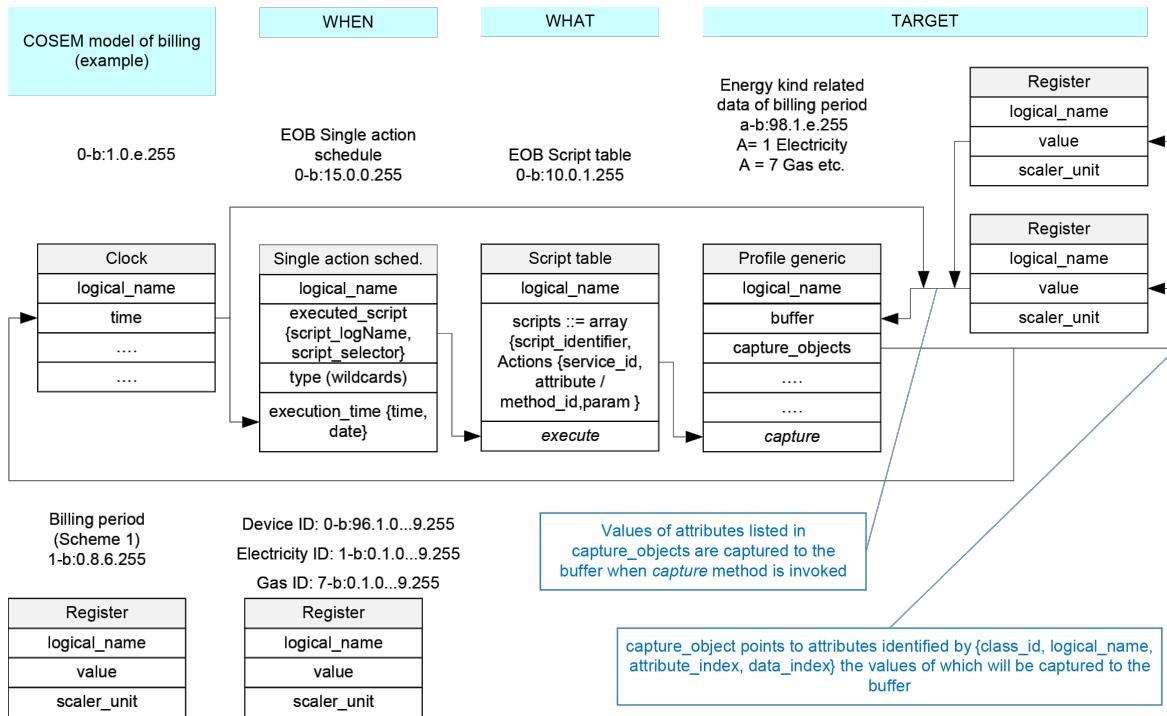
5619

IEC

5620

Figure 20 – COSEM tariffication model (example)

5621 Figure 21 shows an example of modelling parametrization and management of billing using
 5622 COSEM objects.



5623

IEC

5624

Figure 21 – COSEM billing model (example)

5625

5626 **4.6 Payment metering related interface classes**

5627 **4.6.1 Overview of the COSEM accounting model**

5628 The COSEM accounting model contains four interlinked interface classes: “Account”, “Credit”,
5629 “Charge” and the “Token gateway” IC. These classes are concerned with accounting for energy,
5630 not with delivery of that energy. The “Account” is linked to its associated “Credit”, “Charge” and
5631 “Token gateway” objects by use of the value group D and B field such that an “Account” with
5632 D=0 should be linked to a “Token gateway” with D=40 and have a “Credit” objects with D=10
5633 and “Charge” objects with D=20. Whereas an “Account” with D=1 should have “token gateway”
5634 with D=41, “Credit” objects with D=11 and “Charge” objects with D=21, etc. Multiple “Token
5635 gateway”, “Credit” and “Charge” objects related to the same “Account” are identified using
5636 different values in the value group E field.

5637 An “Account” object contains summary information and coordinates information pertaining to
5638 Credits and Charges. There is a single “Account” object per supply, for example, electricity
5639 import has one “Account” object, but a system that also has micro-generation could have a
5640 second “Account” to deal with the export of generated electricity; the second “Account” might
5641 or might not be accessible via the same Application Association (AA) as the first.

5642 A “Credit” object contains detailed information about one source of funds. There is one or
5643 (usually) more “Credit” object(s) associated with an “Account”: for example, one object for token
5644 credit and one object for emergency credit. Both of these objects can receive credit amounts
5645 from tokens, but emergency credit can only receive credit amounts when it has been consumed
5646 (entirely or partially) and when *credit_configuration* has bit 2 (Requires the credit amount to be
5647 paid back) set.

5648 There are several types of credit listed in IEC TR 62055-21:2005, and these are the types
5649 supported by the “Credit” IC. There can be zero or more instances of each type of Credit.

- 5650 • **token_credit:** Credit that is transferred to a meter operating in prepayment mode,
5651 normally in the form of Credit Tokens;

5652 NOTE 1 In a meter operating in credit mode or managed payment mode, a “Credit” object configured with type
5653 *token_credit* is used for recording the amount of credit used since last synchronised with a client.

5654 NOTE 2 The content of the token is not defined by this document and may be an amount of money or another
5655 quantity that can be accounted in a way that is equivalent to the currency used by the meter.

- 5656 • **reserved_credit:** Credit that is held in reserve, which is released under specific
5657 conditions;
- 5658 • **emergency_credit:** Accounting functions that deal with the calculation and transacting of
5659 credit that is released only under emergency situations. Usually the amount of emergency
5660 credit used is recovered from subsequently purchased credit token

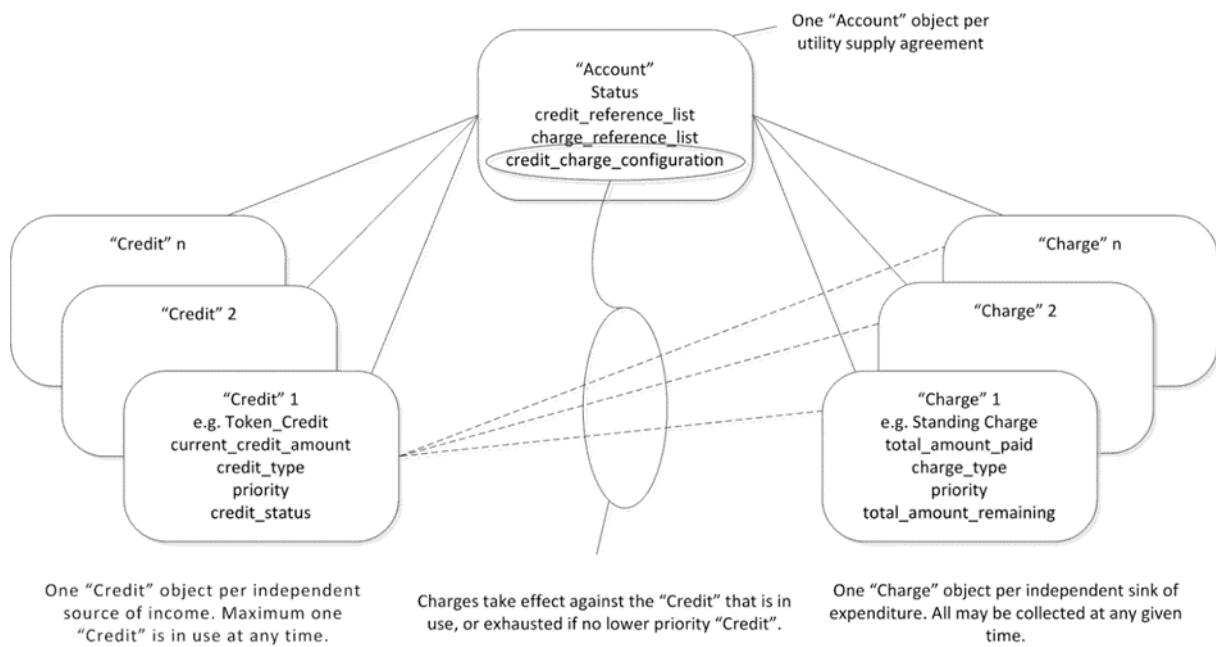
5661 NOTE 3 Emergency above refers to a time when a consumer does not have any token credit, and not to any
5662 safety related situation.

- 5663 • **time_based_credit:** Credit that is released on a scheduled time basis;
- 5664 • **consumption_based_credit:** Credit that is released on the basis of a schedule of
5665 consumption levels. For example if a consumer keeps their consumption below a
5666 threshold, the system may release a predefined amount of credit.

5667 A “Charge” object contains detailed information about one sink of expenditure, that is, one way
5668 in which credit is being used up. There can be one or (usually) more “Charge” objects
5669 associated with an “Account” for instance, one for energy usage, one for standing charge, and
5670 possibly one paying off a debt such as an installation charge.

5671 There are several types of charge listed in IEC TR 62055-21:2005, but the following types,
5672 distinguished by the trigger for collection, are the ones most useful in the COSEM accounting
5673 model. There can be zero or more instances of each type of “Charge” IC.

- 5674 • **consumption_based_collection:** describes charges that are collected according to the
 5675 amount of consumption that has occurred in a tariff. A price per unit is assigned to each
 5676 tariff register of the energy consumed
- 5677 NOTE 4 Tariffs cannot be applied when currency is in time or energy units.
- 5678 • **time_based_collection:** describes charges that are collected regularly according to the
 5679 passage of time, independent of consumption in that period. This may be used to collect
 5680 standing charges, or debt charge to be paid off over a period of time;
- 5681 • **payment_event_based_collection:** describes charges that are collected from every top-
 5682 up that is received, typically for debt repayment. These may be expressed as **amount-
 5683 based**, where a fixed amount is taken from each top-up credit received (for example, the
 5684 consumer pays £2 out of every vend regardless of the vend amount), or
 5685 **percentage_based_collection** where a proportion of the amount of top-up credit received
 5686 is taken (for example, with every vend the consumer pays 20 % of the vend amount). Bit 0
 5687 (Percentage based collection) of the *charge_configuration* attribute of the “Charge” object
 5688 specifies the method of *event_based_collection*. Figure 22 gives a general view of the
 5689 account model.

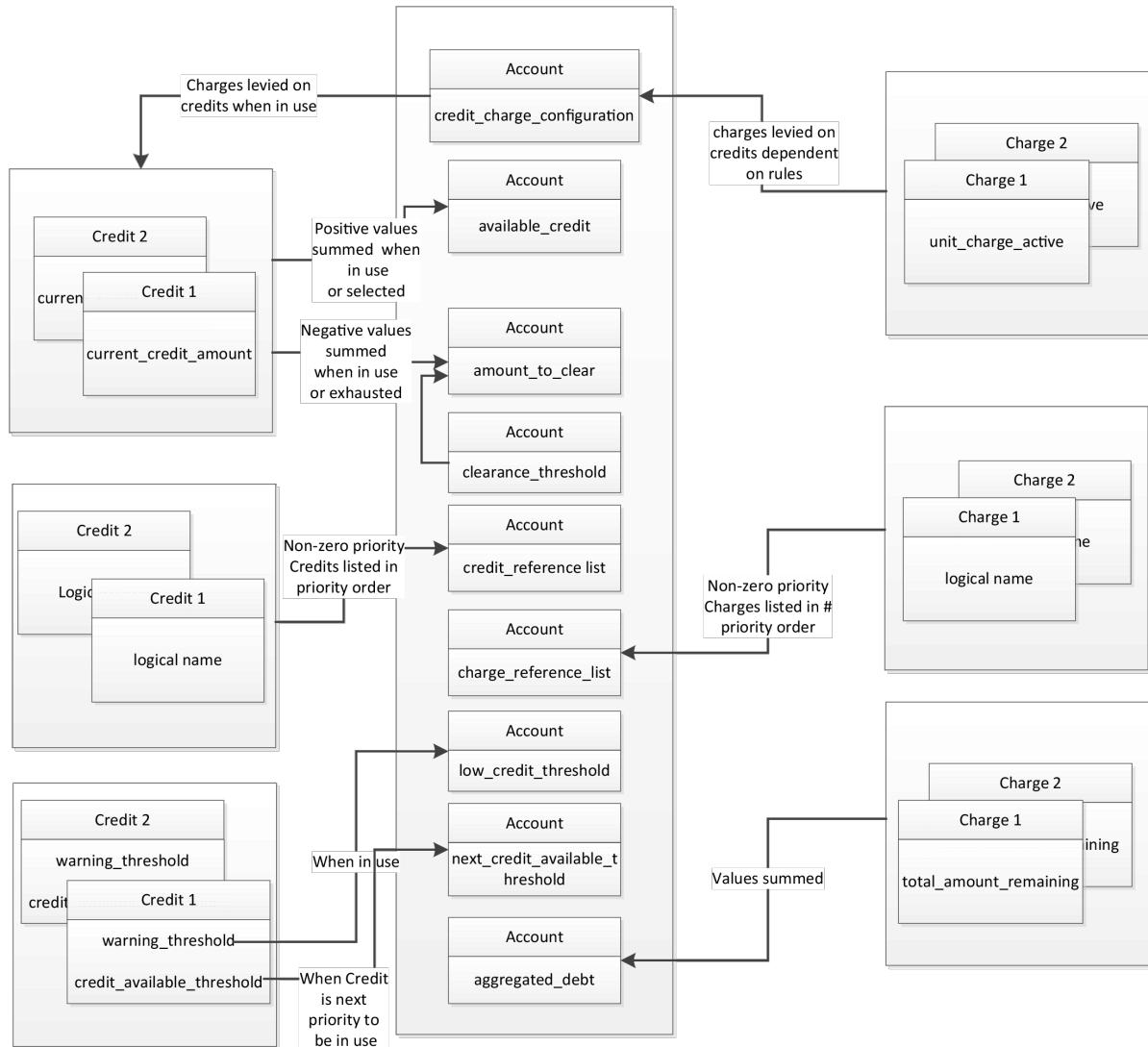


5691 **Figure 22 – Outline Account model**

5692

5693 Figure 23 shows instances of “Account”, “Credit” and “Charge” interface classes with some of
 5694 their attributes and the relationships between those attributes. In this example:

- 5695 a) There is one “Account”, two “Credit” and two “Charge” objects configured;
- 5696 b) “Credit” 1 is of type *token_credit* and the *low_credit_threshold* and *limit* attributes are
 5697 configured to be 0;
- 5698 c) Interaction between multiple classes is covered in this diagram. Detailed configuration of
 5699 individual “Credit” and “Charge” objects is not shown.



Account	0...n	class_id = 111, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string			n/a	x
2. account_mode_and_status	structure			0, 0	x + 0x08
3. current_credit_in_use (dyn.)	unsigned			0	x + 0x10
4. current_credit_status (dyn.)	bit-string			All bits clear	x + 0x18
5. available_credit (dyn.)	double-long			0	x + 0x20
6. amount_to_clear (dyn.)	double-long			0	x + 0x28
7. clearance_threshold (static)	double-long			0	x + 0x30
8. aggregated_debt (dyn.)	double-long			0	x + 0x38
9. credit_reference_list (static)	array			empty	x + 0x40
10. charge_reference_list (static)	array			empty	x + 0x48
11. credit_charge_configuration (static)	array			empty	x + 0x50
12. token_gateway_configuration (static)	array			empty	x + 0x58
13. account_activation_time (static)	octet-string			n/a	x + 0x60
14. account_closure_time (static)	octet-string			n/a	x + 0x68
15. currency (static)	structure			n/a	x + 0x70
16. low_credit_threshold (dyn.)	double-long			0	x + 0x78
17. next_credit_available_threshold (dyn.)	double-long			0	x + 0x80
18. max_provision (static)	long-unsigned			0	x + 0x88
19. max_provision_period (static)	double-long			0	x + 0x90
Specific methods	m/o				
1. activate_account (data)	o				x + 0x98
2. close_account (data)	o				x + 0xA0
3. reset_account (data)	o				x + 0xA8

5717

5718 **4.6.2.2 Attribute description**5719 **4.6.2.2.1 logical_name**

5720 Identifies the “Account” object instance. See 6.2.17.

5721 **4.6.2.2.2 account_mode_and_status**

5722 Defines the payment_mode, enumerated below, and the status of the “Account”, also
 5723 enumerated.

```
5724     account_mode_and_status::= structure
5725     {
5726         payment_mode: enum:
5727             (1) Credit mode,
5728                 (2) Prepayment mode
5729         account_status: enum:
5730             (1) New (inactive) account,
5731                 (2) Account active,
5732                     (3) Account closed
5733     }
```

5734 The payment_mode is an indication of a notional prepayment or credit/managed payment mode.

5735 NOTE 2 Payment_mode does not force some other actions in the meter, or cause a change of behaviour.

5736 The “Credit” and “Charge” objects associated with an “Account” object do not operate unless
 5737 the “Account” object is in the active state. A New (inactive) “Account” object is passive and will
 5738 become active at *account_activation_time* or invocation of the *activate_account* method.

5739 **4.6.2.2.3 current_credit_in_use**

5740 This attribute is an index into the *credit_reference_list* indicating which “Credit” object is *In use*.

5741 **4.6.2.2.4 current_credit_status**

5742 This attribute provides the status of the current “Credit” object *In use* and some information
 5743 about the next priority “Credit” object.

Bit 0 = in_credit, set when the *available_credit* is above zero,
 Bit 1= low_credit, set when the “Account” object’s *available_credit_level* has passed below the “Account” object *low_credit_threshold*,

NOTE 3 The *low_credit_threshold* attribute of the associated “Credit” object *In use* is echoed in that attribute.

Bit 2 = next_credit_enabled, set when the “Account” object *credit_reference_list* contains at least one other “Credit” object with non-zero priority, regardless of credit level,

Bit 3 = next_credit_selectable, set when the next item in the “Account” object *credit_reference_list* contains a lower priority “Credit” object with non-zero priority with a *credit_configuration* bit 1 (Requires confirmation) set such that it requires confirmation (for example “Emergency Credit” that is selectable but has not yet been selected),

NOTE 4 The next “Credit” object becomes selectable when the immediately higher priority “Credit” object has reached the “Account” object’s *next_credit_available_threshold*.

Bit 4 = next_credit_selected, set when the “Account” object *credit_reference_list* contains a “Credit” object of lower priority than the current “Credit” object *In use*, and that lower priority credit object is in the *Selected/Invoked* state,

NOTE 5 This is, for example, an “Emergency Credit” which has been selected but is not yet *In use*.

Bit 5 = selectable_credit_in_use, set when the previously selected credit in the “Account” object *credit_reference_list* is now *In use*,

Bit 6 = out_of_credit, set when the “Credit” object that was *In use* has been exhausted and no further credit is available without an action on the part of the consumer or other actor.

Bit 7 = RESERVED

5744 When the next priority “Credit” object becomes the current “Credit” object then all bits have to
 5745 update.

5746 NOTE 6 A “Credit” object becomes *Selectable* when it is the next priority “Credit” and the
 5747 *next_credit_available_threshold* attribute (reflecting the *credit_available_threshold* attribute in the “Credit”) of the
 5748 “Account” object is greater than or equal to the *available_credit* in the “Account” object. However if the
 5749 *available_credit* becomes greater than the *next_credit_available_threshold* due to the selection of the “Credit” the
 5750 status of the “Credit” remains (2) *Selected*. If the *available_credit* becomes greater than the
 5751 *next_credit_available_threshold* due to a top up or method invocation to increase the *current_credit_amount* of a
 5752 “Credit” objects that is (2) *Selected* or (3) *In use* then this will change the status of the “Credit” object to (0) *Enabled*.

5753 4.6.2.2.5 available_credit

5754 The *available_credit* attribute is the sum of the positive *current_credit_amount* values in the
 5755 instances of the “Credit” class that:

- 5756 – are listed in the “Account” object *credit_reference_list*;
- 5757 – and have a “Credit” object *credit_status* (2) *Selected/Invoked* or (3) *In use*.

5758 This attribute holds the aggregate amount of credit available associated with the “Account”
 5759 object. This value is positive or zero.

5760 NOTE 7 Negative values of “Credit” object *current_credit_amount* attributes are summed in *amount_to_clear*. See
 5761 also the vertical axis in Figure 25.

5762 double-long, scaled according to the *currency* attribute.

5763 4.6.2.2.6 amount_to_clear

5764 This value is the sum of:

- 5765 – all negative values of *current_credit_amount* in “Credit” objects that:
 - 5766 • are listed in the “Account” object *credit_reference_list* and
 - 5767 • have a “Credit” object *credit_status* (4) *Exhausted*,
 - 5768 • have the *credit_configuration* bit 2 (Requires the credit amount to be paid back) is cleared,
- 5770 – the negative (Value * -1) of the amount of credit used from all “Credit” objects where the
 5771 *credit_configuration* bit 2 (Requires the credit amount to be paid back) is set; and
- 5772 – the negative (Value * -1) of the value of the “Account” object *clearance_threshold* attribute;

5773 See also Figure 25.

5774 NOTE 8 “Credit” objects where the *credit_configuration* bit 2 is set are referred to as “repayable” credits.

5775 Credit used is the difference between the *preset_credit_amount* and the *current_credit_amount*
 5776 (thus a positive value) of a “Credit” object when the “Credit” is (3) *In use* or (4) *Exhausted*. In
 5777 the case of non-repayable credits the credit used is not relevant.

5778 NOTE 9 The payment meter’s application process might have specific functional behaviour to allow a consumer to
 5779 live in emergency credit or not. This functionality is not within the scope of this Companion Specification.

5780 This attribute is significant when the meter has accumulated a temporary debt, for instance,
5781 while an emergency credit is *In use*, and/or during a friendly credit period.

5782 This attribute contains the minimum credit that the meter will need to receive (via token receipts
5783 and method invocations on “Credit” objects) in order to clear negative credits and also make
5784 “Credits” marked as repayable (for example an emergency credit), enabled again after they
5785 have been in the (3) *In use*, (2) *Selected/Invoked* or (4) *Exhausted* state.

5786 NOTE 10 For an explanation of the process of distributing top ups between “Credit” objects please refer to attribute
5787 12 *token_gateway_configuration*.

5788 NOTE 11 Where a meter is being used in prepayment mode, *amount_to_clear* is usually the amount needed to
5789 reverse a disconnection. Amount to clear is shown on the vertical axis in Figure 25 as a negative value.

5790 double-long, scaled according to the *currency* attribute (4.6.2.2.15).

5791 **4.6.2.2.7 clearance_threshold**

5792 This attribute is used in conjunction with the *amount_to_clear*, and is included in the description
5793 of that attribute.

5794 This attribute becomes relevant when a meter has consumed all credit sources and has “Credit”
5795 objects with *current_credit_amount* attributes that have values less than zero.

5796 This represents the value of credit that the *available_credit* attribute must reach in order to
5797 make repayable “Credits” enabled again.

5798 To achieve this, it is clear that enough credit has to be added to the meter to ensure that the
5799 *current_credit_available* attribute on all listed “Credit” objects is zero or greater.

5800 double-long, scaled according to the *currency* attribute.

5801 **4.6.2.2.8 aggregated_debt**

5802 This attribute is a simple sum of *total_amount_remaining* of all the “Charge” objects which are
5803 listed in the “Account” object *charge_reference_list*, where bit 1 (Continuous collection) of the
5804 *charge_configuration* is cleared.

5805 NOTE 12 This value is not interchangeable with *amount_to_clear*; it is provided to assist external accounting.

5806 double-long, scaled according to the *currency* attribute.

5807 **4.6.2.2.9 credit_reference_list**

5808 This attribute is an array of logical names, identifying a collection of “Credit” objects that operate
5809 with this “Account” object. The elements in the array shall appear in priority order (priority 1
5810 being first to priority n being last).

5811 Priority 0 credits shall NOT appear in this list, as by definition they are not enabled.

5812 *credit_reference_list*::= array *credit_reference*

5813 *credit_reference*::= octet-string

5814 **4.6.2.2.10 charge_reference_list**

5815 This attribute is an array of logical names, identifying a set of “Charge” objects that operate
5816 with this “Account” object. The elements in the array shall appear in priority order (priority 1

5817 being first to priority n being last). Priority 0 charges shall NOT appear in this list, as by definition
 5818 they are not applied.

5819 charge_reference_list ::= array charge_reference

5820 charge_reference ::= octet-string

5821 **4.6.2.2.11 credit_charge_configuration**

5822 This attribute maps out which Charges are to be collected from which Credits.

5823 If the array has zero elements then it is assumed that any Charge may be collected from any
 5824 Credit in accordance with the “Credit” objects *credit_status* being (3) *In use* or (4) *Exhausted*.

5825 If there are entries in this array then they represent each Charge that can be collected from
 5826 each Credit.

5827 If there is a Credit from which no Charges are collected then that Credit is not consumed and
 5828 will remain unchanged until a Charge is configured.

5829 NOTE 13 Collection of a Charge will cease when reaching zero, in cases where the appropriate “Charge” object
 5830 has bit 1 (continuous collection) of its *charge_configuration* cleared. This will not alter the “Account”
 5831 *credit_charge_configuration*.

5832 credit_charge_configuration ::= array credit_charge_configuration_element

5833
 5834 credit_charge_configuration_element ::= structure
 5835 {
 5836 credit_reference: octet-string,
 5837 charge_reference: octet-string,
 5838 collection_configuration: bit-string
 5839 }

5840 Where *credit_reference* and *charge_reference* contain the *logical_name* of the relevant “Credit”
 5841 and “Charge” object.

5843

5844 collection_configuration ::= bit-string

5845 This element defines behaviour under specific conditions.

5846

Bit 0 = Collect when supply disconnected. When set, the
 “Charge” referred to in *charge_reference* may be
 collected when supply is disconnected. When
 cleared, the “Charge” referred to in *charge_reference*
 shall not be collected when supply is disconnected.

Bit 1 = Collect in load limiting periods. When set, the
 “Charge” referred to in *charge_reference* may be
 collected when supply is in a load limiting period.
 When cleared, the “Charge” referred to in
charge_reference shall not be collected when supply
 is in a load limiting period.

Bit 2 = Collect in friendly credit periods. When set, the
 “Charge” object referred to in *charge_reference* may
 be collected when supply is in a friendly credit period.
 When cleared, the “Charge” referred to in
charge_reference shall not be collected when supply
 is in a friendly credit period.

5847

5848 NOTE 14 The meter application knows internally whether it is in a supply disconnected, load limiting period, or
 5849 friendly credit period, and takes appropriate action based on the value of this attribute.

5850 NOTE 15 It is implicit that charges are also collected at times when these specific conditions are not present.

5851 **4.6.2.2.12 token_gateway_configuration**

5852 This attribute is designed to configure how a new top-up token from the “Token gateway” is to
 5853 be apportioned, such that a configurable percentage of the token amount is distributed to each
 5854 “Credit” object.

5855 NOTE 16 The distribution of token top up amounts is also affected by the values of the *current_credit_amount*,
 5856 *credit_type* and *credit_status* attributes of the “Credit” object.

5857 If there are restrictions on how the credit token received through the “Token gateway” object is
 5858 distributed, then this attribute determines the minimum proportion of the credit token that is
 5859 attributed to each “Credit” object referenced in the array.

5860 The proportions in this array shall not add up to more than 100%. If the sum of the proportions
 5861 is less than 100% then the remainder will be added to the “Credit” objects using the rules below
 5862 for an empty “token_gateway_configuration” attribute.

5863 NOTE 17 It is possible that some “Credits” will get more than their allocated proportion as a result of the process
 5864 highlighted above. The credit token is always 100% distributed across the “Credit” objects.

5865 NOTE 18 The connection between a meter’s one or more “Token gateway” objects and a specific “Account” object
 5866 is not explicitly shown in the model. The management of multiple “Account” objects and multiple token gateways is
 5867 the subject of project specific companion specifications. However in the normal case an “Account” object has one
 5868 “Token gateway” object and all tokens received follow the rules associated with the “Account” object with which it is
 5869 associated (by application of the value group D field; see also 4.6.1).

5870

```
5871     token_gateway_configuration ::= array
5872             token_gateway_configuration_element
5873
5874         token_gateway_configuration_element ::= structure
5875         {
5876             credit_reference      octet-string,
5877             token_proportion     unsigned;
5878         }
```

5879

5880 Where:

- 5881 – *credit_reference* contains the *logical_name* of the relevant “Credit” object;
- 5882 – *token_proportion* is scaled as one percent per integer step from 0 to 100.

5883 If there are no entries in the array then it is assumed that there are no restrictions on processing
 5884 the credit token within the meter’s payment application and credit should be applied first to the
 5885 lowest priority “Credit” object with its *credit_status* set to (3) *In use* or (4) *Exhausted*, then
 5886 apportioned to the other “Credit” objects in ascending order of the “Credit” object *priority*
 5887 (starting with priority n and ending with priority 1).

5888 If the “Credit” object requires a limited amount of top up (for example, an Emergency “Credit”
 5889 object that is being repaid in full) then the credit used (*preset_credit_amount* less
 5890 *current_credit_amount* before the top up) is taken from the Token amount and any surplus is
 5891 applied to higher priority Credits following the same rules. At the point when the credit used is
 5892 repaid the “Credit” will move from (3) *In use* or (4) *Exhausted* to (0) *Enabled*.

5893 See also Note 10.

5894 **4.6.2.2.13 account_activation_time**

5895 Defines the time when the object itself invokes the specific method *activate_account*. A
 5896 definition with "not specified" notation in all fields of the attribute will not be acted upon. Partial
 5897 "not specified" notation in some fields of date and time are not allowed.

5898 When this time is in the past, this field indicates the time at which the "Account" object was
 5899 activated.

5900 octet-string, formatted as specified in 4.1.6.1 for *date_time*.

5901 **4.6.2.2.14 account_closure_time**

5902 Defines the time when the object itself invokes the specific method *close_account*. A definition
 5903 with "not specified" notation in all fields of the attribute will not be acted upon. Partial "not
 5904 specified" notation in just some fields of date and time are not allowed.

5905 When this time is in the past, this field indicates the time at which the "Account" object was
 5906 closed.

5907 octet-string, formatted as specified in 4.1.6.1 for *date_time*.

5908 **4.6.2.2.15 currency**

5909 Defines the "currency" unit used by all functions of an "Account" object. The *charge_per_unit* in
 5910 the *unit_charge* attribute of "Charge" objects has an additional scaling factor that can be applied.

5911 The units could be currency or other units such as kWh, minutes; or other consumption units
 5912 for different types of meters. When the units are monetary, then the name is expressed using
 5913 the 3 character international symbol (GBP, USD, etc.) as defined in ISO 4217.

5914
 5915 currency ::= structure
 5916 {
 5917 currency_name: utf8-string,
 5918 currency_scale: integer,
 5919 currency_unit: enum
 5920 }

5921 Where:

5922 currency_name: utf8-string
 5923 as per ISO 4217

5924 OR

5925 min = minutes,

5926 hrs = hours,

5927 sec = seconds,

5928 kWh = kilowatt hours,

5929 Whr = Watt hours,

5930 mt3 = m³,

5931 ft3 = ft³,

5932 Jle = Joule

5933 OR

5934 Defined by project specific companion specification such that all characters used are lower
5935 case.

5936 The *currency_scale* indicates the exponent of a decimal multiple or submultiple of the base unit
5937 in which the relevant attribute values are expressed. Negative values indicate submultiples.

5938 NOTE 19 For example: if the currency is Euros and the values are expressed in tenths of one cent (one thousandth
5939 of a Euro) then the scalar will have value -3 (minus 3).

5940 *currency_unit*: enum:
5941 (0) time,
5942 (1) consumption,
5943 (2) monetary
5944

5945 The *currency_unit* is an enumerated value that indicates the generic type of currency:

- 5946 – time (in minutes, seconds, hours etc.);
5947 – consumption (in kWhrs, Whrs, m³, Whrs etc.); or
5948 monetary (GBP, USD, EUR, etc.).

5949 **4.6.2.2.16 low_credit_threshold**

5950 This attribute reflects the *warning_threshold* attribute of the “Credit” object currently *In use*.
5951 This threshold can be used to generate a warning to the consumer, for example. It has no other
5952 function.

5953 double-long, scaled according to the *currency* attribute.

5954 NOTE 20 The value of this attribute will change depending on which “Credit” object is *In use* at the time of the
5955 query.

5956 **4.6.2.2.17 next_credit_available_threshold**

5957 This threshold reflects the *credit_available_threshold* attribute of the next-priority “Credit” object
5958 (except when there is no next priority “Credit”; in which case this attribute has the value -2 147
5959 483 648 (largest possible negative value)).

5960 When the *available_credit* attribute of the “Account” object becomes less than or equal to this
5961 value, the *credit_status* attribute of the next priority “Credit” object changes to:

- 5962 (1) *Selectable* in the case when the *credit_configuration* attribute of the “Credit” object
5963 concerned has the bit 1 (Requires confirmation) set;
5964 (2) *Selected/Invoked* in the case where the bit 1 (Requires confirmation) is cleared and the
5965 *next_credit_available_threshold* is greater than zero;
5966 (3) *In use* in the case where the bit 1 (Requires confirmation) is cleared and the
5967 *next_credit_available_threshold* is equal to zero.

5968 NOTE 21 The “next-priority” refers to the next “Credit”, in priority order, that has not yet been selected. This attribute
5969 will change depending on which “Credit” is next in priority order.

5970 The *next_credit_available_threshold* will change depending on which “Credit” is *In use*.

5971 double-long, scaled according to the *currency* attribute.

5972 **4.6.2.2.18 max_provision**

5973 This attribute affects the operation of “Charge” objects that are configured with *charge_type*
5974 equal to (2) *payment_event_based_collection*.

5975 This attribute holds the limit amount scaled as defined in the *currency* attribute across all
5976 “Charge” objects with *charge_type* (2) *payment_event_based_collection*. The limit is related to
5977 the time period defined in *max_provision_period*.

5978 NOTE 22 The combination of this attribute and *max_provision_period* are intended to provide a limit to the total
5979 value of collections from top-ups within, for example, one week.

5980 NOTE 23 If a consumer vends many times in a short period, it is possible that he pays more than is required by the
5981 utility into one of his instantiated “Charge” objects (due to the unpredictability of payment event based collection).

5982 long-unsigned, scaled according to the *currency* attribute.

5983 **4.6.2.2.19 max_provision_period**

5984 This attribute holds the time period applicable for the *max_provision* attribute. This is specified
5985 in seconds.

5986 double-long

5987 **4.6.2.3 Method description**

5988 **4.6.2.3.1 activate_account (data)**

5989 This method allows the activation of the “Account”. The *account_status* element of
5990 *account_mode_and_status* will be set to (2) *Account active*.

5991 NOTE 24 The *account_activation_time* will be set to the time of invoking this method.

5992 data::= integer (0)

5993 **4.6.2.3.2 close_account (data)**

5994 This method forces the closure of the Account.

5995 NOTE 25 This may be done pursuant to a change of supplier or change of tenancy or any other reason.

5996 The *account_status* element of *account_mode_and_status* will be set to (3) *Account closed*.
5997 The *account_closure_time* is set to the time of invoking this method.

5998 NOTE 26 When this method is invoked the Account will no longer be active. As such its attributes will cease to
5999 change along with the attributes of all “Credit” and “Charge” objects listed within the *credit_reference_list* and
6000 *charge_reference_list* until such time that the “Account” object becomes active again, or the “Credit” and “Charge”
6001 objects are referenced from another “Account” object.

6002 data::= integer (0)

6003 **4.6.2.3.3 reset_account (data)**

6004 If the *account_status* of the attribute *account_mode_and_status* is not equal to (2) *Active account*, then this method sets the attributes of the “Account” to default values except where
6005 an attribute does not have a default value, in which case the behaviour shall be project specific.
6006

6007

6008 If the *account_status* element of the attribute *account_mode_and_status* is equal to 1 (New,
6009 inactive account), then this method shall have no effect.

6010 data::= integer (0)

6011

6012

6013

6014

6015 **4.6.3 Credit interface class (class_id = 112, version = 0)**6016 **4.6.3.1 General**

6017 Instances of the “Credit” IC allow the management of a credit that can be consumed by charges.
 6018 There are several different credit types; each “Credit” object characterizes itself by the values
 6019 of its attributes.

6020 All “Credits” associated with one supply are listed in the *credit_reference_list* attribute of the
 6021 “Account” object. “Credits” move between states by:

- 6022 • top-ups;
- 6023 • the adjustment of *current_credit_amount* by method invocation; or
- 6024 • the decrement of credit by charges.

6025 This is explained in the state diagram in 4.6.3.2. Subclause 4.6.1 lists “Credit” types as defined
 6026 in IEC TR 62055-21:2005.

6027 **4.6.3.2 Credit states**

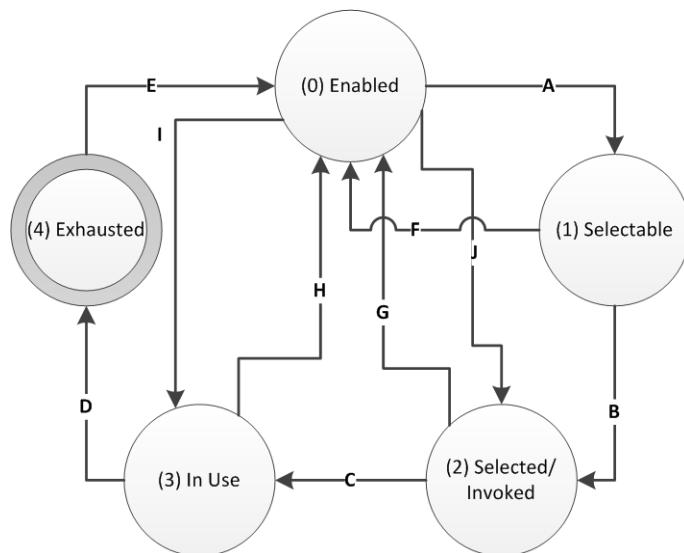
6028 The credit states only have meaning when *priority* is non-zero. They are shown in Table 34 and
 6029 Figure 24. The state transitions are shown in Table 35.

6030

Table 34 – Credit states

Priority	Credit state	Meaning
0	Any	The instance of the “Credit” object is inactive. NOTE When a “Credit” has a non-zero priority, but does not appear in any <i>credit_reference_list</i> then it has the same behaviour as if it had a zero priority.
>0	(0) Enabled	Reference to the “Credit” appears in the <i>credit_reference_list</i> of an active “Account” with a non-zero priority.
>0	(1) Selectable	The “Credit” requires some additional interaction before it can be <i>In use</i> . A credit is selectable only when it is the next priority credit, it is not exhausted and when bit 1 (Requires confirmation) of “Credit” <i>credit_configuration</i> is set.
>0	(2) Selected/Invoked	The “Credit” that was selectable has now been selected but it may not yet be <i>In use</i> (it could be that some of the higher priority “Credit” is still being used i.e. in the case of EMC). Alternatively a “Credit” that was Enabled and did not require selection may arrive here directly if the higher priority credit has become <i>Exhausted</i> .
>0	(3) In use	The “Credit” is being used to pay <i>Charges</i> within the meter.
>0	(4) Exhausted	The “Credit” has run out.

6031



6032

IEC

6033

Figure 24 – Credit States when priority >0

6034

Table 35 – Credit state transitions

Credit state	From	To	Conditions
A	(0) Enabled	(1) Selectable	<p>A “Credit” object becomes selectable when it is the highest priority “Credit” object that is not exhausted and when the “Account” object <i>available_credit</i> reaches the “Account” object <i>next_credit_available_threshold</i>.</p> <p>NOTE 1 This only applies when the <i>credit_configuration</i> attribute of the “Credit” object concerned has bit 1 (Requires confirmation) set.</p>
B	(1) Selectable	(2) Selected/Invoked	<p>The trigger to the transition is external to the COSEM domain such as a button push on the meter or some other stimulus.</p> <p>NOTE 2 This only applies when the <i>credit_configuration</i> attribute of the “Credit” object concerned has bit 1 (Requires confirmation) set.</p>
C	(2) Selected/Invoked	(3) In use	<p>This transition occurs when the previous priority “Credit” object becomes exhausted.</p> <p>NOTE 3 At this point the “Account” object <i>next_credit_available_threshold</i> updates to reflect the <i>credit_available_threshold</i> of the next priority “Credit” object.</p>
D	(3) In use	(4) Exhausted	<p>This transition occurs when the <i>current_credit_amount</i> attribute of the “Credit” object reaches the level set in the <i>limit</i> attribute.</p> <p>NOTE 4 This may indirectly cause a disconnection of supply, in the case where there is no lower priority “Credit” object to become <i>In use</i>.</p> <p>NOTE 5 At the point when the current “Credit” becomes exhausted, <i>credit_status</i> is set to(4) <i>Exhausted</i>.</p>
E	(4) Exhausted	(0) Enabled	<p>This transition occurs when the <i>current_credit_amount</i> becomes larger than the <i>limit</i> attribute., or – in the case of a repayable credits – the credit used has been paid back.</p> <p>NOTE 6 This “Credit” object has received some top-up amount from an incoming token or method invocation.</p>

Credit state	From	To	Conditions
F	(1) Selectable	(0) Enabled	<p>This transition occurs when the “Account” object <i>available_credit</i> becomes larger than the “Account” object <i>next_credit_available_threshold</i>.</p> <p>NOTE 7 This only applies when the <i>credit_configuration</i> attribute of the “Credit” object concerned has the bit 1 (Requires confirmation) set.</p> <p>NOTE 8 This is usually because the <i>current_credit_amount</i> attribute of a “Credit” object that is <i>In use</i> has been increased.</p>
G	(2) Selected / Invoked	(0) Enabled	<p>This transition occurs when the “Account” object <i>available_credit</i> becomes larger than the “Account” object <i>next_credit_available_threshold</i>.</p> <p>NOTE 9 This is usually because the <i>current_credit_amount</i> attribute of a “Credit” object that is <i>In use</i> has been increased.</p>
H	(3) In use	(0) Enabled	<p>This transition occurs when the <i>credit_status</i> of a higher priority “Credit” object becomes <i>In use</i>.</p> <p>NOTE 10 This is usually because the higher priority “Credit” object <i>current_credit_amount</i> has been increased. At this point the <i>next_credit_available_threshold</i> of the “Account” object will also change.</p>
I	(0) Enabled	(3) In use	<p>This transition occurs when the immediately higher priority “Credit” object becomes exhausted.</p> <p>NOTE 11 This only applies when the <i>credit_configuration</i> attribute of the “Credit” object concerned has the bit 1 (Requires confirmation) cleared and the <i>credit_available_threshold</i> of the “Credit” object is set to zero and its <i>current_credit_amount</i> is greater than the <i>limit</i> (a credit cannot be <i>In use</i> until the higher priority Credit is Exhausted).</p>
J	(1) Enabled	(2) Selected/ Invoked	<p>The transition occurs when the <i>available_credit</i> in the “Account” object becomes less than the <i>next_credit_available_threshold</i> on the “Account” object.</p> <p>NOTE 12 This only applies when the <i>credit_configuration</i> attribute of the “Credit” object concerned has the bit 1 (Requires confirmation before it can be selected or <i>In use</i>) cleared, and the <i>credit_available_threshold</i> of this “Credit” object is greater than zero.</p>

6035

6036 **4.6.3.3 Current credit status flags**6037 The significance of the *current_credit_status* flags (this is held by the *current_credit_status* attribute of the “Account” object) is explained in Figure 25 below.
60386039 In Figure 25 an example is given for possible cases involving a payment metering system with
6040 two “Credit” objects: Token Credit and Emergency Credit.6041 NOTE This is one possible implementation. The type and number of credits used in an actual project are subject to
6042 project specific companion specifications.6043 There are two options for selection of selectable credits; either selection of a “Credit” whilst the
6044 current “Credit” is *In use* (case 4) or when the current “Credit” has been exhausted (case 4a).6045 In the case of credit being selected while current credit is *In use* (case 4), credit will start to be
6046 consumed from the next credit immediately after current credit has exhausted. In the case (4a)
6047 the current credit will become exhausted and potentially continue to be decremented by charges
6048 until the consumer takes action by selecting the selectable credit or adding a top up.

6049

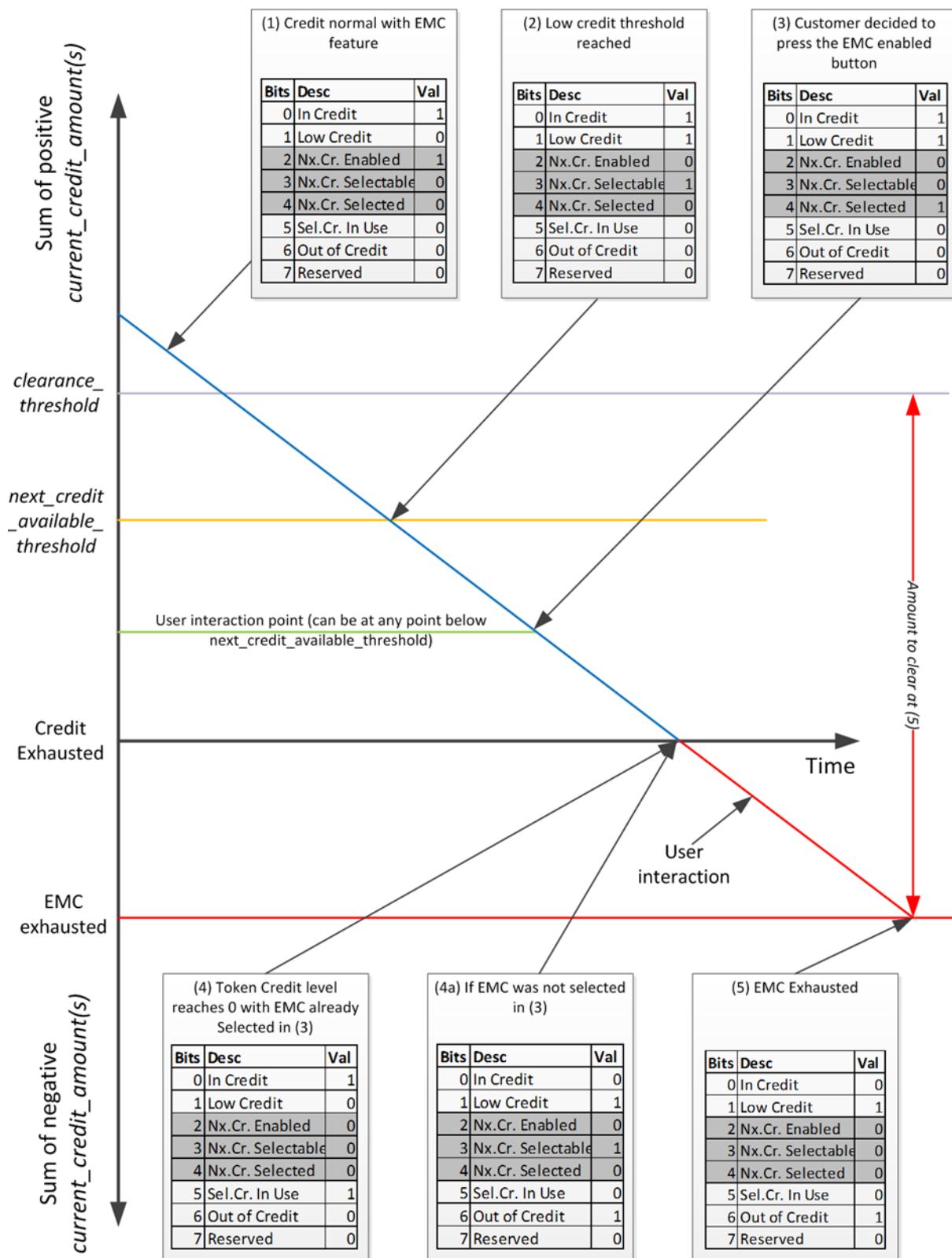


Figure 25 – Operation of current_credit_status flags

6050

6051

6052

6053

Credit	0...n	class_id = 112, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. current_credit_amount (dyn.)	double-long				x + 0x08
3. credit_type (static)	enum				x + 0x10
4. priority (static)	unsigned				x + 0x18
5. warning_threshold (static)	double-long				x + 0x20
6. limit (static)	double-long				x + 0x28
7. credit_configuration (static)	bit-string				x + 0x30
8. credit_status (dyn.)	enum				x + 0x38
<i>The following attribute (9) applies to emergency, some time-based (could be reserved credit), and consumption based credits only.</i>					
9. preset_credit_amount (static)	double-long				x + 0x40
10. credit_available_threshold (static)	double-long				x + 0x48
<i>The following attribute applies to time-based, and consumption based credit only.</i>					
11. period (static)	date-time				x + 0x50
Specific methods	m/o				
1. update_amount (data)	o				x + 0x58
2. set_amount_to_value (data)	o				x + 0x60
<i>The following method applies to emergency, and time-based consumption based credit only.</i>					
3. invoke_credit (data)	o				x + 0x68

6054

6055 **4.6.3.4 Attribute description**6056 **4.6.3.4.1 logical_name**

6057 Identifies the “Credit” object instance. See 6.2.17.

6058 **4.6.3.4.2 current_credit_amount**

6059 Provides the credit value of this particular “Credit” object. (See Figure 26).

6060 NOTE 1 This value is increased and decreased by invoking the methods of this object, the action of top-ups, and
6061 the collection of charges. This value contributes to the *available_credit* attribute in the “Account” object from which
6062 the “Credit” object is referenced.6063 double-long, scaled according to the *currency* attribute of the “Account” object.6064 **4.6.3.4.3 credit_type**6065 This is an enumeration which identifies the type of Credit that this object represents. The type
6066 indicates which attributes are expected to be processed for this “Credit” object, but the actual
6067 processing is directed by values of other attributes including the configuration flags. The types
6068 available are:

6069 enum:
 6070 (0) token_credit,
 6071 (1) reserved_credit,
 6072 (2) emergency_credit,
 6073 (3) time_based_credit,
 6074 (4) consumption_based_credit

6075 **4.6.3.4.4 priority**

6076 Describes the activation priority of this “Credit” object.

6077 Value 1 is the highest priority and value 255 the lowest.

6078 Every “Credit” object shall have a different priority, except in the case of priority 0.

6079 NOTE 2 Behaviour will be undefined if there are multiple “Credit” objects configured with the same non-zero priority.

6080 A value of 0 indicates that the “Credit” object is never activated, although it can still be
6081 configured and its *current_credit_amount* can be adjusted by invoking its methods, but it cannot
6082 receive top ups.

6083 When a “Credit” object of priority zero is configured it shall not appear in the “Account”
6084 *credit_reference_list*, it will be not considered as part of the *available_credit* calculation and it
6085 shall not be decremented by charges.

6086 See the *credit_reference_list* attribute of the “Account” object (4.6.2.2.9) for more information.

6087 **4.6.3.4.5 warning_threshold**

6088 Holds a threshold value for *current_credit_amount*. When *current_credit_amount* is
6089 decremented to the value of *warning_threshold*, a warning is triggered for the consumer that
6090 credit is low.

6091 NOTE 3 The *low_credit_threshold* attribute of the associated “Account” object echoes this attribute of the currently
6092 *In use* “Credit” object.

6093 double-long, scaled according to the *currency* attribute of the “Account” object.

6094 **4.6.3.4.6 limit**

6095 This attribute holds a threshold value for *current_credit_amount*. When *current_credit_amount*
6096 is decremented to the value of *limit*, then *credit_status* becomes (4) *Exhausted*.

6097 NOTE 4 This is typically set to zero in a meter operating in prepayment mode. For a meter operating in credit mode
6098 the highest-priority “Credit” object would normally have a limit equal to the largest possible negative number.

6099 double-long, scaled according to the *currency* attribute of the “Account” object

6100 **4.6.3.4.7 credit_configuration**

6101 Allows configuring the behaviour of the “Credit” object.

6102 If **bit 0** is set then the credit item requires a visual indication on the meter display when it
6103 becomes selected.

6104 If **bit 1** is set, confirmation is required from the consumer before moving into this type of “Credit”
6105 (for example this is the case of an emergency credit object, where the consumer is required to
6106 press a button before the “Credit” is (2) Selected/Invoked)

6107 If **bit 2** is set then this indicates that this “Credit” amount requires repayment and consequently
6108 the credit used will contribute to the *amount_to_clear* attribute in the “Account” object that
6109 references the particular “Credit” object.

6110 If **bit 3** is set then the *current_credit_amount* can be cleared by an end of billing period action
6111 (using a script acting upon the *set_amount_to_value* method. It is not recommended to
6112 configure “Credits” to permit this unless the meter is operating in credit mode.

6113 If **bit 4** is set then this “Credit” object will be permitted to receive credit from tokens.

6114 credit_configuration: bit-string:

- Bit 0 = Requires visual indication,
- Bit 1 = Requires confirmation before it can be selected/invoked,
or
- Bit 2 = Requires the credit amount to be paid back,
- Bit 3 = Resettable,
- Bit 4 = Able to receive credit amounts from tokens

6115

6116 4.6.3.4.8 credit_status

6117 Driven by the prepayment application to indicate the state that the “Credit” object is in.

6118 The states appear in Figure 24 above. Depending on the type of the “Credit” and its
6119 configuration, the value of this attribute is driven by the application based on the values of the
6120 *current_credit_amount*, *limit*, *preset_credit_amount* and *credit_available_threshold* attributes of
6121 the “Credit” objects and the *amount_to_clear* attribute of the “Account”.

6122 When the *amount_to_clear* attribute of the ‘Account’ object referencing this “Credit” object
6123 transitions from a negative value to zero the EMC status shall be (0) *Enabled* again.

6124 NOTE 5 When a “Credit” object has a *credit_status* (4) *Exhausted* then this “Credit” cannot be invoked again until
6125 the “Credit” transitions to (0) *Enabled*. If this is the lowest priority “Credit” object, then charges may still be applied
6126 to this object.

6127 enum :

- (0) The instance of the credit class is in the state of *Enabled*,
- (1) The instance of the credit class is in the *Selectable* state,
- (2) The instance of the credit class is in the *Selected/Invoked* state,
- (3) The instance of the credit class is in the *In use* state and may be consumed by charges,
- (4) The *current_credit_amount* has been entirely consumed and the credit is considered
Exhausted.

6134 For additional explanation see Table 34.

6135 4.6.3.4.9 preset_credit_amount

6136 This attribute is a value that is set as an initial amount of credit that is available for the “Credit”
6137 object.

6138 The value is added to the *current_credit_amount* attribute:

- a) when the *credit_status* becomes (2), *Selected/Invoked* if *credit_configuration* bit 1 (Requires
confirmation) is set, and the application confirmation occurs, or
- d) when the *credit_status* becomes (3) *In use* if *credit_configuration* bit 1 (Requires
confirmation) is cleared unless the *credit_status* is (4) *Exhausted*, or

- 6143 e) when the method *invoke_credit* is invoked, if the *credit_configuration* bit 2 (Requires the
 6144 credit amount to be paid back) is set, or
 6145 f) when the *date_time* in *period* occurs which is implicit.

6146 When a “Credit” does not require a preset amount, the *preset_credit_amount* shall be 0 and the
 6147 “Credit” can receive credit amounts from credit tokens or by invocation of the *update_amount*
 6148 and *set_amount_to_value* methods. The value of *preset credit amount* does not change unless
 6149 a new value is written by the client.

6150 In the case of “Credits” of type *emergency_credit*:

- 6151 – the *preset_credit_amount* attribute shall be used,
- 6152 – the “Credit” shall only be able to receive credit amounts from tokens when:
 - 6153 • the status is (3) *In use* or (4) *Exhausted*;
 - 6154 • some (or all) credit has been used; and
 - 6155 • it is required to be paid back (*credit_configuration* has bit 2 (Requires the credit amount
 6156 to be paid back) set).

6157 NOTE 6 When the *current_credit_amount* has reached the *limit* then the *credit_status* attribute will become (4)
 6158 *Exhausted*. If the *credit_configuration* bit 2 (Requires the credit amount to be paid back) is set, then the *credit used*
 6159 is the difference between *preset_credit_amount* and *current_credit_amount* (positive value), and it would be usual
 6160 that it is repaid by the addition of a credit token or by means of invoking a method. After paying back the “Credit” the
 6161 *current_credit_amount* will be zero but the *credit_status* will be (0) Enabled in the case of *amount_to_clear* = 0 or
 6162 (4) Exhausted in the case where *amount_to_clear* is less than zero.

6163 If this attribute is set to zero there is no behaviour associated with it.

6164 double-long, scaled according to the *currency* attribute of the “Account” object.

6165 **4.6.3.4.10 credit_available_threshold**

6166 A threshold value related to the “Account” object *available_credit*.

6167 When the *available_credit* on the “Account” object decrements to the *credit_available_threshold*
 6168 on the next-priority “Credit” object, the *credit_status* of that object changes to:

- 6169 a) 1 (Selectable) if the *credit_configuration* bit 1 (Requires confirmation) is set, or
- 6170 g) 2 (Selected/Invoked) if the *credit_configuration* bit 1 (Requires confirmation) is cleared.

6171 NOTE 7 This reflects whether the “Credit” requires confirmation before it is put into use.

6172 NOTE 8 A “Credit” will not be *In use* until exhaustion of a higher-priority “Credit”, but if selected before a higher
 6173 priority “Credit” is exhausted the value of the *current_credit_amount* attribute will contribute to *available_credit* in the
 6174 associated “Account”.

6175 double-long, scaled according to the *currency* attribute of the “Account” object.

6176 **4.6.3.4.11 period**

6177 Where the *credit_type* = 3 (time_based_credit) or *credit_type* = 4 (consumption_based_credit),
 6178 this attribute holds the time at which the *current_credit_amount* is set automatically to the
 6179 *preset_credit_amount*.

6180 octet-string, formatted as specified in 4.1.6.1 for *date_time*. Wildcards are allowed. If all fields
 6181 are wildcards, the *preset_credit_amount* will never be added.

6182 **4.6.3.5 Method description**6183 **4.6.3.5.1 update_amount (data)**

6184 This method adjusts the value of the *current_credit_amount* attribute. Positive values adjust the
6185 *current_credit_amount* positively. Negative values are also permitted.

6186 data::= double-long, scaled according to the *currency* attribute of the “Account” object.

6187 **4.6.3.5.2 set_amount_to_value (data)**

6188 This method sets the value of the *current_credit_amount* attribute. The amount previously in
6189 the updated attribute shall be given as the response parameter.

6190 data::= double-long, scaled according to the *currency* attribute of the “Account” object.

6191 Returns double-long, scaled according to the *currency* attribute of the “Account” object.

6192 **4.6.3.5.3 invoke_credit (data)**

6193 This method is invoked to bring this “Credit” object to *credit_status* (2)Selected/Invoked if it has
6194 a *credit_configuration* bit 1 is set (Requires confirmation), and the *credit_status* is (1) Selectable.

6195 The mechanism for selecting the “Credit” is not in the scope of COSEM and shall be specified
6196 by the implementer (e.g. button push, meter process, script etc.)

6197 data::= integer (0)

6198

6199 **4.6.3.6 Additional remarks**

6200 a) Although it is usual that tokens cause the incrementing of *current_credit_amount* and
6201 application of “Charge” objects cause decrement, these inputs are applied by means of
6202 signed addition and it could be possible to accept tokens or “Charge” objects with negative
6203 values.

6204 b) Behaviour of the *emergency_credit* type is determined by the *preset_credit_amount*
6205 attribute and the ‘Requires the credit amount to be paid back’ option in *credit_configuration*
6206 attribute.

6207 When the “Credit” object is selected/invoked, the value of the *preset_credit_amount*
6208 attribute is added to the value of the *current_credit_amount* attribute (which is normally zero
6209 at this point in its lifecycle).

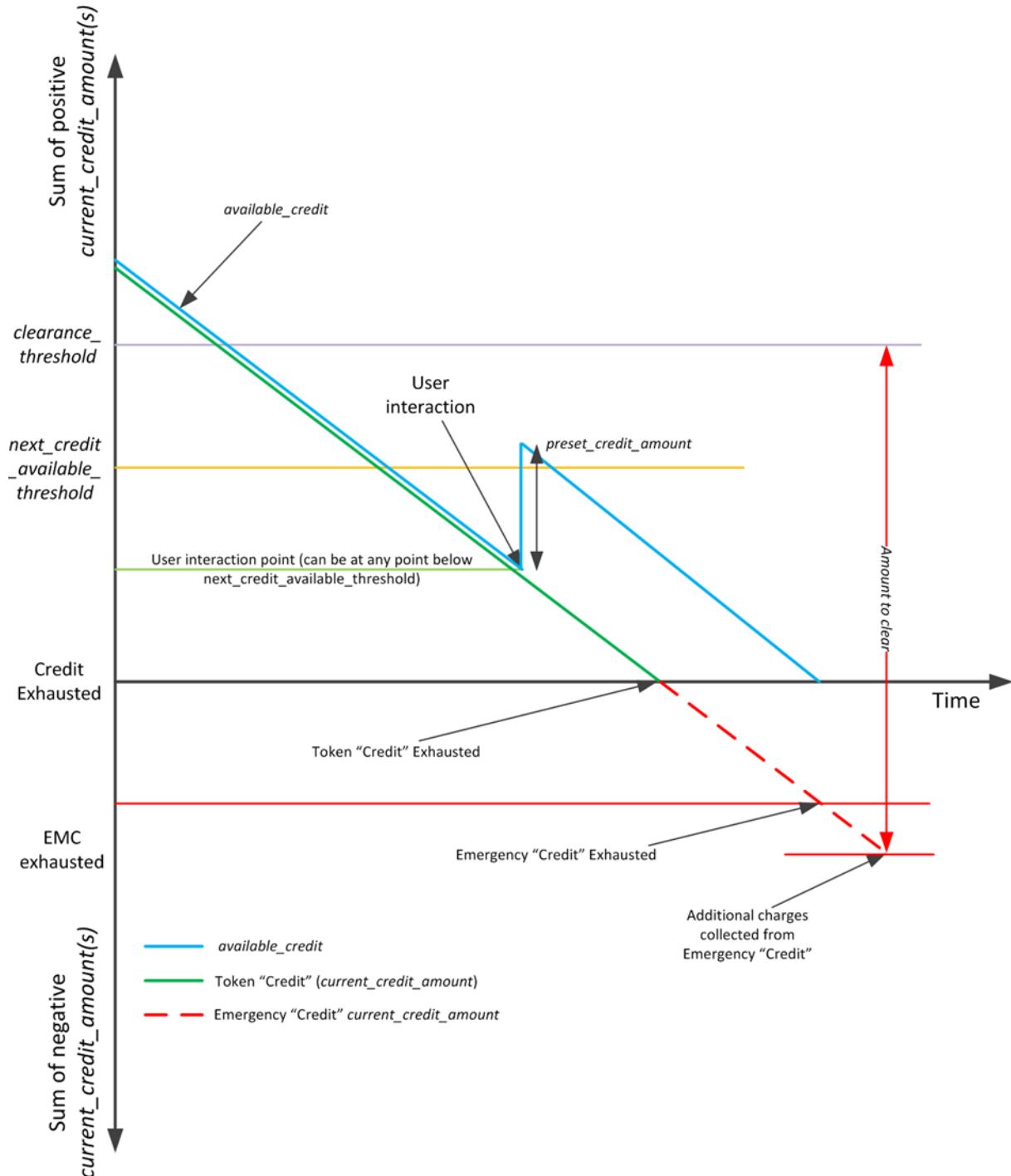
6210 c) To replenish emergency credit and make its status (0) Enabled after being (3) *In use*,
6211 sufficient payment should be received to take the *available_credit* of the “Account” object
6212 up to at least the *clearance_threshold*.

6213 This will necessarily involve providing sufficient funds to repay the negative value of
6214 *current_credit_amount* of the “Credit” objects providing EMC function. To allow for the joint
6215 management of multiple “Credit” instances the credit used values for each “Credit” object
6216 are aggregated into the *amount_to_clear* together with the *clearance_threshold* attribute of
6217 the relevant “Account” object. When *amount_to_clear* reaches zero the status (4) *Exhausted*
6218 is by definition cleared on all associated “Credit” objects.

6219 d) The “Account” object *token_gateway_configuration* attribute determines the distribution of
6220 credit to “Credit” objects.

6221 Figure 26 shows interaction of the *current_credit_amount* attributes of two “Credit” Objects in a
6222 payment metering system. It can be seen that the Emergency Credit, when selected contributes
6223 to the *available_credit* but does not become *In use* until the Token “Credit” has become

6224 *Exhausted*. At the point where Emergency “Credit” is *In use* it starts contributing to the
 6225 *amount_to_clear*. When the Emergency “Credit” is *In use* and contributing to *amount_to_clear*,
 6226 the *amount_to_clear* also takes into consideration the clearance threshold. The clearance
 6227 threshold is only important when Token “Credit” *curren_credit_amount* is zero or below.



6229 **Figure 26 – Interaction of current_credit_amount and available_credit
 6230 with Token “Credit” and Emergency “Credit”**

6231

6232 As a result of invoking the Emergency “Credit” the *available_credit* will become bigger than
 6233 *next_credit_available_threshold* but the Emergency “Credit” will not change *credit_status* from
 6234 (1) *Selected* to (0) *Enabled*. However if the top up or method invocation to *set_amount_to_value*,

6235 forced the *available_credit* (by means of increasing the *current_credit_amount* of a “Credit”
 6236 object) to be more than *next_credit_available_threshold* then the status shall be forced to (0)
 6237 *Enabled*.

6238 **4.6.4 Charge (class_id = 113, version = 0)**

6239 Instances of the “Charge” IC allow the management of a single Charge. Depending on the
 6240 attributes configured such as amount per price and the period, the Charge is taken at
 6241 appropriate times from the “Credit” object *In use*.

6242 NOTE 1 The details of the collection (charge-taking) cycle may be project dependent, and thus they are subject to
 6243 project specific companion specifications since they affect third-party estimates of current values.

6244 Each “Charge” object characterises itself by the values of its attributes.

6245 All “Charges” associated with one supply are referenced in the *charge_reference_list* attribute
 6246 of the related “Account” object (4.6.2.2.10).

Charge	0...n	class_id = 113, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. total_amount_paid (dyn.)	double-long				x + 0x08
3. charge_type (static)	enum				x + 0x10
4. priority (static)	unsigned				x + 0x18
5. unit_charge_active (static)	structure				x + 0x20
6. unit_charge_passive (static)	structure				x + 0x28
7. unit_charge_activation_time (static)	octet-string				x + 0x30
<i>The following attribute relates to time based and consumption based collection only.</i>					
8. period (static)	double-long-unsigned				x + 0x38
<i>The following attributes relate to all charge types.</i>					
9. charge_configuration (static)	bit-string				x + 0x40
10. last_collection_time (dyn.)	date-time				x + 0x48
11. last_collection_amount (dyn.)	double-long				x + 0x50
12. total_amount_remaining (dyn.)	double-long				x + 0x58
<i>The following attributes relate to payment event based collection only.</i>					
13. proportion (static)	long-unsigned				x + 0x60
<i>Specific methods</i>					
1. update_unit_charge (data)	o				x + 0x68
2. activate_passive_unit_charge (data)	o				x + 0x70
<i>The following methods relate to time-based and payment event based collection only.</i>					
3. collect (data)	o				x + 0x78
<i>The following methods relate to payment event based collection only.</i>					
4. update_total_amount_remaining (data)	o				x + 0x80
5. set_total_amount_remaining (data)	o				x + 0x88

6248 **4.6.4.1 Attribute description**6249 **4.6.4.1.1 logical_name**

6250 Identifies the “Charge” object instance. See 6.2.17.

6251 **4.6.4.1.2 total_amount_paid**

6252 Holds the total amount collected for this “Charge” object. It is not normally reset while the
6253 “Account” remains active.

6254 NOTE 2 This value is incremented by the application at the same time and at the same rate as the charge that is
6255 collected.

6256 **4.6.4.1.3 charge_type**

6257 Designates the type of collection associated with this instance of the “Charge” class.

6258 enum:

6259 (0) consumption_based_collection,
6260 (1) time_based_collection,
6261 (2) payment_event_based_collection

6262

6263 The *charge_type* indicates which attributes are expected to be processed for this “Charge”
6264 object. The processing is also influenced by the *charge_configuration* attribute.

6265 NOTE 3 In the case of *payment_event_based_collection* charges this attribute dictates the mechanism of charge
6266 collection. The application collects *payment_event_based_collection* charges from appropriate “Credits” as soon as
6267 credit is distributed from a new token.

6268 **4.6.4.1.4 priority**

6269 Describes the priority of this particular “Charge” when making collection from a “Credit” object.

6270 Every “Charge” object shall have a different priority, except in the case of priority 0.

6271 A value of 1 represents highest priority and a value of 255 the lowest.

6272 A value of 0 indicates that the “Charge” is not activated, though it can still be configured and
6273 its *total_amount_remaining* can be adjusted by invoking the appropriate methods of the “Charge”
6274 object.

6275 “Charge” objects with priority value 0 shall not appear in the *charge_reference_list* of the
6276 “Account” object, but “Charge” objects with *priority* <> 0 do necessarily appear in the
6277 *charge_reference_list* of an “Account”.

6278 **4.6.4.1.5 unit_charge_active**

6279 Defines the active price, that is the amount charged per unit consumed, per unit time, or per
6280 payment received, in terms of the currency attribute of the relevant “Account” instance and
6281 where relevant, the *scaler_unit* attribute of the object identified by the *commodity_reference*
6282 structure.

```
6283                 unit_charge_active::= structure
6284                 {
6285                 charge_per_unit_scaling:    charge_per_unit_scaling_type,
6286                 commodity_reference:      commodity_reference_type,
6287                 charge_table:                charge_table_type
```

6288 }

6289 Where:

```

6290     charge_per_unit_scaling_type::= structure
6291     {
6292         commodity_scale: integer,
6293         price_scale: integer
6294     }
6295     commodity_reference_type::= structure
6296     {
6297         class_id: long-unsigned,
6298         logical_name: octet-string,
6299         attribute_index: integer
6300     }
6301
6302     charge_table_type::= array charge_table_element
6303
6304     charge_table_element::= structure
6305     {
6306         index: octet-string,
6307         charge_per_unit: long
6308     }
6309

```

6310 Explanations related to charge_per_unit_scaling

6311 Where the *charge_type* = (0) *consumption_based_collection* the field in
 6312 *charge_per_unit_scaling* is expressed as the exponent (to the base of 10) of the multiplication
 6313 factor of the base unit in which the relevant attribute values are expressed internally:

- 6314 – *commodity_scale* is applied to the units associated with *commodity_reference*,
- 6315 – *price_scale* is applied to the *currency_scale* of the *currency* attribute of the relevant
 6316 “Account” object.

6317 Where the *charge_type* IS NOT (0) *consumption_based_collection*, the *commodity_scale* shall
 6318 be set to 0 and the *price_scale* is expressed as the exponent (to the base of 10) of the
 6319 *currency_scale* to be applied to the *currency* attribute of the “Account” object.

6320 NOTE 4 For example, suppose the primary *currency_name* is Euros (as determined in the “Account” object) with
 6321 values expressed in units of one cent (one hundredth of a Euro) and the commodity being supplied is active import
 6322 electrical energy with values expressed in units of 1 000 Wh (one kWh) from the *scaler_unit* attribute of the
 6323 *commodity_reference*). Then for a price in tenths of a cent per kilowatt-hour the price scale will have value -3 (minus-
 6324 three) since it is in thousandths of a base unit and the commodity scale will have value 0 (zero) since it is in base
 6325 units (i.e. kWh).

6326 For the case where the primary currency is kWh or similarly related to the *commodity_scale*
 6327 then a TOU tariff is not possible and as such the *charge_table* element of the *unit_charge_active*
 6328 and *unit_charge_passive* shall not be relevant but the *commodity_scale* and *price_scale* shall
 6329 be the same value.

6330 NOTE 5 The *commodity_scale* and *price_scale* do not imply any particular rate of collection.

6331 Explanations related to charge_type

6332 When the *charge_type* = (0) *consumption_based_collection* the *commodity_reference*
 6333 identifies a *scaler_unit* attribute of a total register, measuring whatever is being supplied, for
 6334 instance, electrical import energy.

6335 When the *charge_type* = (1) *time_based_collection* or (2) *payment_event_based_collection*,
 6336 then the *commodity_reference* shall be a structure of zero elements.

6337 Explanations to charge table elements

6338 Each element in the array *charge_table* is a structure identifying what is being charged for and
6339 collected.

6340 Where charge_type = (0) consumption_based_collection:

- index is an identifier of a derivative of the main commodity reference, for example tariff rate registers,
 - charge_per_unit is the charge amount per unit that is scaled by the *charge_per_unit_scaling* factors.

6345 NOTE 6 For example, if the end consumer generates energy then the "Charge" should be negative, meaning
6346 that the utility pays the end consumer for the energy he produces.

6347 NOTE 7 It is also possible to model exported energy using a separate "Account" object and other related
6348 objects.

6349 Where the *charge_type* = (1) *time_based_collection* or (2) *payment_event_based_collection*,
6350 then a single entry into the array *charge_table* is made, containing:

- index is an octet-string of length 0;
 - charge_per_unit is a value equivalent to the amount charged per *period* or per payment event that is scaled by the *charge_per_unit_scaling* factors.

6354

6355 4.6.4.1.6 unit_charge_passive

6356 Holds the details of prices to be applied at the occurrence of the activation date.

6357 Its data structure is the same as the *unit_charge_active*. On activation, the whole structure of
6358 *unit_charge_passive* is copied into *unit_charge_active*.

6359 4.6.4.1.7 unit_charge_activation_time

6360 Defines the time when the object itself invokes the specific method
6361 *activate_unit_charge_passive.*

6362 A definition with "not specified" notation in all fields of the attribute will deactivate this
6363 automatism. Partial "not specified" notation in just some fields of date and time is not allowed.

6364 octet-string, formatted as specified in 4.1.6.1 for *date_time*

6365 4.6.4.1.8 period

6366 Where `charge_type` = (0) `consumption_based_collection` or (1) `time_based_collection` it defines
6367 the period over which the Charge will be collected. If period is set to zero there is no automatic
6368 collection of this "Charge": collection is done by invoking the `collect` method.

6369 NOTE 8 Implementations may vary: it is possible to collect the whole charge in a single collection, or to divide the
6370 collection into parts.

6371 Where charge_type = (2) payment_event_based_collection this attribute is not used.

6372

6373 double-long-unsigned, expressed in seconds.

6374 **4.6.4.1.9 charge_configuration**

6375 This attribute is a bit-string defined as follows:

6376 charge_configuration ::= bit-string

6377 Bit 0 = Percentage based collection,

6378 Bit 1 = Continuous collection

6379 If bit 0 is set and *charge_type* = (2) *payment_event_based_collection* then this “Charge” object
6380 will collect a proportion of each top-up in accordance with the proportion attribute (4.6.4.1.13).
6381 This applies only to token top ups and not to method invocations.

6382 NOTE 9 The proportion attribute of the “Charge” object and the provision attribute of the “Account” object provide
6383 more information on this collection.

6384 If bit 1 is set then collection will not terminate when the *total_amount_remaining* is equal to
6385 zero. If bit 1 is cleared then charge collection will terminate when the *total_amount_remaining*
6386 attribute is equal to zero.

6387 **4.6.4.1.10 last_collection_time**

6388 Holds the *date_time* when the last collection took place.

6389 This includes incremental collection made against consumption.

6390 octet-string, formatted as specified in 4.1.6.1 for *date_time*

6391 **4.6.4.1.11 last_collection_amount**

6392 Holds the last collection amount relating to the *last_collection_time* attribute.

6393 double-long, scaled according to the *price_scale* element of *unit_charge_active*.

6394 **4.6.4.1.12 total_amount_remaining**

6395 Where *charge_configuration* bit 1 (Continuous collection) is cleared, this attribute holds the
6396 total amount remaining for this “Charge” object.

6397 NOTE 10 The value of this attribute is initially configured by invoking the *set_total_amount_remaining* method.

6398 NOTE 11 This attribute is used to limit the collections for repayment of a debt. It is not a direct measure of the
6399 consumer’s liability. The value is not allowed to go negative (it is set to zero instead) and is not incremented by
6400 collection of a “Charge” with a negative price.

6401 Where *charge_configuration* bit 1 (Continuous collection) is set, this attribute is zero.

6402 NOTE 12 See also the Additional Notes at the end of the “Charge” IC specification.

6403 double-long, scaled according to the *price_scale* element of *unit_charge_active*.

6404 **4.6.4.1.13 proportion**

6405 Where the *charge_configuration* bit 0 (Percentage based collection) is set and *charge_type* =
6406 (2) *payment_event_based_collection*, then this attribute is the proportion of each top-up amount
6407 processed within the “Token gateway” object linked via the “Account” object to this “Charge”
6408 object.

6409 The range 0x0000 to 0x2710 (0 to 10,000) represents 0 to 100 %.

6410 NOTE 13 The amount of collection that occurs may be limited by the *max_provision* and *max_provision_period*
6411 attributes of the “Account” object.

6412 If a fixed amount is required to be collected at every top-up (*charge_type* = 2), then the amount
6413 to be taken should be set in the first element of the *charge_table* array in the *unit_charge_active*
6414 / *unit_charge_passive* attribute.

6415 Where either *charge_type* <> (2) *payment_event_based_collection* or *charge_configuration* bit
6416 0 (Percentage based collection) is cleared then this attribute has no effect.

6417 In the case when *charge_type* = (2) *payment_event_based_collection* and the collection of a
6418 fixed amount is required see *unit_charge_active* (4.6.4.1.5) and *unit_charge_passive*
6419 (4.6.4.1.6).

6420 **4.6.4.2 Method description**

6421 **4.6.4.2.1 update_unit_charge (data)**

6422 Allows updating a subset of unit charge values inside the *unit_charge_passive* attribute. The
6423 *charge_per_unit_scaling* and *commodity_reference* values are not affected by this method.

6424 It remains necessary to activate the *unit_charge_passive* so that the values can take effect.

```
6425         data ::= array charge_table_element
6426         charge_table_element ::= structure
6427         {
6428             index:          octet-string,
6429             charge_per_unit: long
6430         }
```

6431 See the *charge_table_element* of the *unit_charge_active* attribute for a description of the
6432 elements.

6433 NOTE 14 In the case where the index exists then the value is overwritten, and if the index does not exist then the
6434 new value is appended to the array.

6435 **4.6.4.2.2 activate_passive_unit_charge (data)**

6436 Copies the whole structure of the *unit_charge_passive* attribute into the *unit_charge_active*
6437 attribute.

6438 NOTE 15 This method has no effect on the collection activity or the priority of the “Charge” object.

6439 data ::= integer (0)

6440 **4.6.4.2.3 collect (data)**

6441 Where *charge_type* <> (0) *consumption_based_collection*, this method makes collection of the
6442 amount defined in *unit_charge_active* when *charge_configuration* bit 0 (percentage based
6443 collection) is cleared.

6444 Where *charge_type* = (0) *consumption_based_collection*, this method has no effect.

6445 data ::= integer (0)

6446 **4.6.4.2.4 update_total_amount_remaining (data)**

6447 Allows the update of the *total_amount_remaining* attribute. The value is to be scaled according
 6448 to the *price_scale* of *unit_charge_active*. The value is added to *total_amount_remaining*. The
 6449 amount previously in *total_amount_remaining* attribute shall be given as the response
 6450 parameter.

6451 NOTE 16 This does not affect *total_amount_paid*.

6452 data ::= double-long, scaled according to the *price_scale* of *unit_charge_active*.

6453 and

6454 returns double-long, scaled according to the *price_scale* of *unit_charge_active*.

6455 **4.6.4.2.5 set_total_amount_remaining (data)**

6456 Sets the *total_amount_remaining* attribute. The value shall be not less than 0. The amount
 6457 previously in *total_amount_remaining* attribute shall be given as the response parameter.

6458 NOTE 17 *total_amount_remaining* is set without affecting *total_amount_paid*.

6459 data ::= double-long, scaled according to the *price_scale* of *unit_charge_active*,

6460 and

6461 returns double-long, scaled according to the *price_scale* of *unit_charge_active*.

6462

6463 **4.6.5 Token gateway (class_id = 115, version = 0)**

6464 An instance of the “Token gateway” IC implements the Token Carrier Interface.

6465 NOTE 1 A single instance of the “Token gateway” object is instantiated for each “Account” object and hence each
 6466 supply contract.

Token gateway	0...n	class_id = 115, version = 0				
Attributes	Data type	Min.	Max.	Def.	Short name	
1. logical_name (static)	octet-string				x	
2. token (dyn.)	octet-string				x + 0x08	
3. token_time (dyn.)	octet-string				x + 0x10	
4. token_description (dyn.)	array				x + 0x18	
5. token_delivery_method (dyn.)	enum				x + 0x20	
6. token_status (dyn.)	structure				x + 0x28	
Specific methods	m/o					
1. enter (data)	m				x + 0x30	

6467

6468 **4.6.5.1 Attribute description**

6469 **4.6.5.1.1 logical_name**

6470 Identifies the “Token gateway” object instance. See 6.2.17.

6471 4.6.5.1.2 token

6472 Contains the unprocessed octet string of the most recently received or token for the purpose of
6473 capture in a history profile.

4.6.5.1.3 token_time

6475 Contains, for the most recently received and processed token, the time and date at which a
6476 received token arrived at the DLMS/COSEM server.

6477 octet-string, formatted as specified in 4.1.6.1 for *date_time*

4.6.5.1.4 token_description

6479 Contains a description of the token for the most recently received and processed token which
6480 is supplied by the meter's application program.

6481 NOTE 2 For example this could contain the type of token (credit or engineering), and/or the token origin.

6482 The use of this attribute is to be described in the token specification or project specific
6483 companion specifications.

```
6484     token_description::= array    token_description_element  
6485  
6486     token_description_element ::= octet-string
```

6488 4.6.5.1.5 token_delivery_method

6489 Reflects the route by which the last token was received.

6490 enum:
6491 (0) via remote communications,
6492 (1) via local communications,
6493 (2) via manual entry

6495 4.6.5.1.6 token status

6496 `token_status` is a structure containing a generic enumeration that shows the status of the last
6497 token operation and an optional bit-string that can be associated with the token status giving
6498 extra information about the token `status code`.

6499 NOTE 3 It is expected that the optional bit string content will be documented in a project specific companion
6500 specification.

NOTE 4 The exact meaning and criteria for evaluation of the *status_code* values will be slightly different depending on the token protocol and format. For example a token conforming to IEC 62055-41 has very precise criteria and meanings for each of the terms Validation, Authentication and Token result, while other specifications may have differing terms.

6505

```
6506     token_status ::= structure
6507     {
6508         status_code:          enum,
6509         data_value:          bit-string
6510     }
6511
6512     status_code: enum:
6513
6514     (0) Token format result OK,
6515     (1) Authentication result OK,
6516     (2) Validation result OK.
```

6517 (3) Token execution result OK,
6518 (4) Token format failure,
6519 (5) Authentication failure,
6520 (6) Validation result failure,
6521 (7) Token execution result failure,
6522 (8) Token received and not yet processed

6524 On receipt of a token the initial state shall be set to 8 while the token is being processed and
6525 until another state can be deduced.

6526 The use of the data_value bit-string field is to be defined in project specific companion
6527 specifications.

6528 4.6.5.2 Method

4.6.5.2.1 enter (data)

6530 This method is invoked to transfer a token to the DLMS/COSEM server in the form of octet-
6531 string. If successful it may return the *token_status* attribute.

6532 data ::= octet-string, and returns *token_status*

6533 4.6.5.3 Additional remarks

6534 There are a number of configurable limits that may be held by “Data” objects and relate to the
6535 behaviour and processing of tokens and some aspects of the payment metering system. Some
6536 of these limits have been defined in this specification, have standard OBIS codes and are
6537 described below:

6538 "Max credit limit": On receipt of a token the meter's application process shall check that the
6539 addition of the credit token's value to the associated credit object(s) will not force the
6540 *available_credit* in the "Account" object above the limit programmed in the "Max credit limit"
6541 object. If the received token is found to force the *available_credit* to be more than this limit then
6542 the meter shall reject the token and return the appropriate status.

6543 "Max vend limit": On receipt of a token the meter's application process shall check that the
6544 value of the credit token is not more than the limit programmed in the "Max vend limit" object. If
6545 the received token is found to be more than this limit then the meter must reject the token and
6546 return the appropriate status.

6547 See also 6.2.17.

6548

6549 **4.7 Interface classes for setting up data exchange via local ports and modems**6550 **4.7.1 IEC local port setup (class_id = 19, version = 1)**6551 **4.7.1.1 Overview**

6552 This IC allows modelling the configuration of communication ports using the protocols specified
 6553 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				

6554

6555 **4.7.1.2 Attribute description**6556 **4.7.1.2.1 logical_name**

6557 Identifies the “IEC local port setup” object instance. 6.2.18.

6558 **4.7.1.2.2 default_mode**

6559 Defines the protocol used by the meter on the port.

6560 enum:
 6561 (0) protocol according to IEC 62056-21:2002 (modes A...E),
 6562 (1) protocol according to IEC 62056-46:2002/AMD1:2006. Using this
 6563 enumeration value all other attributes of this IC are not applicable,
 6564 (2) protocol not specified. Using this enumeration value, attribute 4), prop_baud
 6565 is used for setting the communication speed on the port. All other attributes
 6566 are not applicable.
 6567

6568 **4.7.1.2.3 default_baud**

6569 Defines the baud rate for the opening sequence.

6570 enum:

6571 (0) 300 baud,
 6572 (1) 600 baud,
 6573 (2) 1 200 baud,
 6574 (3) 2 400 baud,
 6575 (4) 4 800 baud,
 6576 (5) 9 600 baud,
 6577 (6) 19 200 baud,
 6578 (7) 38 400 baud,
 6579 (8) 57 600 baud,

6580 (9) 115 200 baud
6581

6582 **4.7.1.2.4 prop_baud**

6583 Defines the baud rate to be proposed by the meter.

6584 enum:

6585 (0) 300 baud,
6586 (1) 600 baud,
6587 (2) 1 200 baud,
6588 (3) 2 400 baud,
6589 (4) 4 800 baud,
6590 (5) 9 600 baud,
6591 (6) 19 200 baud,
6592 (7) 38 400 baud,
6593 (8) 57 600 baud,
6594 (9) 115 200 baud
6595

6597 **4.7.1.2.5 response_time**

6598 Defines the minimum time between the reception of a request (end of request telegram) and
6599 the transmission of the response (begin of response telegram).

6600 enum:

6601 (0) 20 ms,
6602 (1) 200 ms
6603

6604 **4.7.1.2.6 device_addr**

6606 Device address according to IEC 62056-21:2002.

6607 **4.7.1.2.7 pass_p1**

6608 Password 1 according to IEC 62056-21:2002.

6609 **4.7.1.2.8 pass_p2**

6610 Password 2 according to IEC 62056-21:2002.

6611 **4.7.1.2.9 pass_w5**

6612 Password W5 reserved for national applications.

6613

6614

6615

6616

6617

6618

6619

6620

6621 **4.7.2 IEC HDLC setup (class_id = 23, version = 1)**6622 **4.7.2.1 Overview**

6623 This IC allows modelling and configuring communication channels according to IEC 62056-
 6624 46:2002/AMD1:2006. 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	0x3FFD		x + 0x40
Specific methods	m/o				

6625

6626 NOTE 1 The maximum value of the attributes *max_info_field_length_transmit* and *max_info_field_length_receive*
 6627 has been increased from 128 to 2 030 for efficiency reasons.

6628 NOTE 2 A *max_info_field_length_receive* of 128 bytes is needed to ensure a minimal performance.

6629 NOTE 3 The maximum value of the *inter-octet-time-out* attribute has been increased from 1 000 ms to 6 000 ms in
 6630 order to allow using communication media, where long delays may occur. The default value has been changed to 25
 6631 ms to align with 6.4.4.3.4 of IEC 62056-46:2002/AMD1:2006.

6632 **4.7.2.2 Attribute description**6633 **4.7.2.2.1 logical_name**

6634 Identifies the “IEC HDLC setup” object instance. See 6.2.20.

6635 **4.7.2.2.2 comm_speed**

6636 The communication speed supported by the corresponding port.

6637 **enum:**
 6638 (0) 300 baud,
 6639 (1) 600 baud,
 6640 (2) 1 200 baud,
 6641 (3) 2 400 baud,
 6642 (4) 4 800 baud,
 6643 (5) 9 600 baud,
 6644 (6) 19 200 baud,
 6645 (7) 38 400 baud,
 6646 (8) 57 600 baud,
 6647 (9) 115 200 baud

6648 This communication speed can be overridden if the HDLC mode of a device is entered through
 6649 a special mode of another protocol.

6650 **4.7.2.2.3 window_size_transmit**

6651 The maximum number of frames that a device or system can transmit before it needs to receive
6652 an acknowledgement from a corresponding station. During logon, other values can be
6653 negotiated.

6654 **4.7.2.2.4 window_size_receive**

6655 The maximum number of frames that a device or system can receive before it needs to transmit
6656 an acknowledgement to the corresponding station. During logon, other values can be negotiated.

6657 **4.7.2.2.5 max_info_length_transmit**

6658 The maximum information field length that a device can transmit. During logon, a smaller value
6659 can be negotiated.

6660 **4.7.2.2.6 max_info_length_receive**

6661 The maximum information field length that a device can receive. During logon, a smaller value
6662 can be negotiated.

6663 **4.7.2.2.7 inter_octet_time_out**

6664 Defines the time, expressed in milliseconds, over which, when no character is received from
6665 the primary station, the device will treat the already received data as a complete frame.

6666 **4.7.2.2.8 inactivity_time_out**

6667 Defines the time, expressed in seconds over which, when no frame is received from the primary
6668 station, the device will process a disconnection.

6669 When this value is set to 0, this means that the inactivity_time_out is not operational.

6670 **4.7.2.2.9 device_address**

6671 Contains the physical device address of a device.

6672 In the case of one byte addressing:

6673	0x00	NO_STATION Address,
6674	0x01...0x0F	Reserved for future use,
6675	0x10...0x7D	Usable address space,
6676	0x7E	'CALLING' device address,
6677	0x7F	Broadcast address

6678 In the case of two byte addressing:

6679	0x0000	NO_STATION address,
6680	0x0001...0x000F	Reserved for future use,
6681	0x0010...0x3FFD	Usable address space,
6682	0x3FFE	'CALLING' physical device address,
6683	0x3FFF	Broadcast address
6684		
6685		
6686		
6687		

6688 **4.7.3 IEC twisted pair (1) setup (class_id = 24, version = 1)**6689 **4.7.3.1 Overview**

6690 Instances of this IC allow setting up data exchange over the medium *twisted pair with carrier*
6691 *signalling* as specified in IEC 62056-3-1:2013. Several communication channels can be
6692 configured.

6693 The communication medium *twisted pair with carrier signalling* is widely used in metering. The
6694 main advantages of using this medium are the ease of installation and the reliability of
6695 communications due to carrier signalling. This medium can be used:

- 6696 • between Local Network Access Points (LNAPs) and metering end devices (M interface);
- 6697 • between Local Network Access Points (LNAPs) and Neighbourhood Network Access
6698 Points (NNAPs); and
- 6699 • for direct connection between a HHU and the metering end device.

6700 IEC 62056-3-1:2013 specifies three communication profiles using the medium twisted pair with
6701 carrier signalling:

- 6702 • without DLMS;
- 6703 • with DLMS; and
- 6704 • with DLMS/COSEM.

6705 IEC 62056-31:1999 supports only the first two profiles.

6706 The new, DLMS/COSEM profile introduces a Support Manager Layer entity performing the
6707 initialisation of the bus, discovery management, alarm management and communication speed
6708 negotiation. It also allows higher baud rates up to 9 600 Bd. The Transport Layer supports
6709 segmentation and reassembly.

6710 The IC “IEC Twisted pair (1) set up” (class_id = 24, version = 0) supports the first two
6711 communication profiles specified in IEC 62056-31:1999.

6712 The new version 1 supports the DLMS/COSEM profile. With its introduction, the use of version
6713 0 is deprecated.

6714 The use of the communication profiles specified in IEC 62056-3-1:2013 requires using the
6715 registration services provided by the Euridis Association: www.euridis.org.

6716 The following COSEM interface objects are necessary to set up data exchange over the medium
6717 *Twisted pair with carrier signalling*:

- 6718 • “IEC Twisted pair (1) setup”: class_id = 24, version = 1;
- 6719 • “MAC address”: class_id = 43, version = 0;
- 6720 • “Data”: class_id = 1, version = 0.

6721 For OBIS codes, see clause 6.2.21.

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	<i>m/o</i>				

6722

6723 **4.7.3.2 Attribute description**6724 **4.7.3.2.1 logical_name**

6725 Identifies the “IEC twisted pair setup” object instance. See 6.2.21.

6726 **4.7.3.2.2 mode**

6727 This attribute specifies the working mode of this interface

6728 enum:

6729
6730 (0) inactive. The interface ignores all frames received,
6731 (1) always active,
6732 (2)... (127) reserved,
6733 (128)...(250) manufacturer specific.6734 **4.7.3.2.3 comm_speed**

6735 Holds the communication speed supported by the port.

6736 enum:

6737 (2) 1200 baud,
6738 (3) 2400 baud,
6739 (4) 4800 baud,
6740 (5) 9600 baud,
6741 (6) 19200 baud,
6742 (7) 38400 baud

6743 NOTE IEC 62056-3-1:2013 supports baud rates from 1 200 to 9 600.

6744

6745 **4.7.3.2.4 primary_address_list**6746 Holds the list of Primary Station Addresses (ADP) for which each logical device of the real
6747 equipment (the secondary station) has been programmed. See IEC 62056-3-1:2013, 5.2.4.

6748 primary_address_list_type::= array unsigned

6749 **4.7.3.2.5 tabi_list**6750 Represents the list of the TAB(i) for which the real equipment (the secondary station) has been
6751 programmed in the case of forgotten station call (see IEC 62056-3-1:2013).

6752 When using IEC 62056-3-1 profile with DLMS/COSEM, the tabi_list attribute is made of only
 6753 one element of value 0, the value used for the discovery process.

6754 tabi_list_type::= array tabi_element
 6755
 6756 tabi_element: integer
 6757

6758 **4.7.3.3 MAC address**

6759 Secondary stations – typically metering end devices – hold a secondary station address (ADS).
 6760 The length of the ADS shall be 6 octets and consists of the elements shown in Table 36. This
 6761 address shall be worldwide unique and it shall be assigned by the manufacturer to the
 6762 secondary station. The ADS assigned is valid through the lifetime of the secondary station.

6763 **Table 36 – 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 62056-6-1:2021, Table 8.	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.			

6764

6765 **4.7.3.4 Fatal error register**

6766 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
 6767 primary station. The structure of the fatal error register shall be as specified in Table 37.
 6768

6769

Table 37 – 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. No 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.

6770

6771 **4.7.4 Modem configuration (class_id = 27, version = 1)**

6772 **4.7.4.1 Overview**

6773 This IC allows modelling the configuration and initialisation of modems used for data transfer
6774 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					

6775

6776 **4.7.4.2 Attribute description**

6777 **4.7.4.2.1 logical_name**

6778 Identifies the “Modem configuration” object instance. See 6.2.6.

6779 **4.7.4.2.2 comm_speed**

6780 The communication speed between the device and the modem, not necessarily the
6781 communication speed on the WAN.

6782 enum:

6783 300 baud,
6784 600 baud,
6785 1 200 baud,
6786 2 400 baud,
6787 4 800 baud,
6788 9 600 baud,
6789 19 200 baud,
6790 38 400 baud,
6791 57 600 baud,
6792 115 200 baud

6793

6794 **4.7.4.2.3 initialization_string**

6795 Contains all the necessary initialization commands to be sent to the modem in order to configure
 6796 it properly. This may include the configuration of special modem features.

```
6797
6798     array      initialization_string_element
6799
6800         initialization_string_element::= structure
6801         {
6802             request:          octet-string,
6803             response:         octet-string,
6804             delay_after_response: long-unsigned
6805         }
6806
```

6807 If the array contains more than one initialization_string_element, the requests are sent in a
 6808 sequence. The next request is sent after the expected response matching the previous request
 6809 and waiting a delay_after_response time [ms], to allow the modem to execute the request.

6810 NOTE It is assumed that the modem is pre-configured so that it accepts the initialization_string. If no initialization
 6811 is needed, the initialization string is empty.

6812 **4.7.4.2.4 modem_profile**

6813 Defines the mapping from Hayes standard commands/responses to modem specific strings.

```
6814     array      modem_profile_element
6815
6816         modem_profile_element: octet-string
6817
```

6818 The modem_profile array shall contain the corresponding strings for the
 6819 modem used in following order:

```
6820
6821     Element 0:    OK,
6822     Element 1:    CONNECT,
6823     Element 2:    RING,
6824     Element 3:    NO CARRIER,
6825     Element 4:    ERROR,
6826     Element 5:    CONNECT 1 200,
6827     Element 6:    NO DIAL TONE,
6828     Element 7:    BUSY,
6829     Element 8:    NO ANSWER,
6830     Element 9:    CONNECT 600,
6831     Element 10:   CONNECT 2 400,
6832     Element 11:   CONNECT 4 800,
6833     Element 12:   CONNECT 9 600,
6834     Element 13:   CONNECT 14 400,
6835     Element 14:   CONNECT 28 800,
6836     Element 15:   CONNECT 33 600,
6837     Element 16:   CONNECT 56 000
```

6838

6839 **4.7.5 Auto answer (class_id = 28, version = 2)**

6840 NOTE 1 Version 1 of the Auto answer class was an interim version.

6841 Version 0 of the Auto answer class models how the device handles incoming calls to request
 6842 the connection of the modem.

6843 In version 2, new capabilities are added to manage wake-up requests that may be in the form
 6844 of a wake-up call or a wake-up message e.g. an (empty) SMS message. After a successful
 6845 wake-up request, the device connects to the network. See also Annex A.

6846 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
 6847 feature requires the presence of a calling line identification (CLI) service in the communication
 6848 network used.

6850 NOTE 2 The wake-up process is fully decoupled from AL services, i.e. a wake-up message cannot contain any
 6851 xDLMS service requests. This is to avoid creating a backdoor. xDLMS messages may be exchanged in SMS
 6852 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				

6853

6854 **4.7.5.1 Attribute description**

6855 **4.7.5.1.1 logical_name**

6856 Identifies the “Auto answer” object instance. See 6.2.6.

6857 **4.7.5.1.2 mode**

6858 Defines the working mode of the line when the device is auto answering.

6859 enum:

- 6860 (0) line dedicated to the device,
- 6861 (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,
- 6862 (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,
- 6863 (3) currently no modem connected,
 (200...255) manufacturer specific modes

6871 **4.7.5.1.3 listening_window**

6872 Defines the time points when the communication window(s) become active (start_time) and
 6873 inactive (end_time). The start_time implicitly defines the period.

6874 EXAMPLE When the day of month is not specified (equal to 0xFF) this means that we have a daily listening window
 6875 management. Daily, monthly ...window management can be defined.

6876 array window_element
 6877
 6878 window_element::= structure
 6879 {
 6880 start_time: octet-string,
 6881 end_time: octet-string
 6882 }
 6883 start_time and end_time are formatted as specified in 4.1.6.1 for *date-time*.

6884 **4.7.5.1.4 status**

6885 Here the status of the window is defined.

6886 enum:
 6887 (0) Inactive: the device will manage no new incoming call. This status is
 6888 automatically reset to Active when the next listening window starts,
 6889 (1) Active: the device can answer to the next incoming call,
 6890 (2) Locked: This value can be set automatically by the device or by a
 6891 specific client when this client has completed its reading session and
 6892 wants to give the line back to the customer before the end of the window
 6893 duration. This status is automatically reset to Active when the next
 6894 listening window starts.

6895 **4.7.5.1.5 number_of_calls**

6896 This number is the reference used in modes 1 and 2.

6897 When set to 0, this means there is no limit.

6898 **4.7.5.1.6 number_of_rings**

6899 Defines the number of rings before the meter connects the modem. Two cases are distinguished:
 6900 The number of rings within the window defined by the attribute *listening_window* and the number
 6901 of rings outside the *listening_window*.

6902 nr_rings_type::= structure
 6903 {
 6904 nr_rings_in_window: unsigned (0 = no connection in the window),
 6905 nr_rings_out_of_window: unsigned (0 = no connection outside the window)
 6906 }

6907 If the number of rings inside and outside the window is the same, the modem always connects
 6908 regardless of the settings of the *listening_window*.

6909 **4.7.5.1.7 list_of_allowed_callers**

6910 Contains an – optional – list of calling numbers which further limits the connectivity of the
 6911 modem based on the calling number. It also controls the acceptance of wake-up calls or wake-
 6912 up messages (e.g. SMS) from a calling number.

6913 This requires the presence of a calling line identification (CLI) service in the communication
 6914 network used.

6915 list_of_allowed_callers::= array list_of_allowed_callers_element
 6916
 6917 list_of_allowed_callers_element::= structure
 6918 {
 6919 caller_id: octet-string,
 6920 call_type: enum

6921 }

6922

- 6923 – the `caller_id` element holds a calling number from which calls or messages (e.g. SMS) are
6924 accepted. The wild-card characters '?' and '*' are supported. With '?' any single character
6925 matches, with '*' any character string matches. '*' can only be used at the beginning or at
6926 the end of a number, but neither in between nor alone.

6927 Example 1: "+994193500" = only calls from "+994193500" are accepted.

6928 Example 2: "+9941935????" = calls from all numbers in the range of "+99419350000" to "+99419359999" are
6929 accepted.

6930 Example 3: "7777*" = calls from all numbers starting with "7777" are accepted.

6931 Example 4: "**9000" = calls from all numbers ending with "9000" are accepted.

- 6932 – the `call_type` element defines the purpose of the call, i.e. if it's a standard CSD call or a
6933 wake-up call / wake-up message.

6934 enum:

6935 (0) = normal CSD call; the modem only connects if the calling number matches an entry in
6936 the list. This is tested in addition to all other attributes, e.g. `number_of_rings`,
6937 `listening_windows`, etc.

6938 (1) = wake-up request; calls or messages from this calling number are handled as wake-up
6939 requests. The wake-up request is processed immediately regardless of all other
6940 attributes like `number_of_rings` and `listening_window` (except if the calling number is
6941 also present in the list of normal CSD calls, see below).

6942 The received message shall be completely empty; otherwise it is not treated as a wake-up
6943 message.

6944 If the message contains a valid xDLMS APDU from a client in a pre-established AA, the
6945 corresponding xDLMS service is executed instead of processing the wake-up request.

6946 If the message is not empty but does not contain any valid xDLMS APDU from a client
6947 in a pre-established AA then the server does not react.

6948 For a call of type (1) the modem *does not* connect – independently of the outcome of
6949 the wake-up process.

6950 For a call of both type (1) and type (0): see below.

6951 If the same calling number is defined as initiator of a normal CSD call (type (0)) and as
6952 initiator of a wake-up request (type (1)) the following rule applies for the incoming calls:
6953 the wake-up request is only processed if the caller disconnects the line before the
6954 `number_of_rings` criterion (depending on `listening_window`) is reached. If the
6955 `number_of_rings` criterion is met then a modem connection is established.

6956 The `number_of_rings` parameter shall be set large enough to allow the initiator of the
6957 call to control the behaviour of the receiver of the call without any knowledge of the time
6958 instances of the rings at the receiver's side.

6959 NOTE 3 If the `list_of_allowed_callers` is empty (= array [0]) the auto answer function operates in type (0)
6960 = normal CSD call and the modem connects independently of the calling number.

6961

6962

6963

6964

6965 **4.7.6 Auto connect (class_id = 29, version = 2)**6966 **4.7.6.1 Overview**

6967 Version 1 of the “Auto connect” class models how the device performs auto dialling or sends
 6968 messages using various services.

6969 In version 2 new capabilities are added to model the connection of the device to a
 6970 communication network. Network connection may be permanent, within a time window or on
 6971 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				

6972

6973 **4.7.6.2 Attribute description**6974 **4.7.6.2.1 logical_name**

6975 Identifies the “Auto connect” object instance. See 6.2.6.

6976 **4.7.6.2.2 mode**

6977 Controls the auto connect functionality in terms of the timing, the message type and the
 6978 infrastructure to be used.

6979 Modes (1) to (3) are dedicated to CSD services.

6980 Modes (4) to (6) are dedicated to sending specific messages using a specific infrastructure.

6981 Modes (101) to (104) apply to packet switched network connections only (e.g. GPRS).

6982 enum:

- 6983 (0) no auto connect; the device never connects,
- 6984 (1) auto dialling allowed anytime, the values defined in the *calling_window* are ignored,
- 6985 (2) auto dialling allowed within the validity time of the *calling_window*,
- 6986 (3) “regular” auto dialling allowed within the validity time of the *calling_window*; “alarm”
 6987 initiated auto dialling allowed anytime,
- 6988 (4) SMS sending via Public Land Mobile Network (PLMN),
- 6989 (5) SMS sending via PSTN,
- 6990 (6) email sending,
- 6991 (7...99)reserved,
- 6992 (101) the device is permanently connected to the communication network,
- 6993 (102) the device is permanently connected to the communication network within the
 6994 validity time of the calling window. The device is disconnected outside the calling
 6995 window. No connection possible outside the calling window,

- 6996 (103) the device is permanently connected to the communication network within the
 6997 validity time of the calling window. The device is disconnected outside the calling
 6998 window but it connects to the communication network as soon as the connect method
 6999 is invoked,
 7000 (104) the device is usually disconnected. It connects to the communication network as
 7001 soon as the connect method is invoked,
 7002 (105...199) reserved,
 7003 (200...255) manufacturer specific modes.

7004 **4.7.6.2.3 repetitions**

7005 The maximum number of retries in case of unsuccessful connection attempts.

7006 **4.7.6.2.4 repetition_delay**

7007 The time delay, expressed in seconds until an unsuccessful connection attempt can be repeated.

7008 repetition_delay = 0 means delay is not specified

7009 **4.7.6.2.5 calling_window**

7010 Contains the time points when the window becomes active (start_time), and inactive (end_time).
 7011 The start_time implicitly defines the period.

7012 EXAMPLE When the day of month is not specified (equal to 0xFF) this means that the calling window is managed
 7013 on a daily basis. Daily, monthly ...window management can be defined.

```
7014           array      window_element
7015
7016           window_element::= structure
7017           {
7018             start_time: octet-string,
7019             end_time: octet-string
7020           }
```

7021 start_time and end_time are formatted as specified in 4.1.6.1 for *date-time*.

7022 **4.7.6.2.6 destination_list**

7023 Contains the list of destinations (for example phone numbers, email addresses or their
 7024 combinations) where the message(s) have to be sent under certain conditions. The conditions
 7025 and their link to the elements of the array are not defined here.

```
7026           array      destination
7027
7028           destination::= octet-string
7029
```

7030 **4.7.6.3 Method**

7031 **4.7.6.3.1 connect (data)**

7032 Initiates the connection process to the communication network according to the rules defined
 7033 via the mode attribute.

7034 data ::= integer (0)

7035

7036

7037 **4.7.7 GPRS modem setup (class_id = 45, version = 0)**7038 **4.7.7.1 Overview**

7039 This IC allows setting up GPRS modems, by handling all data necessary data for modem
 7040 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>				

7041

7042 **4.7.7.2 Attribute description**7043 **4.7.7.2.1 logical_name**

7044 Identifies the “GPRS modem setup” object instance. See 6.2.23.

7045 **4.7.7.2.2 APN**

7046 Defines the access point name of the network.

7047 **4.7.7.2.3 PIN_code**

7048 Holds the personal identification number.

7049 **4.7.7.2.4 quality_of_service**

7050 Specifies the quality of service parameters. It is a structure of 2 elements:

- 7051 – the first element defines the default or minimum characteristics of the network concerned.
 7052 These parameters have to be set to best effort value;
- 7053 – the second element defines the requested parameters.

```

7054           quality_of_service ::= structure
7055
7056           {
7057             default:          qos_element,
7058             requested:       qos_element
7059           }
7060           qos_element ::= structure
7061           {
7062             precedence:     unsigned,
7063             delay:          unsigned,
7064             reliability:   unsigned,
7065             peak throughput: unsigned,
7066             mean throughput: unsigned
7067           }

```

7068

7069 **4.7.8 GSM diagnostic (class_id: 47, version: 2)**

7070 **4.7.8.1 Overview**

7071 The cellular network is undergoing constant changes in terms of registration status, signal
 7072 quality, etc. It is necessary to monitor and log the relevant parameters in order to obtain
 7073 diagnostic information that allows identifying communication problems in the network.

7074 An instance of the “GSM diagnostic” class stores parameters of the GSM/GPRS, UMTS, CDMA
 7075 or LTE network necessary for analysing the operation of the network.

7076 A GSM diagnostic “Profile generic” object is also available to capture the attributes of the GSM
 7077 diagnostic object, see IEC 62056-6-1:2021, 6.5.

GSM diagnostic	0...n	class_id = 47, version = 2			
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				

7078

7079 **4.7.8.2 Attribute description**

7080 **4.7.8.2.1 logical_name**

7081 Identifies the “GSM Diagnostic” object instance. For logical names, see 6.2.23.

7082 **4.7.8.2.2 operator**

7083 Holds the name of the network operator e.g. “YourNetOp”

7084 **4.7.8.2.3 status**

7085 Indicates the registration status of the modem.

7086 enum:
 7087 (0) not registered,
 7088 (1) registered, home network,
 7089 (2) not registered, but MT is currently searching a new operator to
 7090 register to,
 7091 (3) registration denied,
 7092 (4) unknown,
 7093 (5) registered, roaming,
 7094 (6)... (255) reserved

7095

7096

7097 **4.7.8.2.4 cs_attachment**

7098 Indicates the current circuit switched status.

7099 enum:
 7100 (0) inactive,
 7101 (1) incoming call,
 7102 (2) active,
 7103 (3)...(255) reserved

7104 **4.7.8.2.5 ps_status**

7105 The *ps_status* value field indicates the packet switched status of the modem.

7106 enum:
 7107 (0) inactive,
 7108 (1) GPRS,
 7109 (2) EDGE,
 7110 (3) UMTS,
 7111 (4) HSDPA,
 7112 (5) LTE,
 7113 (6) CDMA,
 7114 (7) LTE Cat M1,
 7115 (8) LTE Cat NB1,
 7116 (9) LTE Cat NB2,
 7117 (10)...(255) reserved
 7118

7119 **4.7.8.2.6 cell_info**

7120 Represents the cell information:

7121 cell_info_type::= structure
 7122 {
 7123 cell_ID: double-long-unsigned,
 7124 location_ID: long-unsigned,
 7125 signal_quality: unsigned,
 7126 ber: unsigned,
 7127 mcc: long-unsigned,
 7128 mnc: long-unsigned,
 7129 channel_number: double-long-unsigned
 7130 }

7131 Where:

- 7132 – cell_ID: Four-byte cell ID in hexadecimal format;
- 7133 – location_ID: Two-byte location area code (LAC) in the case of GSM networks or Tracking Area Code (TAC) in the case of UMTS, CDMA or LTE networks in hexadecimal format (e.g. "00C3" equals 195 in decimal);
- 7136 – signal_quality: Represents the signal quality:

7137 (0) –113 dBm or less,
 7138 (1) –111 dBm,
 7139 (2...30) –109...–53 dBm,
 7140 (31) –51 or greater,
 7141 (99) not known or not detectable;

- 7143 – ber: Bit Error Rate (BER) measurement in percent:
 - (0...7) as RXQUAL_n values specified in ETSI GSM 05.08:1996, 8.2.4
 - (99) not known or not detectable.
- 7147 – mcc: Mobile Country Code of the serving network, as defined in ITU-T E.212 (05.2008);
- 7148 – mnc: Mobile Network Code of the serving network, as defined in ITU-T E.212 (05.2008);

- 7149 – channel_number: Represents the absolute radio-frequency channel number (ARFCN or
 7150 EARFCN for LTE network).

7151 **4.7.8.2.7 adjacent_cells**

```
7152     array      adjacent_cell_info
7153
7154     adjacent_cell_info::= structure
7155     {
7156         cell_ID:      double-long-unsigned,
7157         signal_quality: unsigned
7158     }
```

7159 Where:

- 7160 – cell_ID: Four-byte cell ID in hexadecimal format;
 7161 – signal_quality: Represents the signal quality:

```
7162     (0)      -113 dBm or less,
7163     (1)      -111 dBm,
7164     (2...30)  -109...-53 dBm,
7165     (31)      -51 or greater,
7166     (99)      not known or not detectable.
```

7168 **4.7.8.2.8 capture_time**

7169 Holds the date and time when the data have been last captured.

7170 *date-time* is formatted as specified in 4.1.6.1.

7171

7172 **4.7.9 LTE monitoring (class_id: 151, version: 1)**

7173 **4.7.9.1 Overview**

7174 Instances of the ‘LTE monitoring’ IC allow monitoring LTE modems by handling all data
 7175 necessary data for this purpose.

7176 Version 1 of the LTE monitoring IC also covers LTE Cat M1 and NB-IoT networks.

LTE monitoring	0...n	class_id = 151, version = 1			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. LTE_network_parameters (dyn.)	LTE_network_parameters_type				x + 0x08
3. LTE_quality_of_service (dyn)	LTE_QoS_type				x + 0x10
Specific methods	m/o				

7177

7178

7179 **4.7.9.2 Attribute description**

7180 **4.7.9.2.1 logical_name**

7181 Identifies the “LTE monitoring” object instance. See 6.2.23.

7182 **4.7.9.2.2 LTE_network_parameters**

7183 Represents the network parameters for the LTE network

7184 **LTE_network_parameters_type ::= structure**

```

7185                   {
7186                    T3402:               long-unsigned,
7187                    T3412:               long-unsigned,
7188                    T3412ext2:        double-long-unsigned,
7189                    T3324:               long-unsigned,
7190                    TeDRX:              double-long unsigned,
7191                    TPTW:                long-unsigned,
7192                    qRxlevMin:         integer,
7193                    qRxlevMinCE-r13: integer,
7194                    qRxlevMinCE1-r13 integer
7195                   }

```

7196 **Where:**

7199 **T3402** timer in seconds, used on PLMN selection procedure and sent by the
7200 network to the modem. Refer to 3GPP TS 24.301 V13.11.0 (2018-01) for
7201 details.

7202 **T3412** timer in seconds, used to manage the periodic tracking area updating
7203 procedure and sent by the network to the modem. Refer to 3GPP TS
7204 24.301 V13.11.0 (2018-01) for details.

7205 **T3412ext2** timer in seconds (extended periodic tracking area update timer). Refer to
7206 3GPP TS 24.301 V13.11.0 (2018-01) and 3GPP TS 24.008 V13.7.0
7207 (2016-10) for details.

7208 **T3324** timer in seconds (Power saving mode active timer). Refer to 3GPP TS
7209 24.301 V13.11.0 (2018-01) and 3GPP TS 24.008 V13.7.0 (2016-10) for
7210 details.

7211 **TeDRX** timer (Extended Idle mode DRX cycle timer). Refer to 3GPP TS 24.301
7212 V13.11.0 (2018-01) and 3GPP TS 24.008 V13.7.0 (2016-10) for details.

7213 The double-long unsigned value shall be multiplied by 0,01 to get the real
7214 value in seconds. E.g. 512 represents 5,12 seconds

7215 **TPTW** timer (Extended Idle mode DRX paging time window). Refer to 3GPP TS
7216 24.301 V13.11.0 (2018-01) and 3GPP TS 24.008 V13.7.0 (2016-10) for
7217 details.

7218 The long unsigned value shall be multiplied by 0,01 to get the real value
7219 in seconds. E.g. 512 represents 5,12 seconds

7220 **qRxlevMin** the minimum required Rx level in the cell in dBm as defined in 3GPP TS
7221 36.304 V13.8.0 (2018-01).

7222 **qRxlevMinCE-r13** the minimum required Rx level in enhanced coverage CE Mode A, LTE
7223 Cat M1. For this mode a value from -70...-22 in steps of 1 is required.

7224 NOTE 1 This field is not used in case of LTE Cat NB1 or NB2.

7225 qRxlevMinCE1-r13 the minimum required Rx level in enhanced coverage CE Mode B, LTE
 7226 Cat M1. For this mode a value from -78...-22 in steps of 1.

7227 NOTE 2: This field is not used in case of LTE Cat NB1 or NB2.

7228 4.7.9.2.3 LTE_quality_of_service

7229 Represents the quality of service of the LTE network

```
7230     LTE_QoS_type ::= structure
 7231     {
 7232       (N)RSRQ:           integer,
 7233       (N)RSRP:           integer,
 7234       SNR:              integer,
 7235       Coverage Enhancement: enum
 7236     }
```

7237 Where:

7239 – (N)RSRQ represents the signal quality as defined in 3GPP TS 36.133. The use of this
 7240 parameter can be determined from the GSM diagnostic IC, ps_status attribute:

- 7241 – For LTE Cat M1, a value range from -30 up to 46 is necessary to represent RSRQ.
 7242 Refer to 3GPP TS 36.133 V13.11.0 (2018-04) for details.
- 7243 – For LTE Cat NB1 and LTE Cat NB2 a value range from -30 up to 46 is necessary to
 7244 represent NRSRQ. Refer to 3GPP TS 36.133 V14.4.0 (2017-07) for details.

(-30)	-34 dB,	7246
(-29)	-33,5 dB,	
(-28)..(-1)	-33 dB .. -19,5 dB,	7247
(0)	-19,5 dB or less,	
(1)	-19 dB,	7248
(2)..(31)	-18,5 dB .. -4 dB,	
(32)	-3,5 dB,	7249
(33)	-3 dB,	7250
(34)	-3 dB or less,	7251
(35)	-2,5 dB,	7252
(36)	-2 dB,	7253
(37)..(45)	-1,5 dB .. +2,5 dB,	
(46)	-2,5 dB or better,	7254
(99)	not known or not detectable,	7255
other values	reserved	7256
		7257

7258 – (N)RSRP represents the signal level as defined in 3GPP TS 36.133. The use of this
 7259 parameter is determined from the GSM diagnostic IC, ps_status attribute:

- 7261 – For ps_status = 7 (LTE Cat M1) a value range from -17 up to 97 is necessary to
 7262 represent RSRP. Refer to 3GPP TS 36.133 V13.11.0 (2018-04) for details

(-17)	-156 dBm,
(-16)	-155 dBm,
(-15)..(-1)	-154 dBm .. -140 dBm,
(0)	-140 dBm or less,

(1)	-139 dB,	7264
(2)	-138 dBm,	
(3)..(96)	-137 dBm .. -44 dBm,	7265
(97)	-44 dBm or better,	
(127)	not known or not detectable,	7266
other values	reserved	7267

7268 NOTE 3 depending on the use of Coverage Enhancement modes, a reported value of (0) will be used or not (for
 7269 devices that do not use CE modes, values (-17..-1) will not be used. A Reported value of (0) means in that case a
 7270 RSRP < -140 dBm).

7271

7272 – For ps_status = 8 or 9 (LTE Cat NB1 or LTE Cat NB2) a value range from 0 to 113 is
 7273 necessary to represent NRSRP value. Refer to 3GPP TS 36.133 V14.4.0 (2017-07) for
 7274 details.

7275

(0)	-156 dBm,	7276
(1)	-155 dBm,	7277
(2)	-154 dBm	
(3)..(112)	-153 dBm .. -44 dBm,	7278
(113)	-44 dBm or better,	
(127)	not known or not detectable,	7279
other values	reserved	7280

7281 NOTE 4 For the NRSRP value (NB-IoT) the mapping is different from the mapping for the RSRP value
 7282 (LTE Cat M1).

7283

- 7284 – SNR the Signal to Noise Ratio in a range from -20 dB to 50 dB. Refer to 3GPP TS 36.101
 7285 V15.4.0 (2019-01) for details.
- 7286 – Coverage Enhancement Coverage Enhancement Mode A or B in case of LTE Cat M1, CE
 7287 Level 0,1 or 2 in case of LTE Cat NB1/NB2. Refer to 3GPP TS 36.331 V15.5.1 (2019-05),
 7288 3GPP TS 36.321 V15.5.0 (2019-05) and 3GPP TS 36.213 V15.5.0 (2019-05) for details.

Enum	LTE Cat M1	NB-IoT	7289
(0)	CE Mode A	CE Level 0	
(1)	CE Mode B	CE Level 1	7290
(2)	(not used)	CE Level 2	
other values	reserved		7291

7292

7293

7294 **4.8 Interface classes for setting up data exchange via M-Bus**

7295 **4.8.1 Overview**

7296 The M-Bus related interface classes specified in this subclause 5.7 are used in two different
7297 scenarios:

- 7298 a) a DLMS/COSEM server hosted by a M-Bus master and exchanging dedicated M-Bus APDUs
7299 with M-Bus slaves;
- 7300 h) a DLMS/COSEM client hosted by a M-Bus master and exchanging DLMS/COSEM APDUs
7301 with DLMS/COSEM servers hosted by M-Bus slaves;

7302 In case a) instances of the following M-Bus interface classes are used to set up and manage
7303 the M-Bus media in the DLMS/COSEM server:

- 7304 • M-Bus client (class_id = 72), see 4.8.3;
- 7305 • M-Bus master port setup (class_id = 74), see 4.8.5;
- 7306 • M-Bus diagnostic (class_id = 77, version = 0), see 4.8.7.

7307 In case b) instances of the following M-Bus interface classes are used in the DLMS/COSEM
7308 server:

- 7309 • DLMS/COSEM server M-Bus port setup (class_id = 76), see 4.8.6;
- 7310 • M-Bus slave port setup (class_id = 25), see 4.8.2; and/or
- 7311 • Wireless Mode Q channel (class_id = 73), see 4.8.4;
- 7312 • M-Bus diagnostic (class_id = 77, version = 0), see 4.8.7.

7313

7314 **4.8.2 M-Bus slave port setup (class_id = 25, version = 0)**

7315 **4.8.2.1 Overview**

7316 NOTE 1 The name of this IC has been changed from “M-BUS port setup” to “M-Bus slave port setup”, to indicate
7317 that it serves to set up data exchange when a COSEM server communicates with a COSEM client using wired M-
7318 Bus.

7319 This IC allows modelling and configuring communication channels according to EN 13757-
7320 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					

7321

7322 **4.8.2.2 Attribute description**

7323 **4.8.2.2.1 logical_name**

7324 Identifies the “M-Bus slave port setup” object instance. See 6.2.22.

7325

7326 **4.8.2.2.2 default_baud**

7327 Defines the baud rate for the opening sequence.

7328 enum:

- 7329 (0) 300 baud,
- 7330 (3) 2 400 baud,
- 7331 (5) 9 600 baud

7332

7333 **4.8.2.2.3 avail_baud**

7334 Defines the baud rates that can be negotiated after startup.

7335 enum:

- 7336 (0) 300 baud,
- 7337 (1) 600 baud,
- 7338 (2) 1 200 baud,
- 7339 (3) 2 400 baud,
- 7340 (4) 4 800 baud,
- 7341 (5) 9 600 baud,
- 7342 (6) 19 200 baud,
- 7343 (7) 38 400 baud

7344

7345 **4.8.2.2.4 addr_state**

7346 Defines whether or not the device has been assigned an address since last power up of the device.

7347 enum:

- 7348 (0) Not assigned an address yet,
- 7349 (1) Assigned an address either by manual setting, or by automated method.

7350 **4.8.2.2.5 bus_address**

7351 The currently assigned address on the bus for the device.

7352 NOTE 2 If no bus address is assigned, the value is 0.

7353

7354 **4.8.3 M-Bus client (class_id = 72, version = 1)**7355 **4.8.3.1 Overview**

7356 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.

7359 NOTE 1 Version 1 of the “M-Bus client” IC is in line with EN 13757-3:2013.

7360 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 4.8.5.

7362 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.

7366 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

7368 are captured periodically or on an appropriate trigger. Each data element is stored in an M-Bus
 7369 value object, of IC “Extended register”. M-Bus value objects may be captured in M-Bus “Profile
 7370 generic” objects, eventually along with other, non M-Bus specific objects.

7371 Using the methods of “M-Bus client” objects, M-Bus slave devices can be installed and de-
 7372 installed.

7373 It is also possible to send data to M-Bus slave devices and to perform operations like resetting
 7374 alarms, synchronizing the clock, transferring an encryption key, etc.

7375 Configuration field as defined in EN 13757-3:2013, 5.12 provides information about the
 7376 encryption mode and number of encrypted bytes.

7377 Encryption key status provides information if encryption key has been set, transferred to M-Bus
 7378 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

7379

7380 **4.8.3.2 Attribute description**

7381 **4.8.3.2.1 logical_name**

7382 Identifies the “M-Bus client” object instance. See 6.2.22.

7383 **4.8.3.2.2 mbus_port_reference**

7384 Provides reference to an “M-Bus master port setup” object, used to configure an M-Bus port,
 7385 each interface allowing to exchange data with one or more M-Bus slave devices.

7386 **4.8.3.2.3 capture_definition**

7387 Provides the *capture_definition* for M-Bus slave devices.

7388 NOTE 2 This attribute can be pre-configured or written as part of the installation procedure.

```
7389     array      capture_definition_element
7390
7391     capture_definition_element::= structure
7392     {
7393         data_information_block:    octet-string,
7394         value_information_block:   octet-string
7395     }
```

7396 NOTE 3 The elements *data_information_block* and *value_information_block* correspond to Data Information Block
 7397 (DIB) and Value Information Block (VIB) described in EN 13757-3:2013, 6.2 and Clause 7 respectively.

7398 **4.8.3.2.4 capture_period**

7399 >= 1: Automatic capturing assumed. Specifies the capture period in seconds.

7400 0: No automatic capturing: capturing is triggered externally or capture events occur
 7401 asynchronously.

7402 **4.8.3.2.5 primary_address**

7403 Carries the primary address of the M-Bus slave device. The range is 0...250.

7404 Each M-bus device is bound to a channel of the M-Bus master. However, there is no direct link
 7405 between the primary address and the channel number.

7406 NOTE 4 The specification of the B field of the OBIS codes limits the range to 1...64 within one logical device. See
 7407 IEC 62056-6-1:**2021**, 5.2.

7408 If the slave device is already configured and thus, its primary address is different from 0, then
 7409 this value shall be written to the *primary_address* attribute. From this moment, the data
 7410 exchange with the M-Bus slave device is possible.

7411 Otherwise, the *slave_install* method shall be used; see 4.8.3.3.1 below.

7412 NOTE 5 The *primary_address* attribute cannot be used to store a desired primary address for an unconfigured slave
 7413 device. If the *primary_address* attribute is set, this means that the M-Bus client can immediately operate with this
 7414 primary address, which is not the case with an unconfigured slave device.

7415 **4.8.3.2.6 identification_number**

7416 Carries the Identification Number element of the data header as specified in EN 13757-3:2013,
 7417 5.5.

7418 This attribute, together with attributes 7, 8 and 9 are filled with the values found in the first
 7419 message received after installation.

7420 If in subsequent messages these values are not the same, the message is discarded.

7421 **4.8.3.2.7 manufacturer_id**

7422 Carries the Manufacturer Identification element of the data header as specified in EN 13757-
7423 3:2013, 5.6.

7424 **4.8.3.2.8 version**

7425 Carries the Version element of the data header as specified in
7426 EN 13757-3:2013, 5.7.

7427 **4.8.3.2.9 device_type**

7428 Carries the Device type identification element of the data header as specified in EN 13757-
7429 3:2013, 5.8, Table 6.

7430 **4.8.3.2.10 access_number**

7431 Carries the Access Number element of the data header as specified in
7432 EN 13757-3:2013, 5.9.

7433 **4.8.3.2.11 status**

7434 Carries the Status byte element of the data header as specified in
7435 EN 13757-3:2013, 5.10, Tables 7 and 8.

7436 It is updated with every readout of the M-Bus slave device.

7437 **4.8.3.2.12 alarm**

7438 Carries the Alarm state specified in EN 13757-3:2013, Annex D.

7439 It is updated with every readout of the M-Bus slave device.

7440 **4.8.3.2.13 configuration**

7441 Carries the Configuration field (previously: Signature field) as specified in EN 13757-3:2013,
7442 5.12. It contains information about the encryption mode and the number of encrypted bytes.

7443 It is updated with every readout of the M-Bus slave device.

7444 **4.8.3.2.14 encryption_key_status**

7445 Provides information on the status of the encryption key. See also Annex B.

7446 enum:

- 7447 (0) no encryption key,
- 7448 (1) encryption_key set,
- 7449 (2) encryption_key transferred,
- 7450 (3) encryption_key set and transferred,
- 7451 (4) encryption_key in use.

7452 **4.8.3.3 Method description**

7453 **4.8.3.3.1 slave_install (data)**

7454 Installs a slave device, which is yet unconfigured (its primary address is 0).

7455 data ::= unsigned

7456 This method can be successfully invoked only if the current value of the *primary_address*
7457 attribute is 0. The following actions are performed:

- 7458 – the M-Bus address 0 is checked for presence of a new device;
- 7459 – if no uninstalled M-Bus slave is found, the method invocation fails;
- 7460 – if the *slave_install* method is invoked with “0” as a parameter, then the primary address is
7461 assigned automatically. This is done by checking the *primary_address* attribute of all M-Bus
7462 client objects in the DLMS/COSEM device and then selecting the first unused number. The
7463 *primary_address* attribute is set to this address and it is then transferred to the M-Bus slave
7464 device;
- 7465 – if the *slave_install* method is invoked with a primary address (other than 0) as a parameter,
7466 then the *primary_address* attribute is set to this value and it is then transferred to the M-Bus
7467 slave device.

7468 NOTE 6 Unconfigured slave devices are configured with primary address as specified in EN 13757-3:2013, Annex
7469 E.5.

7470 **4.8.3.3.2 *slave_deinstall* (data)**

7471 De-installs the slave device. The main purpose of this service is to de-install the M-Bus slave
7472 device and to prepare the master for the installation of a new device. The following actions are
7473 performed:

- 7474 – the M-Bus address is set to 0 in the M-Bus slave device;
- 7475 – the encryption key transferred previously to the M-Bus slave device is destroyed; the default
7476 key is not affected;
- 7477 – the *encryption_key_status* is set to (0): no encryption_key;
- 7478 – the attribute *primary_address* is also set to 0.

7479 NOTE 7 A new M-Bus slave can be installed only once the value of the *primary_address* attribute is 0.

7480 data::= unsigned (0)

7481 **4.8.3.3.3 *capture* (data)**

7482 Captures values – as specified by the *capture_definition* attribute – from the M-Bus slave device.

7483 data::= integer (0)

7484 **4.8.3.3.4 *reset_alarm* (data)**

7485 Resets alarm state of the M-Bus slave device.

7486 data::= integer (0)

7487 **4.8.3.3.5 *synchronize_clock* (data)**

7488 Synchronizes the clock of the M-Bus slave device with that of the M-Bus client device.

7489 data::= integer (0)

7490

7491

7492 4.8.4 Wireless Mode Q channel (class_id = 73, version = 1)

7493 4.8.4.1 Overview

7494 Instances of this IC define the operational parameters for communication using the mode Q
7495 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	<i>m/o</i>				

7496

7497 4.8.4.2 Attribute description

7498 4.8.4.2.1 logical_name

7499 Identifies the “Wireless Mode Q channel” object instance. See 6.2.22.

7500 4.8.4.2.2 addr state

7501 Defines whether or not the device has been assigned an address since last power up of the
7502 device.

7503 enum:

7504

7505 (0) not assigned an address yet,
7506 (1) assigned an address either by manual setting or by automated
7507 method.

7508 4.8.4.2.3 device_address

7509 The currently assigned address of the device on the network.

7510 4.8.4.2.4 address_mask

7511 The group address the device will respond to when short form addressing is used.

7512

7513 4.8.5 M-Bus master port setup (class_id = 74, version = 0)

7514 **4.8.5.1** Overview

7515 Instances of this IC define the operational parameters for communication using the EN 13757-2
7516 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

<i>Specific methods</i>	<i>m/o</i>		
-------------------------	------------	--	--

7517

7518 **4.8.5.2 Attribute description**7519 **4.8.5.2.1 logical_name**

7520 Identifies the “M-Bus master port setup” object instance. See 6.2.22.

7521 **4.8.5.2.2 comm_speed**

7522 The communication speed supported by the port

7523 enum:

- 7524 (0) 300 baud,
- 7525 (1) 600 baud,
- 7526 (2) 1 200 baud,
- 7527 (3) 2 400 baud,
- 7528 (4) 4 800 baud,
- 7529 (5) 9 600 baud,
- 7530 (6) 19 200 baud,
- 7531 (7) 38 400 baud

7532

7533

7534

7535 **4.8.6 DLMS/COSEM server M-Bus port setup (class_id = 76, version = 0)**7536 **4.8.6.1 Overview**7537 Instances of the “DLMS/COSEM server M-Bus port setup” are used in DLMS/COSEM servers hosted
7538 by M-Bus slave devices, using the DLMS/COSEM wired or wireless M-Bus (wM-Bus) communication
7539 profile.

DLMS/COSEM server M-Bus port setup		0...n	class_id = 76, version = 0			
Attributes		Data type	Min.	Max.	Def.	Short name
1.	logical_name (static)	octet-string				x
2.	M-Bus_profile_selection (static)	octet-string				x + 0x08
3.	M-Bus_port_communication_state (dyn.)	enum				x + 0x10
4.	M-Bus_Data_Header_Type (dyn.)	enum			2	x + 0x18
5.	primary_address (static)	unsigned			0	x + 0x20
6.	identification_number (static)	double-long-unsigned			0	x + 0x28
7.	manufacturer_id (static)	long-unsigned			0	x + 0x30
8.	version (static)	unsigned			0	x + 0x38
9.	device_type (static)	unsigned			0	x + 0x40
10.	max_pdu_size (static)	long-unsigned				x + 0x48
11.	listening_window (static)	array				x + 0x50
Specific methods		m/o				

7540

7541 **4.8.6.2 Attribute description**

7542 **4.8.6.2.1 logical_name**

7543 Identifies the “DLMS/COSEM server M-Bus port setup” object instance. See 6.2.22.

7544 **4.8.6.2.2 M-Bus_profile_selection**

7545 References an M-Bus communication port setup object describing the physical capabilities for
7546 wired or wireless communication. The referenced object is either an “M-Bus slave port setup”
7547 object (class_id = 25) or a “Wireless Mode Q channel” object (class_id = 73).

7548 **4.8.6.2.3 M-Bus_port_communication_state**

7549 This attribute is relevant only in wM-Bus.

7550 Carries the communication status of the M-Bus node.

7551 See EN 13757-4:2013, 11.6.3, Table 27.

7552 enum:

7553 (0) No access: Meter provides no access windows (unidirectional meter),

7554 (1) Temporary no access: Meter supports bidirectional access in general, but there is no
7555 access window after this transmission (e.g. temporary no access in order to keep duty
7556 cycle limits or to limit energy consumption),

7557 (2) Limited access: Meter provides a short access windows only immediately after this
7558 transmission (e.g. battery operated meter),

7559

7560

7561

7562
7563 (3) Unlimited access: Meter provides unlimited access at least until next transmission (e.g.
7564 mains powered devices)
7565

7566 **4.8.6.2.4 M-Bus_Data_Header_Type**

7567 Carries the type of the M-Bus Data Header, derived from the CITL value from the current
7568 communication.

7569 In the case of a push operation the default value (2) shall be applied.

7570 enum:

7571 (0) M-Bus_Data_Header_Type == None_M-Bus_Data_Header, the value of the CITL field
7572 shall be 0x10 when segmentation is not used, and shall be 0x00 .. 0x1F when
7573 segmentation is used (with the FIN bit set to 0 in all segments except in the last segment);

7574 (1) M-Bus_Data_Header_Type == Short_M-Bus_Data_Header, the value of the CITL field
7575 shall be 0x61 in an M-Bus frame sent by a master and 0x7D in a an M-Bus frame sent by
7576 a slave;

7577 (2) M-Bus_Data_Header_Type == Long_M-Bus_Data_Header, the value of the CITL field
7578 shall be 0x60 in an M-Bus frame sent by a master and 0x7C in a an M-Bus frame sent by
7579 a slave.

7580 **4.8.6.2.5 primary_address**

7581 Carries the primary address of the M-Bus slave device.

7582 See IEC 62056-7-3:2017, Table 1.

7583 If the slave device is already configured and thus, its primary address is different from 0, then
7584 the data exchange with the M-Bus slave device is possible.

7585 **4.8.6.2.6 identification_number**

7586 Carries the Identification Number element of the Data Header as specified in EN 13757-3:2013,
7587 5.5.

7588 In the case of unidirectional communication this value – together with attributes 6, 7, 8 and 9 –
7589 shall be parametrized.

7590 In the case of bi-directional communication this attribute – together with attributes 6, 7, 8 and
7591 9 – are filled with the values found in the first message received after installation by different
7592 M-Bus network management interfaces.

7593 NOTE If in subsequent messages these values are not the same, the message is discarded.

7594 **4.8.6.2.7 manufacturer_id**

7595 Carries the Manufacturer Identification element of the Data Header as specified in EN 13757-
7596 3:2013, 5.6.

7597

7598 **4.8.6.2.8 version**

7599 Carries the Version element of the Data Header as specified in EN 13757-3:2013, 5.7.

7600 **4.8.6.2.9 device_type**

7601 Carries the Device type identification element of the Data Header as specified in EN 13757-
7602 3:2013, 5.8, Table 6.

7603 **4.8.6.2.10 max_pdu_size**

7604 Contains length capability available from M-Bus lower layers (expressed in bytes).

7605 Specifies the maximum length of the DLMS payload (an APDU or a part of it) that can be carried
7606 by one M-Bus frame.

7607 In the case of long messages, either block transfer provided by the DLMS/COSEM application
7608 layer or segmentation provided by the transport layer or both mechanisms can be used.

7609 **4.8.6.2.11 listening_window**

7610 This attribute is relevant only in wM-Bus.

7611 Defines the time points when the point-to-point communication window(s) become active
7612 (start_time) and inactive (end_time). The start_time implicitly defines the period.

7613 EXAMPLE When the day of month is not specified (equal to 0xFF) this means that we have a daily listening window
7614 management. Daily, monthly, etc., window management can be defined.

```
7615         array window_element
7616
7617             window_element::= structure
7618
7619                 {
7620                     start_time: octet-string,
7621                     end_time: octet-string
7622                 }
7623
```

7624 start_time and end_time are formatted as specified in 4.6.1 for *date-time*.

7625

7626 **4.8.7 M-Bus diagnostic (class_id = 77, version = 0)**7627 **4.8.7.1 Overview**

7628 Instances of the IC “M-Bus diagnostic” hold information related to the operation of the M-Bus network,
7629 like current signal strength, channel identifier, link status to the M-Bus network and counters related to
7630 the frame exchange, transmission and frame reception quality.

M-Bus diagnostic	0...n	class_id = 77, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				X
2. received-signal-strength (dyn.)	unsigned				x + 0x08
3. channel_Id (dyn.)	unsigned			0	x + 0x10
4. link_status (dyn.)	enum				x + 0x18
5. broadcast_frames_counter (dyn.)	array			0	x + 0x20
6. transmissions_counter (dyn.)	double-long-unsigned			0	x + 0x28
7. FCS_OK_frames_counter (dyn.)	double-long-unsigned			0	x + 0x30
8. FCS_NOK_frames_counter (dyn.)	double-long-unsigned			0	x + 0x38
9. capture_time (dyn.)	structure				x + 0x40
Specific methods	m/o				
1. reset (data)	o				x + 0x48

7631

7632 4.8.7.2 Attribute description

7633 4.8.7.2.1 logical_name

7634 Identifies the “M-Bus diagnostic” object instance. See 6.2.22.

7635 4.8.7.2.2 received_signal_strength

7636 This attribute is relevant only for wireless bidirectional M-Bus communication.

7637 When the wM-Bus profile is used, this attribute holds the value of signal strength of the last
7638 wM-Bus frame received, expressed in dBm.

7639 4.8.7.2.3 channel_Id

7640 This attribute is relevant only for wM-Bus communication.

7641 When the wM-Bus profile is used, this attribute holds the identification of the channel currently
7642 used.

7643 The default value is 0.

7644 4.8.7.2.4 link_status

7645 This attribute is relevant only for wM-Bus communication.

7646 When the wM-Bus profile is used, this attribute holds the current status of link to the M-Bus
7647 network.

7648 The possible stati are as specified in EN 13757-5:2015, 9.7.4.1.7 Link State.

7649 enum:
7650 (0) Default (data never received),
7651 (1) Link in normal operation,
7652 (2) Link temporarily interrupted,

7653 (3) Link permanently interrupted

7654

7655 NOTE 1 If synchronisation with network is lost, trials start automatically to re-establish the link by performing radio
7656 scans. As long as re-synchronisation attempts are in progress, the link_status is temporarily interrupted.

7657 Link_status changes to normal operation when the link is re-established.

7658 Link_status changes to permanently interrupted , when the manufacturer specified number of attempts is exceeded.

7659 4.8.7.2.5 broadcast_frames_counter

7660 Holds the broadcast-frames-counter values with time stamp of last frame received and
7661 differentiated by client identifiers.

7662 array broadcast frame counter definition

7663 broadcast frame counter definition:= structure

```
7665
7666 {  
7667     client_id:      unsigned,  
7668     counter:        double-long-unsigned,  
7669     time_stamp:     date-time  
7670 }
```

The default value is 0

7673

7674 4.8.7.2.6 transmissions_counter

7675 Counts the number of frames transmitted by the related M-Bus port.

7676 The transmission counter is incremented at the beginning of each transmission phase. A client
7677 system can write this variable to update the counter. When the transmissions counter reaches
7678 the maximum value, it automatically returns to 0 on the next increment.

7679 The default value is 0.

7680 4.8.7.2.7 FCS_OK_frames_counter

Counts the number of frames received with a correct checksum. When the FCS_OK_frames_counter field reaches the maximum value, it automatically returns to 0 on the next increment.

7684 The default value is 0.

7685 4.8.7.2.8 FCS NOK frames counter

Counts the number of frames received with an incorrect checksum. When the FCS_NOK frames counter field reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0.

7689 4.8.7.2.9 capture time

7690 Holds the time stamp of the most recent change of the value of the attributes 2, 4, 6, 7 or 8.

7691 capture time::= structure

7692

```
7693      \ attribute_id:    unsigned,  
7694      time_stamp:     date-time
```

7696 }

7697

7698 **4.8.7.3 Method**

7699 **4.8.7.3.1 reset (data)**

7700 Clears all counters, received_signal_strength, link_status and capture_time.

7701 data ::= integer(0)

7702 NOTE 2 Channel_id management is outside the scope of the specification of this IC.

7703 See EN 13757-4:2013.

7704

7705

7706 **4.9 Interface classes for setting up data exchange over the Internet**

7707 **4.9.1 TCP-UDP setup (class_id = 41, version = 0)**

7708 **4.9.1.1 Overview**

7709 This IC allows modelling the setup of the TCP or UDP sub-layer of the COSEM TCP or UDP
7710 based transport layer of a TCP-UDP/IP based communication profile.

7711 In TCP-UDP/IP based communication profiles, all AAs between a physical device hosting one
7712 or more COSEM client application processes and a physical device hosting one or more COSEM
7713 server APs rely on a single TCP or UDP connection. The TCP or UDP entity is wrapped in the
7714 COSEM TCP-UDP based transport layer. Within a physical device, each AP – client AP or
7715 server logical device – is bound to a Wrapper Port (WPort). The binding is done with the help
7716 of the SAP Assignment object, see 4.4.5.

7717 On the other hand, a COSEM TCP or UDP based transport layer may be capable to support
7718 more than one TCP or UDP connections, between a physical device and several peer physical
7719 devices hosting COSEM APs.

7720 When a COSEM physical device supports various data link layers – for example Ethernet and
7721 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					

7722

7723 **4.9.1.2 Attribute description**

7724 **4.9.1.2.1 logical_name**

7725 Identifies the “TCP-UDP setup” object instance. See 6.2.23.

7726 **4.9.1.2.2 TCP-UDP_port**

7727 Holds the TCP-UDP port number on which the physical device is listening for the DLMS/COSEM
7728 application.

7729 For DLMS/COSEM, the following port numbers have been registered by the IANA. See
7730 <http://www.iana.org/assignments/port-numbers>

7731 dlms/cosem 4059/TCP DLMS/COSEM

7732 dlms/cosem 4059/UDP DLMS/COSEM

7733 **4.9.1.2.3 IP_reference**

7734 References an IP setup object by its logical name. The referenced object contains information
7735 about the IP Address settings of the IP layer supporting the TCP-UDP layer.

7736 **4.9.1.2.4 MSS**

7737 With the help of the Maximum Segment Size (MSS) option, a TCP can indicate the maximum
7738 receive segment size to its partner. Note, that:

- 7739 – this option shall only be sent in the initial connection request (i.e. in segments with the SYN
7740 control bit sent);
- 7741 – if this option is not present, conventionally MSS is considered as its default value, 576;
- 7742 – MSS is not negotiable; its value is indicated by this attribute.

7743 **4.9.1.2.5 nb_of_sim_conn**

7744 The maximum number of simultaneous connections the COSEM TCP-UDP based transport
7745 layer is able to support.

7746 **4.9.1.2.6 inactivity_time_out**

7747 Defines the time, expressed in seconds over which, if no frame is received from the COSEM
7748 client, the inactive TCP connection shall be aborted.

7749 When this value is set to 0, this means that the *inactivity_time_out* is not operational. In other
7750 words, a TCP connection, once established, in normal conditions – no power failure, etc. – will
7751 never be aborted by the COSEM server.

7752 Note, that all actions related to the management of the inactivity time-out function such as
7753 measuring the inactivity time, aborting the TCP connection if the time-out is over, etc. are
7754 managed inside the TCP-UDP layer implementation.

7755 **4.9.2 IPv4 setup (class_id = 42, version = 0)**

7756 **4.9.2.1 Overview**

7757 NOTE 1 Compared to earlier editions of this standard, this specification provides improvements in presenting the
7758 attributes. As this does not constitute technical changes, the version of the IC remains 0.

7759 This IC allows modelling the setup of the IPv4 layer, handling all information related to the IP
7760 Address settings associated to a given device and to a lower layer connection on which these
7761 settings are used.

7762 There shall be an instance of this IC in a device for each different network interface implemented.
7763 For example, if a device has two interfaces (using the TCP-UDP/IPv4 profile on both of them),
7764 there shall be two instances of the IPv4 setup IC in that device: one for each of these interfaces.

7765

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

7766

7767 **4.9.2.2 Attribute description**7768 **4.9.2.2.1 logical_name**

7769 Identifies the “IPv4 setup” object instance. See 6.2.23.

7770 **4.9.2.2.2 DL_reference**

7771 References a Data link layer (e.g. Ethernet or PPP) setup object by its logical name.

7772 The referenced object contains information about the specific settings of the data link layer
7773 supporting the IP layer.7774 **4.9.2.2.3 IP_address**7775 Carries the value of the IP address (IPv4) of this physical device on the network to which the
7776 device is connected via the referenced lower layer interface. It can be either (static) or
7777 (dynamic). In the latter case, dynamic IP address assignment (for example DHCP) is used.

7778 If no IP address is assigned, the value is 0.

7779 EXAMPLE The IPv4 address 192.168.0.1 (in dotted decimal notation) corresponds to C0A80001 (hexa) which gives
7780 3232235521 (double-long-unsigned).7781 **4.9.2.2.4 multicast_IP_address**7782 Contains an array of IP addresses. IP addresses in this array shall fall into the multicast group
7783 address range (“Class D” addresses, including IP addresses in the range of 224.0.0.0 to
7784 239.255.255.255). When a device receives an IP datagram with one of these IP addresses in
7785 the destination IP address field, it shall consider that this datagram is addressed to it.

7786 multicast_IP_address::= array double-long-unsigned

7787 **4.9.2.2.5 IP_options**

7788 Contains the necessary parameters to support the selected IP options – for example Datagram
 7789 time-stamping or security services (IPSec).

```
7790           IP_options ::= array     IP_options_element
7791
7792           IP_options_element ::= structure
7793
7794           {
7795             IP_Option_Type:      unsigned,
7796             IP_Option_Length:    unsigned,
7797             IP_Option_Data:      octet-string
7798           }
```

7799 NOTE 2 In all cases, as specified in RFC 791, the IP_Option_Length field includes the total length of all three fields:
 7800 IP_Option_Type, IP_Option_Length and IP_Option_Data.

7801 Allowed IP_Option_Types:

7802 **Security: IP_Option_Type = 0x82, IP_Option_Length = 11**

7803 If this option is present, the device shall be allowed to send security, compartmentation,
 7804 handling restrictions and TCC (closed user group) parameters within its IP Datagrams. The
 7805 IP_Option_Data shall contain the value of the Security, Compartments, Handling Restrictions
 7806 and Transmission Control Code values, as specified in RFC 791.

7807 **Loose Source and Record Route: IP_Option_Type = 0x83**

7808 If this option is present, the device shall supply routing information to be used by the gateways
 7809 in forwarding the datagram to the destination, and to record the route information. The
 7810 IP_Option_Length and IP_Option_Data values are specified in RFC 791.

7811 **Strict Source and Record Route: IP_Option_Type = 0x89**

7812 If this option is present, the device shall supply routing information to be used by the gateways
 7813 in forwarding the datagram to the destination, and to record the route information. The
 7814 IP_Option_Length and IP_Option_Data values are specified in RFC 791.

7815 **Record Route: IP_Option_Type = 0x07**

7816 If this option is present, the device shall additionally:

- 7817 – send originated IP Datagrams with that option, providing means to record the route of these
 7818 Datagrams;
- 7819 – as a router, send routed IP Datagrams with the route option adjusted according to this
 7820 option.

7821 The IP_Option_Length and IP_Option_Data values are specified in RFC 791.

7822 **Internet Timestamp: IP_Option_Type = 0x44**

7823 If this option is present, the device shall additionally:

- 7824 – send originated IP Datagrams with that option, providing means to time-stamp the datagram
 7825 in the route to its destination;
- 7826 – as a router, send routed IP Datagrams with the time-stamp option adjusted according to this
 7827 option.

7828 The IP_Option_Length and IP_Option_Data values are specified in RFC 791.

7829 **4.9.2.2.6 subnet_mask**

7830 Contains the subnet mask.

7831 When sub-networking is used in a network segment, each device concerned shall behave
7832 conforming to the sub-networking rules. In order to do that, the device, besides of its IP address,
7833 needs also to know, how the IP address is structured within this sub-networked segment. The
7834 *subnet_mask* attribute carries this information.

7835 With IPv4, the *subnet_mask* is a 32 bits word, expressed exactly in the same format as an IP
7836 Address (for example 255.255.255.0), but has another meaning: the '0' bits of the *subnet_mask*
7837 indicate the portion of the IP Address which is still used as Device_ID on a sub-networked IP
7838 Network³.

7839 **4.9.2.2.7 gateway_IP_address**

7840 Contains the IP Address of the gateway device.

7841 In most IP implementations, there is code in the module that handles outgoing datagrams to
7842 decide if a datagram can be sent directly to the destination on the local network or if it shall be
7843 sent to a gateway. In order to be able to send non-local datagrams to the gateway, the device
7844 shall know the IP address of the gateway device assigned to the given network segment.

7845 If no IP address is assigned, the value is 0.

7846 **4.9.2.2.8 use_DHCP_flag**

7847 TRUE: The device uses DHCP (Dynamic Host Configuration Protocol) to dynamically determine
7848 the *IP_address*, *subnet_mask* and *gateway_IP_address* parameters.

7849 FALSE: The *IP_address*, *subnet_mask* and *gateway_IP_address* parameters shall be set locally.

7850 **4.9.2.2.9 primary_DNS_address**

7851 The IP Address of the primary Domain Name Server (DNS).

7852 If no IP address is assigned, the value is 0.

7853 **4.9.2.2.10 secondary_DNS_address**

7854 The IP Address of the secondary Domain Name Server (DNS).

7855 If no IP address is assigned, the value is 0.

7856 **4.9.2.3 Method**

7857 **4.9.2.3.1 add_mc_IP_address (IP_Address)**

7858 Adds one multicast IP address to the *multicast_IP_address* array.

7859 IP_Address ::= double-long-unsigned

7860 **4.9.2.3.2 delete_mc_IP_address (IP_Address)**

7861 Deletes one IP Address from the *multicast_IP_address* array. The IP Address to be deleted is
 7862 identified by its value.

7863 IP_Address ::= double-long-unsigned

7864 **4.9.2.3.3 get_nb_of_mc_IP_addresses (data)**

7865 Returns the number of IP Addresses contained in the *multicast_IP_address* array.

7866 data ::= unsigned

7867

7868 **4.9.3 IPv6 setup (class_id = 48, version = 0)**7869 **4.9.3.1 Overview**

7870 NOTE 1 See also Annex C.

7871 The IPv6 setup IC allows modelling the setup of the IPv6 layer, handling all information related
 7872 to the IPv6 address settings associated to a given device and to a lower layer connection on
 7873 which these settings are used.

7874 There shall be an instance of this IC in a device for each different network interface implemented.
 7875 For example, if a device has two interfaces (using the UDP/IP and/or TCP/IP profile on both of
 7876 them), there shall be two instances of the IPv6 setup IC in that device: one for each of these
 7877 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

7878

7879 **4.9.3.2 Attribute description**7880 **4.9.3.2.1 logical_name**

7881 Identifies the “IPv6 setup” object instance. See 6.2.23.

7882

7883 **4.9.3.2.2 DL_reference**

7884 References a Data link layer setup object by its logical name. The referenced object contains
7885 information about the specific settings of the data link layer supporting the IPv6 layer.

7886 **4.9.3.2.3 address_config_mode**

7887 Defines the IPv6 address configuration mode

7888 enum:

- 7889 (0) Auto-configuration (default),
- 7890 (1) DHCPv6,
- 7891 (2) Manual,
- 7892 (3) ND (Neighbour Discovery)

7893 NOTE 2 The address_config_mode is common for all IPv6 addresses managed by an instance of the IPv6 setup
7894 class.

7895 **4.9.3.2.4 unicast_IPv6_addresses**

7896 Carries unicast IPv6 address(es) assigned to the related interface of the physical device on the
7897 network (unique local unicast, link local unicast and / or global unicast addresses). An IPv6
7898 address can be either (static) or (dynamic) or both.

7899 unicast_IPv6_addresses ::= array octet-string

7900 The format of each unicast IPv6 address shall be as specified in RFC 3513.

7901 If no unicast IPv6 address is assigned to the interface, an array of zero elements is present
7902 (default).

7903 To reset all unicast IPv6 address(es) configured, an array of zero elements shall be written.

7904 **4.9.3.2.5 multicast_IPv6_addresses**

7905 Contains an array of IPv6 addresses used for multicast.

7906 multicast_IPv6_addresses ::= array octet-string

7907 The format of each multicast IPv6 address shall be as specified in RFC 3513 .

7908 If no multicast IPv6 address is assigned to the interface, an array of zero elements is present
7909 (default).

7910 To reset all multicast IPv6 address(es) configured, an array of zero elements shall be written.

7911 **4.9.3.2.6 gateway_IPv6_addresses**

7912 Contains the IPv6 addresses of the IPv6 gateway device.

7913 gateway_IPv6_addresses ::= array octet-string

7914 The format of each gateway IPv6 address shall be as specified in RFC 3513.

7915 If no gateway IPv6 address is assigned to the interface, an array of zero elements is present
7916 (default).

7917 To reset all gateway IPv6 address(es) configured, an array of zero elements shall be written.

7918 **4.9.3.2.7 primary_DNS_address**

7919 Contains the IPv6 address of the primary Domain Name Server (DNS).

7920 If no IPv6 address is assigned, the length of the octet-string shall be 0.

7921 **4.9.3.2.8 secondary_DNS_address**

7922 Contains the IPv6 address of the secondary Domain Name Server (DNS).

7923 If no IPv6 address is assigned, the length of the octet-string shall be 0.

7924 **4.9.3.2.9 traffic_class**

7925 Contains the traffic class element of the IPv6 header. The 6 most-significant bits are used for
 7926 DSCP (Differentiated Services Codepoint), which is used to classify packets as specified in
 7927 RFC 2474:1998, Clause 3.

7928 **4.9.3.2.10 neighbor_discovery_setup**

7929 Contains the configuration to be used for both routers and hosts to support the Neighbor
 7930 Discovery protocol for IPv6 (RFC 4861).

```
7931           array neighbor_discovery_setup
7932
7933           neighbor_discovery_setup ::= structure
7934
7935           {
7936             RS_max_retry:      unsigned,
7937             RS_retry_wait_time: long-unsigned,
7938             RA_send_period:    double-long-unsigned
7939           }
```

7940 Where:

7941 **RS_max_retry**

7942 Gives the maximum number of router solicitation retries to be performed by a
 7943 node if the expected router advertisement has not been received.

7944 Range: 1-255

7945 Default value: 3

7946 **RS_retry_wait_time**

7947 Gives the waiting time in milliseconds between two successive router
 7948 solicitation retries.

7949 Range: 0-65 535

7950 Default value: 10 000

7951 **RA_send_period**

7952 Gives the router advertisement transmission period in seconds.

7953 Range: 0-4 294 967 295

7955 **4.9.3.3 Method description**

7956 **4.9.3.3.1 add_IPv6_address (data)**

7957 Adds one IPv6 address for the physical interface to the IPv6 address array.

7958 data ::= structure

```

7959     {
7960         IPv6_address_type: enum,
7961         IPv6_address: octet-string
7962     }
7963

```

7964 Where IPv6_address_type defines the type of IPv6 address to add:

```

7965             enum: (0) unicast,
7966                 (1) multicast,
7967                 (2) gateway
7968
7969

```

7970 IPv6_address specifies the IPv6 address to add. The new address will be automatically added
7971 at the end of the list. If an address has to be added to a specific position, this can be done by
7972 writing the whole array.

7973 **4.9.3.3.2 remove_IPv6_address (data)**

7974 Removes one IPv6 address for the physical interface to the IPv6 address array.

```

7975             data::= structure
7976
7977             {
7978                 IPv6_address_type: enum,
7979                 IPv6_address: octet-string
7980             }
7981

```

7982 Where IPv6_address_type defines the type of IPv6 address to remove:

```

7983             enum: (0) unicast,
7984                 (1) multicast,
7985                 (2) gateway
7986
7987

```

7988 IPv6_address specifies the IPv6 address to remove.

7989

7990 **4.9.4 MAC address setup (class_id = 43, version = 0)**

7991 **4.9.4.1 Overview**

7992 NOTE 1 The name and the use of this interface class has been changed from “Ethernet setup” to “MAC address
7993 setup” to allow a more general use, without changing the version.

7994 Instances of this IC hold the MAC address of the physical device (or, more generally, a device
7995 or software.) There shall be an instance of this IC for each network interface of a physical
7996 device.

7997 NOTE 2 In the case of the three-layer HDLC based communication profile, the MAC address (lower HDLC address)
7998 is carried by an IEC HDLC setup object.

7999 NOTE 3 In the case of the S-FSK PLC communication profile, the MAC address is carried by a S-FSK Phy&MAC
8000 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				

8001

8002 **4.9.4.2 Attribute description**8003 **4.9.4.2.1 logical_name**

8004 Identifies the “MAC address setup” object instance. See 6.2.21, 6.2.23 and 6.2.27.

8005 **4.9.4.2.2 mac_address**

8006 Holds the MAC address.

8007

8008 **4.9.5 PPP setup (class_id = 44, version = 0)**8009 **4.9.5.1 Overview**

8010 NOTE 1 Compared to earlier editions of this standard, this specification provides improvements in presenting the
8011 attributes. As this does not constitute technical changes, the version of the IC remains 0.

8012 This IC allows modelling the setup of interfaces using the PPP protocol, by handling all
8013 information related to PPP settings associated to a given physical device and to a lower layer
8014 connection on which these settings are used. There shall be an instance of this IC for each
8015 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				

8016

8017 **4.9.5.2 Attribute description**8018 **4.9.5.2.1 logical_name**

8019 Identifies the “PPP setup” object instance. See 6.2.23.

8020 **4.9.5.2.2 PHY_reference**

8021 References another object by its logical_name. The object referenced contains information
8022 about the specific physical layer interface, supporting the PPP layer.

8023 **4.9.5.2.3 LCP_options**

8024 Contains the necessary parameters to support the selected LCP configuration options.

8025 LCP_options_type::= array LCP_options_type_element

8026 LCP_options_type_element::= structure

8027 {

8028 LCP_Option_Type: unsigned,

8029 LCP_Option_Length: unsigned,

8030 LCP_Option_Data: CHOICE

8031

8032

```
8033 {  
8034     structure      [2] -- for Callback-data  
8035     boolean         [3] -- for ProtF-Compr and AdCtr-Compr,  
8036     double-long-unsigned [6] -- for ACCM and Mag-Num,  
8037     unsigned        [17] -- for FCS-Alternatives,  
8038     long-unsigned    [18] -- for MRU and Auth-Prot  
8039 }  
8040 }  
8041
```

8042 NOTE 2 In all cases, as specified in IETF STD 51 / RFC 1661, the LCP_Option_Length field includes the total
8043 length of all three fields: LCP_Option_Type, LCP_Option_Length and LCP_Option_Data.

8044 NOTE 3 For assigned values see Point-to-Point (PPP) Protocol Field Assignments, available at:
8045 <http://www.iana.org/assignments/ppp-numbers/ppp-numbers.xml>

8046 The supported LCP_Option_Types are the following:

8047 Maximum-Receive-Unit (MRU), LCP_Option_Type = 1. See IETF STD 51 / RFC 1661.

8048 This configuration option may be sent to inform the peer that the implementation can receive
8049 larger packets, or to request that peer send smaller packets. The default value is 1 500 octets;

8050 Async-Control-Character-Map (ACCM), LCP_Option_Type = 2. See IETF STD 51 / RFC 1662.

8051 This configuration option provides a method to negotiate the use of control character
8052 transparency on asynchronous links;

8053 Authentication-Protocol, LCP_Option_Type = 3. See IETF STD 51 / RFC 1661.

8054 This configuration option provides a method to negotiate the use of a specific protocol for
8055 authentication. By default, authentication is not required. The value indicates the authentication
8056 protocol used on the given PPP link. Possible values are:

8057 0x0000 – No authentication protocol is used,

8058 0xc023 – The PAP protocol is used,

8059 0xc223 – The CHAP protocol is used,

8060 0xc227 – The EAP protocol is used.

8061 Magic-Number, LCP_Option_Type = 5. See IETF STD 51 / RFC 1661.

8062 This configuration option provides a method to detect looped-back links and other data link
8063 layer anomalies;

8064 Protocol-Field-Compression (PFC), LCP_Option_Type = 7. See IETF STD 51 / RFC 1661.

8065 This configuration option provides a method to negotiate the compression of the PPP protocol
8066 fields;

8067 Address-and-Control-Field-Compression (ACFC), LCP_Option_Type = 8. See IETF STD 51 /
RFC 1661.

8069 This configuration option provides a method to negotiate the compression of the data link layer
8070 address and control fields;

8071 FCS-Alternatives, LCP_Option_Type = 9. See RFC 1570.

8072 This configuration option provides a method for an implementation to specify another FCS
 8073 format to be sent by the peer, or to negotiate away the FCS altogether. The value of the FCS-
 8074 Alter (FCS Alternatives) options field identifies the FCS used. This field is one octet, and is
 8075 comprised of the "logical or" of the following values:

8076 bit 1 Null FCS,

8077 bit 2 CCITT 16-bit FCS,

8078 bit 4 CCITT 32-bit FCS.

8079 Callback, LCP_Option_Type = 13. See RFC 1570.

8080 This configuration option provides a method for an implementation to request a dial-up peer to
 8081 call back. This provides enhanced security by ensuring that the remote site can connect only
 8082 from a single location as defined by the callback number.

```
8083         callback_data ::= structure
8084
8085             {
8086                 callback_active: boolean, // default: false,
8087                 callback_data_length: unsigned,
8088                 callback_operation: unsigned,
8089                 callback_message: octet-string
8090             }
```

8091 Where:

- 8092 – the callback-active field indicates whether the callback option is active on this PPP link;
- 8093 – the callback_operation field indicates the contents of the Message field;

8094 callback_operation: unsigned

8095	(0)	Location determined by user authentication,
8096	(1)	Dialling string,
8097	(2)	Location identifier,
8098	(3)	E.164 number,
8099	(4)	X.500 distinguished name,
8100	(5)	Unassigned,
8101	(6)	Location is determined during CBCP negotiation.

8103
 8104 the callback_message field is zero or more octets, and its general contents are determined by
 8105 the callback_operation field. The actual format of the information is site or application specific.

8106 **4.9.5.2.4 IPCP_options**

8107 Contains the necessary parameters for the IP Control Protocol – the Network Control Protocol
 8108 module of the PPP – that allow the negotiation of desirable Internet Protocol parameters. For
 8109 details on IPCP, please refer to RFC 1332.

8110 IPCP_options_type ::= array IPCP_options_type_element

8111

8112	IPCP_options_type_element ::= structure
8113	{
8114	

```

8115     IPCP_Option_Type:      unsigned,
8116     IPCP_Option_Length:   unsigned,
8117     IPCP_Option_Data:    CHOICE
8118     {
8119         array          [1] -- for Pref-Peer-IP,
8120         -- each IP address is of type double-long-unsigned
8121         boolean         [3] -- for GAO and USIP,
8122         double-long-unsigned [6] -- for Pref-Local-IP,
8123         long-unsigned    [18] -- for IP-Comp-Prot
8124     }
8125 }
```

8126 NOTE 4 In all cases, as specified in RFC 1332, the IPCP_Option_Length field includes the total length of all three
 8127 fields: IPCP_Option_Type, IPCP_Option_Length and IPCP_Option_Data.

8128 The supported IPCP_Option_Types are the following:

8129 [IP-Compression-Protocol \(IP-Comp-Prot\) IPCP_Option_Type = 2. See RFC 1332.](#)

8130 This configuration option provides a way to negotiate the use of a specific compression protocol.
 8131 By default, compression is not enabled. Possible values are:

```

8132     0x0000 – No IP Compression is used (default),
8133     0x002d – Van Jacobson Compressed TCP/IP (RFC 1332),
8134     0x0003 – Robust Header Compression (ROHC) (RFC 3241),
8135     0x0061 – IP Header Compression (RFC 2507, RFC 3544)
8136 
```

8137 [Preferred-Local-IP-Address \(Pref-Local-I\), IPCP_Option_Type = 3. See RFC 1332.](#)

8138 This configuration option provides a way to negotiate the IP address to be used on the local
 8139 end of the link. It allows the sender of the Configure-Request to state, which IP-address is
 8140 desired, or to request that the peer provide the information. The peer can provide this
 8141 information by NAK-ing the option, and returning a valid IP-Address;

8142 NOTE 5 In RFC 1332 the name of option Type 3 is “IP-Address”.

8143 NOTE 6 Option types 20, 21 and 22 have been specified for the purposes of DLMS/COSEM; they are not specified
 8144 in RFC 1332.

8145 [Preferred-Peer-IP-Addresses \(Pref-Peer-IP\), IPCP_Option_Type = 20.](#)

8146 This configuration option provides a way to negotiate the IP Address to be used on the remote
 8147 end of the link. When the Grant-Access-Only-to-Pref-Peer-on-List (GAO) parameter is set to be
 8148 TRUE, the device shall accept PPP connection only with a remote device having one of the IP
 8149 Addresses on this list. When the Use-Static-IP-Pool (USIP) parameter is set to TRUE, the
 8150 COSEM Server device shall try to assign one of these IP Addresses to the remote device;

8151 [Grant-Access-Only-to-Pref-Peer-on-List \(GAO\), IPCP_Option_Type = 21.](#)

8152 This configuration option indicates whether the device can accept PPP connection only with
 8153 peer devices with IP Address on the above list or not. Its default value is FALSE;

8154 [Use-Static-IP-Pool \(USIP\), IPCP_Option_Type = 22.](#)

8155 This configuration option indicates whether the device should try to assign one of the IP
 8156 Addresses of the Preferred-Peer-IP-Addresses to the remote end device during the IP Address
 8157 negotiation phase or not.

8159 **4.9.5.2.5 PPP_authentication**

8160 Contains the parameters required by the PPP authentication procedure used.

```

8161                 PPP_auth_type ::= CHOICE
8162
8163                 {
8164                 null-data [0] -- used when no authentication is required,
8165                 structure [2] -- PAP_login or CHAP_algorithm or EAP_params
8166                 }
8167
8168                 PAP_login ::= structure
8169                 {
8170                 user-name: octet-string,
8171                 PAP-password: octet-string
8172                 }
8173
8174
8175                 CHAP_algorithm ::= structure
8176                 {
8177                 user-name: octet-string,
8178                 algorithm_id: unsigned
8179                 }
8180
8181                 Possible values for CHAP-algorithm-id parameter today are as follows:
8182                 0x05 – CHAP with MD5 (default), see RFC 1994.
8183                 0x06 – SHA-1,
8184                 0x80 – MS-CHAP, see RFC 2433
8185                 0x81 – MS-CHAP-2, see RFC 2759.
8186

```

8187 NOTE 7 New values can be used as become assigned.

8188 When CHAP is used, a “secret” is also required to verify the “challenge” sent by the client. This
 8189 “secret” is not accessible in the PPP setup object.

8190 NOTE 8 The PPP Challenge Handshake Authentication Protocol (CHAP) is specified in RFC 1994.

```

8191                 EAP_params ::= structure
8192
8193                 {
8194                 md5_challenge (Type 4): boolean,
8195                 one_time_password (OTP, Type 5): boolean,
8196                 generic_token_card (GTC, Type 6): boolean
8197                 }
8198

```

8199 NOTE 9 The Extensible Authentication Protocol (EAP) is specified in RFC 3748.

8200

8201

8202

8203

8204

8205

8206 **4.9.6 SMTP setup (class_id = 46, version = 0)**8207 **4.9.6.1 Overview**

8208 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				

8209

8210 **4.9.6.2 Attribute description**8211 **4.9.6.2.1 logical_name**

8212 Identifies the “SMTP setup” object instance. See 6.2.23.

8213 **4.9.6.2.2 server_port**

8214 Defines the value of the TCP-UDP port related to this protocol. By default, this value is the
8215 SMTP port numberID assigned by IANA:

8216 – smtp 25/tcp, smtp 25/udp Simple Mail Transfer

8217 **4.9.6.2.3 user_name**

8218 Defines the user name to be used for the login to the SMTP server.

8219 **4.9.6.2.4 login_password**

8220 Password to be used for login. When the string is void, this means that there is no
8221 authentication.

8222 **4.9.6.2.5 server_address**

8223 Defines the server address as an octet string. This server address can be a name, which shall
8224 be resolvable by the primary DNS or the secondary DNS. In the case when it is directly the IP
8225 address of the server, which is specified here, it shall be a string in dotted format. EXAMPLE:
8226 163.187.45.87.

8227 **4.9.6.2.6 sender_address**

8228 Defines the sender address as an octet string. This sender address can be a name. In the case
8229 when it is directly the IP address of the sender, which is specified here, it will be a string in
8230 dotted format.

8231

8232 **4.9.7 NTP setup (class_id = 100, version = 0)**8233 **4.9.7.1 Overview**

8234 Instances of the “NTP setup” IC allow setting up time synchronisation using the NTP protocol
 8235 as specified in RFC 5905. One or several instances may be configured to support multiple time
 8236 servers.

NTP Setup	0...n	class_id = 100, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. activated (static)	boolean			FALSE	x + 0x08
3. server_address (static)	octet-string				x + 0x10
4. server_port (static)	long-unsigned			123	x + 0x18
5. authentication_method (static)	enum				x + 0x20
6. authentication_keys (static)	array				x + 0x28
7. client_key (static)	octet-string				x + 0x30
Specific methods	m/o				
1. synchronize (data)	m				
2. add_authentication_key (data)	o				
3. delete_authentication_key (data)	o				

8237

8238 **4.9.7.2 Attribute description**

8239 logical_name Identifies the “NTP setup” object instance. See 6.2.23.

8240 **4.9.7.2.1 activated**

8241 Defines if the NTP time synchronisation is active or not.

8242 Synchronisation active = TRUE

8243 **4.9.7.2.2 server_address**

8244 Defines the NTP server address as an octet string. This server address can be a name, which
 8245 must be resolvable by the primary DNS or the secondary DNS. In the case when it is directly
 8246 the IP address of the server the dot-decimal notation shall be used for IPv4 addresses and the
 8247 text format for IPv6 addresses.

8248 Examples: 163.187.45.87 (IPv4) or 2001:db8::1:0:0:1 (IPv6)

8249 **4.9.7.2.3 server_port**

8250 Defines the value of the UDP port related to this protocol. By default, this value is the NTP port
 8251 number ID assigned by IANA:

8252 ntp 123/udp Network Time Protocol

8253 **4.9.7.2.4 authentication_method**

8254 Defines the authentication mode used for NTP protocol

8255 enum:

8256 (0) no_security,
8257 (1) shared_secrets,
8258 (2) auto_key_IFF

8260 4.9.7.2.5 authentication keys

8261 Contains the necessary symmetric keys if shared secrets mode of authentication is used.

```
8262           authentication_keys ::= array authentication_key  
8263  
8264           authentication_key ::= structure  
8265  
8266           {  
8267             key_id    double-long-unsigned,  
8268             key        octet-string  
8269           }
```

8271 4.9.7.2.6 client key

8272 Specifies the client key (NTP server public key) for NTP auto key authentication mechanism
8273 using IFF authentication scheme (auto_key IFF).

8274 4.9.7.3 Method description

8275 4.9.7.3.1 synchronize (data)

8276 Synchronizes the time of the DLMS server with the NTP server.

8277 data::= integer(0)

8278 4.9.7.3.2 add_authentication_key (data)

8279 Adds a new symmetric authentication key to authentication key array.

```
8280                     data ::= structure  
8281  
8282             {  
8283                 key_id      double-long-unsigned,  
8284                 key          octet-string  
8285             }  
8286
```

8287 4.9.7.3.3 delete_authentication_key (data)

Deletes a symmetric authentication key from the key array. The key to be deleted is identified by its `key_id`.

8290 data::= double-long-unsigned

8291

8292 **4.10 Interface classes for setting up data exchange using S-FSK PLC**

8293 **4.10.1 General**

8294 This subclause specifies COSEM interface classes to set up and manage the protocol layers of
 8295 DLMS/COSEM S-FSK PLC communication profile:

- 8296 • the S-FSK Physical layer and the MAC sub-layer as defined in IEC 61334-5-1:2001 and
 8297 IEC 61334-4-512:2001;
- 8298 • the LLC sub-layer as specified in IEC 61334-4-32:1996.

8299 The MIB variables / logical link parameters specified in IEC 61334-4-512:2001, IEC 61334-5-
 8300 1:2001, IEC 61334-4-32:1996 and IEC 62056-8-8:2020, Electricity *metering data exchange -*
 8301 ***The DLMS/COSEM suite - Part 8-8: Communication profile for ISO/IEC 14908 series networks***

8302 ISO/IEC 8802-2:1998 respectively have been mapped to attributes and/or methods of COSEM
 8303 ICs. The specification of these elements has been taken from the above standards and the text
 8304 has been adapted to the DLMS/COSEM environment.

8305 NOTE IEC 61334-4-512:2001 also specifies some management variables to be used on the Client side. However,
 8306 the Client side object model is not covered in this document.

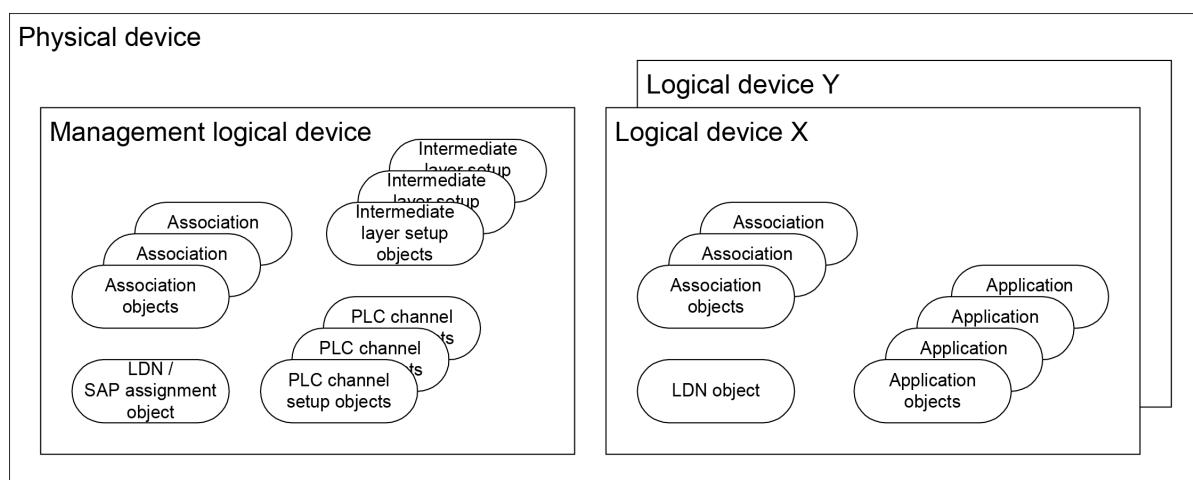
8307 For definitions related to S-FSK PLC profile see 3.2.

8308 **4.10.2 Overview**

8309 COSEM objects for setting up the S-FSK PLC channel and the LLC layer, if implemented, shall
 8310 be located in the Management Logical Device of COSEM servers.

8311 Figure 27 shows an example with a COSEM physical device comprising three logical devices.
 8312 Each logical device shall contain a Logical Device Name (LDN) object. Each logical device
 8313 contains one or more Association objects, one for each client supported.

8314 NOTE As in this example there is more than one logical device, the mandatory Management Logical Device contains
 8315 a SAP Assignment object instead of a Logical Device Name object.



8316

IEC

8317 **Figure 27 – Object model of DLMS/COSEM servers**

8318 The management logical device contains the setup objects of the physical and MAC layers of
 8319 the PLC channel, as well as setup objects for the intermediate layer(s). It may contain further
 8320 application objects.

8321 The other logical devices, in addition to the Association and Logical Device Name objects
 8322 mentioned above, contain further application objects, holding parameters and measurement
 8323 values.

8324 IEC 61334-4-512:2001 uses DLMS named variables to model the MIB objects and specifies
 8325 their DLMS name in the range 8...184. For compatibility with existing implementations, the short
 8326 names 8...400 [sic] are reserved for devices using the IEC 61334-5-1:2001 S-FSK PLC profiles
 8327 without COSEM. Therefore, when mapping the attributes and methods of the COSEM objects
 8328 specified in this document to DLMS named variables (SN mapping) this range shall not be used.

8329 Table 38 shows the mapping of MIB variables to attributes and/or methods of COSEM ICs.

8330 Note that on the one hand, not all MIB variables specified in IEC 61334-4-512:2001 have been
 8331 mapped to attributes and methods of COSEM ICs. On the other hand, some new management
 8332 variables are specified in this document.

8333 **Table 38 – Mapping IEC 61334-4-512:2001 MIB variables to**
 8334 **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	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			
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			

Name	Reference (unless otherwise indicated)	Interface class	class_id / attribute / method
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	S-FSK Reporting system list (class_id = 56, version = 0)	56 / Attr. 2
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

8335

8336 **4.10.3 S-FSK Phy&MAC set-up (class_id = 50, version = 1)**8337 **4.10.3.1 Overview**

8338 NOTE 1 The use of version 0 of this interface class is deprecated.

8339 An instance of the “S-FSK Phy&MAC set-up” class stores the data necessary to set up and
8340 manage the physical and the MAC layer of the PLC S-FSK lower layer profile.

8341

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		m/o				

8342

8343 **4.10.3.2 Attribute description**8344 **4.10.3.2.1 logical_name**

8345 Identifies the “S-FSK Phy&MAC setup” object instance. See 6.2.24.

8346 **4.10.3.2.2 initiator_electrical_phase**

8347 Holds the MIB variable initiator-electrical-phase (variable 18) specified in IEC 61334-4-512:2001, 5.8.

8349 It is written by the client system to indicate the phase to which it is connected.

8350 enum:

- (0) Not defined (default),
- (1) Phase 1,
- (2) Phase 2,
- (3) Phase 3

8355 NOTE 2 This enumeration is different from that of IEC 61334-4-512.

8356 **4.10.3.2.3 delta_electrical_phase**8357 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.

8359 It indicates the phase difference between the client's connecting phase and the server's connecting phase. The following values are predefined:

8361 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.

8365 The phase difference between the server's connecting phase and the client's connecting phase
 8366 is equal to:

- 8367 (2) 60 degrees,
 8368 (3) 120 degrees,
 8369 (4) 180 degrees,
 8370 (5) -120 degrees,
 8371 (6) -60 degrees

8373 **4.10.3.2.4 max_receiving_gain**

8374 Holds the MIB variable *max-receiving-gain* (variable 2) specified in IEC 61334-4-512:2001, 5.2
 8375 and IEC 61334-5-1:2001, 3.5.5.3.

8376 Corresponds to the maximum allowed gain bound to be used by the server system in the
 8377 receiving mode. The default unit is dB.

8378 NOTE 3 In IEC 61334-4-512:2001, no units are specified.

8379 The possible values of the gain may depend on the hardware. Therefore, after writing a value
 8380 to this attribute, the value should be read back to know the actual value.

8381 **4.10.3.2.5 max_transmitting_gain**

8382 Holds the value of the *max-transmitting-gain*.

8383 Corresponds to the maximum attenuation bound to be used by the server system in the
 8384 transmitting mode. The default unit is dB.

8385 The possible values of the gain may depend on the hardware. Therefore, after writing a value
 8386 to this attribute, the value should be read back to know the actual value.

8387 **4.10.3.2.6 search_initiator_threshold**

8388 This attribute is used in the intelligent search initiator process. If the value of the initiator signal
 8389 is above the value of this attribute, a fast synchronization process is possible.

8390 The default value is 98 dB μ V.

8391 **4.10.3.2.7 frequencies**

8392 Contains frequencies required for S-FSK modulation.

```
8393     frequencies_type ::= structure
8394
8395         {
8396             mark_frequency:      double-long-unsigned,
8397             space_frequency:    double-long-unsigned
8398         }
```

8399 The default unit is Hz.

8400 **4.10.3.2.8 mac_address**

8401 Holds the MIB variable *mac-address* (variable 3) specified in IEC 61334-4-512:2001, 5.3 and in
 8402 IEC 61334-5-1:2001, 4.3.7.6.

8403 NOTE 4 MAC addresses are expressed on 12 bits.

8404 Contains the value of the address of the physical attachment (MAC address) associated to the
8405 local system. In the unconfigured state, the MAC address is "NEW-address".

8406 This attribute is locally written by the CIASE when the system is registered (with a Register
8407 service). The value is used in each outgoing or incoming frame. The default value is "NEW-
8408 address".

8409 This attribute is set to NEW:

- 8410 – by the MAC sub-layer, once the time-out-not-addressed delay is exceeded;
- 8411 – when a client system "resets" the server system. See 4.10.4.

8412 When this attribute is set to NEW:

- 8413 – the system loses its synchronization (function of the MAC-sublayer);
- 8414 – the *mac_group_address* attribute is reset (array of 0 elements);
- 8415 – the system automatically releases all AAs which can be released.

8416 NOTE 5 The second item is not present in IEC 61334-4-512:2001.

8417 The predefined MAC addresses are shown in Table 42.

8418 **4.10.3.2.9 mac_group_addresses**

8419 Holds the MIB variable *mac-group-address* (variable 4) specified in IEC 61334-4-512:2001, 5.3
8420 and in IEC 61334-5-1:2001, 4.3.7.6.

8421 Contains a set of MAC group addresses used for broadcast purposes.

8422 array mac-address
8423 mac-address ::= long-unsigned

8424

8425 The ALL-configured-address, ALL-physical-address and NO-BODY addresses are not included
8426 in this list. These ones are internal predefined values.

8427 This attribute shall be written by the initiator using DLMS services to declare specific MAC
8428 group addresses on a server system.

8429 This attribute is locally read by the MAC sublayer when checking the destination address field
8430 of a MAC frame not recognized as an individual address or as one of the three predefined
8431 values (ALL-configured-address, ALL-physical-address and NO-BODY).

8432 **4.10.3.2.10 repeater**

8433 Holds the MIB variable *repeater* (variable 5) specified in IEC 61334-4-512:2001, 5.3 and in IEC
8434 61334-5-1:2001, 4.3.7.6.

8435 It specifies whether the server system effectively repeats all frames or not.

8436 enum:
8437 (0) never repeater,
8438 (1) always repeater,
8439 (2) dynamic repeater

8440

8441 If the *repeater* variable is equal to 0, the server system should never repeat the frames.

8442 If it is set to 1, the server system is a repeater: it has to repeat all frames received without error
 8443 and with a current credit greater than zero.

8444 If it is set to 2, then the repeater status can be dynamically changed by the server itself.

8445 NOTE 6 The value 2 value is not specified in IEC 61334-4-512.

8446 This attribute is internally read by the MAC sub-layer each time a frame is received.

8447 The default value shall be specified in project specific companion specifications.

8448 **4.10.3.2.11 repeater_status**

8449 Holds the current *repeater status* of the device.

8450 boolean:

8451 (0) FALSE = no repeater,
 8452 (1) TRUE = repeater

8455 **4.10.3.2.12 min_delta_credit**

8456 Holds the MIB variable *min-delta-credit* (variable 9) specified in IEC 61334-4-512:2001, 5.3 and
 8457 in IEC 61334-5-1:2001, 4.3.7.6.

8458 NOTE 7 Only the three least significant bits are used.

8459 The Delta Credit (DC) is the subtraction of the Initial Credit (IC) and Current Credit (CC) fields
 8460 of a correct received MAC frame. The delta-credit minimum value of a correct received MAC
 8461 frame, directed to a server system, is held by this variable.

8462 The default value is set to the maximal initial credit (see IEC 61334-5-1:2001 4.2.3.1 for further
 8463 explanations on the credit and the value of MAX_INITIAL_CREDIT). A client system can
 8464 reinitialise this variable by setting its value to the maximal initial credit.

8465 **4.10.3.2.13 initiator_mac_address**

8466 Holds the MIB variable *initiator-mac-address* specified in IEC 61334-5-1:2001, 4.3.7.6.

8467 Its value is either the MAC address of the active-initiator or the NO-BODY address, depending
 8468 on the value of the *synchronization_locked* attribute (see 4.10.3.2.14 below). See also IEC
 8469 61334-5-1:2001 3.5.3, 4.1.6.3 and 4.1.7.2.

8470 **4.10.3.2.14 synchronization_locked**

8471 Holds the MIB variable *synchronization-locked* (variable 10) specified in IEC 61334-4-512:2001,
 8472 5.3.

8473 Controls the synchronization locked / unlocked state. See IEC 61334-5-1:2001 for more details.

8474 If the value of this attribute is equal to TRUE, the system is in the synchronization-locked state.
 8475 In this state, the *initiator-mac-address* is always equal to the MAC address field of the active-
 8476 initiator MIB object. See attribute 2 of the S-FSK Active initiator IC (4.10.4.2.2).

8477 If the value of this attribute is equal to FALSE, the system is in the synchronization-unlocked
 8478 state. In this state, the *initiator_mac_address* attribute is always set to the NO-BODY value: a
 8479 value change in the MAC address field of the active-initiator MIB object does not affect the

8480 content of the *initiator_mac_address* attribute which remains at the NO-BODY value. The
 8481 default value of this variable shall be specified in the implementation specifications.

8482 NOTE 8 In the synchronization-unlocked state, the server synchronizes on any valid frame. In the synchronization
 8483 locked state, the server only synchronizes on frames issued or directed to the client system the MAC address of
 8484 which is equal to the value of the *initiator_mac_address* attribute.

8485 **4.10.3.2.15 transmission_speed**

8486 The transmission speed supported by the physical device. See also IEC 61334-5-1:2001, 3.2.2.

enum:	50 Hz	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

8487

8488

8489

8490 **Table 39 – 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.	

8491

4.10.4 S-FSK Active initiator (class_id = 51, version = 0)

8493 4.10.4.1 Overview

8494 An instance of the “S-FSK Active initiator” IC stores the data of the active initiator. The active
8495 initiator is the client system, which has last registered the server system with a CIASE Register
8496 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 (data)					

8497

4.10.4.2 Attribute description

8499 4.10.4.2.1 logical_name

8500 Identifies the “S-FSK Active initiator” object instance. See 6.2.25.

8501 4.10.4.2.2 active initiator

8502 Holds the MIB variable *active-initiator* (variable 15) specified in IEC 61334-4-512:2001, 5.6.

8503 Contains the identifiers of the active initiator, which has last registered the system with a
8504 Register request. See IEC 61334-4-511:2000, 7.2.

8505 The Initiator system is identified with its System Title, MAC address and L-SAP selector:

```
8506                         initiator_descriptor ::= structure  
8507  
8508                         {  
8509                             system_title:          octet-string,  
8510                             MAC_address:        long-unsigned,  
8511                             L_SAP_selector:    unsigned  
8512                         }
```

8513 The size and the structure of the system title may be specified in system specifications. When
8514 the system title is used as part of the initialisation vector of cryptographic algorithms, then the
8515 size shall meet the requirements applicable for the initialisation vector.

8516 The MAC_address element is used to update the *initiator-mac-address* MAC management
8517 variable when the system is configured in the synchronization-locked state. See the
8518 specification of the *initiator_mac_address* and the *synchronization_locked* attributes of the S-
8519 FSK Phy&MAC setup IC in 5.9.3.

8520 As long as the server is not registered by an active initiator, the L_SAP_selector field is set to
8521 0 and the system_title field is equal to an octet string of 0s.

8522 The default value of the initiator-descriptor is: system_title = octet-string of 0s, MAC_address
8523 = NO-BODY and L SAP selector = 0.

8524 The value of this attribute can be updated by the invocation of the
8525 *reset NEW not synchronized* method or by the CIASE Register service.

8526 **4.10.4.3 Method**8527 **4.10.4.3.1 reset_NEW_not_synchronized (data)**

8528 Holds the MIB variable *reset-NEW-not-synchronized* (variable 17) specified in IEC 61334-4-
 8529 512:2001, 5.8.

8530 Allows a client system to “reset” the server system. The submitted value corresponds to a client
 8531 MAC address. The writing is refused if:

- 8532 – the value does not correspond to a valid client MAC address or the predefined NO-BODY
 8533 address;
- 8534 – the submitted value is different from the NO-BODY address and the *synchronization_locked*
 8535 attribute is not equal to TRUE.

8536 For the description of the Intelligent Search Initiator process, see IEC 62056-8-3:2013, 10.7.

8537 When this method is invoked, the following actions are performed:

- 8538 – the system returns to the unconfigured state (UNC: MAC-address equals NEW-address).
 8539 This transition automatically causes the synchronization lost (function of the MAC sub
 8540 layer);
- 8541 – the system changes the value of the *active_initiator* attribute: the MAC_address is set to the
 8542 submitted value, the L-SAP_selector is set to the value 0 and the system_title is set to an
 8543 octet-string of 0s;
- 8544 – all AAs that can be released are released.

8545

8546 **4.10.5 S-FSK MAC synchronization timeouts (class_id = 52, version = 0)**8547 **4.10.5.1 Overview**

8548 An instance of the “S-FSK MAC synchronization timeouts” IC stores the timeouts related to the
 8549 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					

8550

8551 **4.10.5.2 Attribute description**8552 **4.10.5.2.1 logical_name**

8553 Identifies the “S-FSK synchronization timeouts” object instance. See 6.2.25.

8554 **4.10.5.2.2 search_initiator_timeout**

8555 This timeout supports the intelligent search initiator function.

8556 It defines the value of the time, expressed in seconds, during which the server system is
8557 searching for the initiator with the strongest signal.

8558 During this timeout, all initiators, which may be heard by the servers, are expected to talk.

8559 After the expiry of this timeout, the server will accept a Register request from the initiator having
8560 provided the strongest signal and it will be locked to that initiator.

8561 If the value of the timeout is equal to 0, this means that the feature is not used.

8562 The timeout is started at the beginning of the Search Initiator Phase, when the server receives
8563 the first frame with a valid initiator MAC address. The timeout is restarted when the Search
8564 Initiator Phase is over and the server locks on the initiator. During the Check Initiator Phase, it
8565 is restarted on the reception of each valid frame.

8566 A Fast synchronization may be performed if the level of signal is good enough (Level of initiator
8567 signal >= Search-Initiator-Threshold) and one of the MAC addresses (Source or Destination) is
8568 an Initiator MAC address. This means that the module (the meter) is next to a DC or next to a
8569 module that is already locked on that DC. The module locks in this case on that initiator.

8570 **4.10.5.2.3 synchronization_confirmation_timeout**

8571 Holds the MIB variable *synchronization-confirmation-timeout* (variable 6) specified in IEC
8572 61334-4-512:2001, 5.3 and IEC 61334-5-1:2001, 4.3.7.6.

8573 Defines the value of the time, expressed in seconds, after which a server system which just
8574 gets frame synchronized (detection of a data path equal to AAAA54C7 hex) will automatically
8575 lose its frame synchronization if the MAC sublayer does not identify a valid MAC frame. The
8576 timeout starts after the reception of the first four bytes of a physical frame.

8577 The value of this variable can be modified by a client system. This time-out ensures a fast
8578 desynchronization of a system, which has synchronized on a wrong physical frame. See IEC
8579 61334-5-1:2001, 3.5.3 for more details.

8580 The default value of this variable should be specified in the implementation specifications.

8581 A value equal to 0 is equivalent to cancel the use of the related
8582 *synchronization_confirmation_timeout* counter.

8583 **4.10.5.2.4 time_out_not_addressed**

8584 Holds the MIB variable *time-out-not-addressed* (variable 7) specified in IEC 61334-4-512:2001,
8585 5.3 and in IEC 61334-5-1:2001, 4.3.7.6.

8586 Defines the time, in minutes, after which a server system that has not been individually
8587 addressed:

- 8588 – returns to the non configured state (UNC: MAC-address equals NEW-address): this
8589 transition automatically involves the loss of the synchronization (function of the MAC sub
8590 layer) and releasing all AAs that can be released;
- 8591 – loses its active initiator: the MAC address of the active-initiator is set to NO-BODY, the
8592 LSAP selector is set to the value 00 and the System Title is set to an octet-string of 0s.

8593 Because broadcast addresses are not individual system addresses, the timer associated with
8594 the *time-out-not-addressed* delay ensures that a forgotten system will sooner or later return to
8595 the unconfigured state. It will be then discovered again.

8596 A forgotten system is a system, which has not been individually addressed for more than the
 8597 "time-out-not-addressed" amount of time.

8598 The default value of this variable should be specified in the implementation specifications.

8599 A value equal to 0 is equivalent to cancel the use of the related time-out-not-addressed counter.

8600 **4.10.5.2.5 time_out_frame_not_OK**

8601 Holds the MIB variable *time-out-frame-not-OK* (variable 8), specified in IEC 61334-4-512:2001,
 8602 5.3 and in IEC 61334-5-1:2001, 4.3.7.6.

8603 Defines the time, in seconds, after which a server system that has not received a properly
 8604 formed MAC frame (incorrect NS field, inconsistent number of received sub frames, false Cyclic
 8605 Redundancy Code checking) loses its frame synchronization.

8606 The default value of this variable shall be specified in the implementation specifications.

8607 A value equal to 0 is equivalent to cancel the use of the related *time-out-frame-not-OK* counter.

8608

8609 **4.10.6 S-FSK MAC counters (class_id = 53, version = 0)**

8610 **4.10.6.1 Overview**

8611 An instance of the “S-FSK MAC counters” IC stores counters related to the frame exchange,
 8612 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

8613

8614 **4.10.6.2 Attribute description**

8615 **4.10.6.2.1 logical_name**

8616 Identifies the “S-FSK MAC counters” object instance. See 6.2.25.

8617 **4.10.6.2.2 synchronization_register**

8618 Holds the MIB variable *synchronization-register* (variable 23), specified in IEC 61334-4-
 8619 512:2001, 5.8.

```
8620     array synchronization_couples
8621
8622     synchronization_couples ::= structure
8623
8624     {
8625         mac_address:          long-unsigned,
8626         synchronizations_counter: double-long-unsigned
8627     }
8628
```

8629 This variable counts the number of synchronization processes performed by the system.
 8630 Processes that lead to a synchronization loss due to the detection of a wrong initiator are
 8631 registered. The other processes that lead to a synchronization loss (time-out, management
 8632 writing) **are not registered**.

8633 This variable provides a balance sheet of the different systems on which the server system is
 8634 "potentially" able to synchronize.

8635 A synchronization process is initialized when the Management Application Entity (connection
 8636 manager) receives a MA_Sync.indication (Synchronization State = SYNCHRO_FOUND)
 8637 primitive from the MAC Sublayer Entity. This process is registered in the synchronization-
 8638 register variable only if the MA_Sync.indication (Synchronization State = SYNCHRO_FOUND)
 8639 primitive is followed by one of the three primitives:

- 8640 1) MA_Data.indication (DA, SA, MSDU) primitive;
- 8641 2) MA_Sync.indication (Synchronization State = SYNCHRO_CONF, SA, DA);
- 8642 3) MA_Sync.indication (Synchronization State = SYNCHRO_LOSS, Synchro Loss
 Cause = wrong_initiator, SA, DA)

8644 NOTE The third primitive is only generated if the server system is configured in a synchronization-locked state. See
 8645 4.10.3.

8646 Processes which lead to the generation of MA_Sync.indication (Synchronization State =
 8647 SYNCHRO_LOSS) primitives indicating synchronization loss due to:

- 8648 – the physical layer;
- 8649 – the time-out-not-addressed counter;
- 8650 – setting the mac_address attribute of the S-FSK Phy&MAC setup object to NEW; see 4.10.3;
 or
- 8652 – invoking the reset_NEW_not_synchronized method of the S-FSK Active initiator object; see
 4.10.4 (this is known as Management Writing)

8654 **are not taken** into account in this variable.

8655 For details on the MA_Sync.indication service primitive, see IEC 61334-5-1:2001, 4.1.7.1.

8656 If the synchronization process ends with one of the three primitives listed above, the
 8657 synchronization-register variable is updated by taking into account the SA and DA fields of the
 8658 primitive.

8659 The updating of the *synchronization-register* variable is carried out as follows:

- 8660 First, the Management Entity checks the SA and DA fields.
- 8661 – If one of these fields corresponds to a client MAC address (CMA) the Entity:
 - 8662 • checks if the client MAC address (CMA) appears in one of the couples contained in the synchronization-register variable;
 - 8663 • if it appears, the related synchronizations-counter subfield is incremented;
 - 8664 • if it does not appear, a new (mac-address, synchronizations-counter) couple is added. This couple is initialized to the (CMA, 1) value.
 - 8667 – 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).
- 8672 When a *synchronizations-counter* field reaches the maximum value, it automatically returns to 0 on the next increment.
- 8674 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.
- 8678 The default value of this variable is an empty array.

8679 **4.10.6.2.3 desynchronization_listing**

8680 Holds the MIB variable *desynchronization-listing* (variable 24), specified in IEC 61334-4-512:2001, 5.8.

8682 structure

```
8683       {
8684        nb_physical_layer_desynchronization:           double-long-unsigned,
8685        nb_time_out_not_addressed_desynchronization: double-long-unsigned,
8686        nb_timeout_frame_not_OK_desynchronization:    double-long-unsigned,
8687        nb_write_request_desynchronization:            double-long-unsigned,
8688        nb_wrong_initiator_desynchronization:         double-long-unsigned
8689      }
```

8692 This variable counts the number of desynchronizations that occurred depending on their cause.
 8693 On reception of synchronization loss notification, the Management Entity updates this attribute
 8694 by incrementing the counter related to the cause of the desynchronization.

8695 When one of the counters reaches the maximum value, it automatically returns to 0 on the next
 8696 increment.

8697 The default value of this variable is a structure with all elements equal to 0.

8698 **4.10.6.2.4 broadcast_frames_counter**

8699 Holds the MIB variable *broadcast-frames-counter* (variable 19) specified in IEC 61334-4-512:2001, 5.8.

```
8701           array        broadcast-couples
8702                            broadcast-couples::= structure
8703
8704
```

8705 {
 8706 source-mac-address: long-unsigned,
 8707 frames-counter: double-long-unsigned
 8708 }
 8709

8710 It counts the broadcast frames received by the server system and issued from a client system
 8711 (source-mac-address = any valid client-mac-address, destination-mac address = ALL-physical).
 8712 The number of frames is classified according to the origin of the transmitter. The counter is
 8713 incremented even if the LLC-destination-address is not valid on the server system. When the
 8714 frames-counter field reaches its maximum value, it automatically returns to 0 on the next
 8715 increment.

8716 The maximum number of broadcast-couples {source-mac-address, frames-counter} contained
 8717 in this variable should be specified in the implementation specifications. When this maximum is
 8718 reached, the updating of the variable follows a First-In-First-Out (FIFO) mechanism: only the
 8719 newest source MAC addresses are memorized.

8720 **4.10.6.2.5 repetitions_counter**

8721 Holds the MIB variable *repetitions-counter* (variable 20) specified in IEC 61334-4-512:2001,
 8722 5.8.

8723 Counts the number of repetition phases. The repetition phases following a transmission are not
 8724 counted. If the MAC sub-layer is configured in the no-repeater mode, this variable is not
 8725 updated. The repetitions counter measures the activity of the system as a repeater. A received
 8726 frame repeated five times (from CC=4 to CC=0) is counted only once in the repetitions counter
 8727 since it corresponds to one repetition phase. The counter is incremented at the beginning of
 8728 each repetition phase. When the repetitions counter reaches the maximum value, it
 8729 automatically returns to 0 on the next increment. The default value is 0.

8730 **4.10.6.2.6 transmissions_counter**

8731 Holds the MIB variable *transmissions-counter* (variable 21) specified in IEC 61334-4-512:2001,
 8732 5.8.

8733 Counts the number of transmission phases. A transmission phase is characterized by the
 8734 transmission and the repetition of a frame. A repetition phase, which follows the reception of a
 8735 frame, is not counted. The transmission counter is incremented at the beginning of each
 8736 transmission phase. A client system can write this variable to update the counter. When the
 8737 transmissions counter reaches the maximum value, it automatically returns to 0 on the next
 8738 increment. The default value is 0.

8739 **4.10.6.2.7 CRC_OK_frames_counter**

8740 Holds the MIB variable *CRC-OK-frames-counter* (variable 22) specified in IEC 61334-4-
 8741 512:2001, 5.8

8742 Counts the number of frames received with a correct Frame Check Sequence Field. When the
 8743 CRC OK frames counter field reaches the maximum value, it automatically returns to 0 on the
 8744 next increment. The default value is 0.

8745 **4.10.6.2.8 CRC_NOK_frames_counter**

8746 Counts the number of frames received with an incorrect Frame Check Sequence Field. When
 8747 the CRC NOK frames counter field reaches the maximum value, it automatically returns to 0 on
 8748 the next increment. The default value is 0.

8749 **4.10.6.3 Method**

8750 **4.10.6.3.1 reset (data)**

8751 Clears all counters.

8752 data ::= integer (0)

8753

8754 **4.10.7 IEC 61334-4-32 LLC setup (class_id = 55, version = 1)**

8755 **4.10.7.1 Overview**

8756 An instance of the “IEC 61334-4-32 LLC setup” IC holds parameters necessary to set up and
8757 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				

8758

8759 **4.10.7.2 Attribute description**

8760 **4.10.7.2.1 logical_name**

8761 Identifies the “IEC 61334-4-32 LLC setup” object instance. See 6.2.25.

8762 **4.10.7.2.2 max_frame_length**

8763 Holds the length of the LLC frame in bytes. See IEC 61334-4-32:1996, 5.1.4.

8764 For the S-FSK PLC profile, the minimum / default / maximum values are 26 / 134 / 242 bytes
8765 respectively. See IEC 61334-5-1:2001, 4.2.2.

8766 NOTE For other lower layer profiles, see the corresponding values in the relevant specification.

8767 **4.10.7.2.3 reply_status_list**

8768 Holds the MIB variable *reply-status-list* (variable 11) specified in IEC 61334-4-512:2001, 5.4.

8769 Lists the L-SAPs that have a not empty RDR (Reply Data on Request) buffer, which has not
8770 already been read. The length of a waiting L-SDU is specified in number of sub frames (different
8771 from zero). The variable is locally generated by the LLC sub layer.

8772

8773 array reply_status

8774 reply_status ::= structure

8775 {

8776 L_SAP_selector: unsigned,
8777 length_of_waiting_L_SDU: unsigned
8778

8779

8780 }

8781

8782 length_of_waiting_L_SDU in the case of the S-FSK profile is in number of sub-frames; valid
8783 values are 1 to 7.

8784

8785 **4.10.8 S-FSK Reporting system list (class_id = 56, version = 0)**

8786 **4.10.8.1 Overview**

8787 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
<i>Specific methods</i>	<i>m/o</i>				

8788

8789 **4.10.8.2 Attribute description**

8790 **4.10.8.2.1 logical_name**

8791 Identifies the “S-FSK Reporting system list” object instance. See 6.2.25.

8792 **4.10.8.2.2 reporting_system_list**

8793 Holds the MIB variable *reporting-system-list* (variable 16) specified in IEC 61334-4-512:2001,
8794 5.7.

8795 array system-title

8796

8797 system-title ::= octet-string

8798

8799 Contains the system-titles of the server systems which have made a DiscoverReport request
8800 and which have not already been registered. The list has a finite size and it is sorted upon the
8801 arrival. The first element is the newest one. Once full, the oldest ones are replaced by the new
8802 ones.

8803 The reporting system list is updated:

- 8804 – when a DiscoverReport CI_PDU is received by the server system (whatever its state: non
8805 configured or configured): the CIASE adds the reporting system-title at the beginning of the
8806 list, and verifies that it does not exist anywhere else in the list, if so it destroys the old one.
8807 A system-title can only be present once in the list;
- 8808 – when a Register CI_PDU is received by the server system (whatever its state: non
8809 configured or configured): the CIASE checks the reporting-system list. If a system-title is
8810 present in the reporting-system-list and in the Register CI-PDU, the CIASE deletes the
8811 system-title in the reporting-system-list: this system is no more considered as a reporting
8812 system.

8813

8814 **4.11 Interface classes for setting up the LLC layer for ISO/IEC 8802-2**

8815 **4.11.1 General**

8816 This subclause specifies the ICs available for setting up the ISO/IEC 8802-2 LLC layer, used in
8817 some DLMS/COSEM communication profiles, in the various types of operation.

8818 For definitions related to the ISO/IEEE 8802-2 LLC layer see IEC 62056-8-8:2020, Electricity
8819 *metering data exchange - The DLMS/COSEM suite - Part 8-8: Communication profile for*
8820 *ISO/IEC 14908 series networks*

8821 ISO/IEC 8802-2:1998, 1.4.2

8822 **4.11.2 ISO/IEC 8802-2 LLC Type 1 setup (class_id = 57, version = 0)**

8823 **4.11.2.1 Overview**

8824 An instance of the “ISO/IEC 8802-2 LLC Type 1 setup” IC holds the parameters necessary to
8825 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				

8826

8827 **4.11.2.2 Attribute description**

8828 **4.11.2.2.1 logical_name**

8829 Identifies the “ISO/IEC 8802-2 LLC Type 1 setup” object instance. See 6.2.26.

8830 **4.11.2.2.2 max_octets_ui_pdu**

8831 Refer to the appropriate MAC protocol specification for any limitation on the maximum number
8832 of octets in a UI PDU. No restrictions are imposed by the LLC sublayer. However, in the interest
8833 of having a value that all users of Type 1 LLC may depend upon, all MACs shall at least be
8834 capable of accommodating UI PDUs with information fields up to and including 128 octets in
8835 length.

8836 See ISO/IEC 8802-2:1998, 6.8.1 *Maximum number of octets* in a UI PDU.

8837

8838 **4.11.3 ISO/IEC 8802-2 LLC Type 2 setup (class_id = 58, version = 0)**

8839 **4.11.3.1 Overview**

8840 An instance of the “ISO/IEC 8802-2 LLC Type 2 setup” IC holds the parameters necessary to
8841 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	<i>m/o</i>				

8842

8843 **4.11.3.2 Attribute description**8844 **4.11.3.2.1 logical_name**

8845 Identifies the “ISO/IEC 8802-2 LLC Type 2 setup” object instance. See 6.2.26.

8846 transmit_window_size_k

8847 The transmit window size (k) shall be a data link connection parameter that can never exceed
 8848 127. It shall denote the maximum number of sequentially numbered I PDUs that the sending
 8849 LLC may have outstanding (i.e., unacknowledged). The value of k is the maximum number by
 8850 which the sending LLC send state variable V(S) can exceed the N(R) of the last received I PDU.

8851 See IEC 62056-8-8:2020, Electricity *metering data exchange - The DLMS/COSEM suite - Part*
 8852 *8-8: Communication profile for ISO/IEC 14908 series networks*

8853 ISO/IEC 8802-2:1998, 7.8.4 *Transmit window size, k*.8854 **4.11.3.2.2 receive_window_size_rw**

8855 The receive window size (RW) shall be a data link connection parameter that can never exceed
 8856 127. It shall denote the maximum number of unacknowledged sequentially numbered I PDUs
 8857 that the local LLC allows the remote LLC to have outstanding. It is transmitted in the information
 8858 field of XID (see ISO/IEC 8802-2:1998, 5.4.1.1.2) and applies to the XID sender. The XID
 8859 receiver shall set its transmit window (k) to a value less than or equal to the receive window of
 8860 the XID sender to avoid overrunning the XID sender.

8861 See IEC 62056-8-8:2020, Electricity *metering data exchange - The DLMS/COSEM suite - Part*
 8862 *8-8: Communication profile for ISO/IEC 14908 series networks*

8863 ISO/IEC 8802-2:1998, 7.8.6 *Receive window size, RW*.8864 **4.11.3.2.3 max_octets_i_pdu_n1**

8865 N1 is a data link connection parameter that denotes the maximum number of octets in an I PDU.
 8866 Refer to the various MAC descriptions to determine the precise value of N1 for a given medium
 8867 access method. LLC itself places no restrictions on the value of N1. However, in the interest of
 8868 having a value of N1 that all users of Type 2 LLC may depend upon, all MACs shall at least be
 8869 capable of accommodating I PDUs with information fields up to an including 128 octets in length.

8870 See IEC 62056-8-8:2020, Electricity *metering data exchange - The DLMS/COSEM suite - Part*
8871 *8-8: Communication profile for ISO/IEC 14908 series networks*

8872 ISO/IEC 8802-2:1998, 7.8.3 *Maximum number of octets in an I PDU, N1.*

8873 **4.11.3.2.4 max_number_transmissions_n2**

8874 N2 is a data link connection parameter that indicates the maximum number of times that a PDU
8875 is sent following the running out of the acknowledgment timer, the P-bit timer, the reject timer,
8876 or the busy-state timer.

8877 See IEC 62056-8-8:2020, Electricity *metering data exchange - The DLMS/COSEM suite - Part*
8878 *8-8: Communication profile for ISO/IEC 14908 series networks*

8879 ISO/IEC 8802-2:1998, 7.8.2 *Maximum number of transmissions, N2.*

8880 **4.11.3.2.5 acknowledgement_timer**

8881 The acknowledgment timer is a data link connection parameter that shall define the time interval
8882 during which the LLC shall expect to receive an acknowledgment to one or more outstanding I
8883 PDUs or an expected response PDU to a sent unnumbered command PDU. The unit is seconds.

8884 See IEC 62056-8-8:2020, Electricity *metering data exchange - The DLMS/COSEM suite - Part*
8885 *8-8: Communication profile for ISO/IEC 14908 series networks*

8886 ISO/IEC 8802-2:1998, 7.8.1.1 *Acknowledgement timer.*

8887 **4.11.3.2.6 p_bit_timer**

8888 The P-bit timer is a data link connection parameter that shall define the time interval during
8889 which the LLC shall expect to receive a PDU with the F bit set to “1” in response to a sent Type
8890 2 command with the P bit set to “1”. The unit is seconds.

8891 See IEC 62056-8-8:2020, Electricity *metering data exchange - The DLMS/COSEM suite - Part*
8892 *8-8: Communication profile for ISO/IEC 14908 series networks*

8893 ISO/IEC 8802-2:1998, 7.8.1.2 *P-bit timer.*

8894 **4.11.3.2.7 reject_timer**

8895 The reject timer is a data link connection parameter that shall define the time interval during
8896 which the LLC shall expect to receive a reply to a sent REJ PDU. The unit is seconds.

8897 See IEC 62056-8-8:2020, Electricity *metering data exchange - The DLMS/COSEM suite - Part*
8898 *8-8: Communication profile for ISO/IEC 14908 series networks*

8899 ISO/IEC 8802-2:1998, 7.8.1.3 *Reject timer.*

8900 **4.11.3.2.8 busy_state_timer**

8901 The busy-state timer is a data link connection parameter that shall define the timer interval
8902 during which the LLC shall wait for an indication of the clearance of a busy condition at the
8903 other LLC. The unit is seconds.

8904 See IEC 62056-8-8:2020, Electricity *metering data exchange - The DLMS/COSEM suite - Part*
8905 *8-8: Communication profile for ISO/IEC 14908 series networks*

8906 ISO/IEC 8802-2:1998, 7.8.1.4 *Busy-state timer*.

8907

8908

8909 **4.11.4 ISO/IEC 8802-2 LLC Type 3 setup (class_id = 59, version = 0)**

8910 **4.11.4.1 Overview**

8911 An instance of the “ISO/IEC 8802-2 LLC Type 3 setup” IC holds the parameters necessary to
8912 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				

8913

8914 **4.11.4.2 Attribute description**

8915 **4.11.4.2.1 logical_name**

8916 Identifies the “ISO/IEC 8802-2 LLC Type 3 setup” object instance. See 6.2.26.

8917 **4.11.4.2.2 max_octets_acn_pdu_n3**

8918 N3 is a logical link parameter that denotes the maximum number of octets in an ACn command
8919 PDU. Refer to the various MAC descriptions to determine the precise value of N3 for a given
8920 medium access method. LLC places no restrictions on the value of N3.

8921 See IEC 62056-8-8:2020, Electricity **metering data exchange - The DLMS/COSEM suite - Part**
8922 **8-8: Communication profile for ISO/IEC 14908 series networks**

8923 ISO/IEC 8802-2:1998, 8.6.2 *Maximum number of octets in an ACn command PDU, N3*.

8924 **4.11.4.2.3 max_number_transmissions_n4**

8925 N4 is a logical link parameter that indicates the maximum number of times that an ACn
8926 command PDU is sent by LLC trying to accomplish a successful information exchange.
8927 Normally, N4 is set large enough to overcome the loss of a PDU due to link error conditions. If
8928 the medium access control sublayer has its own retransmission capability, the value of N4 may
8929 be set to one so that LLC does not itself requeue a PDU to the medium access control sublayer.

8930 See IEC 62056-8-8:2020, Electricity **metering data exchange - The DLMS/COSEM suite - Part**
8931 **8-8: Communication profile for ISO/IEC 14908 series networks**

8932 ISO/IEC 8802-2:1998, 8.6.1 *Maximum number of transmissions, N4*.

8933 **4.11.4.2.4 acknowledgement_time_t1**

8934 The acknowledgment time is a logical link parameter that determines the period of the
8935 acknowledgment timers, and as such shall define the time interval during which the LLC shall
8936 expect to receive an ACn response PDU from a specific LLC from which the LLC is awaiting a
8937 response PDU. The acknowledgment time shall take into account any delay introduced by the
8938 MAC sublayer and whether the timer is started at the beginning or at the end of the sending of
8939 the ACn command PDU by the LLC. The proper operation of the procedure shall require that
8940 the acknowledgment time be greater than the normal time between the sending of an ACn
8941 command PDU and the reception of the corresponding ACn response PDU. If the medium
8942 access control sublayer performs its own retransmissions and if the logical link parameter N4
8943 is set to one to prevent LLC from re-queuing a PDU, then the acknowledgment time T1 may be
8944 set to infinity, making the acknowledgment timers unnecessary.

8945 The unit is seconds. Infinity is indicated by all bits set to 1.

8946 See IEC 62056-8-8:2020, Electricity *metering data exchange - The DLMS/COSEM suite - Part*
8947 *8-8: Communication profile for ISO/IEC 14908 series networks*

8948 ISO/IEC 8802-2:1998, 8.6.4, *Acknowledgement time, T1*.

8949

8950 **4.11.4.2.5 receive_lifetime_var_t2**

8951 This time value is a logical link parameter that determines the period of all of the receive variable
8952 lifetime timers. T2 shall be longer by a margin of safety than the longest possible period during
8953 which the first transmission and all retries of a single PDU may occur. The margin of safety
8954 shall take into account anything affecting LLCs perception of the arrival time of PDUs, such as
8955 LLC response time, timer resolution, and variations in the time required for the medium access
8956 control sublayer to pass received PDUs to LLC.

8957 If the destruction of the received state variables is not desired, the value of time T2 may be set
8958 to infinity. In this case the receive variable lifetime timer need not be implemented.

8959 The unit is seconds. Infinity is indicated by all bits set to 1.

8960 See IEC 62056-8-8:2020, Electricity *metering data exchange - The DLMS/COSEM suite - Part*
8961 *8-8: Communication profile for ISO/IEC 14908 series networks*

8962 ISO/IEC 8802-2:1998, 8.6.5 *Receive lifetime variable, T2*.

8963 **4.11.4.2.6 transmit_lifetime_var_t3**

8964 This time value is a logical link parameter that determines the minimum lifetime of the transmit
8965 sequence state variables. T3 shall be longer by a margin of safety than

8966 a) the logical link variable T2 at stations to which ACn commands are sent; and

8967 b) the longest possible lifetime of an ACn command-response pair. The lifetime of an ACn
8968 command-response pair shall take into account the sum of processing time, queuing delays,
8969 and transmission time for the command and response PDUs at the local and remote stations.

8970 If the destruction of the transmit state variables is not desired, the value of time T3 may be set
8971 to infinity. Note, if the receive variable lifetime parameter, T2 is set to infinity at remote stations

8972 to which ACn commands are sent, then the T3 parameter shall be set to infinity at the local
 8973 station.

8974 The unit is seconds. Infinity is indicated by all bits set to 1.

8975 See IEC 62056-8-8:2020, Electricity *metering data exchange - The DLMS/COSEM suite - Part*
 8976 *8-8: Communication profile for ISO/IEC 14908 series networks*

8977 ISO/IEC 8802-2:1998, 8.6.6 *Transmit lifetime variable, T3*.

8978

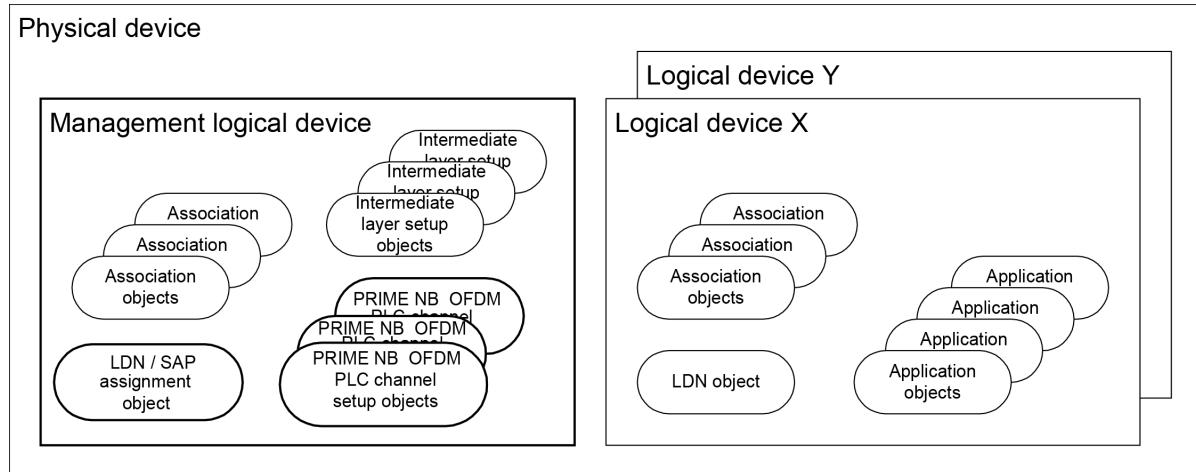
8979 **4.12 Interface classes for setting up and managing DLMS/COSEM narrowband OFDM**
 8980 **PLC profile for PRIME networks**

8981 **4.12.1 Overview**

8982 See also Annex D.

8983 COSEM objects for data exchange using narrowband OFDM PLC profile for PRIME networks,
 8984 if implemented, shall be located in the Management Logical Device of COSEM servers.

8985 Figure 28 shows an example with a COSEM physical device comprising three logical devices.



8986

IEC

8987 **Figure 28 – Object model of DLMS/COSEM servers**

8988 Each logical device shall contain a Logical Device Name (LDN) object.

8989 NOTE As in this example there is more than one logical device, the mandatory Management logical device contains
 8990 a "SAP Assignment" object instead of a Logical Device object.

8991 Each logical device contains one or more "Association" objects, one for each client supported.

8992 The management logical device contains the setup objects of the physical and MAC layers of
 8993 narrowband OFDM PLC profile for PRIME networks as well as setup objects for the intermediate
 8994 layer(s). It may contain further application objects.

8995 The other logical devices, in addition to the "Association" and Logical Device Name objects
 8996 mentioned above, contain further application objects, holding parameters and measurement
 8997 values.

- 8998 To set up and manage the 61334-4-32 LLC SSCS, one IC is specified:
- 8999 • “61334-4-32 LLC SSCS setup”, see 4.12.3
- 9000 To manage the PRIME NB OFDM PLC physical layer (PhL), one IC is specified:
- 9001 • “PRIME NB OFDM PLC Physical layer counters”, see 4.12.5;
- 9002 To set up and manage the PLC PRIME OFDM MAC layer, four ICs are specified:
- 9003 • “PRIME NB OFDM PLC MAC setup”: see 4.12.6;
- 9004 • “PRIME NB OFDM PLC MAC functional parameters”: see 4.12.7;
- 9005 • “PRIME NB OFDM PLC MAC counters”: see 4.12.8;
- 9006 • “PRIME NB OFDM PLC MAC network administration data”: see 4.12.9.
- 9007 For application identification, one IC is specified:
- 9008 • “PRIME NB OFDM PLC Application identification”, see 4.12.11.

9009 **4.12.2 Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes**

9010 ITU-T G.9904:2012 defines variables in Table 10-1 and Table 10-2 for PHY PIB attributes Table
9011 10-3 to Table 10-8 for MAC PIB attributes and Table 10-9 for Applications PIB attributes.

9012 Table 40 shows the mapping of PRIME NB OFDM PLC PIB attributes to attributes of COSEM
9013 ICs. Only variables related to the switch and Terminal nodes are mapped. Variables relevant
9014 for the base node are not mapped, because the base node acts as a client regarding the
9015 distribution network.

9016 **Table 40 – Mapping of PRIME NB OFDM PLC PIB attributes to**
9017 **COSEM IC attributes**

Name	Identifier	Interface class	class_id / attribute
PHY PIB attributes – PHY read-only variable that provide statistical information¹			
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
PHY read-only parameters, providing information on specific implementation²			
phyTxQueueLen	0x00B0		Not modelled
phyRxQueueLen	0x00B1		
phyTxProcessingDelay	0x00B2		
phyRxProcessingDelay	0x00B3		
PhyAgcMinGain	0x00B4		
PhyAgcStepValue	0x00B5		
PhyAgcStepNumber	0x00B6		
MAC read-write variables, read-only variables³			
macMinSwitchSearchTime	0x0010		82 / Attr. 2
macMaxPromotionPdu	0x0011		82 / Attr. 3

Name	Identifier	Interface class	class_id / attribute
macMaxPromotionPduTxPeriod	0x0012	PRIME NB OFDM PLC MAC setup (class_id = 82, version = 0)	82 / Attr. 4
macBeaconsPerFrame	0x0013		82 / Attr. 5
macSCPMaxTxAttempts	0x0014		82 / Attr. 6
macCtlReTxTimer	0x0015		82 / Attr. 7
macMaxCtlReTx	0x0018		82 / Attr. 8
macEMASMOOTHING	0x0019		Not modelled
macSCPRBO	0x0016		Not modelled
macSCPChSenseCount	0x0017		Not modelled
MAC read-only variables that provide functional information⁴			
macLNID	0x0020	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 information⁵			
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
Read-only lists, made available by MAC layer through management interface⁶			
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 attributes⁷			
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

Name	Identifier	Interface class	class_id / attribute
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.			

9018

9019 4.12.3 61334-4-32 LLC SSCS setup (class_id = 80, version = 0)

9020 4.12.3.1 Overview

9021 An instance of the “61334-4-32 LLC SSCS” (Service Specific Convergence Sublayer, 432 CL)
 9022 setup IC holds addresses that are provided by the base node during the opening of the
 9023 convergence layer, as a response to the establish request of the service node. They allow the
 9024 service node to be part of the network managed by the base node.

61334-4-32 LLC SSCS 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				

9025

9026 4.12.3.2 Attribute description

9027 4.12.3.2.1 logical_name

9028 Identifies the “61334-4-32 LLC SSCS setup object instance”. See 6.2.27.

9029 4.12.3.2.2 service_node_address

9030 Holds the value of the address assigned to the service node during its registration by the base
 9031 node.

9032 After deregistration, the value of this address is NEW, meaning 0xFFE.

9033 4.12.3.2.3 base_node_address

9034 Holds the value of the base node address to which the service node is registered.

9035 After deregistration this address is 0.

9036 **4.12.3.3 Method**9037 **4.12.3.3.1 reset (data)**

9038 This method is used for deallocating the service node address.

9039 The value of the *service_node_address* becomes NEW and the value of the
9040 *base_node_address* becomes 0.

9041

9042 **4.12.4 PRIME NB OFDM PLC Physical layer parameters**

9043 The physical layer parameters are not modelled.

9044 **4.12.5 PRIME NB OFDM PLC Physical layer counters (class_id = 81, version = 0)**9045 **4.12.5.1 Overview**

9046 An instance of the “PRIME NB OFDM PLC Physical layer counters” IC stores counters related
9047 to the physical layers exchanges. The objective of these counters is to provide statistical
9048 information for management purposes.

9049 The attributes of instances of this IC shall be read only. They can be reset using the reset
9050 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				

9051

9052 **4.12.5.2 Attribute description**

9053 logical_name

9054 Identifies the “PRIME NB OFDM PLC Physical layer counters” object instance. See 6.2.27.

9055 **4.12.5.2.1 phy_stats_crc_incorrect_count**

9056 PIB attribute 0x00A0: Number of bursts received on the physical layer for which the CRC was
9057 incorrect.

9058 **4.12.5.2.2 phy_stats_crc_failed_count**

9059 PIB attribute 0x00A1: Number of bursts received on the physical layer for which the CRC was
9060 correct, but the Protocol field of PHY header had invalid value. This count would reflect number
9061 of times corrupt data was received and the CRC calculation failed to detect it.

9062 **4.12.5.2.3 phy_stats_tx_drop_count**

9063 PIB attribute 0x00A2: Number of times when PHY layer received new data to transmit
 9064 (PHY_DATA.request) and had to either overwrite on existing data in its transmit queue or drop
 9065 the data in new request due to full queue.

9066 **4.12.5.2.4 phy_stats_rx_drop_count**

9067 PIB attribute 0x00A3: Number of times when the PHY layer received new data on the channel
 9068 and had to either overwrite on existing data in its receive queue or drop the newly received data
 9069 due to full queue.

9070 NOTE When a counter reaches the maximum value (0xFFFF), it is automatically rolled-over.

9071 **4.12.5.3 Method**

9072 **4.12.5.3.1 reset (data)**

9073 This method is used for resetting all the counters held by an instance of this interface.

9074

9075 **4.12.6 PRIME NB OFDM PLC MAC setup (class_id = 82, version = 0)**

9076 **4.12.6.1 Overview**

9077 An instance of the “PRIME NB OFDM PLC MAC setup” IC holds the necessary parameters to
 9078 set up and manage the PRIME NB OFDM PLC MAC layer.

9079 These attributes influence the functional behaviour of an implementation. These attributes may
 9080 be defined external to the MAC, typically by the management entity and implementations may
 9081 allow changes to their values during normal running, i.e. even after the device start-up sequence
 9082 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	<i>m/o</i>				

9083

9084 **4.12.6.2 Attribute description**

9085 **4.12.6.2.1 logical_name**

9086 Identifies the “PRIME NB OFDM PLC MAC setup” object instance. See 6.2.27..

9087 **4.12.6.2.2 mac_min_switch_search_time**

9088 PIB attribute 0x0010: Minimum time for which a service node in *Disconnected* status should
 9089 scan the channel for beacons before it can broadcast PNPDUs. This attribute is not maintained
 9090 in base nodes.

9091 The unit of this attribute is seconds.

9092 **4.12.6.2.3 mac_max_promotion_pdu**

9093 PIB attribute 0x0011: Maximum number of PNPDUs that may be transmitted by a service node
 9094 in a period of *mac_promotion_pdu_tx_period* seconds. This attribute is not maintained in base
 9095 nodes.

9096 **4.12.6.2.4 mac_promotion_pdu_tx_period**

9097 PIB attribute 0x0012: Time quantum for limiting the number of PNPDUs transmitted from a
 9098 service node. No more than *mac_max_promotion_pdu* may be transmitted in a period of
 9099 *mac_promotion_pdu_tx_period*.

9100 The unit of this attribute is seconds.

9101 **4.12.6.2.5 mac_beacons_per_frame**

9102 PIB attribute 0x0013: Maximum number of beacon slots that may be provisioned in a frame.
 9103 This attribute is maintained in base nodes.

9104 **4.12.6.2.6 mac_scp_max_tx_attempts**

9105 PIB attribute 0x0014: Number of times the CSMA algorithm would attempt to transmit requested
 9106 data when a previous attempt was withheld due to PHY indicating channel busy.

9107 **4.12.6.2.7 mac_ctl_re_tx_timer**

9108 PIB attribute 0x0015: Number of seconds for which a MAC entity waits for acknowledgement of
 9109 receipt of MAC control packet from its peer entity. On expiry of this time, the MAC entity may
 9110 retransmit the MAC control packet.

9111 The unit of this attribute is seconds.

9112 **4.12.6.2.8 mac_max_ctl_re_tx**

9113 PIB attribute 0x0018: Maximum number of times a MAC entity will try to retransmit an
 9114 unacknowledged MAC control packet. If the retransmit count reaches this maximum, the MAC
 9115 entity shall abort further attempts to transmit the MAC control packet.

9116 NOTE When a counter reaches the maximum value (0xFFFF), it is automatically rolled-over.

9117

9118 **4.12.7 NB OFDM PLC MAC functional parameters (class_id = 83 version = 0)**9119 **4.12.7.1 Overview**

9120 The attributes of an instance of the “PRIME NB OFDM PLC MAC functional parameters” IC
 9121 belong to the functional behaviour of MAC. They provide information on specific aspects.

9122 The attributes of instances of this IC shall be read only.

9123

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				

9124

9125 **4.12.7.2 Attribute description**9126 **4.12.7.2.1 logical_name**

9127 Identifies the “PRIME NB OFDM PLC MAC functional parameters” object instance. See 6.2.27.

9128 **4.12.7.2.2 mac_LNID**

9129 PIB attribute 0x0020: LNID allocated to this node at time of its registration.

9130 **4.12.7.2.3 mac_LSID**9131 PIB attribute 0x0021: LSID allocated to this node at the time of its promotion. This attribute is
9132 not maintained if the node is in a Terminal functional state.9133 **4.12.7.2.4 mac_SID**9134 PIB attribute 0x0022: SID of the switch node through which this node is connected to the
9135 subnetwork. This attribute is not maintained in a base node.9136 **4.12.7.2.5 mac_SNA**9137 PIB attribute 0x0023: Subnetwork address to which this node is registered. The base node
9138 returns the SNA it is using.9139 **4.12.7.2.6 mac_state**

9140 PIB attribute 0x0024: Present functional state of the node.

9141 enum:
 9142 (0) Disconnected,
 9143 (1) Terminal,
 9144 (2) Switch,
 9145 (3) Base
 9146

9147 **4.12.7.2.7 mac_scp_length**

9148 PIB attribute 0x0025: The SCP length, in symbols, in present frame.

9149 **4.12.7.2.8 mac_node_hierarchy_level**

9150 PIB attribute 0x0026: Level of this node in subnetwork hierarchy.

9151 **4.12.7.2.9 mac_beacon_slot_count**

9152 PIB attribute 0x0027: Number of beacon slots provisioned in present frame structure.

9153 **4.12.7.2.10 mac_beacon_rx_slot**

9154 PIB attribute 0x0028: Beacon slot in which this device's switch node transmits its beacon. This
 9155 attribute is not maintained in a base node.

9156 **4.12.7.2.11 mac_beacon_tx_slot**

9157 PIB attribute 0x0029: Beacon slot in which this device transmits its beacon. This attribute is not
 9158 maintained in service nodes that are in a *Terminal* functional state.

9159 **4.12.7.2.12 mac_beacon_rx_frequency**

9160 PIB attribute 0x002A: Number of frames between receptions of two successive beacons. A
 9161 value of 0x0 indicates beacons are received in every frame. This attribute is not maintained in
 9162 a base node.

9163 **4.12.7.2.13 mac_beacon_tx_frequency**

9164 PIB attribute 0x002B: Number of frames between transmissions of two successive beacons. A
 9165 value of 0x0 indicates beacons are transmitted in every frame. This attribute is not maintained
 9166 in service nodes that are in a *Terminal* functional state.

9167 **4.12.7.2.14 mac_capabilities**

9168 PIB attribute 0x002C: This attribute defines the capabilities of the node. It is a bitmap each bit
 9169 defining a capability.

9170	Bit 0: Switch Capable
9171	Bit 1: Packet Aggregation
9172	Bit 2: Contention Free Period
9173	Bit 3: Direct connection
9174	Bit 4: Multicast
9175	Bit 5: PHY Robustness Management
9176	Bit 6: ARQ
9177	Bit 7: Reserved for future use
9178	Bit 8: Direct Connection Switching
9179	Bit 9: Multicast Switching Capability
9180	Bit 10: PHY Robustness Management Switching Capability
9181	Bit 11: ARQ Buffering Switching Capability
9182	Bit 12 to 15: Reserved for future use
9183	

9184

9185 **4.12.8 PRIME NB OFDM PLC MAC counters (class_id = 84, version = 0)**9186 **4.12.8.1 Overview**

9187 An instance of the “PRIME NB OFDM PLC MAC counters” IC stores statistical information on
 9188 the operation of the MAC layer for management purposes. The attributes of instances of this IC
 9189 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				

9190

9191 **4.12.8.2 Attribute description**9192 **4.12.8.2.1 logical_name**

9193 Identifies the “PRIME NB OFDM PLC MAC counters” object instance. See 6.2.27.

9194 **4.12.8.2.2 mac_tx_data_pkt_count**

9195 PIB attribute 0x0040: Count of successfully transmitted MSDUs.

9196 **4.12.8.2.3 mac_rx_data_pkt_count**9197 PIB attribute 0x0041: Count of successfully received MSDUs whose destination address was
 9198 this node.9199 **4.12.8.2.4 mac_tx_ctrl_pkt_count**

9200 PIB attribute 0x0042: Count of successfully transmitted MAC control packets.

9201 **4.12.8.2.5 mac_rx_ctrl_pkt_count**9202 PIB attribute 0x0043: Count of successfully received MAC control packets whose destination
 9203 was this node.9204 **4.12.8.2.6 mac_csma_fail_count**

9205 PIB attribute 0x0044: Count of failed CSMA transmit attempts.

9206 **4.12.8.2.7 mac_csma_ch_busy_count**

9207 PIB attribute 0x0045: Count of number of times this node has to back off SCP transmission due
 9208 to channel busy state.

9209 NOTE When a counter reaches the maximum value (0xFFFFFFFF), it is automatically rolled-over.

9210 **4.12.8.3 Method**9211 **4.12.8.3.1 reset (data)**

9212 This method is used for resetting all the counters held by an instance of this interface class.

9213 data::= integer (0)

9214

9215 **4.12.9 PRIME NB OFDM PLC MAC network administration data (class_id = 85, version
 9216 = 0)**

9217 **4.12.9.1 Overview**

9218 This IC holds the parameters related to the management of the devices connected to the
 9219 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				

9220

9221 **4.12.9.2 Attribute description**9222 **4.12.9.2.1 logical_name**

9223 Identifies the “PRIME NB OFDM PLC MAC network administration data” object instance. See
 9224 6.2.27.

9225 **4.12.9.2.2 mac_list_multicast_entries**

9226 PIB attribute 0x0052: List of entries in multicast switching table. This list is not maintained in
 9227 service nodes in a *Terminal* functional state.

9228 mac_list_multicast_entries_type::= array mac_list_multicast_entries_element
 9229 mac_list_multicast_entries_element::= structure
 9230

```

9231
9232      {
9233          mcast_entry_LCID:      integer, -- LCID of multicast group
9234          mcast_entry_members: long     -- number of child nodes
9235      }
9236

```

9237 The number of child nodes is the number of the members of this group, including the Node itself.

9238 **4.12.9.2.3 mac_list_switch_table**

9239 PIB attribute 0x0053: Switch table. This table is not maintained by service nodes in a *Terminal*
9240 state.

```

9241          mac switch_table ::= array    stbl_entry_LSID
9242          stbl_entry_LSID ::=        long   -- SID of attached Switch node
9243
9244

```

9245 **4.12.9.2.4 mac_list_direct_table**

9246 PIB attribute 0x0055: Direct table.

```

9247          mac_direct_table ::= array    mac_direct_table_element
9248
9249          mac_direct_table_element ::= structure
9250          {
9251              dconn_entry_src_SID:    long,
9252              dconn_entry_src_LNID:   long,
9253              dconn_entry_src_LCID:   long,
9254              dconn_entry_dst_SID:   long,
9255              dconn_entry_dst_LNID:  long,
9256              dconn_entry_dst_LCID:  long,
9257              dconn_entry_DID:       octet-string (size 6 bytes)
9258          }
9259
9260

```

9261 Where:

- 9262 – dconn_entry_src_SID is the SID of switch through which the source service node is
9263 connected;
- 9264 – dconn_entry_src_LNID is the NID allocated to the source service node;
- 9265 – dconn_entry_src_LCID is the LCID allocated to this connection at the source;
- 9266 – dconn_entry_dst_SID is the SID of the switch through which the destination service node is
9267 connected;
- 9268 – dconn_entry_dst_LNID is the NID allocated to the destination service node;
- 9269 – dconn_entry_dst_LCID is the LCID allocated to this connection at the destination;
- 9270 – dconn_entry_DID is the EUI-48 of the direct switch.

9271 **4.12.9.2.5 mac_list_available_switches**

9272 PIB attribute 0x0056: List of switch nodes whose beacons are received.

```

9273          mac_list_available_switches ::= array  mac_list_available_switches_element
9274
9275          mac_list_available_switches_element ::= structure
9276
9277          {
9278              slist_entry_SNA:        octet-string (size 6 bytes),
9279              slist_entry_LSID:       long,
9280              slist_entry_level:      integer,

```

```

9281             slist_entry_rx_level: integer,
9282             slist_entry_rx_snr: integer
9283         }

```

9284 Where:

- 9285 – slist_entry_SNA is EUI-48 of the subnetwork;
- 9286 – slist_entry_LSID is SID of this switch;
- 9287 – slist_entry_level is level of this switch in subnetwork hierarchy;
- 9288 – slist_entry_rx_level is the received signal level for this Switch;
- 9289 – slist_entry_rx_snr is the signal to noise ratio for this switch.

9290 **4.12.9.2.6 mac_list_phy_comm**

9291 PIB attribute 0x0057: List of PHY communication parameters. It is maintained in every node.
 9292 For terminal nodes it contains only one entry for the switch the node is connected through. For
 9293 other nodes is contains also entries for every directly connected child node.

```

9294     mac_list_phy_comm ::= array      phy_comm_element
9295
9296     phy_comm_element ::= structure
9297
9298     {
9299       phy_Comm_EUI:          octet-string,
9300       phy_Comm_Tx_Pwr:      integer,
9301       phy_Comm_Tx_Cod:      integer,
9302       phy_Comm_Rx_Cod:      integer,
9303       phy_Comm_Rx_Lvl:      integer,
9304       phy_Comm_SNR:         integer,
9305       phy_Comm_Tx_Pwr_Mod: integer,
9306       phy_Comm_Tx_Cod_Mod: integer,
9307       phy_Comm_Rx_Cod_Mod: integer
9308     }

```

9309 Where:

- 9310 – phy_Comm_EUI is the EUI-48 of the other device;
- 9311 – phy_Comm_Tx_Pwr is the Tx power of GPDUs sent to the device;
- 9312 – phy_Comm_Tx_Cod is the Tx coding of GPDUs sent to the device;
- 9313 – phy_Comm_Rx_Cod is the Rx coding of GPDUs received from the device;
- 9314 – phy_Comm_Rx_Lvl is the Rx power level of GPDUs received from the device;
- 9315 – phy_Comm_SNR is the SNR of GPDUs received from the device;
- 9316 – phy_Comm_Tx_Pwr_Mod is the number of times the Tx power was modified;
- 9317 – phy_Comm_Tx_Cod_Mod is the number of times the Tx coding was modified;
- 9318 – phy_Comm_Rx_Cod_Mod is the number of times the Rx coding was modified.
- 9319 –

9320 **4.12.9.3 Method**

9321 **4.12.9.3.1 reset (data)**

9322 This method is used for resetting all the entries (to an array of 0 elements) of the attributes 2
 9323 to 6 of the instance of this interface class.

```

9324     data ::= integer (0)

```

9325

9326 **4.12.10 PRIME NB OFDM PLC MAC address setup (class_id = 43, version = 0)**

9327 An instance of the MAC address setup IC holds the EUI-48 MAC address of the device. The
 9328 size of this octet string is 6 due to the fact that this address is a EUI-48 and is unique. See also
 9329 4.9.4 and 6.2.27.

9330 **4.12.11 PRIME NB OFDM PLC Application identification (class_id = 86, version = 0)**

9331 **4.12.11.1 Overview**

9332 An instance of the “PRIME NB OFDM PLC Application identification IC” holds identification
 9333 information related to administration and maintenance of PRIME NB OFDM PLC devices. They
 9334 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				

9335

9336 **4.12.11.2 Attribute description**

9337 **4.12.11.2.1 logical_name**

9338 Identifies the device setup object instance. See 6.2.27.

9339 **4.12.11.2.2 firmware_version**

9340 PIB attribute 0x0075: Textual description of the firmware version running on the device.

9341 **4.12.11.2.3 vendor_id**

9342 PIB attribute 0x0076: Unique vendor identifier assigned by PRIME Alliance.

9343 **4.12.11.2.4 product_id**

9344 PIB attribute 0x0077: Vendor assigned unique identifier for specific product.

9345

9346 **4.13 Interface classes for setting up and managing the DLMS/COSEM narrowband**
 9347 **OFDM PLC profile for G3-PLC networks**

9348 **4.13.1 Overview**

9349 This subclause 5.12 specifies version 1 of interface classes for setting up and managing the
 9350 MAC and 6LoWPAN Adaptation layers of the DLMS/COSEM G3-PLC profile, based on
 9351 ITU-T G.9903:2014.

9352 NOTE 1 The use of version 0 of these interface classes based on 3GPP TS 36.321 V15.5.0 (2019-05) – see 5.12.2
 9353 to 5.12.4 – is deprecated.

9354 For this purpose, the elements of the PAN Information Base (PIB) have been mapped to
 9355 attributes of COSEM ICs.

9356 COSEM objects for data exchange using G3-PLC, if implemented, shall be located in the
 9357 Management Logical Device of COSEM servers.

9358 To set up and manage the DLMS/COSEM G3-PLC profile layers (including PHY,
 9359 IEEE 802.15.4:2006 MAC and 6LoWPAN), three ICs are specified:

- 9360 • “G3-PLC MAC layer counters”, see 4.13.3;
- 9361 • “G3-PLC MAC setup”, see 4.13.4;
- 9362 • “G3-PLC 6LoWPAN adaptation layer setup”, see 4.13.5.

9363 An instance of the existing COSEM interface class “MAC address” (class_id = 43, version = 0)
 9364 is needed to indicate the EUI-48 MAC address of the G3-PLC modem (corresponding to
 9365 aExtendedAddress constant in IEEE 802.15.4:2006).

9366 IPv6 configuration is provided by an instance of “IPv6 setup” class.

9367 NOTE 2 The PHY layer of ITU-T G.9903:2014 is out of scope of the G3-PLC setup ICs.

9368 **4.13.2 Mapping of G3-PLC PIB attributes to COSEM IC attributes**

9369 In terms of IEEE 802.15.4:2006, a meter is a Reduced Function Device (RFD) while a
 9370 concentrator / Neighbourhood Network Access Point (NNAP) is a Full Function Device (FFD) /
 9371 PAN coordinator. In terms of DLMS/COSEM the meter is the server and the concentrator /
 9372 NNAP is the client (or an agent for a client).

9373 As COSEM models only the server and not the client, the G3-PLC setup classes concern only
 9374 the RFD (Reduced Function Device) and not the PAN coordinator.

9375 Table 41 shows the mapping of G3-PLC PIB attributes to attributes of COSEM interface classes.

9376 **Table 41 – 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 information¹			
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

Name	Identifier	Interface class	class_id / attribute
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
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 = 2)	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

Name	Identifier	Interface class	class_id / attribute
adp_disable_default_routing	0xF0		92 / Att. 18
adp_device_type	0x10		92 / Att. 19
<p>¹ See ITU-T G.9903:2014, 9.3.6.2.2 and 9.3.6.2.3.</p> <p>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>.</p> <p>³ See ITU-T G.9903:2014, 9.4.1.1.</p> <p>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>.</p>			
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.			

9377

9378 4.13.3 G3-PLC MAC layer counters (class_id = 90, version = 1)

9379 4.13.3.1 Overview

9380 An instance of the “G3-PLC MAC layer counters” IC stores counters related to the MAC layer
 9381 exchanges. The objective of these counters is to provide statistical information for management
 9382 purposes.

9383 The attributes of instances of this IC shall be read only. They can be reset using the reset
 9384 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				

9385

9386 4.13.3.2 Attribute description

9387 NOTE When a counter reaches the maximum value (0xFFFFFFFF), it's automatically rolled-over.

9388 4.13.3.2.1 logical_name

9389 Identifies the “G3-PLC MAC layer counters” object instance. See 6.2.28.

9390 4.13.3.2.2 mac_Tx_data_packet_count

9391 PIB attribute 0x0101: Statistic counter of successfully transmitted data packets (MSDUs).

9392 **4.13.3.2.3 mac_Rx_data_packet_count**

9393 PIB attribute 0x0102: Statistic counter of successfully received data packets (MSDUs).

9394 **4.13.3.2.4 mac_Tx_cmd_packet_count**

9395 PIB attribute 0x0103: Statistic counter of successfully transmitted command packets.

9396 **4.13.3.2.5 mac_Rx_cmd_packet_count**

9397 PIB attribute 0x0104: Statistic counter of successfully received command packets.

9398 **4.13.3.2.6 mac_CSMA_fail_count**

9399 PIB attribute 0x0105: Counts the number of times when CSMA backoffs reach
9400 macMaxCSMABackoffs.

9401 **4.13.3.2.7 mac_CSMA_no_ACK_count**

9402 PIB attribute 0x0106: Counts the number of times when an ACK is not received while
9403 transmitting a unicast data frame (The loss of ACK is attributed to collisions).

9404 **4.13.3.2.8 mac_bad_CRC_count**

9405 PIB attribute 0x0109: Statistic counter of the number of frames received with bad CRC.

9406 **4.13.3.2.9 mac_Tx_data_broadcast_count**

9407 PIB attribute 0x0108: Statistic counter of the number of broadcast frames sent.

9408 **4.13.3.2.10 mac_Rx_data_broadcast_count**

9409 PIB attribute 0x0107: Statistic counter of successfully received broadcast packets.

9410 **4.13.3.3 Method**

9411 **4.13.3.3.1 reset (data)**

9412 This method forces a reset of the object. By invoking this method, the value of all counters is
9413 set to 0.

9414 data::= integer (0)

9415

9416 **4.13.4 G3-PLC MAC setup (class_id = 91, version = 1)**

9417 **4.13.4.1 Overview**

9418 An instance of the “G3-PLC MAC setup” IC holds the necessary parameters to set up and
9419 manage the G3-PLC IEEE 802.15.4:2006 MAC sub-layer.

9420 These attributes influence the functional behaviour of an implementation. Implementations may
9421 allow changes to the attributes during normal running, i.e. even after the device start-up
9422 sequence has been executed.

9423

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			0x00000 0000FF FFFFFFF 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	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				x + 0xB0

9424

9425 **4.13.4.2 Attribute description**

9426 **4.13.4.2.1 logical_name**

9427 Identifies the “G3-PLC MAC setup” object instance. See 6.2.28.

9428 **4.13.4.2.2 mac_short_address**

9429 PIB attribute 0x0053: The 16-bit address the device is using to communicate through the PAN.
 9430 Its value shall be equal to 0xFFFF when the device does not have a short address. An
 9431 associated device necessarily has a short address, so that a device cannot be in the state
 9432 where it is associated but does not have a short address.

9433 **4.13.4.2.3 mac_RC_coord**

9434 PIB attribute 0x010F: Route cost to coordinator, to be used in the beacon payload as
 9435 RC_COORD

9436 4.13.4.2.4 mac_PAN_id

9437 PIB attribute 0x0050: The 16-bit identifier of the PAN through which the device is operating. A
9438 value equal to 0xFFFF indicates that the device is not associated.

9439 4.13.4.2.5 mac_key_table

9440 PIB attribute 0x0071: This attribute holds GMK keys required for MAC layer ciphering. The
9441 attribute can hold up to two 16-bytes keys. The Key Identifier value must be different for each
9442 key.

9443 For security reason, the key entries cannot be read, only written.

9444 array mac GMK

```
9446     mac_GMK ::= structure  
9447     {  
9448         key_id:    unsigned,  
9449         key:       octet-string  
9450     }  
9451 }
```

9453 key_id The Key Identifier used to refer to this key, can take the value 0 or 1.

9454 key The AES-128 key used for ciphering the frames exchanged at MAC layer.

4.13.4.2.6 mac_frame_counter

9456 PIB attribute 0x0077: The outgoing frame counter for this device, used when ciphering frames
9457 at MAC layer.

9458 4.13.4.2.7 mac_tone_mask

9459 PIB attribute 0x0110: Defines the tone mask to use during symbol formation.

4.13.4.2.8 mac_TMR_TTL

9461 PIB attribute 0x010D: Maximum time to live of tone map parameters entry in the neighbour table
9462 in minutes.

4.13.4.2.9 mac_max_frame_retries

9464 PIB attribute 0x0059: Maximum number of retransmissions.

4.13.4.2.10 mac_neighbour_table_entry_TTL

9466 PIB attribute 0x010E: Maximum time to live for an entry in the neighbour table in minutes.

4.13.4.2.11 mac_neighbour_table

9468 PIB attribute 0x010A: See ITU-T G.9903:2014 9.3.7.2 for CENELEC and FCC bands.

9469 The neighbour table contains information about all the devices within the POS of the device.
9470 One element of the table represents one PLC direct neighbour of the device.

9471 array neighbour table

neighbour_table::= structure

```
9474
9475    {
9476        short_address: long-unsigned,
9477        payload_modulation_scheme: boolean,
9478        tone_map: bit-string,
9479        modulation: enum,
9480        tx_gain: integer,
9481        tx_res: enum,
9482        tx_coeff: bit-string,
9483        lqi: unsigned,
9484        phase_differential: integer,
9485        TMR_valid_time: unsigned,
9486        neighbour_valid_time: unsigned
9487    }
```

9489 NOTE 1 This table is actualized each time any frame is received from a neighbour device, and each time a Tone
9490 Map Response is received.

9491 short_address The MAC Short Address of the node which this entry refers to.
9492

9493 payload_modulation_scheme Payload Modulation scheme to be used when transmitting
9494 to this neighbour.
9495 FALSE: Differential,
9496 TRUE: Coherent

9497 tone_map
9498
9499
9500
9501 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.

9502 modulation The modulation type to use for communicating with the device.
9503

9504 enum :
9505 (0) Robust Mode,
9506 (1) DBPSK,
9507 (2) DQPSK,
9508 (3) D8PSK,
9509 (4) 16-QAM

9510 NOTE 2 The 16-QAM modulation is optional and only applicable for
9511 FCC band.

tx_gain Defines the Tx Gain to use to transmit frames to that device.

9513 tx_res Defines the Tx Gain resolution corresponding to one gain
9514 step.
9515 0: 6 dB,
9516 1: 3 dB

9517 tx_coeff
9518
9519
9520
9521
9522
9523
9524 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 CFNFI FC-A or a 2-bit value

9525
 9526
 9527
 9528
 9529
 9530
 9531
 9532

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.

9533
 9534
 9535

Example: in CENELEC bands, with gain values equal to [1, -5, 4, -2, 0, 1], tx_coeff encoding is equal to: '0001 1101 0100 1010 0000 0001'

9536
 9537

NOTE 3 One group of tones gathers 6 consecutive tones (or carriers) for CENELEC bands, and 3 consecutive tones for FCC band.

9538 lqi
 9539

Link Quality Indicator of the link to the neighbour (reverse LQI)

9540
 9541
 9542

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.

9543 phase_differential
 9544
 9545
 9546

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.

9547 TMR_valid_time
 9548
 9549

Remaining time in minutes until which the tone map response parameters in the neighbour table are considered valid.

- 9550
 9551
- 9552
 9553
 9554
 9555
- 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.

9556 neighbour_valid_time
 9557
 9558
 9559
 9560
 9561
 9562

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.

9563 **4.13.4.2.12 mac_high_priority_window_size**

9564 PIB attribute 0x0100: The high priority contention window size in number of slots.

9565 **4.13.4.2.13 mac_CSMA_fairness_limit**

9566 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.

9569 **4.13.4.2.14 mac_beacon_randomization_window_length**

9570 PIB attribute 0x0111: Duration time in seconds for the beacon randomization.

9571 **4.13.4.2.15 mac_A**

9572 PIB attribute 0x0112: This parameter controls the adaptive CW linear decrease.

9573 **4.13.4.2.16 mac_K**

9574 PIB attribute 0x0113: Rate adaptation factor for channel access fairness limit. This attribute can
9575 take a value between 1 and macCSMAFairnessLimit.

9576 **4.13.4.2.17 mac_min_CW_attempts**

9577 PIB attribute 0x0114: Number of consecutive attempts while using minimum CW.

9578 **4.13.4.2.18 mac_cenelec_legacy_mode**

9579 PIB attribute 0x0115: This read only attribute indicates the capability of the node.

9580 0: The following configuration is used (legacy mode):

- 9581 – Elementary interleaving;
- 9582 – Interleaver parameters n_i and n_j are not swapped when $I(i,j) = 0$.

9583 1: The following configuration is used (non legacy mode):

- 9584 – Full Block interleaving;
- 9585 – Interleaver parameters n_i and n_j are swapped when $I(i,j) = 0$.

9586 **4.13.4.2.19 mac_FCC_legacy_mode**

9587 PIB attribute 0x0116: This read only attribute indicates the capability of the node.

9588 0: The following configuration is used (legacy mode):

- 9589 – Differential FCH modulation;
- 9590 – Elementary interleaving;
- 9591 – Interleaver parameters n_i and n_j are not swapped when $I(i,j) = 0$;
- 9592 – Single RS block.

9593 1: The following configuration is used (non legacy mode):

- 9594 – Coherent FCH modulation;
- 9595 – Full Block interleaving;
- 9596 – Interleaver parameters n_i and n_j are swapped when $I(i,j) = 0$;
- 9597 – Two RS blocks.

9598 **4.13.4.2.20 mac_max_BE**

9599 PIB attribute 0x0047: Maximum value of backoff exponent. It should always be greater than
9600 macMinBE.

9601 **4.13.4.2.21 mac_max_CSMA_backoffs**

9602 PIB attribute 0x004E: Maximum number of backoff attempts.

9603 4.13.4.2.22 mac_min_BE

9604 PIB attribute 0x004F: Minimum value of backoff exponent.

4.13.4.3 Method description

9606 4.13.4.3.1 mac_get_neighbour_table_entry (data)

9607 This method is used to retrieve the mac neighbour table for one MAC short address. It may be
9608 used to perform topology monitoring by the client.

9609 The method invocation parameter contains a mac_short_address.

9610 data ::= long-unsigned

9611 The response parameter includes the neighbour table for this mac_short_address.

9612 data::= array neighbour_table

9613 where mac_neighbour_table is as defined in the *mac_neighbour_table* attribute of the present
9614 IC.

4.13.5 G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 2)

4.13.5.1 Overview

9617 An instance of the “G3-PLC 6LoWPAN adaptation layer setup” IC holds the necessary
9618 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 = 2			
<i>Attribute (s)</i>		<i>Data type</i>	<i>Min.</i>	<i>Max.</i>	<i>Def.</i>	<i>Short name</i>
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	5	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	1023	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
20.	adp_default_coord_route _enabled (static)	boolean			FALSE	x + 0x98
21.	adp_destination_address _set (dyn)	array				x + 0xA0
<i>Specific methods</i>		<i>m/o</i>				

9622

4.13.5.2 Attribute description

4.13.5.2.1 logical_name

Identifies the “G3-PLC OFDM 6LoWPAN adaptation layer setup” object instance. See 6.2.28.

4.13.5.2.2 adp_max_hops

PIB attribute 0x0F: Defines the maximum number of hops to be used by the routing algorithm.

4.13.5.2.3 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.

4.13.5.2.4 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.

9634 **4.13.5.2.5 adp_prefix_table**

9635 PIB attribute 0x01: Contains the list of prefixes defined on this PAN.

9636 **NOTE 1** it is assumed that the link local IPv6 address exists independently and is not affected by the prefixes defined
 9637 in the prefix table.

9638 **4.13.5.2.6 adp_routing_configuration**

9639 The routing configuration element specifies all parameters linked to the routing mechanism
 9640 described in ITU-T G.9903:2014. The elements are specified in 9.4.1.2 of that Recommendation.

9641 **NOTE 2** The Link cost calculation is provided in ITU-T G.9903:2014 Annex B.

```
9642                 array routing_configuration
9643                 routing_configuration ::= structure
9644                 {
9645                 adp_net_traversal_time:             unsigned,
9646                 adp_routing_table_entry_TTL:       long-unsigned,
9647                 adp_Kr:                             unsigned,
9648                 adp_Km:                             unsigned,
9649                 adp_Kc:                             unsigned,
9650                 adp_Kq:                             unsigned,
9651                 adp_Kh:                             unsigned,
9652                 adp_Krt:                          unsigned,
9653                 adp_RREQ_retries:                 unsigned,
9654                 adp_RREQ_wait:                     unsigned,
9655                 adp_blacklist_table_entry_TTL:    long-unsigned,
9656                 adp_unicast_RREQ_gen_enable:    boolean,
9657                 adp_RLC_Enabled:                     boolean,
9658                 adp_add_rev_link_cost:             unsigned
9659         }
```

9660 Where

9661 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.
9662	Range : 0-255
9663	Default value : 20

9666 adp_routing_table_entry_TTL	PIB attribute 0x12: Maximum time-to-live of a routing table entry (in minutes).
9667	Range : 0-65 535
9668	Default value : 60

9670 adp_Kr	PIB attribute 0x13: A weight factor for the Robust Mode to calculate link cost.
9671	Range : 0-31
9672	Default value : 0

9674 adp_Km	PIB attribute 0x14: A weight factor for modulation to calculate link cost.
9675	Range : 0-31
9676	Default value : 0

9678 adp_Kc	PIB attribute 0x15: A weight factor for number of active tones to calculate link cost.
9679	Range : 0-31
9680	Default value : 0

9682	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.
9687	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.
9692	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
9696	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
9700	adp_RREQ_wait	PIB attribute 0x19: The number of seconds to wait between two consecutive RREQ – RERR generations. Range : 0-255 Default value : 30
9704	NOTE 3 : The title and definition of this item has been modified from previous version	
9705	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
9709	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
9714	adp_RLC_enabled	PIB attribute 0x09: Enable the sending of RLCREQ frame by the device. Default value : FALSE
9717	adp_add_rev_ink_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
9721	4.13.5.2.7 adp_broadcast_log_table_entry_TTL	
9722	PIB attribute 0x02: Maximum time to live of an adpBroadcastLogTable entry (in minutes).	
9723	4.13.5.2.8 adp_routing_table	
9724	PIB attribute 0x0C: Contains the routing table.	

```

9725     array routing_table
9726
9727     routing_table ::= structure
9728
9729     {
9730         destination_address: long-unsigned,
9731         next_hop_address: long-unsigned,
9732         route_cost: long-unsigned,
9733         hop_count: unsigned,
9734         weak_link_count: unsigned,
9735         valid_time: long-unsigned
9736     }
9737

```

9738 NOTE 4 This table is actualized each time a route is built or updated (triggered by data traffic) and each time the
 9739 TTL timer expires.

9740 Where:

9741 destination_address Address of the destination.

9742 next_hop_address Address of the next hop on the route towards the destination.

9743 route_cost Cumulative link cost along the route towards the destination.

9744 hop_count Number of hops of the selected route to the destination.
 9745 Range: 0-14

9746 NOTE 5 Practically the maximum allowed value is limited by adp_max_hops.

9747 weak_link_count Number of weak links to destination.
 9748 Range: 0-14

9749 NOTE 6 Practically the maximum allowed value is limited by adp_max_hops.

9750 valid_time Remaining time in minutes until when this entry in the routing table is
 9751 considered valid.

9752 4.13.5.2.9 adp_context_information_table

9753 PIB attribute 0x07: Contains the context information associated to each CID extension field.
 9754 See ITU-T G.9903:2017. The elements are specified in Table 9-30 of that Recommendation.

```

9755     array context_information_table
9756
9757     context_information_table ::= structure
9758
9759     {
9760         CID: bit-string,
9761         context_length: unsigned,
9762         context: octet-string,
9763         C: boolean,
9764         valid_lifetime: long-unsigned
9765     }

```

9766 Where:

9767 CID Corresponds to the 4-bit context information used for source and
 9768 destination addresses (SCI, DCI).
 9769 The first bit in the bit-string (bit 0) corresponds to the least

9770		significant bit of CID in G.9903:2017, Table 9-30.
9771		Range: 0x00-0x0F
9772	context_length	Indicates the length of the carried context (up to 128-bit contexts may be carried). Range: 0-128
9775	context	Corresponds to the carried context used for compression/decompression purposes.
9777		
9778	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.
9788	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
9793	4.13.5.2.10 adp_blacklist_table	
9794	PIB attribute 0x1E: Contains the list of the blacklisted neighbours.	
9795		array blacklisted_neighbour_set
9796		
9797		blacklisted_neighbour_set::= structure
9798		
9799		{
9800		blacklisted_neighbour_address: long-unsigned,
9801		valid_time: long-unsigned
9802		}
9803	Where:	
9804	blacklisted_neighbour_address	The 16-bit address of the blacklisted neighbour.
9805	valid_time	Remaining time in minutes until which this entry in the blacklisted neighbour table is considered valid.
9807	4.13.5.2.11 adp_broadcast_log_table	
9808	PIB attribute 0x0B: Contains the broadcast log table.	
9809	NOTE 7	This table provides a list of the broadcast packets recently received by this device.
9810		array broadcast_log_table
9811		
9812		broadcast_log_table::= structure
9813		
9814		{

```
9815             source_address:    long-unsigned,  
9816             sequence_number:   unsigned,  
9817             valid_time:       long-unsigned  
9818         }
```

9820 Where:

9821 source_address The 16-bit source address of a broadcast packet. This is the
9822 address of the broadcast initiator.

9823 sequence number The sequence number contained in the BC0 header.

9824 valid_time Remaining time in minutes until when this entry in the broadcast log table is considered valid.
9825

4.13.5.2.12 adp_group_table

9827 PIB attribute 0x0E: Contains the group addresses to which the device belongs.

9828 array group_address

9830 group address::= long-unsigned

group_address – Group address to which this node has been subscribed.

9833 4 13 5 2 13 adp max join wait time

9834 PIB attribute 0x20: Network join timeout in seconds for LBD

9835 4 13 5 2 14 adp path discovery time

9836 PIB attribute 0x21: Timeout for path discovery in seconds

9837 4.13.5.2.15 adp active key index

9838 PIB attribute 0x22: Index of the active GMK to be used for data transmission

9839 4.13.5.2.16 adp metric type

9840 PIB attribute 0x03: Metric Type to be used for routing purposes

9841 4.13.5.2.17 adp coord short address

9842 PIB attribute 0x08: Defines the short address of the coordinator.

4.13.5.2.18 adp_disable_default_routing

9844 PIB attribute 0xF0: If TRUE, the default routing (LOADng) is disabled. If FALSE, the default
9845 routing (LOADng) is enabled.

9846 4.13.5.2.19 adp_device_type

9847 PIB attribute 0x10: Defines the type of the device connected to the modem:

9848 enum:

9849 (0) PAN device,
9850 (1) PAN coordinator,
9851 (2) Not Defined

9852

9853 **4.13.5.2.20 adp_default_coord_route_enabled**

9854 PIB attribute 0x24: If TRUE, the adaptation layer adds a default route to the coordinator after
9855 successful completion of the bootstrapping procedure. If FALSE no default route will be created.

9856 **4.13.5.2.21 adp_destination_address_set**

9857 PIB attribute 0x23: Contains the list of the addresses of the devices for which this LOADng
9858 router is providing connectivity.

9859 array D_address

9860 D_address ::= long-unsigned

9861 Where:

9862 D_Address The 16-bit address of a destination node attached to this LOADng router and
9863 for which this LOADng router provides connectivity.

9864

9865

9866

9867 **4.14 Interface classes for setting up and managing DLMS/COSEM HS-PLC ISO/IEC
9868 12139-1 neighbourhood networks**

9869 **4.14.1 Overview**

9870 COSEM objects for data exchange using DLMS/COSEM HS-PLC ISO/IEC 12139-1
9871 neighbourhood networks, if implemented, shall be located in the Management Logical Device
9872 of COSEM servers.

9873 For setting up and managing DLMS/COSEM HS-PLC ISO/IEC 12139-1 neighbourhood
9874 networks the following ICs are specified:

- 9875 • HS-PLC ISO/IEC 12139-1 MAC setup, see 4.14.2;
- 9876 • HS-PLC ISO/IEC 12139-1 CPAS setup, see 4.14.3;
- 9877 • HS-PLC ISO/IEC 12139-1 IP SSAS setup, see 4.14.4;
- 9878 • HS-PLC ISO/IEC 12139-1 HDLC SSAS setup, see 4.14.5.

9879 **4.14.2 HS-PLC ISO/IEC 12139-1 MAC setup (class_id = 140, version = 0)**

9880 **4.14.2.1 Overview**

9881 Instances of the "HS-PLC ISO/IEC 12139-1 MAC setup" IC hold parameters necessary to set
9882 up and manage the MAC layer of the HS-PLC ISO/IEC 12139-1 profile.

HS-PLC ISO/IEC 12139-1 MAC setup	0...n	class_id = 140 version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name	octet-string				x
2. group_id (static)	long64-unsigned	0	2^{46}		x + 0x08
3. secondary_group_id (static)	long64-unsigned	0	2^{46}		x + 0x10
4. station_id (static)	long64-unsigned	0	2^{48}		x + 0x18
5. parent_station_id (static)	long64-unsigned	0	2^{48}		x + 0x20
6. repeater_status (static)	boolean				x + 0x28
7. encryption_mode (static)	enum	1	2	1	x + 0x30
8. initial_encryption_key (static)	octet-string				x + 0x38
9. rts/cts (static)	boolean	0	1	0	x + 0x40
Specific methods	m/o				

9883

9884 **4.14.2.2 Attribute description**

9885 **4.14.2.2.1 logical_name**

9886 Identifies the "HS-PLC ISO/IEC 12139-1 MAC setup" object instance. See 6.2.31.

9887 **4.14.2.2.2 group_id**

9888 Holds the group identifier of HS-PLC ISO/IEC 12139-1. Refer to ISO/IEC 12139-1:2009, Clause
9889 7 for detailed information.

9890 **4.14.2.2.3 secondary_group_id**

9891 Holds the secondary group identifier of HS-PLC ISO/IEC 12139-1. Refer to ISO/IEC 12139-
9892 1:2009, Clause 7 for detailed information.

9893 **4.14.2.2.4 station_id**

9894 Holds the station identifier of HS-PLC ISO/IEC 12139-1. Refer to ISO/IEC 12139-1:2009,
9895 Clause 7 for detailed information.

9896 **4.14.2.2.5 parent_station_id**

9897 Holds the parent station identifier* of HS-PLC ISO/IEC 12139-1. Refer to ISO/IEC 12139-
9898 1:2009, Clause 7 for detailed information.

9899 * The parent station means the directly neighbouring station from a station itself in a multi-stage HS-PLC ISO/IEC
9900 12139-1 link from the station to NNAP (a source station – Repeater(s) –NNAP).

9901 **4.14.2.2.6 repeater_status**

9902 Refer to ISO/IEC 12139-1:2009, Clause 7 for detailed information.

9903 Holds the current repeater status of the device.

9904 boolean:

9905 FALSE = no repeater,
9906 TRUE = repeater

9908

9909 **4.14.2.2.7 encryption_mode**

9910 Holds the encryption mode the PLC station uses. Refer to ISO/IEC 12139-1:2009, Clause 7 for
9911 detailed information.

9912 enum:

9913 (0) AES128,
9914 (1) Reserved.

9915

9916 **4.14.2.2.8 initial_encryption_key**

9917 Encryption key which is used initially during PLC registration (cell join) period. Refer to ISO/IEC
9918 12139-1:2009, Clause 7 for detailed information.

9919 **4.14.2.2.9 rts/cts**

9920 Holds the status of the RTS/CTS parameter. The RTS/CTS (Request To Send / Clear To Send)
9921 parameter is used to prevent collision caused by hidden HS-PLC ISO/IEC 12139-1 stations).
9922 Refer to ISO/IEC 12139-1:2009, Clause 7 for detailed information.

9923 RTS/CTS enable/disable

9924 boolean:

9925 FALSE = Disable,
9926 TRUE = Enable

9927

9928

9929

9930

9931

9932 **4.14.3 HS-PLC ISO/IEC 12139-1 CPAS setup (class_id = 141, version = 0)**

9933 **4.14.3.1 Overview**

9934 Instances of the "HS-PLC ISO/IEC 12139-1 CPAS setup" IC hold parameters necessary to set
9935 up and manage the CPAS layer of the HS-PLC ISO/IEC 12139-1 profile.

HS-PLC ISO/IEC 12139-1 CPAS setup	0...n	class_id = 141 version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name	octet-string				x
2. cpas_address (static)	long64-unsigned	0	2^{48}		x + 0x08
3. cpas_ether_type (static)	long-unsigned				x + 0x10
4. master_station_cpas_address (static)	long64-unsigned	0	2^{48}		x + 0x18
Specific methods	m/o				

9936

9937 **4.14.3.2 Attribute description**

9938 **4.14.3.2.1 logical_name**

9939 Identifies the "HS-PLC ISO/IEC 12139-1 CPAS setup" object instance. See 6.2.31.

9940 **4.14.3.2.2 cpas_address**

9941 Holds the CPAS address of the HS-PLC ISO/IEC 12139-1 profile.

9942 See IEC 62056-8-6:2017, 5.4.2.

9943 **4.14.3.2.3 cpas_ether_type**

9944 Holds the EtherType value of the CPAS sublayer.

9945 See IEC 62056-8-6:2017, 5.4.2.

9946 **4.14.3.2.4 master_station_cpas_address**

9947 Holds the master station's* CPAS address of the HS-PLC ISO/IEC 12139-1 profile.

9948 * The master station means the destination station of the CPAS frame (i.e. typically NNAP) in a multi-stage HS-
9949 PLC ISO/IEC 12139-1 link (a source station – Repeater(s) – A destination station).

9950

9951 **4.14.4 HS-PLC ISO/IEC 12139-1 IP SSAS setup (class_id = 142, version = 0)**

9952 **4.14.4.1 Overview**

9953 Instances of the "HS-PLC ISO/IEC 12139-1IP SSAS setup" IC hold parameters necessary to
9954 set up and manage the IP SSAS of the HS-PLC ISO/IEC 12139-1 profile.

HS-PLC ISO/IEC 12139-1 IP SSAS setup	0...n	class_id = 142, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name	octet-string				x
2. ip_header_comp_type (static)	enum				x + 0x08
3. ip_alive_time (static)	long-unsigned		0xFFFF	0	x + 0x10
Specific methods	m/o				

9955

9956 **4.14.4.2 Attribute description**

9957 **4.14.4.2.1 logical_name**

9958 Identifies the "HS-PLC ISO/IEC 12139-1 IP SSAS setup" object instance. See 6.2.31.

9959 **4.14.4.2.2 ip_header_comp_type**

9960 Holds the IP_Header_Comp_Type value as specified in IEC 62056-8-6:2017, Table 3.

9961 enum:
 9962 (0) General IPv4 packet (No compression),
 9963 (1) General IPv6 packet (No compression),
 9964 (2) Van Jacobson header compression (RFC 1144),
 9965 (3) IP header compression (RFC 2508),
 9966 (4) ROHC (RFC 3095).
 9967

9968 **4.14.4.2.3 ip_alive_time**

9969 Holds the IP SSAS alive time value in seconds.

9970 0: No expiry

9971

9972 **4.14.5 HS-PLC ISO/IEC 12139-1 HDLC SSAS setup (class_id = 143, version = 0)**

9973 **4.14.5.1 Overview**

9974 Instances of the "HS-PLC ISO/IEC 12139-1 HDLC SSAS setup" IC hold parameters necessary
 9975 to set up and manage the HDLC SSAS of the HS-PLC ISO/IEC 12139-1 profile.

HS-PLC ISO/IEC 12139-1 HDLC SSAS setup	0...n	class_id = 143, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name	octet-string				x
2. master_station_id (static)	long64-unsigned	0	2^{48}		x + 0x08
Specific methods	m/o				

9976

9977 **4.14.5.2 Attribute description**

9978 **4.14.5.2.1 logical_name**

9979 Identifies the "HS-PLC ISO/IEC 12139-1 HDLC SSAS setup" object instance. See 6.2.31.

9980 **4.14.5.2.2 master_station_id**

9981 Holds the master station identifier of the HS-PLC ISO/IEC 12139-1 network.

9982

9983

9984 **4.15 ZigBee® setup classes**9985 **4.15.1 Overview**

9986 This subclause 4.14 specifies COSEM interface classes required for the external configuration
9987 and management of a ZigBee® network to allow interfacing with a multi-part installation that
9988 internally uses ZigBee® communications. ZigBee® is a low-power radio communications
9989 technology and open standard that is operated by the ZigBee® Alliance, see www.zigbee.org.

9990 **ZigBee® is a registered trademark of the ZigBee® Alliance.**

9991 NOTE 1 A multi-part installation is one where the meter provides information and/or services to the householder on
9992 behalf of the utility. For example, the meter interacts with an in home display, and/or an external load control switch,
9993 and/or a smart appliance, to inform the customer of their usage in real time, to control heating devices, and possibly
9994 to disconnect peak loads when supply is constrained. While it is possible that the consumer will control the ZigBee®
9995 network, in normal operations the utility will control the radio system. This is to ensure that security is maintained for
9996 PAN, so that ZigBee® devices such as load switches controlled by the utility operate in a secure manner.

9997 ZigBee® defines a local network of devices linked by radio, with routing and forwarding of
9998 messages and with encryption for privacy at network-level.

9999 NOTE 2 Such a local network is known as a PAN – a Personal Area Network. This name is used in the ZigBee®
10000 community as ZigBee® is underpinned by the IEEE 802.15.4:2006 standard, which uses the term PAN. This is
10001 broadly equivalent to a HAN (Home Area Network – name used in the context of smart metering in the UK) or PAN.

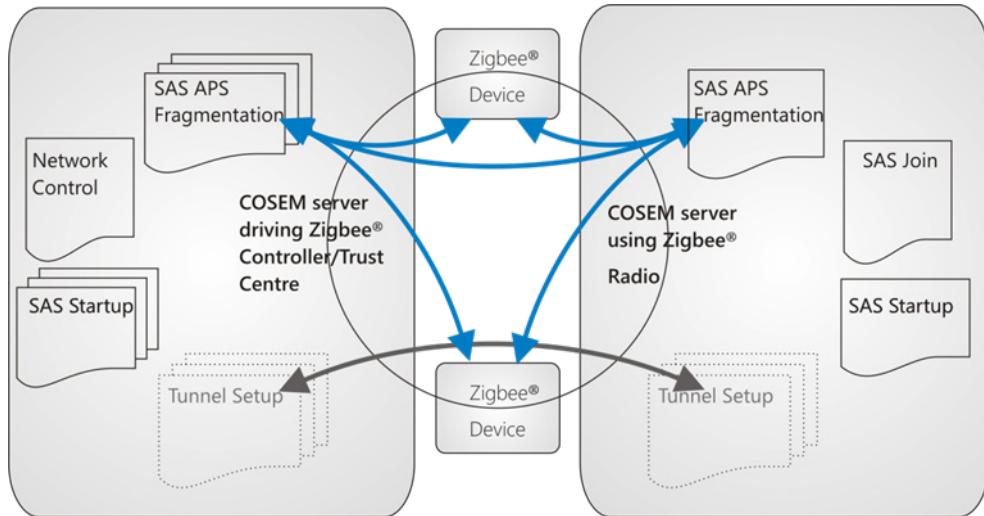
10002 Each PAN has one device designated as ZigBee® coordinator, which has responsibility for
10003 creating and managing the network, and which normally acts also as a ZigBee® Trust Center
10004 for the management of ZigBee® network, PCLK's and APS Link keys.

10005 There is a process of PAN creation (and corresponding destruction) which is performed by the
10006 coordinator; this declares the existence of the network without any devices apart from the
10007 coordinator forming part of it. Other ZigBee® devices can join a network created with
10008 cooperation of the coordinator, and equally can choose to leave, or can be invited to leave by
10009 the coordinator (this is not currently enforceable). Normally devices are members of the network
10010 indefinitely; they do not repeatedly join and leave. To create a PAN the coordinator has to
10011 receive an external trigger and needs to have setup information including:

- 10012 • extended PAN ID;
10013 • link keys or install code (for initial communication with new devices);
10014 • radio channel information.

10015 During the process of creating the PAN, the coordinator scans for nearby radio devices,
10016 “exchanges” keys, chooses the short addresses, and confirms use of radio channels. Details of
10017 the information available from the ZigBee® servers on each device are also exchanged.

10018 NOTE 3 Full details of the joining process are documented in ZigBee® 053474, the ZigBee® specification. More
10019 information on ZigBee® technology can be sought at <http://www.zigbee.org/>.



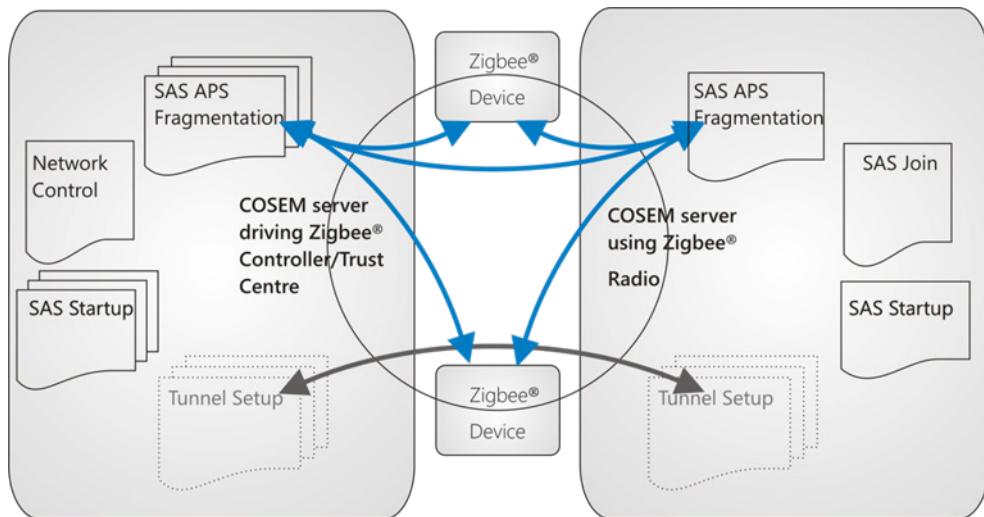
10020

Figure 29 – Example of a ZigBee® network

10022 shows an example architecture with a Comms hub on the left, that comprises a DLMS/COSEM
 10023 server as well as the ZigBee® coordinator. The DLMS/COSEM server has interface objects
 10024 needed to set up and control the ZigBee® network and may have other COSEM objects to
 10025 support a metering application. Further, there are two native ZigBee® devices and another
 10026 DLMS/COSEM server – an electricity meter or another meter type – on the right which is also
 10027 a “normal” ZigBee® network device.

10028 Any ZigBee® device can be “joined” to the network by remote control.

10029 It is assumed that the Comms hub will also have a further network connection to the WAN, and
 10030 it is assumed that this is managed by existing DLMS structures – e.g. PSTN, GSM/3G, PLC etc.
 10031 – but this is out of scope of this Technical Specification.



10032

Figure 29 – Example of a ZigBee® network

10034 The role of COSEM interface objects in the process of creating / destructing the PAN is to set
 10035 up ZigBee® parameters and allow a DLMS/COSEM client to trigger actions, typically when
 10036 commissioning the installation, in a system where WAN communications between a central
 10037 system and a smart meter installation is by means of DLMS.

10038 Operation of the ZigBee® network is not the responsibility of DLMS/COSEM. The
 10039 DLMS/COSEM server is merely the vehicle for controlling the ZigBee® network by an external
 10040 manager (DLMS/COSEM Client).

10041 The use of ZigBee® setup classes in the ZigBee® coordinator and other DLMS/COSEM
 10042 ZigBee® devices is shown in Table 42.

10043 **Table 42 – Use of ZigBee® setup COSEM interface classes**

ZigBee® coordinator	Other DLMS/COSEM ZigBee® devices	Reference
ZigBee® SAS startup	ZigBee® SAS startup	4.15.2
–	ZigBee® SAS join	4.15.3
ZigBee® SAS APS fragmentation	ZigBee® SAS APS fragmentation	4.15.4
ZigBee® network control	–	4.15.5
Optionally: ZigBee® tunnel setup	Optionally: ZigBee® tunnel setup	4.15.6

10044

10045 This set of COSEM ICs supports the ZigBee® 2007 and ZigBee® PRO protocol stacks. The
 10046 ZigBee® IP protocol stack is not supported at this time.

10047 **4.15.2 ZigBee® SAS startup (class_id = 101, version = 0)**

10048 **4.15.2.1 Overview**

10049 NOTE 1 In the specification of the ZigBee® COSEM ICs, the length of the octet-strings is indicated for information.

10050 Instances of this IC are used to configure a ZigBee® PRO device with information necessary to
 10051 create or join the network. The functionality that is driven by this object and the effect on the
 10052 network depends on whether the object is located in a ZigBee® coordinator or in another
 10053 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				

10054

10055 **4.15.2.2 Attribute description**

10056 **4.15.2.2.1 logical_name**

10057 Identifies the “ZigBee® SAS startup” object instance. See 6.2.29.

10058 **4.15.2.2.2 short_address**

10059 Defines the 16-bit address by which this device is known locally on the ZigBee® network. Value
10060 0x0000 is given to a coordinator / Trust Center device, and only to these devices.

10061 NOTE 2 This short address will be issued to the device by the coordinator when the device joins the network.

10062 **4.15.2.2.3 extended_pan_ID**

10063 Defines the unique address by which the network is known externally.

10064 The length of the octet-string is 8 octets.

10065 **4.15.2.2.4 pan_ID**

10066 Defines the 16-bit address by which network is known locally.

10067 **4.15.2.2.5 channel_mask**

10068 Defines (as a bit mask) the set of radio channels which this PAN is permitted to use. Actual
10069 usage of channels within this set is determined by the coordinator on the basis of local
10070 conditions.

10071 The mask is as defined in the ZigBee® specification.

10072 **4.15.2.2.6 protocol_version**

10073 Defines the version of the ZigBee® protocol to be used on the network.

10074 **4.15.2.2.7 stack_profile**

10075 Identifies the capabilities of the ZigBee ® stack.

10076 enum:

- 10077 (1) ZigBee®,
10078 (2) ZigBee® PRO

10079 **4.15.2.2.8 start_up_control**

10080 Indicates the commissioning state of the ZigBee® device:

10081 2: un-commissioned. Indicates that the device will seek to join the network if/when it is
10082 established;

10083 0: commissioned. Indicates that the device should consider itself a part of the network
10084 indicated by the extended_pan_id attribute. In this case it will not perform any explicit join
10085 or rejoin operation. See NOTE 3 below.

10086 **4.15.2.2.9 trust_center_address**

10087 Defines the unique extended address of the Trust Center used for this network. This is often,
10088 but not necessarily, the address of the ZigBee® coordinator.

10089 The length of the octet-string is 8 octets.

10090 **4.15.2.2.10 link_key**

10091 Defines the key value used to secure point-to-point communications between particular devices
10092 within the Smart Energy Profile. The way in which the raw key value is protected is not defined
10093 in this specification.

10094 This is the APS link key or the PCLK. It's down to the implementation if this attribute is Hashed
10095 or in the clear or even if this is used.

10096 If this is the Trust Center then this is not usually implemented.

10097 This is usually read only, but may need to be writable for testing.

10098 The length of the octet-string is 16 octets.

10099 **4.15.2.2.11 network_key**

10100 Defines the key value used to secure general communications between devices. The way in
10101 which the raw key value is protected is not defined in this specification. The network key value
10102 may be set internally by the coordinator.

10103 It is down to the implementation if this attribute is Hashed or in the clear or even if this is used.

10104 If this is the TC then this is not usually implemented.

10105 This is read only.

10106 The length of the octet-string is 16 octets.

10107 **4.15.2.2.12 use_insecure_join**

10108 Indicates whether the coordinator is permitted to allow insecure joining as defined by ZigBee®
10109 specification.

10110 NOTE 3 The “ZigBee® SAS startup” object reflects the state of the ZigBee® HAN to the WAN (or a diagnostic tool
10111 connected to a ZigBee® connected DLMS/COSEM Server). For example, if the object is in a Comms hub, then the
10112 attribute can be read out to show that the ZigBee® HAN has not been commissioned. The main function of this object
10113 is to provide information about a ZigBee® network to a client on the WAN (or via the optical port). The reason that
10114 there are multiple instances of the “ZigBee® SAS startup” object in Figure 29 is to express the idea that there may
10115 be multiple ZigBee® networks operated from a single Comms Hub.

10116

10117 **4.15.3 ZigBee® SAS join (class_id = 102, version = 0)**

10118 **4.15.3.1 Overview**

10119 Instances of this IC configure the behaviour of a ZigBee® PRO device on joining or loss of
10120 connection to the network. “ZigBee® SAS join” objects are present in all devices where a
10121 DLMS/COSEM server controls the ZigBee® Radio behaviour, but it is not used when a device
10122 is acting as coordinator (as the coordinator device creates rather than joins a network).
10123 “ZigBee® SAS join” objects can be factory configured, or configured using another
10124 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	<i>m/o</i>				

10125

10126 **4.15.3.2 Attribute description**10127 **4.15.3.2.1 logical_name**

10128 Identifies the “ZigBee® SAS join” object instance. See 6.2.29.

10129 **4.15.3.2.2 scan_attempts**10130 Defines the number of consecutive scans that a ZigBee® device will perform on each attempt
10131 to rejoin a network, in the event of losing contact.10132 **4.15.3.2.3 time_between_scans**10133 Defines the period in seconds between consecutive scans in a single attempt to rejoin a
10134 network.10135 **4.15.3.2.4 rejoin_interval**10136 Defines the period in seconds that the device should wait after apparently becoming
10137 disconnected from a network, before attempting to rejoin.10138 **4.15.3.2.5 rejoin_retry_interval**10139 Defines the period in seconds for which the device should wait after a failed attempt to rejoin a
10140 network before trying again.

10141 Remarks on usage

10142 At boot time or when instructed to join a network, the device should complete up to three (3)
10143 (Note: as per the default) scan attempts to find a ZigBee® coordinator or router with which to
10144 associate.10145 If a device has not been commissioned, this means that when the user presses a button or uses
10146 another methodology to instruct the device to join a network, it will scan all of the channels up
10147 to three times (as per the default) to find a network that allows joining. If it has already been
10148 commissioned, it should scan up to three times (Note: as per the default) to find its original PAN
10149 to join. (ZigBee® Pro devices should try to find a network using their original extended PAN ID
10150 and ZigBee® devices can only try to find a network using their original PAN ID).10151 Remark on the *rejoin_retry_interval*10152 Imposes an upper bound on the *rejoin_interval* parameter – this is restarted if device is touched
10153 by human user, i.e. by a button press. This parameter is intended to restrict how often a device
10154 will scan to find its network in case the network is no longer present and therefore a scan

10155 attempt by the device would always fail (i.e., if a device finds it has lost network connectivity, it
 10156 will try to re-join the network, scanning all channels if necessary). If the scan fails to successfully
 10157 re-join, the device will wait for 15 min before attempting to re-join again. To be network friendly,
 10158 it would be recommended to adaptively extend this time period if successive re-joins fail. It
 10159 would also be recommended the device should try a re-join when triggered (via a control, button,
 10160 etc.) and fall back to the *rejoin_retry_interval* if attempts to re-join fail again.

10161

10162 **4.15.4 ZigBee® SAS APS fragmentation (class_id = 103, version = 0)**10163 **4.15.4.1 Overview**

10164 Instances of this IC configure the fragmentation feature of ZigBee® PRO transport layer. This
 10165 fragmentation is not of concern to COSEM; the object merely allows configuration of the
 10166 fragmentation function by an external manager (DLMS/COSEM client).

10167 Instances of this IC are present in all devices where a DLMS/COSEM server controls the
 10168 ZigBee® Radio behaviour.

ZigBee® SAS APS fragmentation	0...n	class_id = 103, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. aps_interframe_delay (static)	long-unsigned			50	x + 0x08
3. aps_max_window_size (static)	long-unsigned			1	x + 0x10
Specific methods	m/o				

10169

10170 **4.15.4.2 Attribute description**10171 **4.15.4.2.1 logical_name**

10172 Identifies the “ZigBee® SAS APS fragmentation” object instance. See 6.2.29.

10173 **4.15.4.2.2 aps_interframe_delay**

10174 Defines the delay in milliseconds between sending two blocks of a fragmented transmission.

10175 **4.15.4.2.3 aps_max_window_size**

10176 Defines the maximum number of unacknowledged frames that can be transmitted consecutively.

10177 NOTE These initial values will allow for installation, and setting these values will depend on the specification of the
 10178 ZigBee® Interface.

10179

10180 **4.15.5 ZigBee® network control (class_id = 104, version = 0)**10181 **4.15.5.1 Overview**

10182 There will be a single instance of the “ZigBee® network control” IC in any device that can act
 10183 as a ZigBee® coordinator controlled by the DLMS/COSEM client. This class allows interaction
 10184 between a DLMS/COSEM client (head-end system) and a ZigBee® coordinator at times such
 10185 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				
2. unregister_device (data)	m				
3. unregister_all_devices (data)	o				
4. backup_PAN (data)	o				
5. restore_PAN (data)	o				
6. identify_device (data)	o				
7. remove_mirror (data)	o				
8. update_network_key (data)	o				
9. update_link_key (data)	o				
10. create_PAN (data)	m				
11. remove_PAN (data)	m				

10186

10187 **4.15.5.2 Attribute description**10188 **4.15.5.2.1 logical_name**

10189 Identifies the “ZigBee® network control” object instance. See 6.2.29.

10190 **4.15.5.2.2 enable_disable_joining**10191 A flag controlling whether devices are allowed to join the ZigBee® network. The flag is normally
10192 FALSE (joining disabled). At certain times – when a new device is expected to join – the flag is
10193 set externally to TRUE and then remains set for the duration of the *join_timeout*, or until
10194 externally reset if that happens sooner.10195 **4.15.5.2.3 join_timeout**10196 Defines the time period in seconds during which the coordinator device will permit joining of
10197 new devices, following setting of the *enable_disable_joining* flag.10198 **4.15.5.2.4 active_devices**

10199 This attribute shows all of the currently authorised devices within the ZigBee® PAN.

```

10200           array      active_device
10201
10202           active_device ::= structure
10203           {
10204             mac_address:          octet-string,
10205             status:               bit-string,
10206             maxRSSI:              integer,
10207             averageRSSI:          integer,
10208             minRSSI:              integer,
10209             maxLQI:               unsigned,
10210             averageLQI:            unsigned,

```

```

10211      minLQI:          unsigned,
10212      last_communication_date-time: octet-string,
10213      number_of_hops:        unsigned,
10214      transmission_failures: unsigned,
10215      transmission_successes: unsigned,
10216      application_version:  unsigned,
10217      stack_version:       unsigned
10218      }

```

10219 The length of the mac_address is 8 octets.

10220 The length of the status is 8 bits.

10221 The Max/Average/Min values are taken over the last 24 h period for each device. Period is from
10222 00:00:01 to 00:00:00. They are always historical i.e. they refer to the last day.

```

10223      status ::= bit-string[8]
10224
10225      Bit 0 = Authorised on PAN
10226      Bit 1 = Actively reporting on PAN
10227      Bit 2 = Unauthorised on PAN but has reported
10228      Bit 3 = Authorised after swap-out
10229      Bit 4 = SEP Transmitting
10230      Bit 4 = Reserved
10231      Bit 6 = Reserved
10232      Bit 7 = Reserved

```

10233 Other elements are detailed in the ZigBee® specification.

10234 **4.15.5.3 Method description**

10235 **4.15.5.3.1 register_device (data)**

10236 This method is called externally to instruct the coordinator that a device should be added to the
10237 list of authorised devices. This method does not actually join the devices to the network, they
10238 are just authorised to join at some time in the future.

```

10239      data ::= structure
10240
10241      {
10242      ieee_address:          octet-string,
10243      key_type:             enum,
10244      key:                  octet-string,
10245      device_type:          enum,
10246      }

```

10247

10248 Where:

- 10249 – ieee_address holds the IEEE address of the device. The length of the octet-string is 8 octets;
- 10250 – key_type determines the type of content of key.

```

10251      enum:
10252          (0) Pre-configured Link Key,
10253          (1) Install code,
10254          (2)...(255) Reserved,
10255

```

10256 NOTE 1 Install Codes will be subject to agreement across the members who are implementing a ZigBee® network.
10257 Therefore the length of install codes, the use of a CRC, and how they are padded when transported in this method
10258 are subject to agreement as part of a project specific companion specification.

10259 – key is a pre-configured link key or install code. This will depend on the device being
10260 authorised to join the network. Its maximum length is 16 octets,

10261 – device_type is linked to the enumeration shown below so that the coordinator has a
10262 method of understanding the possible services that the joining device may require.

10263 NOTE 2 For example, it is likely that an implementation would require Mirror function for Gas, Water and Heat
10264 Meters connected by ZigBee®. However, determination of which ZigBee® support is needed for which device will be
10265 dependent on the organisation designing the smart metering solution, perhaps a government or a large utility.

10266 device-type::= enum
10267
10268 (0) Electricity Meter
10269 (1) Gas Meter
10270 (2) Water Meter
10271 (3) Thermal Meter
10272 (4) Pressure Meter
10273 (5) Heat Meter
10274 (6) Cooling Meter
10275 (7) Electric Vehicle charging Meter
10276 (8) PV Generation Meter
10277 (9) Wind Turbine Generation Meter
10278 (10) Water Turbine Generation Meter
10279 (11) Micro Generation Meter
10280 (12) Solar Hot Water Generation Meter
10281 (13) ZigBee® Controlled Load Switch
10282 (14) ZigBee® Based Boost Button
10283 (128) IHD
10284 (129) Range extender
10285 (130) CAD (Consumer Access Device)
10286 (131) Thermostat
10287 (132) Prepayment Terminal
10288 (133) ZigBee® Controlled Load Switch
10289 (134) ZigBee® Based Boost Button

10290

10291 **4.15.5.3.2 unregister_device (data)**

10292 This method is called externally to instruct the coordinator that a device should leave the
10293 network.

10294 data::= octet-string

10295 It holds the ieee_address. The length of the octet-string is 8 octets.

10296 **4.15.5.3.3 unregister_all_devices (data)**

10297 This method is called externally to instruct the coordinator that all devices should leave the
10298 network.

10299 data::= integer (0)

10300 NOTE 3 The likely use of this function is to ensure a network is empty of devices prior to destroying it.

10301 **4.15.5.3.3.1 backup_PAN (data)**

10302 This method instructs the coordinator to create a back-up of information that would be
10303 necessary to re-create the PAN.

10304 NOTE 4 The storage location of the back-up is not currently defined and is an internal function of the DLMS/COSEM
 10305 server.

10306 data ::= integer (0)

10307

10308 Method invocation return parameters are detailed below:

```
10309                   data ::= structure
10310
10311                   {
10312                    date_time:                   octet-string,
10313                    extended_PAN_ID:           octet-string,
10314                    devices_to_backup:         array device_to_backup
10315                   }
```

10316

10317 Where:

- 10318 – date-time refers to the time at which the backup process is due to begin. It is formatted as
 10319 specified in 4.6.1;
- 10320 – extended_PAN_ID identifies the PAN. The length of the octet-string is 8.

10321

```
10322                   device_to_backup ::= structure
10323
10324                   {
10325                    MAC_address:                octet-string,
10326                    hashed_TC_link_key:        octet-string
10327                   }
```

10328

10329 Where:

- 10330 – MAC_address hold the MAC address. The length of the octet-string is 8;
- 10331 – the length of octet-string holding the hashed_TC_link_key is 16 octets.

10332 NOTE 5 The method of hashing the link key is not part of this specification; it is defined in the ZigBee® Smart
 10333 Energy Specification. MMO is currently used.

10334 **4.15.5.3.4 restore_PAN (data)**

10335 This method instructs the coordinator to restore a PAN using backup information. The storage
 10336 location of the back-up is not currently defined and is an internal function of the DLMS/COSEM
 10337 Server.

```
10338                   data ::= structure
10339
10340                   {
10341                    extended_PAN_ID:           octet-string,
10342                    devices_to_restore:        array device_to_restore
10343                   }
```

10344 Where:

- 10345 – extended_PAN_ID identifies the PAN. The length of the octet-string is 8;

10346 device_to_restore ::= structure
 10347 {
 10348 MAC_address: octet-string,
 10349 hashed_TC_link_key: octet-string
 10350 }

10351 Where:

- 10352 – MAC_address holds the MAC address. The length of the octet-string is 8;
 10353 – the length of octet-string holding the hashed_TC_link_key is 16 octets.

10354

10355 NOTE 6 The method of hashing the link key is not part of this specification; it is defined in the ZigBee® Smart
 10356 Energy Specification. MMO is currently used.

10357 **4.15.5.3.5 identify_device (data)**

10358 This method is called externally to instruct a device to identify itself to an engineer present on
 10359 site, for example by sounding a buzzer.

10360 data ::= ieee_address

10361

10362 ieee_address: octet-string

10363

10364 ieee_address holds the IEEE address of the device. The length of the octet-string is 8 octets.

10365 **4.15.5.3.6 remove_mirror (data)**

10366 This method causes the removal of a ZigBee® mirror that reflects the real device identified by
 10367 the mac_address parameter.

10368 data ::= structure
 10369 {
 10370 mac_address: octet-string,
 10371 mirror_control: bit-string
 10372 }

10373 Where:

- 10374 – MAC_address holds the MAC address. The length of the octet-string is 8;
 10375 – the mirror_control parameter is provided to support the execution of implementation-specific
 10376 actions, which should be defined in a project specific companion specification.

10377 EXAMPLE The following example is one of the possible options for the bit-string functionality.

10378 0 = Force Gas Meter Removal.
 10379 NOTE Where a device removal is forced, the keys values for this
 10380 device are removed; an APS ack from the device is not required.
 10381 1 = Clear all Mirror Data
 10382 2 = Clear Consumption registers / indexes
 10383 3 = Clear Demand & Max Demand registers
 10384 4 = Clear ZigBee® attributes
 10385 5 = Clear MPAN

10386 6 = Clear Billing Information
10387 7 = Clear Logs
10388 8 = Clear OTA Firmware waiting
10389 9...14 = Reserved
10390 15 = Action all

10391 Should the bit be set then the action is carried out.

10392 The full meaning and actions that the DLMS/COSEM server should undertake are
10393 implementation-specific actions, which should be defined in a project specific companion
10394 specification.

10395 **4.15.5.3.7 update_network_key (data)**

10396 This method requests that the ZigBee® coordinator updates the network key and propagate it
10397 to all devices on the PAN. For details of ZigBee® key management, see the ZigBee®
10398 specification.

10399 data::= integer (0)

10400 **4.15.5.3.8 update_link_key (data)**

10401 This method requests that the ZigBee® coordinator updates the link key and propagate it to the
10402 identified device on the PAN. For details of ZigBee® key management, see the ZigBee®
10403 specification.

10404 data::= ieee_address

10405 ieee_address: octet-string

10406 ieee_address holds the IEEE address of the device. The length of the octet-string is 8 octets.

10407 **4.15.5.3.9 create_PAN (data)**

10408 This method is called externally to instruct a coordinator to create a network using the
10409 configuration held in the ZigBee® SAS startup object.

10410 data::= integer (0)

10411 **4.15.5.3.10 remove_PAN (data)**

10412 This method is called externally to instruct a coordinator to destroy a network by turning off the
10413 ZigBee® radio and removing all of the settings associated with the current PAN.

10414 data::= integer (0)

10415

10416 **4.15.6 ZigBee® tunnel setup (class_id = 105, version = 0)**

10417 **4.15.6.1 Overview**

10418 A ZigBee® tunnel is established between two ZigBee® PRO devices to allow DLMS APDUs to
10419 be transferred between them. The tunnel in effect extends WAN connectivity to ZigBee®
10420 devices not connected to the WAN through a ZigBee® device connected to the same ZigBee®
10421 network and connected to the WAN.

10422 The ZigBee® tunnel setup objects would be present on the coordinator and on all other
 10423 DLMS/COSEM devices that are not connected to the WAN.

10424 Creation of the tunnel is managed on demand and invisibly from the point of view of the
 10425 DLMS/COSEM client. The target device is implicitly identified by the COSEM addressing
 10426 information.

10427 NOTE 1 See also the Gateway specification in IEC 62056-5-3:**2021**, Annex C.

10428

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	m/o				

10429

10430 4.15.6.2 Attribute description

10431 4.15.6.2.1 logical_name

10432 Identifies the “ZigBee® tunnel setup” object instance. See 6.2.29.

10433 4.15.6.2.2 maximum_incoming_transfer_size

10434 Defines the maximum size, in octets, of the data packet that can be transferred to the tunnel
 10435 client in the payload of a single **TransferData** (TransferData is a ZigBee® parameter)
 10436 command. Behaviour on receipt of a larger packet is not defined in this specification.

10437 4.15.6.2.3 maximum_outgoing_transfer_size

10438 Defines the maximum size, in octets, of the data packet that can be sent from the tunnel client
 10439 in the payload of a single **TransferData** command. Behaviour on transmission of a larger packet
 10440 is not defined in this specification. See NOTE 2 below.

10441 4.15.6.2.4 protocol_address

10442 The *protocol_address* is implementation-specific, which should be defined in a project specific
 10443 companion specification.

10444 The length of the octet-string is 6 octets.

10445 4.15.6.2.5 close_tunnel_timeout

10446 Defines the time, in seconds that the ZigBee® server waits before closing an inactive tunnel on
 10447 its own (without waiting for the *CloseTunnel* Command from the client within the ZigBee®
 10448 specification) and freeing its resources

10449 The timer is re-started with each reception of a command.

10450 NOTE 2 The word *outcoming* is unusual, but is adopted for commonality with ZigBee® specifications.

10451 **4.16 Interface classes for setting up and managing the DLMS/COSEM profile for**
 10452 **LPWAN networks**

10453 **4.16.1 General**

10454 This clause specifies ICs for setting up devices using the DLMS/COSEM LPWAN
 10455 communication profile and to diagnose the LPWAN network:

- 10456 • the ICs specified in 4.16.2 are used to define setup and diagnostic objects for the SCHC-
 10457 LPWAN part;
- 10458 • the ICs specified in 4.17 are used to define setup and diagnostic objects for the lower
 10459 layers.

10460 For each specific LPWAN technology, a setup and a diagnostic IC should be specified. ICs for
 10461 the LoRaWAN 1.0.3 technology are specified in 4.16.3.2.

10462 **4.16.2 Generic interface classes**

10463 **4.16.2.1 SCHC-LPWAN setup (class_id = 126, version = 0)**

10464 **4.16.2.1.1 Overview**

10465 Instances of this IC are available for configuring the parameters needed to set up a LPWAN
 10466 device.

SCHC-LPWAN setup	0..n	class_id = 126 , version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. ipwan_reference (static)	octet-string				x + 0x08
3. schc_cd_rules (static)	array				x + 0x10
4. schc_fr_param (static)	structure				x + 0x18
Specific methods (if required)	m/o				

10467

10468 **4.16.2.1.2 Attribute description**

10469 **4.16.2.1.2.1 logical_name**

10470 Identifies the “LPWAN setup” object instance. See 6.2.23.

10471 **4.16.2.1.2.2 ipwan_reference**

10472 References an LPWAN technology specific setup object.

10473 **4.16.2.1.2.3 schc_cd_rules**

10474 Contains the necessary parameters to support LPWAN Static Context Header Compression
 10475 (SCHC) compression and decompression rules

```

10476           array schc_cd_rules_element
10477
10478           schc_cd_rules_element ::= structure
10479
10480           {
10481             rule_id:          unsigned,
10482             field_descriptors: array field_descriptor
10483           }
```

```
10484
10485     field_descriptor ::= structure
10486
10487     {
10488         field_id:           field_id_type,
10489         field_length:      unsigned,
10490         field_position:    unsigned,
10491         direction_indicator: direction_indicator_type,
10492
10493         target_value:      CHOICE
10494
10495     {
10496         -- simple data types
10497         null-data          [0],
10498         boolean             [3],
10499         bit-string          [4],
10500         double-long          [5],
10501         double-long-unsigned [6],
10502         octet-string        [9],
10503         visible-string      [10],
10504         utf8-string         [12],
10505         bcd                 [13],
10506         integer              [15],
10507         long                [16],
10508         unsigned             [17],
10509         long-unsigned        [18],
10510         long64               [20],
10511         long64-unsigned      [21],
10512         enum                [22],
10513         float32             [23],
10514         float64             [24],
10515         date-time            [25],
10516         date                [26],
10517         time                [27],
10518
10519         -- complex data types
10520         array               [1],
10521         structure            [2],
10522         compact-array         [19]
10523     }
10524
10525     matching_operator:   matching_operator_type,
10526     compression_decompression_action:
10527     compression_decompression_action_type
10528
10529
10530     field_id_type ::= enum
10531
10532     {
10533         (0)          IPv6-version,
10534         (1)          IPv6-DiffServ,
10535         (2)          IPv6-FlowLabel,
10536         (3)          IPv6-Length,
10537         (4)          IPv6-NextHeader,
10538         (5)          IPv6-HopLimit,
10539         (6)          IPv6-Dev-Prefix,
10540         (7)          IPv6-DevIID,
10541         (8)          IPv6-AppPrefix,
10542         (9)          IPv6-AppIID,
10543         (10)         UDP-DevPort,
10544         (11)         UDP-AppPort,
10545         (12)         UDP-Length,
```

```

10546          (13)      UDP-C checksum,
10547          (14)      DLMS-Wrapper-version,
10548          (15)      DLMS-Wrapper-SSAP,
10549          (16)      DLMS-Wrapper-CSAP,
10550          (17)      DLMS-Wrapper-Length
10551      }
10552
10553      direction_indicator_type ::= enum
10554      {
10555          (0)      Uplink,
10556          (1)      Downlink,
10557          (2)      bidirectional
10558      }
10559
10560      matching_operator_type ::= enum
10561      {
10562          (0)      equal,
10563          (1)      ignore,
10564          (2)      MSBx,
10565          (3)      match-mapping
10566      }
10567
10568      compression_decompression_action_type ::= enum
10569      {
10570          (0)      not-sent,
10571          (1)      value-sent,
10572          (2)      mapping-sent,
10573          (3)      LSB,
10574          (4)      compute-length,
10575          (5)      compute-checksum,
10576          (6)      DevIID,
10577          (7)      ApplID,
10578      }
10579

```

4.16.2.1.2.4 schc_fr_param

Contains the necessary parameters to support LPWAN Static Context Header Compression (SCHC) fragmentation and reassembly parameters.

```

10583      schc_fr_param ::= structure
10584
10585      {
10586          rule_id_scheme:      enum
10587          {
10588              (0)      fixed-size,
10589              (1)      variable-size
10590          },
10591          max_packet_size:    long-unsigned,
10592          padding_l2_word_size: unsigned,
10593          padding_bits_value:  unsigned,
10594          delay_after_transmission: long-unsigned,
10595          interleaved_packet_tran: boolean,
10596          window_size:        unsigned,
10597          rule_params:         array rule_param_element
10598
10599      }
10600
10601      rule_param_element ::= structure
10602      {
10603          rule_id:            unsigned,
10604          rule_id_len:         unsigned,

```

```

10605 dtag_len:           unsigned,
10606 w_len:             unsigned,
10607 fcn_len:            unsigned,
10608 rcs_algorithm:      enum
10609         {
10610             (0)  None,
10611             (1)  CRC32 using 0xEDB88320
10612         },
10613 reliability_mode:  enum
10614         {
10615             (0)  No-ACK,
10616             (1)  ACK-on-Error,
10617             (2)  ACK-Always
10618         },
10619 retransmission_timer: long-unsigned,
10620 inactivity_timer:    long-unsigned,
10621 max_ack_request:    unsigned
10622 }
10623

```

10624 Where:

10625 rule_id_scheme	The RuleID numbering scheme, fixed-size or variable-size Rule IDs, the way the Rule ID is transmitted;
10627 max_packet_size	The maximum packet size that should ever be reconstructed by SCHC Decompression [octets];
10629 padding_l2_word_size	The size of the L2 Word [bits];
10630 padding_bits_value	The value of the padding bits (0 or 1);
10631 delay_after_transmission	The delay to be added after transmission [milliseconds];
10632 interleaved_packet_tran	The support for interleaved packet transmission;
10633 window_size	The WINDOW_SIZE, for modes that use windows;
10634 rule_id_len	The length in bits for the RuleID header field of SCHC F/R message [bits];
10636 dtag_len	The length in bits for the Datagram Tag (DTag) header field of SCHC F/R message [bits];
10638 w_len	The length in bits for the W header field of SCHC F/R message [bits];
10639 fcn_len	The length in bits for the Fragment Compressed Number (FCN) header field of SCHC F/R message [bits];
10641 rcs_algorithm	The algorithm for the calculation of Reassembly Check Sequence (RCS) field of SCHC F/R message;
10643 reliability_mode	The reliability mode used;
10644 retransmission_timer	The Retransmission Timer duration [milliseconds];
10645 inactivity_timer	The Inactivity Timer duration [milliseconds];

10646 max_ack_request The MAX_ACK_REQUEST value.

10647 C/D rules defined for the DLMS over SCHC profile are listed in Table 51.

10648

Table 43 – C/D Rule 1

field_id	field_length	field_position	direction_indicator	target_value	matching_operator	compression_decompression_action
IPv6-version	4	1	bidirectional	6	equal	not-sent
IPv6-DiffServ	8	1	bidirectional	0	equal	not-sent
IPv6-FlowLabel	20	1	bidirectional	0	equal	not-sent
IPv6-Length	16	1	bidirectional		ignore	comp-length
IPv6-NextHeader	8	1	bidirectional	17	equal	not-sent
IPv6-HopLimit	8	1	bidirectional	255	equal	not-sent
IPv6-Dev-Prefix	64	1	bidirectional	FE80::/64	equal	not-sent
IPv6-DevlID	64	1	bidirectional		ignore	DevlID
IPv6-AppPrefix	64	1	bidirectional	FE80::/64	equal	not-sent
IPv6-AppIID	64	1	bidirectional	1	ignore	not-sent
UDP-DevPort	16	1	bidirectional	4059	equal	not-sent
UDP-AppPort	16	1	bidirectional	4059	equal	not-sent
UDP-Length	16	1	bidirectional		ignore	comp-length
UDP-C checksum	16	1	bidirectional		ignore	comp-chk
DLMS-Wrapper-version	16	1	bidirectional	1	equal	not-sent
DLMS-Wrapper-SSAP	16	1	bidirectional	1	equal	not-sent
DLMS-Wrapper-CSAP	16	1	bidirectional	1	equal	not-sent
DLMS-Wrapper-Length	16	1	bidirectional		ignore	comp-length

NOTE If the device has to manage a second client, a second rule can be defined where the only field-id that is different will be DLMS-Wrapper-CSAP.

10649

10650

4.16.2.2 SCHC-LPWAN diagnostic (class_id = 127, version = 0)

4.16.2.2.1 Overview

10653 Instances of this IC are intended to provide diagnostic information regarding the transport of
10654 SCHC packets over the LPWAN network that can be useful at the application layer to investigate
10655 potential communications problems and service performance.

SCHC-LPWAN diagnostic	0..n	class_id = 127 , version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name. (static)	octet-string				x
2. schc_packets_tx_counter (dynamic)	double-long-unsigned			0	x + 0x08
3. schc_packets_rx_counter (dynamic)	double-long-unsigned			0	x + 0x10
4. schc_fragments_tx_counter (dynamic)	double-long-unsigned			0	x + 0x18
5. schc_fragments_rx_counter (dynamic)	double-long-unsigned			0	x + 0x20
6. schc_ack_tx_counter (dynamic)	double-long-unsigned			0	x + 0x28

7. schc_ack_rx_counter	(dynamic)	double-long-unsigned			0	x + 0x30
Specific methods (if required)		m/o				
1. reset (data)		...				x + 0x40

10656

10657 **4.16.2.3 Attribute description**10658 **4.16.2.3.1 logical_name**

10659 Identifies the “LPWAN Diagnostic” object instance. See 6.2.23.

10660 **4.16.2.3.2 schc_packets_tx_counter**

10661 It counts the SCHC non fragmented packet transmitted by the device.

10662 **4.16.2.3.3 schc_packets_rx_counter**

10663 It counts the SCHC non fragmented packet received by the device.

10664 **4.16.2.3.4 schc_fragments_tx_counter**

10665 It counts the SCHC fragments transmitted by the device.

10666 **4.16.2.3.5 schc_fragments_rx_counter**

10667 It counts the SCHC fragments received by the device.

10668 **4.16.2.3.6 schc_ack_tx_counter**

10669 It counts the ACK frame transmitted by the device.

10670 **4.16.2.3.7 schc_ack_rx_counter**

10671 It counts the ACK frame received by the device.

10672 **4.16.2.4 Method description**10673 **4.16.2.4.1 reset (data)**

10674 This method resets all counters to 0.

10675 data::= integer (0)

10676

10677

10678 **4.17 LPWAN specific interface classes**10679 **4.17.1 General**10680 Interface classes specified in this clause are related to a given LPWAN technology. Currently,
10681 interface classes related to LoRaWAN 1.0.3 network are specified. ICs for other LPWAN
10682 technologies will be added later.

10683 **4.17.2 LoRaWAN® interface classes**10684 **4.17.2.1 General**

10685 These ICs define the LoRaWAN 1.0.3 parameters used during registration and deregistration
 10686 process, needed at the device side.

10687 NOTE 1 DEVEUI is defined using the MAC address setup interface class (class_id = 43).

10688 NOTE 2 ABP (Activation by Personalization is not permitted).

10689

10690 **4.17.2.2 LoRaWAN setup (class_id = 128, version = 0)**10691 **4.17.2.2.1 Overview**

10692 Instances of this interface class hold the parameters needed to set up a LoRaWAN 1.0.3 device.

10693 This interface class mirrors the parametrization published in LoRa Alliance document:
 10694 *LoRaWAN-TR006 DLMS End Device Monitoring Guidelines.doc*.

LoRaWAN® setup	0..n	class_id = 128 , version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. class (dynamic)	enum				x + 0x08
3. state (dynamic)	enum				x + 0x10
4. max_transmit_EIRP_setting (static)	integer				x + 0x18
5. ADR_mode (dynamic)	boolean				x + 0x20
6. regional_parameters (dynamic)	enum				x + 0x28
7. device_operation (dynamic)	structure				x + 0x38
8. modem_versions (static)	structure				x + 0x40
9. devAddr (dynamic)	double-long-unsigned				x + 0x48
10. join_strategy (dynamic)	enum				x + 0x50
11. multicasts_parameters (dynamic)	array				x + 0x58
Specific methods (if required)	m/o				
1. disconnect_from_network()	...				x + 060
2. change_class(data)	...				x + 068
3. change_region(data)	...				x + 0x70

10695

10696 **4.17.2.2.2 Attribute description**10697 **4.17.2.2.2.1 logical_name**

10698 Identifies the “LoRaWAN setup” object instance. See 6.2.23.

10699 **4.17.2.2.2.2 class**

10700 Defines the class of the device.

10701 Read only attribute, use change_class method to change the value.

10702 **enum:**
 10703 (0) Class A,
 10704 (1) Class B,
 10705 (2) Class C

10706

4.17.2.2.2.3 state

10708 Shows the state of the network connection process of the device.

10709 **enum:**
 10710 (0) not connected,
 10711 (1) connecting state (joining),
 10712 (2) not connected, connected at least one time,
 10713 (3) connected,
 10714 (4) connection error (join error)

10715

4.17.2.2.2.4 max_transmit_EIRP_setting

10717 Read only parameter. Maximum transmission capability of the device in dBm.

4.17.2.2.2.5 ADR_mode

10719 **Adaptive Data Rate** is a LoRaWAN 1.0.3 parameter allowing the network server to control the
 10720 device datarate and TX power. Recommended for static devices (devices expected to not roam
 10721 between networks); it optimises energy consumption and network capacity.

10722 ADR is disabled when ADR_Mode is set to FALSE.

10723 ADR is enabled when ADR_Mode is set to TRUE.

4.17.2.2.2.6 regional_parameters

10725 Identifies the list of region names that a network and a device need to know, to open a
 10726 LoRaWAN service. Those regions are used to support local radio regulation specifics, like duty
 10727 cycle (or LBT), max time over air per message, max power, ...

10728 They also govern the selection of default radio channels, default radio settings, the channel
 10729 mode (fixed or dynamic, ...)

```
10730                   regional_parameters := enum
10731                   {
10732                   (0) EU868,
10733                   (1) US915,
10734                   (2) CN779 (Deprecates 2021-01-01),
10735                   (3) EU433,
10736                   (4) AU915,
10737                   (5) CN470,
10738                   (6) AS923-1,
10739                   (7) AS923-2,
10740                   (8) AS923-3,
10741                   (9) KR920,
10742                   (10) IN865,
10743                   (11) RU864
10744                   }
```

10745

10746 **4.17.2.2.2.7 device_operation**

10747 Counters that can help to identify issues.

```
10748     device_operation ::= structure
10749     {
10750         TotalJoinRequestCounter: long unsigned,
10751         TimeSinceLastJoinRequest: long unsigned,
10752         TimeSinceLastJoinAccept: long unsigned
10753     }
```

10754 Where:

10755 TotalJoinRequestCounter: Number of join requests since last reset.

10756 TimeSinceLastJoinRequest: Number of seconds elapsed since the device initiated the
10757 connection process.

10758 TimeSinceLastJoinAccept: Number of seconds elapsed since the device connected to the
10759 network.

10760 **4.17.2.2.8 modem_versions**

10761 Identifies the modem hardware, software and protocol versions and the version of the regional
10762 parameters.

```
10763     versions ::= structure
10764
10765     {
10766         HardwareVersion: CHOICE
10767         {
10768             octet-string,
10769             visible-string
10770         },
10771
10772         SoftwareVersion: CHOICE
10773         {
10774             octet-string,
10775             visible-string
10776         },
10777
10778         RegionalParametersVersion: CHOICE
10779         {
10780             octet-string,
10781             visible-string
10782         },
10783
10784         ProtocolVersion: CHOICE
10785         {
10786             octet-string,
10787             visible-string
10788         }
10789     }
```

10791 **4.17.2.2.9 devAddr**

10792 32 bits device address identifying the device inside the current network. Assigned by the
10793 Network Server of the device. Should be 0 when not joined.

10794 **4.17.2.2.10 join_strategy**

10795 Defines the algorithm, timing/DR to apply during the joining.

10796 To be documented by vendor. Vendor can assign a code to a Join strategy (where applying).

10797 **4.17.2.2.11 multicasts_parameters**

10798 Both DLMS, IP and LoRaWAN are multicast technologies, LoRaWAN 1.0.3 supporting up to
10799 four multicast groups. One or more group(s) can be created for a set of DLMS devices then
10800 SCHC gateway will forward all IP multicast data corresponding to that group to the end devices.

10801 LoRa Alliance has published a recommended application layer messaging definition allowing to
10802 remotely manage devices groups, and program class B or C distribution window for a given
10803 group.

```
multicasts_parameters ::= array multicast_parameters
multicast_parameters ::= structure
{
  mc_addr:          double-long-unsigned,
  mc_key:           octet-string,
  mc_min_fcount:   double-long-unsigned,
  mc_max_fcount:   double-long-unsigned,
  mc_start_time:   date-time,
  mc_duration:     long-unsigned,
  mc_class:         enum,
  mc_downlink_datarate: unsigned-integer
  mc_downlink_frequency: double-long-unsigned
}
```

10819 Where:

10820 **mc_addr:** multicast device address;

10821 Note: Should be 0 when multicast group is not used,32 bits

10822 **mc_key:** key used to encode the payload of the multicast group;

10823 Note: 128 bits

10824 **mc_min_fcount:** minimum multicast Fcounter, device should drop payload with lower FCount;

10825 **mc_max_fcount:** maximum multicast Fcounter, device should drop payload with higher FCount

10826 **mc_start_time:** Time to start the multicast session, if applicable

10827 **mc_duration:** Maximum duration of the multicast session

10828 **mc_class:** Device shall switch to the class during the length of the multicast window

```
enum:
  Class B,
  Class C
  All others reserved
```

10833

10834 **mc_downlink_datarate:** Datarate of the multicast session

10835 **mc_downlink_frequency:** Frequency of the multicast session, in Hz

10836

10837 An end-device supports a maximum of 4 simultaneous groups, and a minimum of 0. Up to 4
 10838 elements may be used in the array.

10839 **4.17.2.2.3 Method description**

10840 **4.17.2.2.3.1 disconnect_from_network()**

10841 This method corresponds to a LoRaWAN 1.0.3 new join procedure.

10842 Data ::= integer (0)

10843 **4.17.2.2.3.2 change_class (data)**

10844 This method is used to change the class attribute. Data is an enum.

10845 **4.17.2.2.3.3 change_region (data)**

10846 This method is used to change locally the Regional Parameter. Can be used only in
 10847 disconnected state.

10848

10849 **4.17.2.3 LoRaWAN diagnostic (class_id = 129, version = 0)**

10850 **4.17.2.3.1 Overview**

10851 Instances of this IC are intended to provide diagnostic information regarding the LoRaWAN
 10852 1.0.3 lower layers to investigate potential communications problems and service performance.

LoRaWAN diagnostic	0..n	class_id = 129 , version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. internal_error_code (dynamic)	enum				x + 0x08
3. out_frames_u_counter (dynamic)	double-long-unsigned			0	x + 0x10
4. out_frames_c_counter (dynamic)	double-long-unsigned			0	x + 0x18
5. in_frames_u_counter (dynamic)	double-long-unsigned			0	x + 0x20
6. in_frames_c_counter (dynamic)	double-long-unsigned			0	x + 0x28
7. in_mac_command_counter (dynamic)	double-long-unsigned			0	x + 0x30
8. in_mac_ans_error_counter (dynamic)	double-long-unsigned			0	x + 0x38
9. in_mac_ignored_counter (dynamic)	double-long-unsigned			0	x + 0x40
10. in_per (dynamic)	integer				x + 0x48
11. in_mean_rssi_rx1 (dynamic)	integer				x + 0x50
12. in_mean_snr_rx1 (dynamic)	integer				x + 0x58
13. in_mean_rssi_rx2 (dynamic)	integer				x + 0x60
14. in_mean_snr_rx2 (dynamic)	integer				x + 0x68
Specific methods (if required)	m/o				
1. reset (data)	...				

10853

10854

10855 **4.17.2.3.2 Attribute description**10856 **4.17.2.3.2.1 logical_name**

10857 Identifies the “LPWAN Diagnostic” object instance. See 6.2.23.

10858 **4.17.2.3.2.2 internal_error_code**

10859 Fault condition identifier.

10860 To be documented by vendor. When an end-device is faulty, vendor can assign a code to the
10861 fault condition.10862 **4.17.2.3.2.3 out_frames_u_counter**

10863 Number of frames sent unconfirmed.

10864 **4.17.2.3.2.4 out_frames_c_counter**

10865 Number of frames sent confirmed.

10866 **4.17.2.3.2.5 in_frames_u_counter**

10867 Number of frames received unconfirmed.

10868 **4.17.2.3.2.6 in_frames_c_counter**

10869 Number of frames received confirmed.

10870 **4.17.2.3.2.7 in_mac_command_counter**

10871 Number of MAC command received from the network. Note : all commands are answered.

10872 **4.17.2.3.2.8 in_mac_ans_error_counter**

10873 Number of MAC answers replied with error.

10874 **4.17.2.3.2.9 in_mac_ans_ignored_counter**

10875 Number of MAC commands ignored.

10876 **4.17.2.3.2.10 in_per**

10877 Per : packet error rate. Computed over the 5 received downlink.

10878 Unit is percentage (56).

10879 **4.17.2.3.2.11 in_mean_rssi_rx1**

10880 Computed over the last 5 received downlink on rx1.

10881 Unit is dBm

10882 **4.17.2.3.2.12 in_mean_snr_rx1**

10883 Computed over the last 5 received downlink on rx1.

10884 Unit is dB

10885 **4.17.2.3.2.13 in_mean_rssi_rx2**

10886 Computed over the last 5 received downlink on rx2.

10887 Unit is dBm

10888 **4.17.2.3.2.14 in_mean_snr_rx2**

10889 Computed over the last 5 received downlink on rx2.

10890 Unit is dB

10891 **4.17.2.3.3 Method description**10892 **4.17.2.3.3.1 reset (data)**

10893 This method resets all counters to 0.

10894 data:: = Integer (0)

10895

10896 **4.18 Interface classes for setting up and managing the DLMS/COSEM profile for Wi-**
10897 **SUN networks**10898 **4.18.1 Wi-SUN setup (class_id = 95, version 0)**10899 **4.18.1.1 Overview**

10900 This IC has the purpose of modelling the Wi-SUN network setup parameters.

10901 All attributes in [FANSPEC] and [PHYSPEC] are written in camelCase and for the benefit of
10902 easy traceability these parameters are transposed into snake_case (underscore notation) within
10903 the DLMS/COSEM specification.

Wi-SUN setup	0..n	class_id = 95, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. network_name (static)	octet-string				x + 0x08
3. routing_method (dynamic)	enum	0	1	1	x + 0x10
4. pan_id (dynamic)	long-unsigned	0	65534		x + 0x18
5. disc_imin (static)	unsigned	1	255		x + 0x20
6. disc_imax (static)	unsigned	1	8		x + 0x28
7. data_message_imin (static)	unsigned	1	255	10	x + 0x30
8. data_message_imax (static)	unsigned	1	8	3	x + 0x38
9. default_dio_interval_min (static)	unsigned	1	255		x + 0x40
10. default_dio_interval_doublings (static)	unsigned	1	8		x + 0x48

11. channel_plan	(static)	structure				x + 0x50
12. channel_function	(static)	enum	0	3		x + 0x58
13. excluded_channels	(static)	CHOICE				x + 0x60
14. join_state	(dynamic)	enum	1	5		x + 0x68
15. reg_channel_exclusions	(static)	structure				x + 0x70
Specific methods (if required)		m/o				
1. reset_join_state (data)		o				x + 0x78

10904

10905 4.18.1.2 Attribute description

10906 4.18.1.2.1 logical_name

10907 Identifies the “Wi-SUN setup” object instance. See clause 6.2.32.

10908 4.18.1.2.2 network_name

10909 Provides the name of the network that the device is allowed to join, or is a member of.

10910 This is represented by ASCII characters.

10911 NOTE 1 The network name has a maximum of 32 octets.

10912 4.18.1.2.3 routing_method

10913 Provides the routing method employed by the device, two options are available:

10914 enum: (2) 1

- (0) Layer 2 routing,
- (1) Layer 3 routing

10917 NOTE 2 Default value according to [FANSPEC] is 1.

10918

10919 4.18.1.2.4 pan_id

Provides the PAN ID (configured at the Border Router and is known by every node on the network. This attribute is not controlled in the application layer.

10922 NOTE 3 See [FANSPEC], 6.3.1.1.

10923 NOTE 4 IEEE 802.15.4 reserves the value 65 535 for the broadcast PAN ID.

10924 4.18.1.2.5 disc_imin

10925 Provides the time in seconds of the time-out for network discovery.

10926 NOTE 5 Typical values of 60s for large PANs (larger than 100 devices) or 15s for small PANs (less than 100
10927 devices) may be employed.

10928 4.18.1.2.6 disc_imax

10929 Provides the number of doublings of the value of "imin" (1-8 doublings of disc_imin).

10930 NOTE 6 Typical value of 4 for large PANs (approximately 16 minutes), value of 2 for small PANs (approximately
10931 60 seconds).

10932 **4.18.1.2.7 data_message_imin**

10933 Provides the network broadcast timeout minimum value in seconds.

10934 NOTE 7 Typical default value of 10s.

10935 **4.18.1.2.8 data_message_imax**

10936 Provides the maximum broadcast timeout as a function of data_message_imin (1-8 doublings
10937 of data_message_imin).

10938 NOTE 8 Typical default value of 3 (80 seconds).

10939 **4.18.1.2.9 default_dio_interval_min**

10940 Provides the ROLL RPL DIO minimum Interval (imin) The DIO trickle timer imin value is defined
10941 as 2 to the power of this value in ms (milliseconds).

10942 NOTE 9 Typical default value for a large network: 19 (an imin of approximately 8 minutes) and a small network:
10943 15 (an imin of approximately 32 seconds).

10944 **4.18.1.2.10 default_dio_interval_doublings**

10945 Provides the ROLL RPL DIO Interval IMAX and is a number of doublings of the value of
10946 default_dio_interval_min (1-8 doublings). A value of 1 for large PANs (approximately 16 minutes
10947 i.e. 8 x 2 minutes), value of 2 for small PANs (approximately 128 seconds i.e 32 x 2 x 2).

10948 **4.18.1.2.11 channel_plan**

10949 Regional spectrum settings as specified in [PHYSPEC], Table 3. The values of
10950 regulatory_domain and operating_class uniquely identify the combination of operating
10951 parameters that the implementation is programmed to sustain. These parameters are subject
10952 to geographic regulation and may change from time to time depending on when regional
10953 spectrum authorities update their national frequency tables.

```
10954                 channel_plan ::= structure
10955                 {
10956                 regulatory_domain_identifier:    unsigned,
10957                 operating_class_designator:    unsigned
10958         }
```

10959 Where:

10960 – regulatory_domain_identifier identifies the regulatory domain that the product is operating
10961 in, thereby identifying various PHY layer parameters including frequency bands, PHY
10962 modes, and channel spacing according to [PHYSPEC], Table 3,

10963 – operating_class_designator identifies the operating class of the device according to
10964 [PHYSPEC], Table 3. Operating class is unique only within the regulatory domain.

10965 NOTE 10 The above sets the channel spacing, channel 0 frequency and number of channels plus the symbol
10966 rate.

10967 **4.18.1.2.12 channel_function**

10968 Provides the channel function. There are currently 4 channel functions: fixed channel or a
10969 channel hopping mode where certain other parameters are specified as enumerated below.

10970 enum:

Value	Description	Note
0	Fixed Channel	The transmitting node uses a single, fixed channel from the available channels.
1	TR51CF	The transmitting node uses the channel function described in ANSI/TIA-4957.200 (TR51CF), 7.1.
2	DH1CF	The transmitting node uses the Direct Hash channel function (DH1CF described in [FANSPEC], 6.3.4.5).
3	Vendor defined	The transmitting node uses a vendor defined channel function explicitly provided by the Hop Count and Hop List (defined in [FANSPEC] 6.3.2.3.2.1.4).
Other	reserved	

10971

10972 **4.18.1.2.13 excluded_channels**

10973 Provides a means via which a node can have either a range (excluded_channel_ranges) or
 10974 individual channels (excluded_channel_mask) notched out (not visited) of the node's channel
 10975 hop sequence. Excluded_channel_ranges and excluded_channel_mask are configured for
 10976 device specific purposes and shall be a subset of the channel_plan that fulfils regulatory
 10977 channel restriction requirements. If no excluded channel ranges are set then the attribute shall
 10978 be set to null-data.

10979 Either excluded_channel_range OR excluded_channel_mask may be configured in a mutually
 10980 exclusive way.

```
10981     excluded_channels: CHOICE
10982     {
10983         null-data [0],
10984         excluded_channel_ranges,
10985         excluded_channel_mask
10986     }
10987
10988
10989     null-data is selected when there are no excluded channels.
10990
10991     excluded_channel_ranges ::= structure
10992     {
10993         number_of_ranges:    integer,
10994         ranges_list:          array channel_range
10995     }
10996     channel_range ::= structure
10997     {
10998         start_channel_number:    long unsigned,
10999         end_channel_number:      long unsigned
11000     }
```

11001

11002 Where:

- 11003 – number_of_ranges is the number of ranges in the ranges_list;
- 11004 – ranges_list is an array of start and end channel numbers representing one or multiple
 11005 excluded channel ranges;
- 11006 – channel_range is a structure of start and end channel numbers that together represent an
 11007 excluded channel range. The start and end channel shall also be considered to be included
 11008 in the excluded range.

11009 NOTE 11 Channels which are illegal or invalid may still not be used even if they are not included in a
 11010 channel_range.

11011

11012

11013 excluded_channel_mask ::= bit-string

11014 Excluded_channel_mask is a variable length bit_string (varying in multiples of
 11015 8) representing the series of channels that are included or not included in the
 11016 device's channel hop sequence. The Excluded Channel Mask field provides
 11017 a means via which a node explicitly reports, for all channels within its channel
 11018 hop sequence, which channels are and are not visited.

Bit	Channel
0 (first bit)	0
1	1
2	2
3	3
....
n	n

11019 NOTE 12 [FANSPEC] states "If a bit in the mask is set to 1, the corresponding channel MUST NOT be used.
 11020 Bits that are not valid channels (unused following rounding to a multiple of 8) MUST be set to 1 on transmit and
 11021 ignored on receipt."

11022 This attribute is to be used for efficiently notching out individual channels rather than a range
 11023 of channels within a band plan.

11024 **4.18.1.2.14 join_state**

11025 Provides the join state of the device.

11026 enum: (0) Initialisation state,
 11027 (1) Select PAN,
 11028 (2) Authenticate,
 11029 (3) Acquire PAN Config,
 11030 (4) Configure Routing,
 11031 (5) Operational,
 11032 (6)...(255) reserved

11033 NOTE 13 Applications on a device should only send or receive traffic when the device is in join state 5.

11034 **4.18.1.2.15 reg_channel_exclusions**

11035 Regional channel exclusions as designated in [FANSPEC] . The values of regulatory_domain
 11036 and operating_class from channel_plan uniquely identify the combination of operating
 11037 parameters that the implementation is programmed to sustain. This field identifies any specific
 11038 channels that must not be used when operating in a specific country.

11039 reg_channel_exclusions ::= structure
 11040 {
 11041 num_channels: long_unsigned,
 11042 excluded_channels: bit-string
 11043 }
 11044

11045 Where:

11046 num_channels is defined as the total number of channels, according to [PHYSPEC], Table 3
 11047 when operating with the regulatory_domain and operating_class defined in channel_plan,

11048 excluded_channels is a bit-string, sized as num_channels, where the first bit, index 0 is Channel
 11049 0 and the last bit is “num_channels – 1”. For each index into excluded_channels, a value of “0”
 11050 indicates the channel can be used and a value of “1” indicates the channel must not be used.

11051 NOTE 14 Unlike the Channel Exclusion in the [FANSPEC] , the regulatory channel exclusion is permanent
 11052 and cannot be modified after configuration. These channels represent regulatory channel exclusions that apply to
 11053 the particular country or region in that the configured device is deployed.

11054 **4.18.1.3 Method description**

11055 **4.18.1.3.1 reset_join_state (data)**

11056 Resets join state attribute to 0.

11057

11058 **4.18.2 Wi-SUN diagnostic (class_id = 96, version 0)**

11059 **4.18.2.1 Overview**

11060 Instances of this IC are intended to provide diagnostic information regarding the Wi-SUN
 11061 network that can be useful to investigate potential communication problems and service
 11062 performance.

11063 To facilitate management from the application, the following diagnostics are provided.

Wi-SUN diagnostics	0..n	class_id = 96, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. sysdescr (dynamic)	octet-string				x + 0x08
3. ifnumber (dynamic)	integer			1	x + 0x10
4. iftable (dynamic)	array				x + 0x18
5. neighbour_table_information (dynamic)	array				x + 0x20
6. transmission_information (dynamic)	array				x + 0x28
<i>Specific methods (if required)</i>	m/o				
1. reset	o				

11064

11065 **4.18.2.2 Attribute description**

11066 **4.18.2.2.1 logical_name**

11067 Identifies the “Wi-SUN diagnostic” object instance. See clause 6.2.32.

11068 **4.18.2.2.2 sysdescr**

11069 A textual description of the entity. This value should include the full name and version
 11070 identification of the system’s hardware type, software operating-system, and networking
 11071 software. It is mandatory that this only contains printable ASCII characters.

11072 **4.18.2.2.3 ifnumber**

11073 The number of interfaces (ifentry) present within the physical device as per RFC 1213. Each
 11074 interface shall have a value of ifentry defined in iftable.

11075 **4.18.2.2.4 iftable**

11076 A list of interface entries as per RFC 1213. The number of entries in the ifentry array is given
 11077 by the value of attribute ifnumber; normally this shall be 1.

```

11078          array ifentry
11079
11080          ifentry::=structure
11081          {
11082              ifdescr          octet-string,
11083              ifphysaddress    octet-string,
11084              ifinoctets        double-long-unsigned,
11085              ifinucastpackets double-long-unsigned,
11086              ifinnucastpackets double-long-unsigned,
11087              ifinerrors        double-long-unsigned,
11088              ifinunknownprotos double-long-unsigned,
11089              ifoutoctets       double-long-unsigned,
11090              ifoutucastpackets double-long-unsigned,
11091              ifoutnucastpackets double-long-unsigned,
11092              ifouterrors       double-long-unsigned,
11093              ifoutQlen         double-long-unsigned,
11094          }
  
```

11095

11096 Where:

Item	description
ifdescr	An ASCII description of the interface as per RFC 1213. A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the hardware interface.
ifphysaddress	The interface's address at the protocol layer immediately 'below' the network layer in the protocol stack. For interfaces which do not have such an address (e.g., a serial line), this object should contain an empty octet string. In the case of Wi-SUN this entry shall be the EUI64.
ifinoctets	The total number of octets received on the interface, including framing characters.
ifinucastpackets	The number of subnetwork-unicast packets delivered to a higher-layer protocol. In the case of Wi-SUN diagnostic this shall be the number of packets passed to the DLMS/COSEM application layer as per RFC 1213.
ifinnucastpackets	The number of non-unicast (i.e., subnetwork-broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol as per RFC 1213.
ifinerrors	The number of inbound packets that contained errors preventing them from being deliverable to a higher layer protocol as per RFC 1213.
ifinunknownprotos	The number of packets received via the interface at the network layer which were discarded because of an unknown or unsupported protocol as per RFC 1213.

<code>ifoutoctets</code>	The total number of octets transmitted out of the interface at the network layer, including framing characters as per RFC 1213.
<code>ifoutunicastpackets</code>	The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or not sent. In the case of Wi-SUN diagnostic higher layer protocols shall be defined as the DLMS/COSEM application layer.
<code>ifoutnucastpackets</code>	The total number of packets that higher-level protocols requested be transmitted to a non-unicast (i.e., a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent. In the case of Wi-SUN diagnostic higher layer protocol shall be defined as the DLMS/COSEM application layer as per RFC 1213.
<code>ifouterrors</code>	The number of outbound packets that could not be transmitted because of errors.
<code>ifoutqlen</code>	The length of the output packet queue (in packets) at the network layer,

11097

11098 **4.18.2.2.5 neighbour_table_**11099 **information** The neighbour table keeps track of devices no more than 1 hop away.

```

11100      array neighbour
11101
11102      neighbour ::= structure
11103
11104      {
11105          neighbour_id: octet-string,
11106          neighbour_type: enum,
11107          device_rank: unsigned,
11108          rssi: long-unsigned,
11109          etx_to_parent: double-long-unsigned
11110      }

```

11111

11112 Where:

- 11113 – neighbour_id is the EUI 64 of the neighbour device. The EUI-64 shall be of the modified
11114 EUI-64 format described in RFC 4291;
- 11115 – neighbour_type is an enumeration defining whether the neighbour device is a parent, child
11116 or sibling.

```

11117          enum:
11118              (0) Undetermined,
11119              (1) Parent,
11120              (2) Child,
11121              (3) Sibling

```

- 11122 – device_rank is a function of distance in hops of the device to the root device in the DODAG.
The smaller the number the closer the device is to the DODAG root device;
- 11123 – rssi is a measure of the signal strength of the neighbour;
- 11124 – etx_to_parent is the Expected Number of transmissions required from the device to the
11125 destination via the parent in question required to transmit a packet. 1 is the perfect situation
11126 and FFFFFFFF is assumed to be equal to infinity and therefore a non-functioning link.

11129

11130 The first entry in the array of neighbours shall relate to the preferred parent and all subsequent
 11131 entries shall be secondary parents.

11132 NOTE Parent represents the node in the routing nearer to the Border Router (and may include the Border Router),
 11133 child represents a node in the routing that has some dependency on this node, and sibling represents a node where
 11134 communications is possible but is neither the parent or child of this node.

11135 **4.18.2.2.6 transmission_information**

11136 Provides transmission information and quality of service per interface.

```
11137     array      transmission_element
11138
11139     transmission_element ::= structure
11140
11141     {
11142         neighbour_id          octet-string,
11143         number_of_receptions   double-long-unsigned,
11144         number_of_sucessful_trans double-long-unsigned,
11145         number_of_failed_trans    double-long-unsigned
11146     }
```

11147

11148 Where:

- 11149 – neighbour_id is the EUI-64 of the FAN interface. The EUI-64 shall be of the modified EUI-
 11150 64 format described in RFC 4291;
- 11151 – number_of_receptions is a simple counter of the total number of receptions from the device
 11152 identified by interface_id;
- 11153 – number_of_sucessful_trans is a simple counter of the total number of successful
 11154 transmissions acknowledged by the device identified by interface_id;
- 11155 – number_of_failed_trans is a simple counter of the total number of failed transmissions
 11156 acknowledged by the device identified by interface_id.

11157

11158 **4.18.2.3 Method description**

11159 **4.18.2.3.1 reset (data)**

11160 Clears attributes 2...4 of the object instance, that is:

```
11161     - sysdescr,
11162     - ifnumber,
11163     - iftable
11164     data ::= integer (0)
```

11165

11166 **4.18.3 RPL diagnostic (class_id = 97, version 0)**

11167 **4.18.3.1 Overview**

11168 As per [FANSPEC] 6.2.3.1.8 RPL is supported for the dissemination of IPv6 multicast
 11169 packets. Instances of this IC are intended to provide diagnostic information regarding the Wi-

11170 SUN network from RPL diagnostic parameters as per RFC 6550 that can be useful to investigate
 11171 potential communication problems and service performance issues.

RPL diagnostic	0..n	class_id = 97, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. rpl_instance_id (dynamic)	unsigned	0		191	x + 0x08
3. dodag_version_number (dynamic)	unsigned				x + 0x10
4. dodag_rank (dynamic)	long				x + 0x18
5. grounded (dynamic)	boolean				x + 0x20
6. mode_of_operation (dynamic)	enum				x + 0x28
7. dodag_prf (dynamic)	unsigned	0	7		x + 0x30
8. dodag_dtsn (dynamic)	unsigned				x + 0x38
9. dodag_id (dynamic)	octet-string				x + 0x40
Specific methods (if required)	m/o				
1. reset (data)	o				x + 0x48

11172

4.18.3.2 Attribute description

4.18.3.2.1 logical_name

11173 Identifies the “RPL diagnostic” object instance. See clause 6.2.32.

4.18.3.2.2 rpl_instance_id

11174 Holds the RPL instance ID as defined in RFC 6550. The values below are defined in RFC 6550 to identify whether or not the RPL Instance ID relates to a Global Instance ID, as in the case of coordinated networks or Local Instance ID as issued by a DODAG in the case of unilateral networks.

11175 Bits 0 and 1 of the unsigned value act as flags where bit 0 when clear defines that the rpl_instance_id is global and has 127 possible values. Bit 0 set means that the rpl_instance_id is local, and in such case bit 1 shall always be clear assuming that the server is the recipient of messages from the rpl_instance_id allocating DODAG.

Instance ID Type	Start value	End value
Global Instance ID	0	127
Local Instance ID	128	191
Not Used	192	255

11176

4.18.3.2.3 dodag_version_number

11177 8-bit unsigned integer set by the DODAG root to the DODAGVersionNumber. RFC 6550, 8.2 describes the rules for DODAGVersionNumbers and how they affect DIO processing.

4.18.3.2.4 dodag_rank

11178 Indicates the DODAG Rank of the node sending the DIO message. RFC 6550, 8.2 describes how Rank is set and how it affects DIO processing.

4.18.3.2.5 grounded

The Grounded 'G' flag indicates whether the DODAG advertised can satisfy the application-defined goal. If the flag is set, the DODAG is grounded. If the flag is cleared, the DODAG is floating.

4.18.3.2.6 mode_of_operation

The Mode of Operation (MOP) field identifies the mode of operation of the RPL instance as administratively provisioned at and distributed by the DODAG root. All nodes that join the DODAG shall honor the MOP in order to fully participate as a router, or else they must only join as a leaf. MOP is encoded as detailed below:

enum: (0) No Downward routes maintained by RPL,
 (1) Non-Storing Mode of Operation,
 (2) Storing Mode of Operation with no multicast support,
 (3) Storing Mode of Operation with multicast support.

4.18.3.2.7 dodag_prf

Defines how preferable the root of this DODAG is compared to other DODAG roots within the instance. DAG Preference ranges from 0x00 (least preferred) to 0x07 (most preferred). The default is 0 (least preferred). RFC 6550, 8.2 describes how DAG Preference affects DIO processing.

4.18.3.2.8 dodag_dtsn

Destination Advertisement Trigger Sequence Number (DTSN) 8-bit unsigned integer set by the node issuing the DIO message. The Destination Advertisement Trigger Sequence Number (DTSN) flag is used as part of the procedure to maintain Downward routes. The details of this process are described in RFC 6550, 9.

4.18.3.2.9 dodag_id

128-bit IPv6 address represented as octet-string (length 16) set by a DODAG root that uniquely identifies a DODAG. The DODAGID MUST be a routable IPv6 address belonging to the DODAG root.

4.18.3.3 Method description

4.18.3.3.1 reset (data)

Clears or (sets to default value) all dynamic fields of the object instance.

data ::= integer (0)

4.18.4 MPI diagnostic (class_id = 98, version 0)

4.18.4.1 Overview

Instances of this IC are intended to provide diagnostic information regarding the Wi-SUN network from MPL diagnostic parameters as per RFC 7731 that can be useful at the application layer to investigate potential communication problems and service performance issues.

MPL diagnostic	0..n	class_id =98 , version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. proactive_forwarding (dynamic)	boolean				x + 0x08
3. z (dynamic)	unsigned	0	0	0	x + 0x18
4. tunit (dynamic)	unsigned	0x01	0xFE		x + 0x20
5. se_lifetime (dynamic)	long-unsigned	0x0001	0xFFFFE		x + 0x28
6. dm_k (dynamic)	unsigned				x + 0x30
7. dm_imin (dynamic)	long-unsigned	0x0001	0xFFFFE		x + 0x38
8. dm_imax (dynamic)	unsigned	0x01	0xFE		x + 0x40
9. dm_t_exp (dynamic)	long-unsigned	0x0001	0xFFFFE		x + 0x48
10. c_k (dynamic)	unsigned				x + 0x50
11. c_imin (dynamic)	long-unsigned	0x0001	0xFFFFE		x + 0x58
12. c_imax (dynamic)	unsigned	0x01	0xFE		x + 0x60
13. c_t_exp (dynamic)	long-unsigned	0x0001	0xFFFFE		x + 0x68
Specific methods (if required)	m/o				
1. reset (data)	o				x + 0x70

11230

4.18.4.2 Attribute description**4.18.4.2.1 logical_name**

Identifies the “MPL diagnostic” object instance. See clause 6.2.32.

4.18.4.2.2 proactive_forwarding

A flag to indicate PROACTIVE_FORWARDING.

NOTE 1 This flag is set if PROACTIVE_FORWARDING = TRUE, as per RFC 7774, 2.1.

4.18.4.2.3 z

Reserved for future use. Defined as per RFC 7774, 2.1.

NOTE 2 This is set to zero in all DHCP Servers, DHCP Clients shall ignore any none zero values.

4.18.4.2.4 tunit

Unit time of timer parameters (SE_LIFETIME and *_IMIN) in this option.

NOTE 3 0 and 0xFF are reserved and are not be used, as per RFC 7774, 2.1.

4.18.4.2.5 se_lifetime

SEED_SET_ENTRY_LIFETIME/TUNIT, in milliseconds (ms).

NOTE 4 0 and 0xFFFF are reserved and are not to be used, as per RFC 7774, 2.1.

4.18.4.2.6 dm_k

DATA_MESSAGE_K, as per RFC 7774, 2.1.

4.18.4.2.7 dm_imin

DATA_MESSAGE_IMIN/TUNIT, in milliseconds (ms).

NOTE 5 0 and 0xFFFF are reserved and are not to be used, as per RFC 7774, 2.1.

4.18.4.2.8 dm_imax

DATA_MESSAGE_IMAX. The actual maximum timeout is described as a number of doublings of DATA_MESSAGE_IMIN, as described in RFC 6206, 4.1.

NOTE 6 0 and 0xFF are reserved and MUST NOT be used, as per RFC 7774, 2.1.

4.18.4.2.9 dm_t_exp

DATA_MESSAGE_TIMER_EXPIRATIONS, as per RFC 7774, 2.1.

NOTE 7 Values 0 and 0xFFFF are reserved and are not to be used, as per RFC 7774, 2.1.

4.18.4.2.10 c_k

CONTROL_MESSAGE_K, as per RFC 7774, 2.1.

4.18.4.2.11 c_imin

CONTROL_MESSAGE_IMIN/TUNIT, in milliseconds (ms).

NOTE 8 0 and 0xFFFF are reserved and are not to be used, as per RFC 7774.

4.18.4.2.12 c_imax

CONTROL_MESSAGE_IMAX. The actual maximum timeout is described as a number of doublings of CONTROL_MESSAGE_IMIN.

NOTE 9 0 and 0xFF are reserved and are not to be used, as per RFC 7774.

4.18.4.2.13 c_t_exp

CONTROL_MESSAGE_TIMER_EXPIRATIONS.

NOTE 10 0 and 0xFFFF are reserved and are not to be used, as per RFC 7774.

11270

4.18.4.3 Method description**4.18.4.3.1 reset (data)**

Clears all dynamic fields of the object instance.

data ::= integer (0)

11275

11276 **4.19 Interface classes for setting up and managing the DLMS/COSEM profile for**
 11277 **ISO/IEC 14908 PLC networks**

11278 **4.19.1 General**

11279 The interface classes specified here are used in devices implementing the IEC 62056-8-8:2020
 11280 DLMS/COSEM communication profile for ISO/IEC 14908 series networks.

11281 **4.19.2 ISO/IEC 14908 identification (class_id = 130, version = 0)**

11282 **4.19.2.1 Overview**

11283 Instances of the ISO/IEC 14908 identification IC allow the identification of the device and the
 11284 network to which the device is connected.

ISO/IEC14908 identification	0...n	class_id = 130, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. node_ID (static.)	unsigned	1	127		x + 0x08
3. subnet_ID (static)	unsigned	1	255		x + 0x10
4. domain_ID (static)	octet-string				x + 0x18
5. program_ID (static)	octet-string				x + 0x20
6. unique_node_ID (static)	octet-string				x + 0x28
Specific methods	m/o				

11285

11286 **4.19.2.2 Attribute description**

11287 **4.19.2.2.1 logical_name**

11288 Identifies the “ISO/IEC 14908 identification” object instance. See 6.2.30.

11289 **4.19.2.2.2 node_ID**

11290 Identification of the node.

11291 **4.19.2.2.3 subnet_ID**

11292 Identification of the subnet to which the device is connected.

11293 **4.19.2.2.4 domain_ID**

11294 Identification of the network. It shall be 3 bytes.

11295 **4.19.2.2.5 program_ID**

11296 Uniquely identifies the functionalities and version of the firmware running on the device. It shall
 11297 be 8 bytes.

11298 **4.19.2.2.6 unique_node_ID**

11299 Identification of the physical connection module connected to the ISO/IEC 14908-1:2012
 11300 network. Each compliant module is assigned a unique 48-bit identifier called Unique_Node_ID.

11301 This ID is unique worldwide and is set at the time of manufacture. The value of this identifier
 11302 does not change from the time of manufacture.

11303

11304

11305 **4.19.3 ISO/IEC 14908 protocol setup (class_id = 131, version = 0)**

11306 **4.19.3.1 Overview**

11307 Instances of the ISO/IEC 14908 protocol setup IC allow the configuration of the ISO/IEC 14908
 11308 device.

ISO/IEC 14908 protocol setup	0...n	class_id = 131, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. inactivity_timeout (static)	long-unsigned				x + 0x08
Specific methods	m/o				

11309

11310 **4.19.3.2 Attribute description**

11311 **4.19.3.2.1 logical_name**

11312 Identifies the “ISO/IEC 14908 protocol setup” object instance. See 6.2.30.

11313 **4.19.3.3 inactivity_timeout**

11314 Minutes of absence of adaptation layer messages from the NNAP before the LNAP is
 11315 considered to be out of communication; see IEC 62056-8-8:2020, F4.3.

11316

11317 **4.19.4 ISO/IEC 14908 protocol status (class_id = 132, version = 0)**

11318 **4.19.4.1 Overview**

11319 Instances of the ISO/IEC 14908 protocol status IC allow the status of the protocol in the ISO/IEC
 11320 14908 device to be determined.

ISO/IEC 14908 protocol status	0...n	class_id = 132, version = 1			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. transmission_errors (dyn)	long-unsigned				x + 0x08
3. transmit_tx_failure (dyn)	long-unsigned				x + 0x10
4. transmit_tx_retries (dyn)	long-unsigned				x + 0x18
5. receive_tx_full (dyn)	long-unsigned				x + 0x20
6. lost_messages (dyn)	long-unsigned				x + 0x28
7. missed_messages (dyn)	long-unsigned				x + 0x30
8. layer2_received (dyn)	long-unsigned				x + 0x38
9. layer3_received (dyn)	long-unsigned				x + 0x40
10. messages_received (dyn)	double-long-unsigned				x + 0x48
11. messages_validated (dyn)	double-long-unsigned				x + 0x50
Specific methods	m/o				
1. reset (data)	o				x + 0x70

11321

11322 **4.19.4.2 Attribute description**11323 **4.19.4.2.1 logical_name**

11324 Identifies the “ISO/IEC 14908 protocol status” object instance. See 6.2.30.

11325 **4.19.4.2.2 transmission_errors**

11326 This is a count of the number of transmission errors that have occurred on the network. A transmission error is detected via a CRC error during packet reception or as a packet that is less than 8 bytes long. This could result from a collision, a noisy medium, signal attenuation, etc.

11330 See ISO/IEC 14908-1:2012, 13.8.2.

11331 **4.19.4.2.3 transmit_tx_failure**

11332 The number of times that the node failed to receive expected acknowledgments or responses after retrying the configured number of times.

11334 See ISO/IEC 14908-1:2012B.8.

11335 **4.19.4.2.4 transmit_tx_retries**

11336 The number of retries sent by this node. This does not include retries used for messages sent with repeated service.

11338 See ISO/IEC 14908-1:2012, B.8.

11339 **4.19.4.2.5 receive_tx_full**

11340 The number of times that an incoming packet was discarded because there was no room in the transaction database.

11341

11342 See ISO/IEC 14908-1:2012, B.8.

11343 **4.19.4.2.6 lost_messages**

11344 This is the number of messages that were addressed to the node that were thrown away
11345 because there was no application buffer available for the message;

11346 See ISO/IEC 14908-1:2012, 13.8.2.

11347 **4.19.4.2.7 missed_messages**

11348 This is the number of messages that were on the network but could not be received because
11349 there was no network buffer (packet buffer) available for the message or the network buffer was
11350 too small to receive the message;

11351 See ISO/IEC 14908-1:2012, 13.8.2.

11352 **4.19.4.2.8 layer2_received**

11353 The number of Layer-2 messages received by this node. Layer-2 messages are those that have
11354 correct CRC and can be addressed to any node.

11355 See ISO/IEC 14908-1:2012, B.8.

11356 **4.19.4.2.9 layer3_received**

11357 The number of messages transmitted from layer 3 of the protocol processor. These can include
11358 network variable updates, explicit messages, acknowledgments, retries, reminders, service pin
11359 messages, and any other type of message.

11360 See ISO/IEC 14908-1:2012, B.8.

11361 **4.19.4.2.10 messages_received**

11362 Number of messages received, which are delivered to the Adaptation layer.

11363 **4.19.4.2.11 messages_validated**

11364 Number of correct messages received at Adaptation layer level.

11365

11366 **4.19.4.3 Method description**

11367 **4.19.4.3.1 reset (data)**

11368 Forces a reset of the object. By invoking this method, the values are set to the default value.
11369 The default value is an instance specific constant.

11370 data ::= integer (0)

11371 NOTE: IEC 62056-8-8:2020 specifies version = 0 which does not support the reset method.

11372

11373

11374 **4.19.5 ISO/IEC 14908 diagnostic (class_id = 133, version = 0)**

11375 **4.19.5.1 Overview**

11376 Instances of the ISO/IEC 14908 diagnostic IC provides information about the device status
11377 inside the PLC network.

ISO/IEC 14908 diagnostic	0...n	class_id = 133, version = 1			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. plc_signal_quality_status (dyn)	enum				x + 0x08
3. com_module_state (dyn)	enum				x + 0x10
4. received_message_status (dyn)	unsigned				x + 0x18
5. no_receive_buffer (dyn)	long-unsigned				x + 0x20
6. transmit_no_data (dyn)	long-unsigned				x + 0x28
7. backlog_overflows (dyn)	long-unsigned				x + 0x30
8. late_ack (dyn)	long-unsigned				x + 0x38
9. frequency_invalid (dyn)	long-unsigned				x + 0x40
Specific methods	m/o				
1. reset (data)	o				x + 0x60

11378

11379 **4.19.5.2 Attribute description**

11380 **4.19.5.2.1 logical_name**

11381 Identifies the “ISO/IEC 14908 diagnostic” object instance. See 6.2.30.

11382 **4.19.5.2.2 plc_signal_quality_status**

11383 Provides the PLC signal quality status:

- 11384 enum:
 11385 (0) No PLC traffic detected,
 11386 (1) PLC traffic detected, but no traffic addressed to this device,
 11387 (2) A packet addressed to this device has been received. The signal
 11388 margin is less than or equal to 9 dB,
 11389 (3) A packet addressed to this device has been received. The signal
 11390 margin is between 12 dB or 15 dB,
 11391 (4) A packet addressed to this device has been received. The signal
 11392 margin is greater than or equal to 18 dB,
 11393 (5) Meter is commissioned but its inactivity_timeout as defined in 4.18.1
 11394 has expired. The last packet addressed to this meter had a signal
 11395 margin is less than or equal to 9 dB,
 11396 (6) Meter is commissioned but its inactivity_timeout as defined in 4.18.1
 11397 has expired. The last packet addressed to this meter had a signal
 11398 margin at 12 dB or 15 dB,
 11399 (7) Meter is commissioned but its inactivity_timeout as defined in 4.18.1
 11400 has expired. The last packet addressed to this meter had a signal
 11401 margin greater than or equal to 18 dB.

11402 Note. This parameter is updated every time a packet is received by the device.

11403 **4.19.5.2.3 com_module_state**

11404 Last received state of the device's power line IEC 62056-8-8:2020 communication module.

11405 enum:
 11406 (0) Invalid state,
 11407 (1) Invalid state,
 11408 (2) Unconfigured,
 11409 (3) Applicationless,
 11410 (4) Configured,
 11411 (6) Hard-offline,
 11412 (12) Configured, soft offline,
 11413 (140) Configured, in bypass mode,
 11414 (255) No reply when status requested,
 11415 other Invalid states.

11416 This field is updated on each device power-up, each re-synch request from the NNAP and each
 11417 communication module reset. Any value which is not listed above is invalid state. See IEC
 11418 62056-8-8:2020, 13.4.

11419 NOTE Configured soft offline state (12) and Configured, in bypass mode state (140) are sub states of Configured
 11420 state (4). See IEC 62056-8-8:2020, 13.4 and 13.8

11421

11422 **4.19.5.2.4 received_message_status**

11423 Unsigned integer, constructed as:

Bit 7 (MSB)	b6	b5	b4	b3	b2	b1	Bit 0 (LSB)
Transceiver phase	reserved						See below

11424 Transceiver phase bits are used as below:

	Bit 7	b6	Bit 5
No phase inversion present	Not set	Not set	Not set
In phase normal	Not set	Not set	Set
Plus 120 degrees inverted	Not set	Set	Not set
Minus 120 degrees normal	Not set	Set	Set
Phase 180 degrees inverted	Set	Not set	Not set
Plus 120 degrees normal	Set	Not set	Set
Minus 120 degrees inverted	Set	Set	Not set
Reserved	Set	Set	Set

11425

11426 **4.19.5.2.5 no_receive_buffer**

11427 Number of times the IEC 62056-8-8:2020 lower layers detected a network inbound packet
 11428 addressed to it, but has no buffer to store it. Hence the message could not be made available
 11429 to the adaptation layer.

11430 4.19.5.2.6 transmit_no_data

11431 Number of times the adaptation layer attempted to transmit data to the IEC 62056-8-8:2020
11432 stack but no buffer is available. Updated on occurrence. IEC 62056-8-8:2020, 13.4.

11433 4.19.5.2.7 backlog_overflows

11434 Number of times the MAC layer predictive backlog counter overflowed. See IEC 62056-8-
11435 8:2020, B.8.

11436 4.19.5.2.8 late_ack

11437 Number of times an ACK or response was received for a transaction that has been already
11438 completed. Typically, this occurs due to too short transaction timer values or network
11439 congestion.

11440 4.19.5.2.9 frequency_invalid

11441 Number of times the IEC 62056-8-8:2020 stack detects an invalid frequency. Updated on
11442 occurrence.

11443

11444 4.19.5.3 Method description**11445 4.19.5.3.1 reset (data)**

11446 Forces a reset of the object. By invoking this method, the values of attributes 5..9 are set to the
11447 default value. The default value is an instance specific constant.

11448 data ::= integer (0)

11449 NOTE IEC 62056-8-8:2020 specifies version = 0 which does not support the reset method.

11450

11451

11452

11453 **5 Previous versions of interface classes**

11454 **5.1 General**

11455 This Clause 5 lists those IC specifications which were included in previous editions of this
 11456 document. The previous IC versions differ from the current versions by at least one attribute
 11457 and/or method and by the version number.

11458 For new implementations in metering devices, only the current versions should be used.

11459 Communication drivers at the client side should also support previous versions.

11460 **5.2 Previous versions of interface classes for parameters and measurement data**

11461 **5.2.1 Profile generic (class_id = 7, version = 0)**

11462 **5.2.1.1 Overview**

11463 The version listed here was valid in Edition 1 and replaced with effect from Edition 2

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 (data)					x + 0x58
2. capture (data)					x + 0x60
3. get_buffer_by_range (data)					x + 0x68
4. get_buffer_by_index (data)					x + 0x70

11464

11465 **5.2.1.2 Attribute description**

11466 **5.2.1.2.1 logical_name**

11467 Identifies the “Profile generic” object instance. For examples, see 6.2.19, 6.2.21, 6.2.46, etc.

11468 **5.2.1.2.2 buffer**

11469 The buffer attribute contains a sequence of entries. Each entry contains values of the captured
 11470 objects (as returned by calling read (*current_value*)). The sequence is ordered according to the
 11471 specified sort method. The buffer gets filled by subsequent calls to *capture* ().

11472 array entry

11473 entry ::= structure

11474

Instance Specific

11475

Default. The buffer is empty at installation.

11476

Remark: Reading the entire buffer delivers only those entries which are “in use”

11477

11478 **5.2.1.2.3 capture_objects**

11479 Specifies the list of capture objects (registers, clocks and profiles) that are assigned to this
11480 profile object. Upon a call of the *capture ()* service, the specified attributes of these objects are
11481 copied into the buffer of the profile.

11482 array ObjectDefinition

11483

11484 ObjectDefinition::= structure

11485 {

11486 logical_name: octet-string,

11487 class_id: long-unsigned,

11488 attribute_index: unsigned

11489 }

11490 where attribute_index is a pointer to the attribute within the object. attribute_index = 1 refers to
11491 the first attribute (i.e. *logical_name*), attribute_index = 2 to the 2nd, etc.)

11492 **5.2.1.2.4 capture_period**

11493 >= 1: Automatic capturing assumed. Specifies the capturing period in seconds

11494 0: no automatic capturing, capturing is trigger externally or capture events occur
11495 asynchronously

11496 **5.2.1.2.5 sort_method**

11497 If the profile is unsorted, it works as a „first in first out“ buffer (it is hence sorted by capturing,
11498 and not necessarily by the time maintained in the clock object). If the buffer is full, the next call
11499 to *capture ()* will push out the first (oldest) entry of the buffer to make space for the new entry.

11500 If the profile is sorted, a call to *capture ()* will store the new entry at the appropriate position in
11501 the buffer, moving all following entries and probably losing the least interesting entry. If the new
11502 entry would enter the buffer after the last entry and if the buffer is already full, the new entry
11503 will not be retained at all.

11504 enum:

11505 1: fifo,

11506 2: lifo (last in first out)

11507 3: largest,

11508 4: smallest,

11509 5: nearest_to_zero,

11510 6: farest_from_zero

11511 Default fifo

11512 **5.2.1.2.6** sort_object

11513 If the profile is sorted, this attribute specifies the register or clock that the ordering is based
11514 upon.

11515 ObjectDefinition see 5.2.1.2.3

11516 Default no object to sort by (only possible with *sort_method* = fifo or lifo)

11517 5.2.1.2.7 entries_in_use

Counts the number of entries stored in the buffer. After a call of `reset()` the buffer does not contain any entries, and this value is zero. Upon each subsequent call of `capture()`, this value will be incremented up to the maximum number of entries that will get stored (see 5.2.1.2.8 `profile_entries`).

11522 double-long-unsigned 0...profile_entries

11523 Default 0

11524 5.2.1.2.8 profile_entries

11525 Specifies, how many entries should be retained in the buffer.

11526 double-long-unsigned 1... (limited by physical size)

11527 Default 1

11528 5.2.1.3 Method description

11529 5.2.1.3.1 reset (data)

11530 Clears the buffer. The buffer has no valid entries afterwards; *entries_in_use* is zero after this
11531 call. This call does not trigger any additional operations of the capture objects, specifically, it
11532 does not reset any captured buffers or registers.

11533 data = integer (0)

11534 5.2.1.3.2 capture (data)

11535 Copies the values of the objects to capture into the buffer by calling read (<object_attribute>)
11536 of each capture object. Depending on the *sort_method* and the actual state of the buffer this
11537 produces a new entry or a replacement for the less significant entry. As long as not all entries
11538 are already used, the *entries_in_use* attribute will be incremented.

11539 This call does not trigger any additional operations within the capture objects such as *capture*
 11540 () or *reset* ().

11541 Note that only some attributes of the captured objects might be stored, not the complete object.

11542 data = integer (0)

11543 **5.2.1.3.3 write (...)**

11544 Any write access to one of the attributes describing the static structure of the buffer will
 11545 automatically call a *reset* () and this call will propagate to all other profiles capturing this profile.

11546 If writing to *profile_entries* is attempted with a value too large for the buffer, it will be set to the
 11547 maximum possible value (restricted by physical size, typically laid down in the firmware).

11548 **5.2.1.3.4 get_buffer_by_range (data)**

11549 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 5.2.1.2.7 <i>entries_in_use</i> above.
array (of instance_specific_value)	out	Sorted sequence of entries containing the requested values of the capture objects.
data ::= structure {restricting_object; from_value; to_value; selected_values}		
In the response “data” is an array of instance_specific_value.		

11550

11551

11552 **5.2.1.3.5 get_buffer_by_index (data)**

11553 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 5.2.1.2.7 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*.

11554

11555

11556 **5.2.2 Compact data (class_id = 62, version = 0)**11557 **5.2.2.1 Overview**

11558 Instances of the “Compact data” IC allow capturing the values of COSEM object attributes as
 11559 determined by the *capture_objects* attribute. Capturing can take place:

- 11560 • on an external trigger (explicit capturing); or
- 11561 • upon reading the *compact_buffer* attribute (implicit capturing)

11562 as determined by the *capture_method* attribute.

11563 The values are stored in the *compact_buffer* attribute as an octet-string.

11564 The set of data types is identified by the *template_id* attribute. The data type of each attribute
 11565 captured is held by the *template_description* attribute.

11566 The client can reconstruct the data in the uncompacted form – i.e. including the COSEM
 11567 attribute descriptor, the data type and the data values – using the *capture_objects*, *template_id*
 11568 and *template_description* attributes.

Compact data	0...n	class_id = 62, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. compact_buffer (dyn.)	octet-string				x + 0x08
3. capture_objects (static)	array				x + 0x10
4. template_id (static)	unsigned				x + 0x18
5. template_description (dyn.)	octet-string				x + 0x20
6. capture_method (static)	enum				x + 0x28
Specific methods	m/o				
1. reset (data)	o				
2. capture (data)	o				

11569

11570 **5.2.2.2 Attribute description**11571 **5.2.2.2.1 logical_name**

11572 Identifies the “Compact data” object instance. See 6.2.41.

11573 **5.2.2.2.2 compact_buffer**11574 Contains the values of the attributes captured as an *octet-string*.11575 When the data captured is of type *octet-string*, *bit-string*, *visible-string*, *utf8-string* or *array* the
11576 length is also included here.11577 **5.2.2.2.3 capture_objects**11578 Specifies the list of COSEM object attributes that are assigned to the “Compact data” object
11579 instance.11580 The *template_id* attribute shall be the first element in the *capture_objects* array.11581 Upon an explicit or implicit invocation of the *capture (data)* method the values of the selected
11582 attributes are captured into the *compact_buffer*.

```

11583           array      capture_object_definition
11584           capture_object_definition ::= structure
11585           {
11586               class_id:      long-unsigned,
11587               logical_name:    octet-string,
11588               attribute_index: integer,
11589               data_index:      long-unsigned
11590           }

```

11591 Where:

- 11592 – attribute_index is a pointer to the attribute within the object, identified by class_id and
11593 logical_name: attribute_index 1 refers to the 1st attribute (i.e. the *logical_name*),
11594 attribute_index 2 to the 2nd attribute etc.; attribute_index 0 refers to all public attributes;
- 11595 – data_index is a pointer selecting one or several specific elements of an attribute with a
11596 complex data type (structure or array).
 - 11597 • if the data type of the attribute is simple, then data_index has no meaning;
 - 11598 • if the data type of the attribute is a structure or an array, then data_index points to one
11599 or several specific elements in the structure or array;
 - 11600 • when the attribute is the *buffer* of a “Profile generic” object, the data_index carries
11601 selective access parameters.

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

- 11602 • 0x0000 = identifies the whole attribute;
- 11603 • 0x0001 to 0x0FFF = identifies one element in the complex attribute. The first element
11604 in the complex attribute is identified by data_index 1;
- 11605 • 0x1000 to 0xFFFF = selective access to the array holding the buffer of a “Profile
11606 generic” object. The data-index selects entries within a number of last (recent) time
11607 periods, or a number of last (recent) entries, as well as the columns in the array.

11609 The encoding is specified in [Table 16](#).

11610 **5.2.2.2.4 template_id**

11611 Contains the identifier of the template.

11612 It shall uniquely identify the instance of the “Compact data” IC and the *template_description*.

11613 **5.2.2.2.5 template_description**

11614 Provides the data type of each attribute captured. It is an *octet-string* generated automatically
11615 by the server upon the programming of the *capture_objects* and it has the following structure:

- 11616 – the first octet is 0x02 (the tag of a structure);
- 11617 – this is followed by the number of elements in the structure – the same as the number of
11618 elements in the *capture_objects* array – encoded as a variable length integer;
- 11619 – this is followed by the data type of each attribute, in the same order as in the *capture_object*
11620 array:
 - 11621 • in the case of attributes with simple data type, the data type is represented by a single
11622 octet, carrying the tag of the data type. In the case of bit-string [4], octet-string [9],
11623 visible-string [10], utf8-string [12] the length of the string is part of the data held in the
11624 *compact_buffer*;
 - 11625 • in the case of an array [1], the data type is represented by a single octet 0x01. This is
11626 followed by the type description of the elements in the array. The number of elements
11627 in the array is part of the data held in the *compact_buffer*;
 - 11628 • in the case of a structure [2], the data type is represented by a single octet 0x02,
11629 followed by the number of elements inside the structure followed by the tag of each
11630 element of the structure.

11631 **5.2.2.2.6 capture_method**

11632 Defines the way the *compact_buffer* is updated.

11633 enum:

11634 (0) Capture upon invoking the *capture (data)* method. This may
11635 occur remotely or locally (explicit capturing),

11636 (1) Capture upon reading the *compact_buffer* attribute (implicit
11637 capturing).

11638 **5.2.2.3 Method description**

11639 **5.2.2.3.1 reset (data)**

11640 Clears the *compact_buffer*. After invoking this method the *compact_buffer* holds an octet-string
11641 of 0 length, until a new capture takes place.

11642 This call does not trigger any additional operations of the capture objects. Specifically, it does
11643 not reset any captured attributes.

11644 data::= integer(0)

11645 **5.2.2.3.2 capture (data)**

11646 Copies the values of the attributes into the *compact_buffer* by reading each capture object.

11647 This call does not trigger any additional operations within the capture objects such as capture
11648 () or reset ().

11649 data::= integer(0)

11650

11651 **5.2.2.4 Behaviour of the object after modification of certain attributes:**

11652 Any modification of the *capture_objects* resets the *compact_buffer* and automatically updates
11653 the *template_description*.

11654 **5.2.2.5 Restrictions**

11655 When defining the *capture_object* attribute, circular references shall be avoided.

11656

11657

11658 **5.3 Previous versions of interface classes for access control and management**

11659 **5.3.1 Association SN (class_id = 12, version = 0)**

11660 **5.3.1.1 Overview**

11661 The version listed here was valid in Edition 1 and it is replaced with effect from Edition 2.

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 (data)	o				
2. getobj_by_logicalname (data)	o				
3. read_by_logicalname (data)	o				
4. get_attributes&services (data)	o				
5. change_LLS_secret (data)	o				
6. change_HLS_secret (data)	o				
7. get_HLS_challenge (data)	o				
8. reply_to_HLS_challenge (data)	o				

11662

11663 **5.3.1.2 Attribute description**11664 **5.3.1.2.1 logical_name**

11665 Identifies the type of the client-server association. See 6.2.33.

11666 **5.3.1.2.2 object_list**11667 Contains the list of all objects with their short_name (DLMS ObjectName of the first attribute),
11668 class_id, version^b and logical_name.

11669 Objlist_type::= array objlist_element

11670 objlist_element::= structure

11671 {

11672 short_name: long,

11673 class_id: long-unsigned,

11674 version = unsigned,

11675 logical_name: octet-string

11676 }

11677

11678 **5.3.1.3 Method description**11679 **5.3.1.3.1 getlist_by_classid (data)**

11680 Delivers the subset of the object_list for a specific class_id.

11681 data::= class_id: long-unsigned

11682 For the response: data::= objlist_type

11683 **5.3.1.3.2 getobj_by_logicalname (data)**

11684 Delivers the entry of the object_list for a specific logical_name and class_id.

11685 data::= structure

11686 {

11687 class_id: long-unsigned,

11688 logical_name: octet-string

11689 }

11690 For the response: data::= objlist_element

11691 **5.3.1.3.3 read_by_logicalname (data)**

11692 Reads attributes for specific objects. The objects are specified by their class_id and their
11693 logical_name.

11694 data::= arrayAttributeldentification

11695 Attributeldentification::= structure

11696 {

11697 class_id: long-unsigned,

11698 logical_name: octet-string,

11699 attribute_index: unsigned

11700 }

11701 where:

11702 – attribute_index is a pointer (i.e. offset) to the attribute within the object.

11703 – attribute_index = 0 delivers all attributes^c, attribute_index = 1 delivers the first attribute (i.e.
11704 logical_name), etc.

11705 For the response: data is according to the type of the attribute.

11706 **5.3.1.3.4 get_attributes&services (data)**

11707 Delivers information about the access rights to the attributes and services within the actual
11708 association. The objects are specified by their class_id and their logical_name.

11709 array ObjectIdentification

11710

```

11711          ObjectIdentification ::= structure
11712          {
11713              class_id:      long-unsigned,
11714              logical_name: octet-string
11715          }
11716          For the response
11717          data ::= array      AccessDescription
11718          AccessDescription ::= structure
11719          {
11720              read_attributes: bit-string,
11721              write_attributes: bit-string,
11722              services:        bit-string
11723          }

```

11724 The position in the bit-string identifies the attribute/service (first position ↔ first attribute, first
 11725 position ↔ first service) and the value of the bit specifies whether the attribute/service is
 11726 available (bit set) or not available (bit clear).

11727 **5.3.1.3.5 change_LLS_secret (data)**

11728 Changes the LLS secret (for example password).

```
11729          data ::= octet-string      new LLS secret
```

11730 **5.3.1.3.6 change_HLS_secret (data)**

11731 Changes the HLS secret (for example encryption key).

```
11732          data ::= octet-stringd new HLS secret
```

11733 **5.3.1.3.7 get_HLS_challenge (data)**

11734 Asks the server for the client “challenge” (for example random number).

```
11735          data ::= octet-string      client challenge
```

11736 **5.3.1.3.8 reply_to_HLS_challenge (data)**

11737 Delivers the “secretly” processed “challenge” back to the server.

```
11738          data ::= octet-string      client's response to the challenge
```

11739 If the authentication is accepted, then the response is successful [0]. If the authentication is not
 11740 accepted, then the response is set to data-access-error [1].

11741 NOTES

11742 a Per client server association.

11743 b Within an client-server association, there are never two objects with the same class_id and logical_name
 11744 differing only in the versions.

11745 c If at least one attribute has no read access right under the current association, then a read_by_logicalname
 11746 () to attribute index = 0 reveals the error message “scope-of-access-violated” (comp. IEC 61334-4-41:1996, Clause
 11747 A.12 (p. 221)).

11748 d The structure of the “new secret” depends on the security mechanism implemented. The “new secret” may
 11749 contain additional checkbits and it may be encrypted.

11750

11751

11752 **5.3.2 Association SN (class_id = 12, version = 1)**

11753 **5.3.2.1 Overview**

11754 This IC allows modelling of AAs between a server and a client, using the application context
 11755 *short name referencing*. A COSEM logical device may have one instance of this IC for each
 11756 association the device is able to support.

11757 The **short_name** of the current “Association SN” object itself is fixed within the COSEM context.
 11758 It is given in 4.1.3 as 0xFA00.

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				
4. get_attributes&methods (data)	o				
5. change_LLS_secret (data)	o				
6. change_HLS_secret (data)	o				
7. reserved from previous versions					
8. reply_to_HLS_authentication (data)	o				

11759

11760 **5.3.2.2 Attribute description**

11761 **5.3.2.2.1 logical_name**

11762 See 5.3.1.2.1.

11763 **5.3.2.2.2 object_list**

11764 Contains the list of all objects with their base_names (short_name), class_id, version and
 11765 logical_name. The base_name is the DLMS objectName of the first attribute (logical_name).

```
11766     objlist_type::=array objlist_element
11767
11768         objlist_element ::=  structure
11769             {
11770                 base_name: long,
11771                 class_id:      long-unsigned,
11772                 version =      unsigned,
11773                 logical_name: octet-string
11774             }
11775     selective access (see 4.1.4) to the attribute object_list may be available (optional). The selective
11776     access parameters are as defined in Table 44.
```

11777

11778

Table 44 – 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

11779

11780 **5.3.2.3 Method description**11781 **5.3.2.3.1 read_by_logicalname (data)**

11782 See 5.3.1.3.2.

11783 **5.3.2.3.2 get_attributes& methods (data)**

11784 Delivers information about the access rights to the attributes and methods within the actual
 11785 association. The objects are specified by their *class_id* and their *logical_name*. With this method,
 11786 the parameterized access feature can also be used.

11787

11788 data::= arrayobject_identification

11789

11790 object_identification::= structure

11791 {

11792 class_id: long-unsigned,

11793 logical_name: octet-string

11794 }

11795 For the response

11796 data::= arrayaccess_description

11797

11798 access_description::= structure

11799 {

11800 read_attributes: bit-string,

11801 write_attributes: bit-string,

11802 methods: bit-string

11803 }

11804 The position in the bit-string identifies the attribute/method (first position ↔ first attribute, first
 11805 position ↔ first method) and the value of the bit specifies whether the attribute/method is
 11806 available (bit set) or not available (bit clear).

11807 Depending on the implementation, some attributes or methods of some objects may not be needed.
 11808 In this case, such attributes or methods may not be accessible (neither read access,
 11809 nor write access to attributes, no access to methods).

11810 **5.3.2.3.3 change_LLS_secret (data)**

11811 See 5.3.1.3.5.

11812 **5.3.2.3.4 change_HLS_secret (data)**

11813 See 5.3.1.3.6.^b

11814 **5.3.2.3.5 reply_to_HLS_authentication (data)**

11815 The remote invocation of this method delivers the client's "secretly" processed "challenge StoC"
 11816 (f/StoC)) back to the server as the data service parameter of the invoked Write.request service.

11817 data::= octet-string client's response to the challenge

11818 data::= octet-string server's response to the challenge

11819 If the authentication is not accepted, then the result parameter in the response shall contain a
 11820 non-OK value, and no data shall be sent back.

11821 NOTES

11822 a If at least one attribute has no read access right under the current association, then a read_by_logicalname
 11823 () to attribute index 0 reveals the error message "scope of access violation" (see IEC 61334-4-41:1996, Clause A.12
 11824 (p. 221)).

11825 b The structure of the "new secret" depends on the security mechanism implemented. The "new secret" may
 11826 contain additional check bits and it may be encrypted.

11827

11828

11829 **5.3.3 Association SN (class_id = 12, version = 2)**

11830 **5.3.3.1 Overview**

11831 COSEM logical devices able to establish AAs within a COSEM context using SN referencing,
 11832 model the AAs using instances of the "Association SN" IC. A COSEM logical device may have
 11833 one instance of this IC for each AA the device is able to support.

11834 The **short_name** of the "Association SN" object itself is fixed within the COSEM context.
 11835 See 4.1.3.

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.	<i>reserved from previous versions</i>	o					
2.	<i>reserved from previous versions</i>	o					
3.	read_by_logicalname (data)	o					x + 0x30
4.	<i>reserved from previous versions</i>	o					
5.	change_secret (data)	o					x + 0x40
6.	<i>reserved from previous versions</i>	o					
7.	<i>reserved from previous versions</i>						
8.	reply_to_HLS_authentication (data)	o					x + 0x58

11836

11837 **5.3.3.2 Attribute description**11838 **5.3.3.2.1 logical_name**

11839 See 5.3.1.2.1.

11840 **5.3.3.2.2 object_list**

11841 Contains the list of all objects with their *base_name* (*short_name*), *class_id*, *version* and
 11842 *logical_name*. The *base_name* is the DLMS *objectName* of the first attribute (*logical_name*).

```

 11843           objlist_type::= array          objlist_element
 11844           objlist_element::= structure
 11845           {
 11846             base_name:      long,
 11847             class_id:       long-unsigned,
 11848             version:        unsigned,
 11849             logical_name:   octet-string
 11850           }
```

11851 selective access (see 4.1.4) to the attribute *object_list* may be available. The access selector
 11852 values and their parameters are as defined in Table 45.

11853 **5.3.3.2.3 access_rights_list**

11854 Contains the access rights to attributes and methods.

11855 The link between the *object_list* and the *access_rights_list* is the *base_name*, present in both
 11856 the *objlist_element* structure and the *access_right_element* structure. Therefore, the
 11857 *base_names* on the two lists shall be the same. The number – and preferably, the order – of
 11858 the elements in the array of *objlist_element* and the array of *access_right_element* shall also
 11859 be the same.

```
11860           access_rights_type ::= array          access_rights_element
11861           access_rights_element ::= structure
11862           {
11863               base_name:           long,
11864               attribute_access:    attribute_access_descriptor,
11865               method_access:       method_access_descriptor
11866           }
11867           attribute_access_descriptor ::= array attribute_access_item
11868           attribute_access_item ::= structure
11869           {
11870               attribute_id:        integer,
11871               access_mode:         enum:
11872                   (0)   no_access,
11873                   (1)   read_only,
11874                   (2)   write_only,
11875                   (3)   read_and_write,
11876                   (4)   authenticated_read_only,
11877                   (5)   authenticated_write_only,
11878                   (6)   authenticated_read_and_write
11879
11880           access_selectors:      CHOICE
11881           {
11882               null-data [0],
11883               array integer [1]
11884           }
11885       }
11886       method_access_descriptor ::= array          method_access_item
```

11887

11888 method_access_item ::= structure

11889 {

11890 method_id: integer,

11891 access_mode: enum:

11892 (0) no_access,

11893 (1) access,

11894 (2) authenticated_access

11895 }

11896 selective access (see 4.1.4) to the attribute *access_rights_list* may be available (optional). The
11897 access selector values and their parameters are as defined in Table 45.

11898 **5.3.3.2.4 security_setup_ reference**

11899 References the “Security setup” object by its *logical_name*. The referenced object manages
11900 security for a given “Association SN” object instance.

11901

11902

11903 **Table 45 – Parameters for selective access to the *object_list* and *access_rights_list*
11904 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 class_id. For the response: data::= objlist_type
2	structure { class_id: long-unsigned, logical_name: octet-string }	2	Delivers the entry of the <i>object_list</i> 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 <i>object_list</i> for a specific base_name. 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 base_name. For the response: data::= access_rights_element

11905 **5.3.3.3 Method description**

11906 **5.3.3.3.1 read_by_logicalname (data)**

11907 See 5.3.1.3.3.

5.3.3.3.2 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

5.3.3.3 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.

11926

11927

5.3.4 Association SN (Class_id = 12, version =3)**5.3.4.1 Overview**

NOTE 1 This new version 3 of the “Association SN” IC supports the client user identification process, see 4.4.2.

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.

11934 The **short_name** of the “Association SN” object itself is fixed within the COSEM context.
11935 See 4.1.3.

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

11936

11937 5.3.4.2 Attribute description

11938 5.3.4.2.1 logical_name

11939 See 5.3.1.2.1.

11940 5.3.4.2.2 object_list

11941 See 5.3.3.2.2.

11942 5.3.4.2.3 access_rights_list

11943 See 5.3.3.2.3.

11944 5.3.4.2.4 security_setup_reference

11945 See 5.3.3.2.4.

11946 5.3.4.2.5 user_list

11947 Contains the list of users allowed to use the AA managed by the given instance of the
11948 "Association SN" IC.

11949 array user_list_entry

11950

11951 user_list_entry ::= structure

```
11952          {  
11953              user_id:    unsigned,  
11954              user_name:   visible-string  
11955          }  
11956 Where:  
11957     – user_id is the identifier of the user (this value is carried by the calling-AE-invocation-id field  
11958     of the AARQ);  
11959     – user_name is the name of the user.  
11960 If the user_list attribute is empty – i.e. it is an array of 0 elements – any user can use the AA,  
11961 i.e. the calling-AE-invocation-id field of the AARQ is ignored.  
11962 If the user_list attribute is not empty then only the users in the list can establish the AA, i.e. the  
11963 calling-AE-invocation-id field of the AARQ shall be present and its value shall match one of  
11964 user_ids in the user_list or else, the AA is not established.
```

11965 **5.3.4.2.6 current_user**

11966 Holds the identifier of the current user.

```
11967         current_user::= user_list_entry (see 5.3.4.2.5)
```

11968 If the user_list is empty, then current_user shall be a structure {user_id: unsigned 0, user_name:
11969 visible string of 0 elements}

11970 **5.3.4.3 Method description**

11971 **5.3.4.3.1 read_by_logicalname (data)**

11972 See 5.3.1.3.3.

11973 **5.3.4.3.2 change_secret (data)**

11974 See 5.3.3.3.2.

11975 **5.3.4.3.3 reply_to_HLS_authentication (data)**

11976 See 5.3.3.3.3.

11977 **5.3.4.3.4 add_user (data)**

11978 Adds a user to the user_list.

```
11979         data::= user_list_entry (see 5.3.4.2.5)
```

11980 **5.3.4.3.5 remove_user (data)**

11981 Removes a user from the user_list.

```
11982         data::= user_list_entry (see 5.3.4.2.5)
```

11983

11984 **5.3.5 Association LN (class_id = 15, version = 0)**11985 **5.3.5.1 Overview**

11986 This IC allows modelling of AAs between a server and a client, using the application context
 11987 *logical name referencing*. Each AA is modelled by one Association LN object.

Association LN	0...MaxNbofAss.	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

11988

11989 **5.3.5.2 Attribute description**11990 **5.3.5.2.1 logical_name**

11991 Identifies the “Association LN” object instance. See 6.2.33.

11992 **5.3.5.2.2 object_list**

11993 Contains the list of visible COSEM objects with their class_id, version, logical name and the
 11994 access rights to their attributes and methods within the given application association.

```

11995          object_list_type ::= array object_list_element
11996          object_list_element ::= structure
11997          {
11998              class_id:           long-unsigned,
11999              version:            unsigned,
12000              logical_name:       octet-string,
12001              access_rights:      access_right
12002          }
  
```

```
12003
12004     access_right ::= structure
12005     {
12006         attribute_access:      attribute_access_descriptor,
12007         method_access:        method_access_descriptor
12008     }
12009
12010     attribute_access_descriptor ::= array attribute_access_item
12011     attribute_access_item ::= structure
12012     {
12013         attribute_id:        integer,
12014         access_mode:        enum:
12015             (0)    no_access,
12016             (1)    read_only,
12017             (2)    write_only,
12018             (3)    read_and_write
12019         access_selectors:   CHOICE
12020     {
12021         null-data          [0],
12022         array integer       [1]
12023     }
12024 }
12025
12026     method_access_descriptor ::= array method_access_item
12027     method_access_item ::= structure
12028     {
12029         method_id: integer,
```

12030 access_mode: boolean

12031 }

12032 Where:

12033 – the attribute_access_descriptor and the method_access_ descriptor always contain all
12034 implemented attributes or methods;

12035 – access_selectors contain a list of the supported selector values;

12036 selective access (see 4.1.4) to the attribute object_list may be available (optional). The
12037 selective access parameters are as defined in 5.3.5.2.9.

12038 **5.3.5.2.3 associated_partners_id**

12039 Contains the identifiers of the COSEM client and server (logical device) APs within the physical
12040 devices hosting these processes, which belong to the AA modelled by the “Association LN”
12041 object.

12042 associated_partners_type::= structure

12043 {

12044 client_SAP: integer,

12045 server_SAP: long-unsigned

12046 }

12047 The range for the client_SAP is 0...0x7F.

12048 The range for the server_SAP is 0x000...0x3FFF.

12049 **5.3.5.2.4 application_context_name**

12050 In the COSEM environment, it is intended that an application context pre-exists and is
12051 referenced by its name during the establishment of an AA. This attribute contains the name of
12052 the application context for that AA.

12053 context_name_type::= CHOICE

12054 {

12055 context_name_structure [2],

12056 octet-string [9]

12057 }

12058 The application context name is specified as OBJECT IDENTIFIER in IEC 62056-5-3:**2021**,
12059 **7.2.2.2**

12060

12061 When the context_name_type is encoded as a structure, it includes the arc labels of the
 12062 OBJECT IDENTIFIER.

```

12063     context_name_structure ::= structure
12064     {
12065         joint_iso_ctt_element:           unsigned,
12066         country_element:               unsigned,
12067         country_name_element:          long-unsigned,
12068         identified_organization_element: unsigned,
12069         DLMS_UA_element:               unsigned,
12070         application_context_element:   unsigned,
12071         context_id_element:            unsigned
12072     }
```

12073 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
 12074 01 (all values are hexadecimal).

12075 When the context_name_type is encoded as an octet-string, it holds the value of the OBJECT
 12076 IDENTIFIER. See IEC 62056-5-3:**2021**, Clause D.4.

12077 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
 12078 hexadecimal).

12079 **5.3.5.2.5 xDLMS_context_info**

12080 Contains all the necessary information on the xDLMS context for the given AA.

```

12081     xDLMS-context-type ::= structure
12082     {
12083         conformance:           bit-string,
12084         max_receive_pdu_size: long-unsigned,
12085         max_send_pdu_size:    long-unsigned,
12086         dlms_version_number:  unsigned,
12087         quality_of_service:   integer,
12088         cyphering_info:        octet-string
12089     }
```

12090 Where:

- 12091 – the conformance element contains the xDLMS conformance block supported by the server;
- 12092 – 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 62056-5-3:2021 Clause 8);
- 12096 – 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 62056-5-3:2021, Clause 8);
- 12100 – the dlms_version_number element contains the DLMS version number supported by the server;
- 12102 – the quality_of_service element is not used;
- 12103 – the cyphering_info, in an active association, contains the dedicated key parameter of the xDLMS initiateRequest APDU (see IEC 62056-5-3:2021, Clause 8).

12105

12106 **5.3.5.2.6 authentication_mechanism_name**

12107 Contains the name of the authentication mechanism for the AA.

12108 mechanism_name_type ::= CHOICE
12109 {
12110 mechanism_name_structure [2],
12111 octet-string [9]
12112 }

12113 The authentication mechanism name is specified as an OBJECT IDENTIFIER in IEC 62056-5-3:2021, 7.2.2.3.

12115 When the mechanism_name_type is encoded as a structure, it includes the arc labels of the
12116 OBJECT IDENTIFIER.

12117
12118 mechanism_name_structure ::= structure
12119 {
12120 joint_iso_ctt_element: unsigned,
12121 country_element: unsigned,
12122 country_name_element: long-unsigned,
12123 identified_organization_element: unsigned,
12124 DLMS_UA_element: unsigned,

12125 authentication_mechanism_name_element: unsigned,
12126 mechanism_id_element: unsigned
12127 }
12128
12129 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
12130 11 01 (all values are hexadecimal):
12131 When the mechanism_name_type is encoded as an octet-string, it holds the value of the
12132 OBJECT IDENTIFIER. See IEC 62056-5-3:**2021**, Clause D.4.
12133 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
12134 hexadecimal).
12135 No mechanism_name is required when no authentication is used.
12136 **5.3.5.2.7 LLS_secret**
12137 Contains the authentication value for the LLS authentication process.
12138 **5.3.5.2.8 association_status**
12139 Indicates the current status of the association, which is modelled by the object.
12140 enum: (0) non-associated,
12141 (1) association-pending,
12142 (2) associated
12143
12144
12145 **5.3.5.2.9 Parameters for selective access to the *object_list* attribute**
12146 • if no selective access is requested, (no Access_Selection_Parameters parameter is
12147 present in the GET.request (.indication) service primitive for the object_list attribute) the
12148 corresponding .response (.confirmation) service shall contain all object_list_element of the
12149 *object_list* attribute;
12150 • when selective access is requested to the *object_list* attribute (the
12151 Access_Selection_Parameters parameter is present), the response shall contain a
12152 ‘filtered’ list of object_list_element, as shown in Table 46:
12153

12154

Table 46 – Parameters for selective access to the *object_list* attribute

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. In this case, only those object_list_element 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: See above.

12155

5.3.5.3 Method description**5.3.5.3.1 reply_to_HLS_authentication (data)**

12158 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.

12161 data ::= octet-string client's response to the challenge

12162 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.

12165 data ::= octet-string server's response to the challenge

12166 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.

5.3.5.3.2 change_HLS_secret (data)

12169 Changes the HLS secret (for example encryption key).

12170 data ::= octet-string^a new HLS secret

12171 NOTE 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.

5.3.5.3.3 add_object (data)

12174 Adds the referenced object to the object_list.

12175 data ::= object_list_element (see 5.3.5.2.2)

12176 **5.3.5.3.4 remove_object (data)**

12177 Removes the referenced object from the object_list.

12178 data::= object_list_element (see 5.3.5.2.2)

12179

12180 **5.3.6 Association LN (class_id = 15, version = 1)**12181 **5.3.6.1 Overview**12182 COSEM logical devices able to establish AAs within a COSEM context using LN referencing,
12183 model the AAs through instances of the “Association LN” IC. A COSEM logical device has one
12184 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

12185

12186 **5.3.6.2 Attribute description**12187 **5.3.6.2.1 logical_name**

12188 See 5.3.5.2.1.

12189 **5.3.6.2.2 object_list**12190 Contains the list of visible COSEM objects with their class_id, version, logical name and the
12191 access rights to their attributes and methods within the given application association.

12192 object_list_type::= array object_list_element

12193 object_list_element::= structure

12194 {

```
12195     class_id:          long-unsigned,  
12196     version:           unsigned,  
12197     logical_name:      octet-string,  
12198     access_rights:    access_right  
12199 }  
12200  
12201     access_right ::= structure  
12202 {  
12203     attribute_access: attribute_access_descriptor,  
12204     method_access:    method_access_descriptor  
12205 }  
12206     attribute_access_descriptor ::= array attribute_access_item  
12207     attribute_access_item ::= structure  
12208 {  
12209     attribute_id:      integer,  
12210     access_mode:      enum:  
12211         (0)    no_access,  
12212         (1)    read_only,  
12213         (2)    write_only,  
12214         (3)    read_and_write,  
12215         (4)    authenticated_read_only,  
12216         (5)    authenticated_write_only,  
12217         (6)    authenticated_read_and_write  
12218     access_selectors: CHOICE  
12219 {  
12220     null-data [0],  
12221     array integer [1]
```

```

12222         }
12223         }
12224         method_access_descriptor ::= array method_access_item
12225
12226         method_access_item ::= structure
12227         {
12228             method_id: integer,
12229             access_mode: enum:
12230                 (0)    no_access,
12231                 (1)    access,
12232                 (2)    authenticated_access
12233         }

```

12234 Where:

- 12235 – the attribute_access_descriptor and the method_access_descriptor elements always
12236 contain all implemented attributes or methods;
- 12237 – access_selectors contain a list of the supported selector values.

12238 selective access (see 4.1.4) to the attribute object_list may be available (optional). The
12239 selective access parameters are as defined in 5.3.5.2.9.

12240 **5.3.6.2.3 associated_partners_id**

12241 Contains the identifiers of the COSEM client and server (logical device) APs within the physical
12242 devices hosting these APs, which belong to the AA modelled by the “Association LN” object.

```

12243         associated_partners_type ::= structure
12244         {
12245             client_SAP:integer,
12246             server_SAP:    long-unsigned
12247         }

```

12248 The range for the client_SAP is 0...0x7F.

12249 The range for the server_SAP is 0x0000...0x3FFF.

12250 The SAPs shall be in the range allowed by the data type and the media.

12251 **5.3.6.2.4 application_context_name**

12252 See 5.3.5.2.4.

12253 **5.3.6.2.5 xDLMS_context_info**

12254 See 5.3.5.2.5.

12255 **5.3.6.2.6 authentication_mechanism_name**

12256 See 5.3.5.2.6.

12257 **5.3.6.2.7 secret**

12258 Contains the secret for the LLS or HLS authentication process.

12259 NOTE In the case of HLS with GMAC, the (HLS_)secret is held by the Security setup object referenced in attribute
12260 9.

12261 **5.3.6.2.8 association_status**

12262 See 5.3.5.2.8.

12263 **5.3.6.2.9 security_setup_reference**

12264 References the Security setup object by its logical name. The referenced object manages
12265 security for a given Association LN object instance.

12266

12267 A SET operation on an attribute of an association LN object becomes effective when this
12268 association object is used to establish a new association.

12269

12270 **5.3.6.3 Method description**12271 **5.3.6.3.1 reply_to_HLS_authentication (data)**

12272 The remote invocation of this method delivers to the server the result of the secret processing
12273 by the client of the server's challenge to the client, f(StoC), as the *data* service parameter of
12274 the ACTION.request primitive invoked.

12275 data ::= octet-string client's response to the challenge

12276 If the authentication is accepted, then the response (ACTION.confirm primitive) contains Result
12277 == OK and the result of the secret processing by the server of the client's challenge to the
12278 server, f(CtoS) in the *data* service parameter of the response service.

12279 data ::= octet-string server's response to the challenge

12280 If the authentication is not accepted, then the result parameter in the response shall contain a
12281 non-OK value, and no data shall be sent back.

12282 **5.3.6.3.2 change_HLS_secret (data)**

12283 Changes the HLS secret (for example encryption key).

12284 data ::= octet-string new HLS secret

12285 The structure of the “new secret” depends on the security mechanism implemented. The “new
12286 secret” may contain additional check bits and it may be encrypted.

12287 5.3.6.3.3 add_object (data)

12288 See 5.3.5.3.3.

12289 5.3.6.3.4 remove_object (data)

12290 See 5.3.5.3.4.

12291

12292 5.3.7 Association LN (class_id = 15, version = 2)

12293 **5.3.7.1** Overview

12294 NOTE 1 This version 2 of the "Association LN" IC supports the client user identification process, see 4.4.2.

12295 COSEM logical devices able to establish AAs within a COSEM context using LN referencing,
12296 model the AAs through instances of the “Association LN” IC. A COSEM logical device has one
12297 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_0mechanism_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				
2. change_HLS_secret (data)	o				
3. add_object (data)	o				
4. remove_object (data)	o				
5. add_user (data)	o				
6. remove_user (data)	o				

12298

12299 **5.3.7.2 Attribute description**12300 **5.3.7.2.1 logical_name**

12301 Identifies the “Association LN” object instance. See 6.2.33.

12302 **5.3.7.2.2 object_list**

12303 Contains the list of visible COSEM objects with their class_id, version, *logical_name* and the
12304 access rights to their attributes and methods within the given AA.

12305 object_list_type ::= array object_list_element

12306 object_list_element ::= structure

12307 {

12308 class_id: long-unsigned,

12309 version: unsigned,

12310 logical_name: octet-string,

12311 access_rights: access_right

12312 }

12313 access_right ::= structure

12314 {

12315 attribute_access: attribute_access_descriptor,

12316 method_access: method_access_descriptor

12317 }

12318

12319 attribute_access_descriptor ::= array attribute_access_item

12320

12321 attribute_access_item ::= structure

12322 {

12323 attribute_id: integer,

12324 access_mode: enum:

12325 (0) no_access,

12326 (1) read_only,

```
12327          (2)  write_only,  
12328          (3)  read_and_write,  
12329          (4)  authenticated_read_only,  
12330          (5)  authenticated_write_only,  
12331          (6)  authenticated_read_and_write  
12332  
12333      access_selectors: CHOICE  
12334      {  
12335          null-data [0],  
12336          array integer [1]  
12337      }  
12338  }  
12339  
12340  method_access_descriptor ::= array method_access_item  
12341  
12342  method_access_item ::= structure  
12343  {  
12344      method_id: integer,  
12345      access_mode: enum:  
12346          (0)  no_access,  
12347          (1)  access,  
12348          (2)  authenticated_access  
12349  }  
12350  Where:  
12351  – the attribute_access_descriptor and the method_access_descriptor elements always  
12352  contain all implemented attributes or methods;  
12353  – access_selectors contain a list of the supported selector values.
```

12354 selective access (see 4.1.4) to the attribute object_list may be available (optional). The
12355 selective access parameters are as defined in 5.3.5.2.9.

12356 **5.3.7.2.3 associated_partners_id**

12357 See 5.3.6.2.3.

12358 **5.3.7.2.4 application_context_name**

12359 See 5.3.6.2.4.

12360 **5.3.7.2.5 xDLMS_context_info**

12361 See 5.3.6.2.5.

12362 **5.3.7.2.6 authentication_mechanism_name**

12363 See 5.3.6.2.6.

12364 **5.3.7.2.7 secret**

12365 See 5.3.6.2.7.

12366 NOTE In the case of HLS with GMAC, the (HLS_)secret is held by the “Security setup” object referenced in attribute
12367 9, security_setup_reference.

12368

12369 **5.3.7.2.8 association_status**

12370 See 5.3.6.2.8.

12371 **5.3.7.2.9 security_setup_reference**

12372 See 5.3.6.2.9.

12373 **5.3.7.2.10 user_list**

12374 Contains the list of users allowed to use the AA managed by the given instance of the
12375 “Association LN” IC.

12376 array user_list_entry

12377 user_list_entry::= structure

12378 {

12379 user_id: unsigned,

12380 user_name: visible-string

12381 }

12382 Where:

- 12383 – user_id is the identifier of the user (this value is carried by the calling-AE-invocation-id field
12384 of the AARQ);

12385 – user_name is the name of the user.

12386 If the *user_list* attribute is empty – i.e. it is an array of 0 elements – any user can use the AA,
12387 i.e. the calling-AE-invocation-id field of the AARQ is ignored.

12388 If the *user_list* attribute is not empty then only the users in the list can establish the AA, i.e. the
12389 calling-AE-invocation-id field of the AARQ shall be present and its value shall match one of
12390 user_ids in the *user_list* or else the AA is not established.

12391 **5.3.7.2.11 current_user**

12392 Holds the identifier of the current user.

12393 current_user::= user_list_entry (see 5.3.7.2.10)

12394 If the *user_list* is empty, then *current_user* shall be a structure {user_id: unsigned 0, user_name:
12395 visible string of 0 elements}

12396 **5.3.7.3 Method description**

12397 **5.3.7.3.1 reply_to_HLS_authentication (data)**

12398 See 5.3.6.3.1.

12399 **5.3.7.3.2 change_HLS_secret (data)**

12400 See 5.3.6.3.2.

12401 **5.3.7.3.3 add_object (data)**

12402 See 5.3.6.3.3.

12403 **5.3.7.3.4 remove_object (data)**

12404 See 5.3.6.3.4.

12405 **5.3.7.3.5 add_user (data)**

12406 Adds a user to the *user_list*.

12407 data::= user_list_entry (see 5.3.7.2.10)

12408 **5.3.7.3.6 remove_user (data)**

12409 Removes a user from the *user_list*.

12410 data::= user_list_entry (see 5.3.7.2.10)

12411

12412 **5.3.8 Security setup (class_id = 64, version = 0)**

12413 **5.3.8.1 Overview**

12414 Instances of this IC contain the necessary information on the security policy applicable and the
12415 security suite in use within a particular AA, between two systems identified by their client system
12416 title and server system title respectively. They also contain methods to increase the level of
12417 security and to transfer the global keys. See IEC 62056-5-3:2021, 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 (data)	o				
2. global_key_transfer (data)	o				

12418

12419 5.3.8.2 Attribute description

12420 5.3.8.2.1 logical_name

12421 Identifies the “Security setup” object instance. See 6.2.36.

12422 5.3.8.2.2 security_policy

12423 Enforces authentication and/or encryption algorithm provided with security_suite.

12424 enum: (0) nothing,

12425 (1) all messages to be authenticated,

12426 (2) all messages to be encrypted,

12427 (3) all messages to be authenticated and encrypted

12428 (4) ... (15) reserved

12429 5.3.8.2.3 security_suite

12430 Specifies authentication, encryption and key transport algorithm.

12431 enum: (0) AES-GCM-128 for authenticated encryption and AES-128 for key wrapping

12432 (1) ... (15) reserved

12433 5.3.8.2.4 client_system_title

12434 Carries the (current) client system title:

12435 – in the S-FSK PLC environment, the active initiator sends its system title using the CIASE protocol;

12437 NOTE 1 It is also held by the *active_initiator* attribute of the S-FSK Active initiator object; see 4.10.4;

12438 – during confirmed or unconfirmed AA establishment, it is carried by the calling-AP-title field of the AARQ APDU;

12440 – 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.

- 12443 – in a pre-established AA, it can be written by the client using an unsecured SET / Write
12444 service.

12445 **5.3.8.2.5 server_system_title**

12446 Carries the server system title.

- 12447 – in the S-FSK PLC environment, the server sends its system title during the discover process,
12448 using the CIASE protocol;
- 12449 – during confirmed AA establishment, it is carried by the responding-AP-title field of the AARE
12450 APDU.

12451 This attribute shall be read only.

12452 **5.3.8.3 Method description**

12453 **5.3.8.3.1 security_activate (data)**

12454 Activates and strengthens the security policy:

12455 enum: (0) nothing,

12456 (1) all messages to be authenticated,

12457 (2) all messages to be encrypted,

12458 (3) all messages to be authenticated and encrypted

12459 The new security policy applies as soon as the method invocation has been confirmed with
12460 success.

12461 NOTE 2 The security policy can only be strengthened.

12462 **5.3.8.3.2 global_key_transfer (data)**

12463 Updates one or more global keys. The *data* parameter includes wrapped key data. Key data
12464 include the key identifiers and the keys themselves.

12465 array key_data

12466

12467 key_data::= structure

12468 {

12469 key_id: enum: (0) global unicast encryption key,

12470 (1) global broadcast encryption key,

12471 (2) authentication key

12472 key_wrapped: octet-string

12473 }

12474 The key wrapping algorithm is as specified by the security suite. The KEK is the master key.

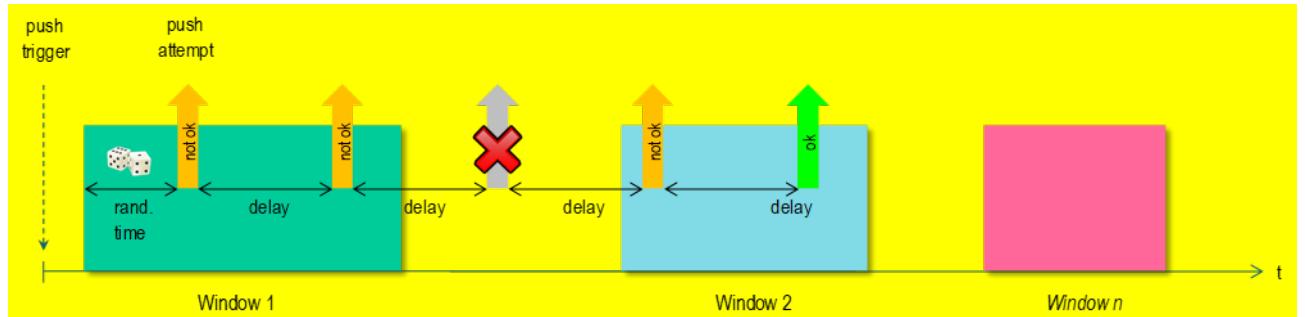
12475 The new key(s) are valid as soon as the method invocation has been confirmed with success.

12476 5.3.9 Push Setup (class_id = 40, version = 0)

12477 5.3.9.1 Overview

12478 The "Push setup" IC contains a list of references to COSEM object attributes to be pushed. It
12479 also contains the push destination and method as well as the communication time windows and
12480 the handling of retries.

12481 The push takes place upon invoking the *push* method, triggered by a Push "Single action
12482 schedule" object, by an alarm "Register monitor" object, by a dedicated internal event or
12483 externally. After the push operation has been triggered, it is executed according to the settings
12484 made in the given "Push setup" object. Depending on the communication window settings, the
12485 push is executed immediately or as soon as a communication window becomes active, after a
12486 random delay. If the push was not successful, retries are made. Push windows, delays and
12487 retries are shown in Figure 30.



12488

12489 **Figure 30 – Push windows and delays**

12490

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				

12491

12492 **5.3.9.2 Attribute description**

12493 **5.3.9.2.1 logical_name**

12494 Identifies the “Push setup” object instance. See 6.2.24.

12495 **5.3.9.2.2 push_object_list**

12496 Defines the list of attributes to be pushed.

12497 Upon invocation of the *push* (data) method the items are sent to the destination defined in the
12498 *send_destination_and_method* attribute.

```
12499           array      object_definition
12500
12501           object_definition ::=   structure
12502
12503           {
12504           class_id:          long-unsigned,
12505           logical_name:       octet-string,
12506           attribute_index:    integer,
12507           data_index:         long-unsigned
12508           }
```

12509 Where:

- 12510 – attribute_index is a pointer to the attribute within the object, identified by class_id and
12511 logical_name: attribute_index 1 refers to the 1st attribute (i.e. the logical_name),
12512 attribute_index 2 to the 2nd attribute etc.; attribute_index 0 refers to all public attributes;
- 12513 – data_index is a pointer selecting one or several specific elements of an attribute with a
12514 complex data type (structure or array).
- 12515 –

data_index	MS-Byte		LS-Byte
	Upper nibble	Lower nibble	

12516

12517 If the data_type of the attribute is simple, then data_index has no meaning.

12518 If the attribute is a structure or an array, then data_index points to an element in the structure
12519 or array. The first element in the complex attribute is identified by data_index 1.

12520 When the attribute is the *buffer* of a “Profile generic” object, the data_index carries selective
12521 access parameters.

- 12522 – 0x0000 = identifies the whole attribute;
- 12523 – 0x0001 to 0xFFFF = identifies one element in the complex attribute;
- 12524 – 0x1000 to 0xFFFF = selective access to the array holding the buffer of a “Profile generic”
12525 object. The data-index selects entries within a number of last (recent) time periods, or a
12526 number of last (recent) entries, as well as the columns in the array.

12527 The encoding is specified in Table 47.

12528 NOTE 1 If the push_object_list array is empty, the push operation is disabled.

12529 NOTE 2 The push_object_list attribute itself can be pushed as well to clearly identify the data pushed.

12530 Similarly to the case of the “Profile generic” IC, all attributes included in the push_object_list
 12531 attribute are pushed regardless of the access rights to them. Therefore, writing of the
 12532 push_object_list attribute should be restricted to clients with appropriate access rights.

12533 **5.3.9.2.3 send_destination_and_method**

12534 Contains the destination address (e.g. phone number, email address, IP address) where the
 12535 data specified by the push_object_list has to be sent, as well as the sending method.

```
12536     send_destination_and_method ::= structure
12537
12538     {
12539         transport_service:    transport_service_type,
12540         destination:          octet-string,
12541         message:              message_type
12542     }
```

12543 Where:

12544 – the transport_service element defines the type of service used to push the data:

```
12545     transport_service_type ::= enum:
12546             (0)      TCP,
12547             (1)      UDP,
12548             (2)      reserved for FTP,
12549             (3)      reserved for SMTP,
12550             (4)      SMS,
12551             (5)      HDLC,
12552             (6)      reserved for M-Bus,
12553             (7)      reserved for ZigBee®,
12554             (200...255) manufacturer specific
12555
```

12556 – the destination element contains the target address where the data has to be sent. The
 12557 elements of the target address depend on the transport service used.

12558 Each “Push setup” object instance specifies a single destination. If it is required to push
 12559 data to several destinations, several “Push setup” objects have to be instantiated,

12560 – the message_type element identifies the encoding of the xDLMS APDU used.

```
12561     message_type ::= enum:
12562             (2)      A-XDR encoded xDLMS APDU,
12563             (3)      XML encoded xDLMS APDU,
12564             (128...255) manufacturer specific
12565
```

12566 – all other transport_service_type and message_type values are reserved for future use.

12567 **5.3.9.2.4 communication_window**

12568 Defines the time points when the communication window(s) for the push become(s) active
 12569 (start_time) and inactive (end_time). See Figure 31.

```
12570     array      window_element
12571
12572     window_element ::= structure
12573
12574     {
12575         start_time: octet-string,
12576         end_time:   octet-string
12577     }
```

12579 start_time and end_time are formatted as specified in 4.1.6.1 for date-time including wildcards.

12580 If the end of a communication window is reached an already started push operation is completed.

12581 If no communication windows are defined (array [0]) the push operation is always possible.

12582 **5.3.9.2.5 randomisation_start_interval**

12583 To avoid simultaneous push operations by a lot of devices at exactly the same point in time, a
 12584 randomisation interval – in seconds – can be defined. This means that the push operation is
 12585 not started immediately after the invocation of the *push* method but randomly delayed within
 12586 the interval.

12587 The *randomisation_start_interval* attribute defines the maximum value which can be obtained
 12588 from a randomizing algorithm. The resulting value is used to delay the first push operation. It is
 12589 not used anymore in case of retries.

12590 If no communication window is active when invoking the *push* method, the delay starts at the
 12591 beginning of the next communication window. If the *push* method is invoked during an active
 12592 communication window, the push operation is still delayed.

12593 The *randomisation_start_interval* is only active for the first push attempt. If no
 12594 *communication_window* is defined, the *randomisation_start_interval* is active for every push
 12595 attempt.

12596 If the *randomisation_start_interval* is set to 0 no delay is active.

12597 **5.3.9.2.6 number_of_retries**

12598 Defines the maximum number of retries in the case of unsuccessful or skipped push attempts.
 12599 After a successful push operation no further push attempts are made until the push operation
 12600 is triggered again. A push is treated as successful if a lower layer transmission confirmation
 12601 has been received.

12602 The conditions of detecting an unsuccessful push attempt may depend on the communication
 12603 profile used and on the implementation and it is therefore out of the Scope of this document.

12604 **5.3.9.2.7 repetition_delay**

12605 The time delay, expressed in seconds until the next push attempt is started after an
 12606 unsuccessful push.

12607 NOTE 3 The repetition delay itself is not influenced by the communication window. But a retry only can be
 12608 made if a communication window is active at that time. Otherwise it is handled like an unsuccessful push attempt.

12609 NOTE 4 The push data is not stored in an intermediate buffer. In the case of retries, the current values of the
 12610 attributes may change with every push attempt.

12611

12612 **5.3.9.3 Method description**

12613 **5.3.9.3.1 push (data)**

12614 Activates the push process leading to the elaboration and the sending of the push data taking
 12615 into account the values of the attributes defined in the given instance of this IC.

12616 **data ::= integer (0)**

12617

Table 47 – 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: <ul style="list-style-type: none"> - in the case of selective access by time period, (i.e. number of month, days, hours ...) 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.

12618

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.

12619

12620

12621 **5.3.10 Push Setup (class_id = 40, version = 1)**12622 **5.3.10.1 Overview**

12623 See 5.3.9.

12624 In version 1 the possibility of data protection has been added offering the same options as
12625 defined in the “Data protection” IC.12626 This version of the interface class is intended to facilitate the use of relative and absolute data
12627 selection. It would be common practice to use an instance of the Push setup IC with relative
12628 data selection for routine data push, and an instance of the Push setup IC with absolute data
12629 for special cases.

12630

Push setup	0...n	class_id = 40, version = 1			
Attributes	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
8. port_reference (static)	octet-string				x + 0x38
9. push_client_SAP (static)	integer				x + 0x40
10. push_protection_parameters (static)	array				x + 0x48
Specific methods	m/o				
1. push (data)	m				x + 0x58

12631

12632 **5.3.10.2 Attribute description**12633 **5.3.10.2.1 logical_name**

12634 Identifies the “Push setup” object instance. See 6.2.24.

12635

12636 **5.3.10.2.2 push_object_list**

12637 Defines the list of attributes to be pushed.

12638 Upon invocation of the *push* (data) method the items are sent to the destination defined in the
12639 *send_destination_and_method* attribute.

12640

12641 Two, mutually exclusive selective access mechanisms are available :

- 12642 – relative selective access, i.e. entries defined relative to current date or entry are returned; this mechanism is controlled by the *data_index* element; or
- 12644 – absolute selective access, i.e. entries in an explicitly defined date range or entry range are returned: entries selected by this mechanism are controlled by the *restriction_element* and the columns are controlled by the lower nibble of the MS byte in the *data_index*.

12647 array push_object_definition

12648

12649 push_object_definition ::= structure

12650

```
12651 {
12652   class_id: long-unsigned,
12653   logical_name: octet-string,
12654   attribute_index: integer,
12655   data_index: long-unsigned,
12656   restriction: restriction_element
12657 }
```

12658

12659 Where:

- 12660 – *attribute_index* is a pointer to the attribute within the object, identified by *class_id* and *logical_name*: *attribute_index* 1 refers to the first attribute (i.e. the *logical_name*), *attribute_index* 2 to the second attribute etc.; *attribute_index* 0 refers to all public attributes, noting IEC 62056-5-3:2021, 4.2.4.3.7;
- 12664 – *data_index* is a pointer selecting a specific element of an attribute with a complex data type (structure or array).
- 12666 – if the data type of the attribute is simple, then *data_index* has no meaning;
- 12667 – if the data type of the attribute is a structure or an array – other than the buffer of a Profile generic object – then *data_index* points to one or several specific elements in the structure or array;
- 12670 – when the attribute is the *buffer* of a “Profile generic” object, the *data_index* carries selective access parameters relative to current date or entry.

12672

data_index	MS-Byte		LS-Byte
	Upper nibble	Lower nibble	

12673

12674

12675 – 0x0000 = the whole attribute is pushed.

12676 – 0x0001 to 0xFFFF = identifies one element in the complex attribute;

12677 – 0x1000 to 0xFFFF = selective access to the array holding the buffer of a “Profile generic”
 12678 object. The data-index selects entries within a number of last (recent) time periods, or a
 12679 number of last (recent) entries, as well as the columns in the array.

12680 The encoding is specified in Table 47.

12681 When the attribute is the buffer of a “Profile generic” object, then restriction_element specifies
 12682 absolute selective access parameters in an explicitly defined date range or entry range.

```

12683             restriction_element ::= structure
12684
12685             {
12686                 restriction_type: enum:
12687                     (0) none,
12688                     (1) range by date,
12689                     (2) range by entry
12690
12691                 restriction_value: CHOICE
12692
12693                     {
12694                         null-data,           // no restrictions apply
12695                         restriction_by_date,
12696                         restriction_by_entry
12697                     }
12698
12699
12700             restriction_by_date ::= structure
12701             {
12702                 from_date: octet-string,
12703                 to_date: octet-string
12704             }
12705
12706             restriction_by_entry ::= structure
12707
12708             {
12709                 from_entry: double-long-unsigned,
12710                 to_entry: double-long-unsigned
12711             }

```

12712 – restriction_element defines absolute selective access to a “Profile generic” buffer by date
 12713 range (from_date to to_date) or by entries (from_entry to to_entry). To use this absolute
 12714 selective access mechanism, data_index shall be :

- 12715 – MS Byte upper nibble set to 0x0;
- 12716 – MS Byte lower nibble set to 0x0 to 0xF in accordance with Table 47;
- 12717 – lower byte set to 0x00.
- 12718 – restriction_element is composed of restriction_type and restriction_value:
 - 12719 – for restriction by date range the restriction_type element holds (1) restriction by date
 and the restriction_value element holds restriction_by_date structure;
 - 12721 – for restriction by entries the restriction_type element holds (2) restriction by entry and
 the restriction_value element holds restriction_by_entry structure;
 - 12723 – otherwise the restriction_type element holds (0) none and the restriction_value
 element holds null-data. This choice shall be taken also if relative selective access is
 to be used.

12726 NOTE 1 If the push_object_list array is empty, the push operation is disabled.

12727 NOTE 2 The push_object_list attribute itself can be also pushed to identify the data pushed.

12728 Similarly to the case of the “Profile generic” IC, all attributes included in the `push_object_list`
 12729 attribute are pushed regardless of the access rights to them. Therefore, writing of the
 12730 `push_object_list` attribute should be restricted to clients with appropriate access rights.

12731 **5.3.10.2.3 send_destination_and_method**

12732 Contains the destination address (e.g. phone number, email address, IP address) where the
 12733 data specified by the `push_object_list` has to be sent, as well as the sending method.

```
12734           send_destination_and_method ::= structure
12735
12736           {
12737             transport_service:   transport_service_type,
12738             destination:        octet-string,
12739             message:            message_type
12740           }
```

12741 Where:

- 12742 – the `transport_service` element defines the type of service used to push the data:

```
12743     transport_service_type ::= enum:
12744         (0)      TCP,
12745         (1)      UDP,
12746         (2)      reserved for FTP,
12747         (3)      reserved for SMTP,
12748         (4)      SMS,
12749         (5)      HDLC,
12750         (6)      reserved for M-Bus,
12751         (7)      reserved for ZigBee®,
12752         (8)      DLMS Gateway
12753         (200...255) manufacturer specific
```

- 12754 – 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.

12755 Each “Push setup” object instance specifies a single destination. If it is required to push
 12756 data to several destinations, several “Push setup” objects have to be instantiated,

- 12757 – the `message_type` element identifies the encoding of the xDLMS APDU used.

```
12758     message_type ::= enum:
12759         (2)      A-XDR encoded xDLMS APDU,
12760         (3)      XML encoded xDLMS APDU,
12761         (128...255) manufacturer specific
```

- 12762 – all other `transport_service_type` and `message_type` values are reserved for future use.

12763 **5.3.10.2.4 communication_window**

12764 See 5.3.9.2.4.

12765 **5.3.10.2.5 randomisation_start_interval**

12766 To avoid simultaneous push operations by a lot of devices at exactly the same point in time, a
 12767 randomisation interval – in seconds – can be defined. This means that the push operation is
 12768 not started immediately after the invocation of the `push` method but randomly delayed within
 12769 the interval.

12770 The `randomisation_start_interval` attribute defines the maximum value which can be obtained
 12771 from a randomizing algorithm. The resulting value is used to delay the first push operation. It is
 12772 not used anymore in case of retries.

12776 If no communication window is active when invoking the *push* method, the delay starts at the
 12777 beginning of the next communication window. If the *push* method is invoked during an active
 12778 communication window, the push operation is still delayed.

12779 The *randomisation_start_interval* is only active for the first push attempt. If no
 12780 *communication_window* is defined, the *randomisation_start_interval* is active for every push
 12781 attempt.

12782 If the *randomisation_start_interval* is set to 0 no delay is active.

12783 If the randomisation time is longer than the first communication window, the first push attempt
 12784 will be made during any later window.

12785 **5.3.10.2.6 number_of_retries**

12786 See 5.3.9.2.6.

12787 **5.3.10.2.7 repetition_delay**

12788 See 5.3.9.2.7.

12789 **5.3.10.2.8 port_reference**

12790 Contains the logical name of a communication port setup object allowing the selection of a
 12791 specific communication channel for the push based on the *transport_service_type*. This mainly
 12792 applies in cases where several channels of the same type are supported.

12793 If this information is not available or not needed, the attribute may be left empty (octet-string
 12794 [0]).

12795 **5.3.10.2.9 push_client_SAP**

12796 Defines the client SAP where the push is directed to in the supporting layer of the
 12797 DataNotification service. The push process takes place within the application context of the AA
 12798 which is linked to the *push_client_SAP* attribute. The security context is determined by the
 12799 "Security setup" object referenced in the related "Association" object.

12800 **5.3.10.2.10 push_protection_parameters**

12801 Specifies all protection parameters to be applied to the DataNotification APDU when the push
 12802 data is sent. It offers the same options as the "Data Protection" IC but it is bound to the sending
 12803 of the data defined in the *push_object_list* attribute.

```

12804         array protection_parameters_element
12805
12806             protection_parameters_element ::= structure
12807
12808             {
12809                 protection_type: enum:
12810                     (0) authentication,
12811                     (1) encryption,
12812                     (2) authentication and encryption,
12813                     (3) digital signature
12814
12815             protection_options: structure
12816
12817             {
12818                 transaction_id: octet-string,
```

```

12819     originator_system_title: octet-string,
12820     recipient_system_title: octet-string,
12821     other_information: octet-string,
12822     key_info:           key_info_element
12823   }
12824 }
```

Where:

- transaction_id holds the identifier of the transaction;
- originator_system_title holds the system title of the originator that applies the protection;
- recipient_system_title holds the system title of the recipient which will check and remove the given protection;
- other_information carries other information. Its contents may be specified in project specific companion specifications. An octet-string of length 0 indicates that this field is not used; key_info holds the information necessary for the recipient to obtain the right key for checking and removing authentication and encryption. In the case of digital signature, key_info is not necessary and it shall be a structure of 0 elements.

The fields transaction-idother-information are A-XDR encoded OCTET STRINGS. The length and the value of each field are included in the AAD when applicable.

```

12837
12838     key_info_element ::= structure
12839   {
12840     key_info_type: enum:
12841
12842       (0) identified_key,
12843         -- used with identified_key_info_options
12844       (1) wrapped_key,
12845         -- used with wrapped_key_info_options
12846       (2) agreed_key
12847         -- used with agreed_key_info_options
12848
12849     key_info_options: CHOICE
12850
12851   {
12852     identified_key_info_options,
12853     wrapped_key_info_options,
12854     agreed_key_info_options
12855   }
12856 }
12857
12858     identified_key_info_options ::= enum:
12859
12860       (0) global_unicast_encryption_key,
12861       (1) global_broadcast_encryption_key
12862
12863     wrapped_key_info_options ::= structure
12864   {
12865     kek_id: enum:
12866
12867       (0) master_key,
12868       key_ciphered_data: octet-string
12869   }
12870
12871     agreed_key_info_options ::= structure
12872   {
12873     key_parameters: octet-string,
12874     key_ciphered_data: octet-string
12875   }
12876 }
```

12877 This attribute is first written by the client. The server may need to fill in some elements.

12878 For the use of the various elements see Table 22 and Table 23 of the "Data protection" IC
12879 (class_id = 30, version = 0).

12880 5.3.10.3 Method description

12881 5.3.10.3.1 push (data)

12882 See 5.3.9.3.1.

12883

12884 5.4 Previous versions of interface classes for time- and event-bound control

12885 5.4.1 Parameter monitor (class_id = 65, version = 0)

12886 5.4.1.1 Overview

12887 Instances of the "Parameter monitor" IC monitor a list of COSEM object attributes holding
12888 parameters.

12889 The parameters can be changed as usual. If the value of an attribute changes and this attribute
12890 is present in the *parameter_list* attribute, the identifier and the value of that attribute is
12891 automatically captured to the *changed_parameter* attribute. The time when the change of the
12892 parameter occurred is captured in the *capture_time* attribute. These attributes may be captured
12893 then by a "Profile generic" object. In this way, a log of all parameter changes can be built. For
12894 the OBIS code of the Parameter monitor log objects, see IEC 62056-6-1:2021, 6.5.

12895 NOTE 1 In the case of simultaneous or quasi simultaneous parameter changes the order of capturing and logging
12896 the changed parameters has to be managed by the application.

12897 Several "Parameter monitor" objects and corresponding "Profile generic" objects can be
12898 instantiated to manage a number of parameter groups. The link between the "Parameter monitor"
12899 object and the corresponding "Profile generic" object is via the *capture_object* attribute of the
12900 "Profile generic" object.

12901 NOTE 2 As the various parameters may be of different type and length, the entries in the profile column holding the
12902 parameters will be also of different type and length. This can be managed for example by capturing different kind of
12903 parameters into different Parameter list "Profile generic" objects and parameter logs.

12904 NOTE 3 The "Profile generic" object holding the parameter change log may capture other suitable object attributes,
12905 like the *time* attribute of the "Clock" object, and any other relevant values.

12906

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	m/o				
1. add_parameter (data)	o				x + 0x20
2. delete_parameter (data)	o				x + 0x28

12907

12908 **5.4.1.2 Attribute description**12909 **5.4.1.2.1 logical_name**

12910 Identifies the “Parameter monitor” object instance. See 6.2.14.

12911 **5.4.1.2.2 changed_parameter**

12912 Holds the identifier and the value of the most recently changed parameter.

```

12913             structure
12914             {
12915                 class_id:      long-unsigned,
12916                 logical_name: octet-string,
12917                 attribute_index: integer,
12918                 attribute_value: CHOICE
12919                 -- CHOICE as specified in the “Data” interface class
12920             }
12921

```

12922 **5.4.1.2.3 capture_time**12923 Provides data and time information showing when the value of the *changed_parameter* attribute
12924 has been captured.12925 *date-time* is formatted as specified in 4.1.6.1.12926 **5.4.1.2.4 parameter_list**

12927 Holds the list of parameters managed by a given instance of the “Parameter monitor” IC.

```

12928         parameter_list ::= array parameter_list_element
12929
12930         parameter_list_element ::= structure
12931         {
12932             class_id:      long-unsigned,
12933             logical_name: octet-string,
12934             attribute_index: integer
12935         }

```

12936 NOTE 4 The list of parameters monitored may be changed by using the *add_parameter* or *delete_parameter*
12937 methods or writing this attribute.12938 **5.4.1.3 Method description**12939 **5.4.1.3.1 add_parameter (data)**12940 Adds one parameter to the *parameter_list*.12941 *data*::= parameter_list_element12942 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
12943 the *buffer* of the “Profile generic” object capturing the *changed_parameter* attribute.12944 **5.4.1.3.2 delete_parameter (data)**12945 Deletes one parameter from the *parameter_list*.12946 *data*::= parameter_list_element12947 NOTE 6 When a parameter is deleted from the parameter list, its changes will not be logged any more. Removing
12948 an element from the list does not affect the *buffer* of the “Profile generic” object capturing the *changed_parameter*
12949 attribute.

12950

12951

12952 **5.5 Previous versions of payment metering related interface classes**

12953 There are no previous versions to report.

12954 **5.6 Previous versions of interface classes for setting up data exchange via local
12955 ports and modems**12956 **5.6.1 IEC local port setup (class_id = 19, version = 0)**12957 **5.6.1.1 Overview**12958 Instances of this IC define the operational parameters for communication using IEC 62056-
12959 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				

12960

12961 **5.6.1.2 Attribute description**12962 **5.6.1.2.1 logical_name**

12963 Identifies the “IEC local port setup” object instance. See 6.2.18.

12964 **5.6.1.2.2 default_mode**

12965 Defines the protocol used by the meter on the port.

12966 enum:

12967 (0) protocol according to IEC 62056-21:2002 (modes A...E)

12968 (1) protocol according to IEC 62056-46:2002/AMD1:2006, Clause 8.
12969 Using this enumeration value all other attributes of this IC are not
12970 applicable.12971 **5.6.1.2.3 default_baud**

12972 Defines the baud rate for the opening sequence

12973	enum:	(0)	300 baud,
12974		(1)	600 baud,
12975		(2)	1 200 baud,
12976		(3)	2 400 baud,
12977		(4)	4 800 baud,
12978		(5)	9 600 baud,
12979		(6)	19 200 baud,
12980		(7)	38 400 baud,
12981		(8)	57 600 baud,
12982		(9)	115 200 baud

12983 5.6.1.2.4 prop baud

12984 Defines the baud rate to be proposed by the meter

12985	enum:	(0)	300 baud,
12986		(1)	600 baud,
12987		(2)	1 200 baud,
12988		(3)	2 400 baud,
12989		(4)	4 800 baud,
12990		(5)	9 600 baud,
12991		(6)	19 200 baud,
12992		(7)	38 400 baud,
12993		(8)	57 600 baud,
12994		(9)	115 200 baud

12995 **5.6.1.2.5** **response_time**

12996 Defines the minimum time between the reception of a request (end of request telegram) and
12997 the transmission of the response (begin of response telegram).

12998 enum:
12999 (0) 20 ms,

13000 (1) 200 ms

13001 **5.6.1.2.6 device_addr**

13002 Device address according to IEC 62056-21:2002.

13003 **5.6.1.2.7 pass_p1**

13004 Password 1 according to IEC 62056-21:2002.

13005 **5.6.1.2.8 pass_p2**

13006 Password 2 according to IEC 62056-21:2002.

13007 **5.6.1.2.9 pass_w5**

13008 Password W5 reserved for national applications.

13009

13010 **5.6.2 IEC HDLC setup, (class_id = 23, version = 0)**

13011 **5.6.2.1 Overview**

13012 An instance of the “IEC HDLC setup” contains all data necessary to set up a communication channel according to IEC 62056-46:2002/AMD1:2006. 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					

13015

13016 **5.6.2.2 Attribute description**

13017 **5.6.2.2.1 logical_name**

13018 Identifies the “IEC HDLC setup” object instance. See 6.2.20.

13019 **5.6.2.2.2 comm_speed**

13020 The communication speed supported by the corresponding port:

13021 enum:

- 13022 (0) 300 baud,
- 13023 (1) 600 baud,
- 13024 (2) 1 200 baud,
- 13025 (3) 2 400 baud,
- 13026 (4) 4 800 baud,
- 13027 (5) 9 600 baud,
- 13028 (6) 19 200 baud,
- 13029 (7) 38 400 baud,
- 13030 (8) 57 600 baud,
- 13031 (9) 115 200 baud
- 13032 This communication speed can be overridden if the HDLC mode of a device is entered through
13033 a special mode of another protocol.
- 13034 **5.6.2.2.3 window_size_transmit**
- 13035 The maximum number of frames that a device or system can transmit before it needs to receive
13036 an acknowledgement from a corresponding station. During logon, other values can be
13037 negotiated.
- 13038 **5.6.2.2.4 window_size_receive**
- 13039 The maximum number of frames that a device or system can receive before it needs to transmit
13040 an acknowledgement to the corresponding station. During logon, other values can be
13041 negotiated.
- 13042 **5.6.2.2.5 max_info_length_transmit**
- 13043 The maximum information field length that a device can transmit. During logon, a smaller value
13044 can be negotiated.
- 13045 **5.6.2.2.6 max_info_length_receive**
- 13046 The maximum information field length that a device can receive. During logon, a smaller value
13047 can be negotiated.
- 13048 **5.6.2.2.7 inter_octet_time_out**
- 13049 Defines the time, expressed in milliseconds, over which, when no character is received from
13050 the primary station, the device will treat the already received data as a complete frame.
- 13051 **5.6.2.2.8 inactivity_time_out**
- 13052 Defines the time, expressed in seconds over which, when no frame is received from the primary
13053 station, the device will process a disconnection.
- 13054 When this value is set to 0, this means that the *inactivity_time_out* is not operational.

13055 **5.6.2.2.9 device_address**

13056 Contains the physical device address of a device:

13057 In the case of single byte addressing:

13058 0x00 NO_STATION Address,

13059 0x01...0x0F Reserved for future use,

13060 0x10...0x7D Usable address space,

13061 0x7E 'CALLING' device address,

13062 0x7F Broadcast address

13063 In the case of double byte addressing:

13064 0x0000 NO_STATION address,

13065 0x0001...0x000F Reserved for future use,

13066 0x0010...0x3FFD Usable address space,

13067 0x3FFE 'CALLING' physical device address,

13068 0x3FFF Broadcast address

13069

13070

13071

13072 **5.6.3 IEC twisted pair (1) setup (class_id = 24, version = 0)**13073 **5.6.3.1 Overview**

13074 NOTE The use of version 0 of the IEC twisted pair (1) setup IC is deprecated.

13075 This IC allows modelling and configuring communication channels according to IEC 62056-31:1999. Several communication channels can be configured.
13076

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				

13077

13078 **5.6.3.2 Attribute description**13079 **5.6.3.2.1 logical_name**

13080 Identifies the “IEC twisted pair setup” object instance. See 6.2.21.

13081 **5.6.3.2.2 secondary_address**

13082 *Secondary_address* memorizes the ADS of the secondary station that corresponds to the real equipment.

13084 octet-string (SIZE(6))

13085 **5.6.3.2.3 primary_address_list**

13086 *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.

13088 primary_address_list_type::= array primary_address_element

13089 primary_address_element::= octet-string

13090 The length of the octet-string is one octet.

13091 **5.6.3.2.4 tabi_list**

13092 *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.

13094 tabi_list_type::= array tabi_element

13095 tabi_element::= integer

13096 **5.6.3.2.5 fatal_error**

13097 *fatal_error* represents the last occurrence of one of the fatal errors of the protocols described in IEC 62056-31:1999.

13099 The initial default value of this variable is 0x00. Then, each fatal error is spotted.

13100 enum:

13101 (0) No-error,

13102 (1) t-EP-1F,

13103 (2) t-EP-2F,

13104 (3) t-EL-4F,

13105 (4) t-EL-5F,

13106 (5) eT-1F,

13107 (6) eT-2F,

- 13108 (7) e-EP-3F,
 13109 (8) e-EP-4F,
 13110 (9) e-EP-5F,
 13111 (10) e-EL-2F
 13112

13113 **5.6.4 PSTN modem configuration (class_id = 27, version = 0)**

13114 **5.6.4.1 Overview**

13115 NOTE The name of this IC was changed to “Modem configuration” in Edition 2.

13116 An instance of the “PSTN modem configuration” IC stores data related to the initialization of
 13117 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
<i>Specific methods</i>	<i>m/o</i>				

13118

13119 **5.6.4.2 Attribute description**

13120 **5.6.4.2.1 logical_name**

13121 Identifies the “PSTN modem configuration” object instance. See 6.2.6.

13122 **5.6.4.2.2 comm_speed**

13123 The communication speed between the device and the modem, not necessarily the
 13124 communication speed on the WAN.

13125 enum:

- 13126 (0) 300 baud,
 13127 (1) 600 baud,
 13128 (2) 1 200 baud,
 13129 (3) 2 400 baud,
 13130 (4) 4 800 baud,
 13131 (5) 9 600 baud,
 13132 (6) 19 200 baud,

- 13133 (7) 38 400 baud,
13134 (8) 57 600 baud,
13135 (9) 115 200 baud

13136 **5.6.4.2.3 initialization_string**

13137 This data contains all the necessary initialization commands to be sent to the modem in order
13138 to configure it properly. This may include the configuration of special modem features.

```
13139     array initialization_string_element  
13140  
13141     initialization_string_element ::= structure  
13142     {  
13143         request: octet-string,  
13144         response: octet-string  
13145     }
```

13146 If the array contains more than one initialization string element, they are subsequently sent to
13147 the modem after receiving an answer matching the defined response.

13148 REMARK It is assumed that the modem is pre-configured so that it accepts the initialization_string. If no initialization
13149 is needed, the initialization string is empty.

13150 **5.6.4.2.4 modem_profile**

13151 This data defines the mapping from Hayes standard commands/responses to modem specific
13152 strings.

```
13153     array     modem_profile_element  
13154     modem_profile_element: octet-string
```

13155

13156 The *modem_profile* array shall contain the corresponding strings for the modem used in
13157 following order:

```
13158     Element 0:      OK,  
13159     Element 1:      CONNECT,  
13160     Element 2:      RING,  
13161     Element 3:      NO CARRIER,  
13162     Element 4:      ERROR,
```

13163 Element 5: CONNECT 1 200,

13164 Element 6 NO DIAL TONE,

13165 Element 7: BUSY,

13166 Element 8: NO ANSWER,

13167 Element 9: CONNECT 600,

13168 Element 10: CONNECT 2 400,

13169 Element 11: CONNECT 4 800,

13170 Element 12 CONNECT 9 600,

13171 Element 13: CONNECT 14 400,

13172 Element 14: CONNECT 28 800,

13173 Element 15: CONNECT 36 600,

13174 Element 16: CONNECT 56 000

13175

5.6.5 Auto answer (class_id = 28, version = 0)

5.6.5.1 Overview

13178 This IC allows modelling how the device manages the “Auto answer” function of the modem, i.e.
13179 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					

13180

5.6.5.2 Attribute description

5.6.5.2.1 logical_name

13183 Identifies the “Auto answer” object instance. See 6.2.6.

5.6.5.2.2 mode

13185 Defines the working mode of the line when the device is auto answer.

13186

13187 enum:

13188 (0) line dedicated to the device,

13189 (1) shared line management with a limited number of calls
13190 allowed. Once the number of calls is reached, the window
13191 status becomes inactive until the next start date, whatever
13192 the result of the call,

13193 (2) shared line management with a limited number of
13194 successful calls allowed. Once the number of successful
13195 communications is reached, the window status becomes
13196 inactive until the next start date,

13197 (3) currently no modem connected,

13198 (200...255) manufacturer specific modes

13199 **5.6.5.2.3 listening_window**

13200 Contains the start and end instant when the window becomes active (for the start instant), and
13201 inactive (for the end instant). The start_date defines implicitly the period.

13202 EXAMPLE When the day of month is not specified (equal to 0xFF) this means that we have a daily share line
13203 management. Daily, monthly ...window management can be defined.

13204 array window_element

13205

13206 window_element::= structure

13207 {

13208 start_time: octet-string,

13209 end_time: octet-string

13210 }

13211 start_time and end_time are formatted as specified in 4.6.1 for *date-time*.

13212 **5.6.5.2.4 status**

13213 Here is defined the status of the window.

13214 enum:

13215 (0) Inactive: the device will manage no new incoming call. This
13216 status is automatically reset to Active when the next listening
13217 window starts,

13218 (1) Active: the device can answer to the next incoming call,

13219 (2) Locked: This value can be set automatically by the device or by
13220 a specific client when this client has completed its reading
13221 session and wants to give the line back to the customer before
13222 the end of the window duration. This status is automatically
13223 reset to Active when the next listening window starts.

5.6.5.2.5 number_of_calls

13225 This number is the reference used in modes 1 and 2.

13226 When set to 0, this means there is no limit.

5.6.5.2.6 number_of_rings

13228 Defines the number of rings before the meter connects the modem. Two cases are distinguished:
13229 number of rings within the window defined by attribute *listening_window* and number of rings outside
13230 the *listening_window*.

13231 nr_rings_type::= structure

13232 {

13233 nr_rings_in_window: unsigned, (0: no connect in window)

13234 nr rings out of window: unsigned (0: no connect out of window)

13235 }

13236

5.6.6 PSTN auto dial (class_id = 29, version = 0)

13238 **5.6.6.1** Overview

13239 NOTE The name of this IC was changed to "Auto connect" in Edition 2.

13240 An instance of the "PSTN auto dial" IC stores data related to the management data transfer
13241 between the device and the modem to perform auto dialling. Several modems can be configured.

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				

13242

5.6.6.2 Attribute description

13244 5.6.6.2.1 logical name

13245 Identifies the “PSTN auto dial” object instance. See 6.2.6.

13246 5.6.6.2.2 mode

13247 Defines if the device can perform auto-dialling.

13248 enum:

13249 (0) no auto dialling,

13250 (1) auto dialling allowed anytime,

13251 (2) auto dialling allowed within the validity time of the calling
13252 window,

13253 (3) “regular” auto dialling allowed within the validity time of the
13254 calling window; “alarm” initiated auto dialling allowed anytime,

13255 (200...255) manufacturer specific modes

13256 5.6.6.2.3 repetitions

13257 The maximum number of trials in the case of unsuccessful dialling attempts.

13258 5.6.6.2.4 repetition_delay

13259 The time delay, expressed in seconds until an unsuccessful dial attempt can be repeated.

13260 repetition_delay 0 means delay is not specified

13261 5.6.6.2.5 calling_window

13262 Contains the start and end date/time stamp when the window becomes active (for the start
13263 instant), or inactive (for the end instant). The start_date defines implicitly the period.

13264 EXAMPLE When day of month is not specified (equal to 0xFF) this means that we have a daily share line
13265 management. Daily, monthly ...window management can be defined.

13266 array window_element

13267 window_element::= structure

13268 {

13269 start_time: octet-string,

13270 end_time: octet-string

13271 }

13272 start_time and end_time are formatted as specified in 4.6.1 for *date-time*.

13273 5.6.6.2.6 phone_list

13274 Contains phone numbers, the device modem has to call under certain conditions. The link
13275 between entries in the array and the conditions are not contained in here.

13276 array phone_number

13277 phone_number ::= octet-string

13278

13279 5.6.7 Auto connect (class_id = 29, version = 1)

13280 **5.6.7.1** **Overview**

13281 This IC allows modelling the management of data transfer from the device to one or several
13282 destinations.

13283 The messages to be sent, the conditions on which they shall be sent and the relation between
13284 the various modes, the calling windows and destinations are not defined here.

13285 Depending on the mode, one or more instances of this IC may be necessary to perform the
13286 function of sending out messages.

13287

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				

13288

13289 5.6.7.2 Attribute description

13290 5.6.7.2.1 logical_name

13291 See 5.6.6.2.1.

13292 5.6.7.2.2 mode

13293 Defines the mode controlling the auto dial functionality concerning the timing, the message type
13294 to be sent and the infrastructure to be used.

13295 enum:

13296 (0) no auto dialling,

13297 (1) auto dialling allowed anytime,

13298 (2) auto dialling allowed within the validity time of the calling
13299 window,

13300 (3) "regular" auto dialling allowed within the validity time of the calling window; "alarm" initiated auto dialling allowed anytime,
13301

13302 (4) SMS sending via Public Land Mobile Network (PLMN),

- 13303 (5) SMS sending via PSTN,
 13304 (6) email sending,
 13305 (200..255) manufacturer specific modes

13306 **5.6.7.2.3 repetitions**

13307 See 5.6.6.2.3.

13308 **5.6.7.2.4 repetition_delay**

13309 See 5.6.6.2.4.

13310 **5.6.7.2.5 calling_window**

13311 See 5.6.6.2.5.

13312 **5.6.7.2.6 destination_list**

13313 Contains the list of destinations (for example phone numbers, email addresses or their
 13314 combinations) where the message(s) have to be sent under certain conditions. The conditions
 13315 and their link to the elements of the array are not defined here.

13316 array destination

13317 destination ::= octet-string

13318

13319 **5.6.8 GSM diagnostic (class_id = 47, version = 0)**

13320 **5.6.8.1 Overview**

13321 The GSM/GPRS network is undergoing constant changes in terms of registration status, signal
 13322 quality etc. It is necessary to monitor and log the relevant parameters in order to obtain
 13323 diagnostic information that allows identifying communication problems in the network.

13324 An instance of the “GSM diagnostic” class stores parameters of the GSM/GPRS network
 13325 necessary for analysing the operation of the network.

13326 A GSM diagnostic “Profile generic” object is also available to capture the attributes of the GSM
 13327 diagnostic object, see IEC 62056-6-1:2021, 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	

<i>Specific methods</i>	<i>m/o</i>		
-------------------------	------------	--	--

13328

13329 **5.6.8.2 Attribute description**13330 **5.6.8.2.1 logical_name**

13331 Identifies the “GSM Diagnostic” object instance. For logical names, see 6.2.23.

13332 **5.6.8.2.2 operator**

13333 Holds the name of the network operator e.g. “YourNetOp”

13334 **5.6.8.2.3 status**

13335 Indicates the registration status of the modem.

13336 enum:

13337 (0) not registered,

13338 (1) registered, home network,

13339 (2) not registered, but MT is currently searching a new
13340 operator to register to,

13341 (3) registration denied,

13342 (4) unknown,

13343 (5) registered, roaming

13344 (6) ... (255) reserved

13345 **5.6.8.2.4 cs_attachment**

13346 Indicates the current circuit switched status.

13347 enum:

13348 (0) inactive,

13349 (1) incoming call,

13350 (2) active,

13351 (3) ... (255) reserved

13352 **5.6.8.2.5 ps_status**

13353 The ps_status value field indicates the packet switched status of the modem.

13354 enum:

13355 (0) inactive,

13356 (1) GPRS,
13357 (2) EDGE,
13358 (3) UMTS,
13359 (4) HSDPA,
13360 (5) ... (255) reserved

13361 **5.6.8.2.6 cell_info**

13362 Represents the cell information:

```
13363     cell_info_type ::= structure  
13364     {  
13365         cell_ID:      long-unsigned,  
13366         location_ID:   long-unsigned,  
13367         signal_quality: unsigned,  
13368         ber:          unsigned  
13369     }
```

13370 Where:

13371 – cell_ID: Two-byte cell ID in hexadecimal format;
13372 – location_ID: Two-byte location area code (LAC) in hexadecimal format (e.g. "00C3" equals
13373 195 in decimal);
13374 – signal_quality: Represents the signal quality:

13375 (0) -113 dBm or less,
13376 (1) -111 dBm,
13377 (2...30) -109...-53 dBm,
13378 (31) -51 or greater,
13379 (99) not known or not detectable;
13380 – ber: Bit Error Rate (BER) measurement in percent:
13381 (0...7) as RXQUAL_n values specified in ETSI GSM 05.08:1996,8.2.4.
13382 (99) not known or not detectable.

13383 **5.6.8.2.7 adjacent_cells**

13384 array adjacent_cell_info

13385
 13386 adjacent_cell_info ::= structure
 13387 {
 13388 cell_ID: long-unsigned,
 13389 signal_quality: unsigned
 13390 }
 13391 Where:

- 13392 – cell_ID: Two-byte cell ID in hexadecimal format;
 - 13393 – signal_quality: Represents the signal quality:
- 13394 (0) –113 dBm or less,
- 13395 (1) –111 dBm,
- 13396 (2...30) –109...–53 dBm,
- 13397 (31) –51 or greater,
- 13398 (99) not known or not detectable.

13399 **5.6.8.2.8 capture_time**

13400 Holds the date and time when the data have been last captured.

13401 *date-time* is formatted as specified in 4.6.1.

13402 **5.6.9 GSM diagnostic (class_id: 47, version: 1)**

13403 **5.6.9.1 Overview**

13404 The cellular network is undergoing constant changes in terms of registration status, signal
 13405 quality, etc. It is necessary to monitor and log the relevant parameters in order to obtain
 13406 diagnostic information that allows identifying communication problems in the network.

13407 An instance of the “GSM diagnostic” class stores parameters of the GSM/GPRS, UMTS, CDMA
 13408 or LTE network necessary for analysing the operation of the network.

13409 A GSM diagnostic “Profile generic” object is also available to capture the attributes of the GSM
 13410 diagnostic object, see IEC 62056-6-1:2021, 6.5.

GSM diagnostic	0...n	class_id = 47, version = 1				
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				

13411

13412 **5.6.9.2 Attribute description**13413 **5.6.9.2.1 logical_name**

13414 See 5.6.8.2.1.

13415 **5.6.9.2.2 operator**

13416 See 5.6.8.2.2.

13417 **5.6.9.2.3 status**

13418 See 5.6.8.2.3.

13419 **5.6.9.2.4 cs_attachment**

13420 See 5.6.8.2.4.

13421 **5.6.9.2.5 ps_status**13422 The *ps_status* value field indicates the packet switched status of the modem.

```

13423         enum:
13424             (0)  inactive,
13425             (1)  GPRS,
13426             (2)  EDGE,
13427             (3)  UMTS,
13428             (4)  HSDPA,
13429             (5)  LTE,
13430             (6)  CDMA,
13431             (7)...(255) reserved

```

13432 **5.6.9.2.6 cell_info**

13433 Represents the cell information:

```

13434     cell_info_type::= structure
13435     {
13436         cell_ID:          double-long-unsigned,
13437         location_ID:      long-unsigned,
13438         signal_quality:   unsigned,
13439         ber:              unsigned,
13440         mcc:              long-unsigned,
13441         mnc:              long-unsigned,
13442         channel_number:   double-long-unsigned
13443     }

```

13444 Where:

13445 – cell_ID: Four-byte cell ID in hexadecimal format;

13446 – location_ID: Two-byte location area code (LAC) in the case of GSM networks or Tracking
 13447 Area Code (TAC) in the case of UMTS, CDMA or LTE networks in hexadecimal format (e.g.
 13448 "00C3" equals 195 in decimal);

13449 – **signal_quality**: Represents the signal quality:

13450 (0) –113 dBm or less,
 13451 (1) –111 dBm,
 13452 (2...30) –109...–53 dBm,
 13453 (31) –51 or greater,
 13454 (99) not known or not detectable;

13455 – **ber**: Bit Error Rate (BER) measurement in percent:

13456 (0...7) as RXQUAL_n values specified in ETSI GSM 05.08:1996, 8.2.4
 13457 (99) not known or not detectable.

13458

13459 – **mcc**: Mobile Country Code of the serving network, as defined in ITU-T E.212 (05.2008);

13460 – **mnc**: Mobile Network Code of the serving network, as defined in ITU-T E.212 (05.2008);

13461 – **channel_number**: Represents the absolute radio-frequency channel number (ARFCN or eaRFCN for LTE network).

13463 **5.6.9.2.7 adjacent_cells**

13464 See 5.6.8.2.7.

13465 **5.6.9.2.8 capture_time**

13466 See 5.6.8.2.8.

13467 **5.6.10 LTE monitoring (class_id: 151, version: 0)**

13468 **5.6.10.1 Overview**

13469 Instances of the “LTE monitoring” IC allow monitoring LTE modems by handling all data
 13470 necessary data for this purpose.

LTE monitoring	0...n	class_id = 151, version = 0			
Attributes	Data type	Min.	Max.	Def.	Short name
1. logical_name (static)	octet-string				x
2. lte_quality_of_service (dyn.)	LTE_qos_type				x + 0x08
Specific methods	m/o				

13471

13472 **5.6.10.2 Attribute description**

13473 **5.6.10.2.1 logical_name**

13474 Identifies the “LTE monitoring” object instance. See 6.2.23.

13475 **5.6.10.2.2 lte_quality_of_service**

13476 Represents the quality of service for the LTE network

```
13477     LTE_qos_type ::= structure
13478     {
13479         T3402:           long-unsigned,
13480         T3412:           long-unsigned,
13481         RSRQ:            unsigned,
13482         RSRP:            unsigned,
13483         qRxlevMin:       integer
13484     }
```

13485 Where:

- 13486 – T3402: timer in seconds, used on PLMN selection procedure and sent by the network to the
 13487 modem. Refer to 3GPP TS 24.008 V13.7.0 (2016-10), *Technical Specification Digital cellular*
 13488 *telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications*
 13489 *System (UMTS); LTE; Mobile radio interface Layer 3 specification; Core network protocols;*
 13490 *Stage 3*
- 13491 – 3GPP TS 24.301 V13.4.0 (2016-01) for details;
- 13492 – T3412: timer in seconds used to manage the periodic tracking area updating procedure and
 13493 sent by the network to the modem. Refer to 3GPP TS 24.008 V13.7.0 (2016-10), *Technical*
 13494 *Specification Digital cellular telecommunications system (Phase 2+) (GSM); Universal*
 13495 *Mobile Telecommunications System (UMTS); LTE; Mobile radio interface Layer 3*
 13496 *specification; Core network protocols; Stage 3*
- 13497 – 3GPP TS 24.301 V13.4.0 (2016-01) for details;
- 13498 – RSRQ: represents the signal quality as defined in 3GPP TS 24.008 V13.7.0 (2016-10),
 13499 *Technical Specification Digital cellular telecommunications system (Phase 2+) (GSM);*
 13500 *Universal Mobile Telecommunications System (UMTS); LTE; Mobile radio interface Layer 3*
 13501 *specification; Core network protocols; Stage 3*
- 13502 – 3GPP TS 24.301 V13.4.0 (2016-01):
 - 13503 (0) –19,5dB,
 13504 (1) –19 dB,
 13505 (2...31) –18,5...-3,5 dB,
 13506 (32) –3dB,
 13507 (99) Not known or not detectable;
- 13508 – RSRP: represents the signal level as defined in 3GPP TS 24.008 V13.7.0 (2016-10),
 13509 *Technical Specification Digital cellular telecommunications system (Phase 2+) (GSM);*
 13510 *Universal Mobile Telecommunications System (UMTS); LTE; Mobile radio interface Layer 3*
 13511 *specification; Core network protocols; Stage 3*
- 13512 – 3GPP TS 24.301 V13.4.0 (2016-01):
 - 13513 (0) –140dBm,
 13514 (1) –139 dBm,
 13515 (2... 94) –138...-45 dBm,
 13516 (95) –44dBm,
 13517 (99) Not known or not detectable;
- 13519 – qRxlevMin: specifies the minimum required Rx level in the cell in dBm as defined in 3GPP
 13520 TS 24.008 V13.7.0 (2016-10), *Technical Specification Digital cellular telecommunications*
 13521 *system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE;*
 13522 *Mobile radio interface Layer 3 specification; Core network protocols; Stage 3*
- 13523 – 3GPP TS 24.301 V13.4.0 (2016-01).

13524

13525 **5.7 Previous versions of interface classes for setting up data exchange via M-Bus**13526 **5.7.1 M-Bus client (class_id = 72, version = 0)**13527 **5.7.1.1 Overview**

13528 Instances of this IC allow setting up M-Bus slave devices and to exchange data with them. Each
 13529 M-Bus client object controls one M-Bus slave device. For details on the M-Bus dedicated
 13530 application layer, see EN 13757-3:2004.

13531 The M-Bus client device may have one or more physical M-Bus interfaces, which can be
 13532 configured using instances of the M-Bus master port setup IC, see 4.8.5.

13533 An M-Bus slave device is identified with its Primary Address, Identification Number,
 13534 Manufacturer ID etc. as defined in EN 13757-3:2004 Clause 5, *Variable Data respond*. These
 13535 parameters are carried by the respective attributes of the M-Bus client IC, see 4.8.3.

13536 Values to be captured from an M-Bus slave device are identified by the *capture_definition*
 13537 attribute, containing a list of data identifiers (DIB, VIB) for the M-Bus slave device. The data
 13538 are captured periodically or on an appropriate trigger. Each data element is stored in an M-Bus
 13539 value object, of IC “Extended register”. M-Bus value objects may be captured in M-Bus Profile
 13540 generic objects, eventually along with other, not M-Bus specific objects.

13541 Using the methods of M-Bus client objects, M-Bus slave devices can be installed and de-
 13542 installed. It is also possible to send data to M-Bus slave devices and to perform operations like
 13543 resetting alarms, setting the clock, transferring an encryption key etc.

13544

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

13545

13546 5.7.1.2 Attribute description

13547 5.7.1.2.1 logical_name

13548 Identifies the “M-Bus client” object instance. See 6.2.22.

13549 5.7.1.2.2 mbus_port_reference

13550 Provides reference to an “M-Bus master port setup” object, used to configure an M-Bus port,
 13551 each interface allowing to exchange data with one or more M-Bus slave devices.

13552 **5.7.1.2.3 capture_definition**

13553 Provides the capture_definition for M-Bus slave devices.

```
13554                 array     capture_definition_element  
13555                 capture_definition_element::= structure  
13556                 {  
13557                 data_information_block:     octet-string,  
13558                 value_information_block:     octet-string  
13559                 }
```

13560 NOTE 2 The elements data_information_block and value_information_block correspond to Data Information Block
13561 (DIB) and Value Information Block (VIB) described in EN 13757-3:2013, 6.2 and Clause 7 respectively.

13562

13563 **5.7.1.2.4 capture_period**

13564 >= 1: Automatic capturing assumed. Specifies the capture period in seconds.

13565 0: No automatic capturing: capturing is triggered externally or capture events occur
13566 asynchronously.

13567 **5.7.1.2.5 primary_address**

13568 Carries the primary address of the M-Bus slave device, in the range 0...250.

13569 If the slave device is already configured and thus, its primary address is different from 0, then
13570 this value shall be written to the *primary_address* attribute. From this moment, the data
13571 exchange with the M-Bus slave device is possible.

13572 Otherwise, the *slave_install* method shall be used; see 5.7.1.3.1.

13573 NOTE 3 The *primary_address* attribute cannot be used to store a desired primary address for an unconfigured slave
13574 device. If the primary address attribute is set, this means that the M-Bus client can immediately operate with this
13575 primary address, which is not the case with an unconfigured slave device.

13576 **5.7.1.2.6 identification_number**

13577 Carries the Identification Number element of the data header as specified in EN 13757-3:2004,
13578 5.4.

13579 It is either a fixed fabrication number or a number changeable by the customer, coded with 8
13580 BCD packed digits (4 Byte), and which thus runs from 00 000 000 to 99 999 999. It can be
13581 preset at fabrication time with a unique number, but could be changeable afterwards, especially
13582 if in addition a unique and not changeable fabrication number (DIF = 0x0C, VIF = 0x78 is
13583 provided.

13584 **5.7.1.2.7 manufacturer_id**

13585 Carries the Manufacturer Identification element of the data header as specified in EN 13757-
13586 3:2004, 5.5.

13587 It is coded unsigned binary with 2 bytes. This *manufacturer_id* is calculated from the ASCII
13588 code of the IEC 62056-21 manufacturer ID (three uppercase letters), using the formula specified
13589 in EN 13757-3:2004, 5.5.

13590 **5.7.1.2.8 version**

13591 Carries the Version element of the data header as specified in EN 13757-3:2004, 5.6. It
 13592 specifies the generation or version of the meter and depends on the manufacturer. It can be
 13593 used to make sure, that within each version number the identification # is unique.

13594 **5.7.1.2.9 device_type**

13595 Carries the Device type identification element of the data header as specified in EN 13757-
 13596 3:2004, 5.7, Table 3.

13597 **5.7.1.2.10 access_number**

13598 Carries the Access Number element of the data header as specified in EN 13757-3:2004, 5.8.

13599 It has unsigned binary coding, and it is incremented (modulo 256) by one before or after each
 13600 RSP-UD from the slave. Since it can also be used to enable private end users to detect an
 13601 unwanted over-frequently readout of their consumption meters, it should not be resettable by
 13602 any bus communication.

13603 **5.7.1.2.11 status**

13604 Carries the Status byte element of the data header as specified in

13605 EN 13757-3:2004, 5.9, Table 4 and 5.

13606 **5.7.1.2.12 alarm**

13607 Carries the Alarm state specified in EN 13757-3:2004 Annex D. It is coded with data type D
 13608 (Boolean, in this case 8 bit). Set bits signal alarm bits or alarm codes. The meaning of these
 13609 bits is manufacturer specific.

13610 **5.7.1.3 Method description**13611 **5.7.1.3.1 slave_install (data)**

13612 Installs a slave device, which is yet unconfigured (its primary address is 0).

13613 This method can be successfully invoked only if the value of the primary_address attribute is 0.
 13614 The following actions are performed:

13615 the M-Bus address 0 is checked for presence of a new device.

- 13616 – if no uninstalled M-Bus slave is found, the method invocation fails;
- 13617 – if the *slave_install* method is invoked with no parameter, then the primary address is
 13618 assigned automatically. This is done by checking the *primary_address* attribute of all M-Bus
 13619 client objects in the DLMS/COSEM device and then selecting the first unused number. The
 13620 *primary_address* attribute is set to this address and it is then transferred to the M-Bus slave
 13621 device;
- 13622 – if the *slave_install* method is invoked with a primary address as a parameter, then the
 13623 *primary_address* attribute is set to this value and it is then transferred to the M-Bus slave
 13624 device.

13625 *data::= unsigned (no data, or a valid primary address)*

13626 NOTE 4 Unconfigured slave devices are configured with primary address as specified in EN 13757-3:2004, Annex
 13627 E.5.

13628

13629 **5.7.1.3.2 slave_deinstall (data)**

13630 De-installs the slave device. The main purpose of this service is to uninstall the M Bus slave
13631 device and to prepare the master for the installation of a new device. The following actions are
13632 performed:

- 13633 – the M-Bus address is set to 0 in the M-Bus slave device;
- 13634 – the encryption key transferred previously to the M-Bus slave device is destroyed; the default
13635 key is not affected.
- 13636 – the attribute *primary_address* is also set to 0.

13637 NOTE 5 A new M-Bus slave can be installed only, once the value of the *primary_address* attribute is 0.

13638 data ::= integer (0)

13639 **5.7.1.3.3 capture**

13640 Capture values – as specified by the *capture_definition* attribute – from the M-Bus slave device.

13641 data ::= integer (0)

13642 **5.7.1.3.4 reset_alarm**

13643 Reset alarm state of the M-Bus slave device.

13644 data ::= integer (0)

13645 **5.7.1.3.5 synchronize_clock**

13646 Synchronize the clock of the M-Bus slave device with that of the M-Bus client device.

13647 data ::= integer (0)

13648 **5.7.1.3.6 data_send**

13649 Send data to the M-Bus slave device.

13650 data ::= array *data_definition_element*
13651
13652 *data_definition_element* ::= structure
13653 {
13654 *data_information_block*: octet-string,
13655 *value_information_block*: octet-string
13656 *data*: CHOICE
13657 {
13658 -- simple data types
13659 *null-data*: [0],
13660 *bit-string*: [4],
13661 *double-long*: [5],
13662 *double-long-unsigned*: [6],
13663 *octet-string*: [9],
13664 *visible-string*: [10],
13665 *utf8-string*: [12],
13666 *integer*: [15],
13667 *long*: [16],
13668 *unsigned*: [17],
13669 *long-unsigned*: [18],

```
13670           long64          [20],  
13671           long64-unsigned  [21],  
13672           float32         [23],  
13673           float64         [24]  
13674       }  
13675     }  
13676
```

13677 **5.7.1.3.7 set_encryption_key**

13678 Sets encryption key to be used with the M-Bus slave device.

13679 Changing the encryption key requires two steps: First, the key is sent to the M-Bus slave,
13680 encrypted with the master key, using the *transfer_key* method. Second, the key is set in the M-
13681 Bus master using the *set_encryption_key* method.

```
13682           data::= octet-string (encryption_key)
```

13683 After the installation of the M-Bus slave, the M-Bus client holds an empty encryption key. With
13684 this, encryption of M-Bus telegrams is disabled. Encryption can be disabled by invoking the
13685 *set_encryption_key* method with null- data as a parameter.

13686 **5.7.1.3.8 transfer_key**

13687 Transfers an encryption key to the M-Bus slave.

```
13688           data::= octet-string (encrypted_key)
```

13689 Each M-Bus slave device shall be delivered with a default encryption key.

13690 Before encrypted M-Bus telegrams can be used, an operational encryption key has to be sent
13691 to the M-Bus slave, by invoking the *transfer_key* method. The method invocation parameter is
13692 the operational encryption key encrypted with the default key of the M-Bus slave device. The
13693 M-Bus telegram sent is not encrypted. After the execution, the encryption is enabled and all
13694 further telegrams are encrypted.

13695 A new encryption key can be set in the M-Bus client by invoking the *set_encryption_key* method
13696 with the new encryption key as a parameter.

13697 With further invocations of the *transfer_key* method, new encryption keys can be sent to the M-
13698 Bus slave. The method invocation parameter is the new encryption key encrypted with the
13699 default key. The M-Bus telegram is encrypted.

13700 When an M-Bus slave is de-installed, the encryption key is destroyed, but the default key is not
13701 affected. Encryption remains disabled until a new encryption is transferred.

13702

13703 **5.8 Previous versions of interface classes for setting up data exchange over the** 13704 **internet**

13705 There are no previous versions to report.

13706 **5.9 Previous versions of interface classes for data exchange using S-FSK PLC**

13707 **5.9.1 S-FSK Phy&MAC setup (class_id = 50, version = 0)**

13708 **5.9.1.1 Overview**

13709 NOTE 1 The use of this version is deprecated.

13710 An instance of the “S-FSK Phy&MAC setup” IC stores the data necessary to set up and manage
13711 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				

13712

13713 **5.9.1.2 Attribute description**

13714 **5.9.1.2.1 logical_name**

13715 Identifies the “S-FSK Phy&MAC setup” object instance. See 6.2.24.

13716 **5.9.1.2.2 initiator_electrical_phase**

13717 Holds the MIB variable initiator-electrical-phase (variable 18) specified in IEC 61334-4-
13718 512:2001, 5.8.

13719 It is written by the client system to indicate the phase to which it is connected.

13720 enum:
13721 (0) Not defined (default),
13722 (1) Phase 1,
13723 (2) Phase 2,
13724 (3) Phase 3.

13725 NOTE 2 This enumeration is different from that of IEC 61334-4-512:2001.

13726

13727 5.9.1.2.3 delta_electrical_phase

13728 Holds the MIB variable delta-electrical-phase (variable 1) specified in IEC 61334-4-512:2001,
13729 5.2 and IEC 61334-5-1:2001, 3.5.5.3.

13730 It indicates the phase difference between the client's connecting phase and the server's
13731 connecting phase. The following values are predefined:

13732 enum:
13733 (0) Not defined: the server is temporarily not able to determine the phase
13734 difference,
13735 (1) The server system is connected to the same phase as the client system.

13736
13737 The phase difference between the server's connecting phase and the client's
13738 connecting phase is equal to:

13739
13740 (2) 60 degrees,
13741 (3) 120 degrees,
13742 (4) 180 degrees,
13743 (5) -120 degrees,
13744 (6) -60 degrees.
13745

13746 5.9.1.2.4 max_receiving_gain

13747 Holds the MIB variable max-receiving-gain (variable 2) specified in

13748 IEC 61334-4-512:2001, 5.2 and in IEC 61334-5-1:2001, 3.5.5.3.

13749

13750 Corresponds to the maximum allowed gain bound to be used by the server system in the
13751 receiving mode. The default unit is dB.

13752 NOTE 3 In IEC 61334-4-512:2001, no units are specified.

13753 The possible values of the gain may depend on the hardware. Therefore, after writing a value
13754 to this attribute, the value should be read back to know the actual value.

13755 5.9.1.2.5 max_transmitting_gain

13756 Holds the value of the max-transmitting-gain.

13757 Corresponds to the maximum attenuation bound to be used by the server system in the
13758 transmitting mode. The default unit is dB.

13759 The possible values of the gain may depend on the hardware. Therefore, after writing a value
13760 to this attribute, the value should be read back to know the actual value.

13761 5.9.1.2.6 search_initiator_gain

13762 This attribute is used in the intelligent search initiator process. If the value of the
13763 max_receiving_gain is below the value of this attribute, a fast synchronization process is
13764 possible.

13765 5.9.1.2.7 frequencies

13766 Contains frequencies required for S-FSK modulation.

13767 frequencies_type ::= structure
13768 {
13769 mark_frequency: double-long-unsigned,
13770 space_frequency: double-long-unsigned
13771 }

13772 The default unit is Hz.

13773 **5.9.1.2.8 mac_address**

13774 Holds the MIB variable mac-address (variable 3) specified in

13775 IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6.

13776 NOTE 4 MAC addresses are expressed on 12 bits.

13777 Contains the value of the address of the physical attachment (MAC address) associated to the
13778 local system. In the unconfigured state, the MAC address is “NEW-address”.

13779 This attribute is locally written by the CIASE when the system is registered (with a Register
13780 service). The value is used in each outgoing or incoming frame. The default value is "NEW-
13781 address".

13782 This attribute is set to NEW:

- 13783 – by the MAC sub-layer, once the time-out-not-addressed delay is exceeded;
- 13784 – when a client system “resets” the server system. See the “S-FSK Active initiator” IC in
13785 Clause 4.10.4.

13786 When this attribute is set to NEW:

- 13787 – the system loses its synchronization (function of the MAC-sublayer);
- 13788 – the mac_group_address attribute is reset (array of 0 elements);
- 13789 – the system automatically releases all AAs which can be released.

13790 NOTE 5 The second item is not present in IEC 61334-4-512:2001.

13791 The predefined MAC addresses are shown in Table 33.

13792

13793 **5.9.1.2.9 mac_group_addresses**

13794 Holds the MIB variable mac-group-address (variable 4) specified in

13795 IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6.

13796 Contains a set of MAC group addresses used for broadcast purposes.

13797 array mac-address

13798 mac-address ::= long-unsigned

13799 The ALL-configured-address, ALL-physical-address and NO-BODY addresses are not included
13800 in this list. These ones are internal predefined values.

13801 This attribute shall be written by the initiator using DLMS services to declare specific MAC
 13802 group addresses on a server system.

13803 This attribute is locally read by the MAC sublayer when checking the destination address field
 13804 of a MAC frame not recognized as an individual address or as one of the three predefined
 13805 values (ALL-configured-address, ALL-physical-address and NO-BODY).

13806 **5.9.1.2.10 repeater**

13807 Holds the MIB variable repeater (variable 5) specified in

13808 IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6.

13809 It specifies whether the server system effectively repeats all frames or not.

13810 enum:

- 13811 (0) never repeater,
- 13812 (1) always repeater,
- 13813 (2) dynamic repeater

13814

13815 If the repeater variable is equal to 0, the server system should never repeat the frames.

13816 If it is set to 1, the server system is a repeater: it has to repeat all frames received without error
 13817 and with a current credit greater than zero.

13818 If it is set to 2, then the repeater status can be dynamically changed by the server itself.

13819 NOTE 6 The value 2 value is not specified in IEC 61334-4-512:2001.

13820 This attribute is internally read by the MAC sub-layer each time a frame is received. The default
 13821 value is 2, and the server system is a repeater (*repeater_status* = TRUE).

13822 **5.9.1.2.11 repeater_status**

13823 Holds the current repeater status of the device.

13824 boolean:

- 13825 FALSE = no repeater,
- 13826 TRUE = repeater

13828 **5.9.1.2.12 min_delta_credit**

13829 Holds the MIB variable min-delta-credit (variable 9) specified in IEC 61334-4-512:2001, 5.3 and
 13830 in IEC 61334-5-1:2001, 4.3.7.6.

13831 NOTE 7 Only the three least significant bits are used.

13832 The Delta Credit (DC) is the subtraction of the Initial Credit (IC) and Current Credit (CC) fields
 13833 of a correct received MAC frame. The delta-credit minimum value of a correct received MAC
 13834 frame, directed to a server system, is held by this variable.

13835 The default value is set to the maximal initial credit (see IEC 61334-5-1:2001, 4.2.3.1 for further
 13836 explanations on the credit and the value of MAX_INITIAL_CREDIT). A client system can
 13837 reinitialise this variable by setting its value to the maximal initial credit.

13838 **5.9.1.2.13 initiator_mac_address**

13839 Holds the MIB variable initiator-mac-address specified in IEC 61334-5-1:2001, 4.3.7.6.

13840 Its value is either the MAC address of the active-initiator or the NO-BODY address, depending
 13841 on the value of the synchronization_locked attribute (see 5.9.1.2.14). See also IEC 61334-5-
 13842 1:2001, 3.5.3, 4.1.6.3 and 4.1.7.2.

13843 If the value NO-BODY is written then the server mac_address (see the *mac_address* attribute)
 13844 has to be set to NEW.

13845 **5.9.1.2.14 synchronization_locked**

13846 Holds the MIB variable synchronization-locked (variable 10) specified in IEC 61334-4-512:2001,
 13847 5.3.

13848 Controls the synchronization locked / unlocked state. See IEC 61334-5-1:2001 for more details.

13849 If the value of this attribute is equal to TRUE, the system is in the synchronization-locked state.
 13850 In this state, the initiator-mac-address is always equal to the MAC address field of the active-
 13851 initiator MIB object. See attribute 2 of the S-FSK Active initiator IC, in 4.10.4.

13852 If the value of this attribute is equal to FALSE, the system is in the synchronization-unlocked
 13853 state. In this state, the *initiator_mac_address* attribute is always set to the NO-BODY value: a
 13854 value change in the MAC address field of the active-initiator MIB object does not affect the
 13855 content of the *initiator_mac_address* attribute which remains at the NO-BODY value. The
 13856 default value of this variable shall be specified in the implementation specifications.

13857 NOTE 8 In the synchronization-unlocked state, the server synchronizes on any valid frame. In the synchronization
 13858 locked state, the server only synchronizes on frames issued or directed to the client system the MAC address of
 13859 which is equal to the value of the *initiator_mac_address* attribute.

13860

13861 **5.9.2 S-FSK IEC 61334-4-32 LLC setup (class_id = 55, version = 0)**

13862 **5.9.2.1 Overview**

13863 An instance of the “S-FSK IEC 61334-4-32 LLC setup” IC holds parameters necessary to set
 13864 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					

13865

13866 **5.9.2.2 Attribute description**

13867 **5.9.2.2.1 logical_name**

13868 Identifies the “S-FSK IEC 61334-4-32 LLC setup” object instance. See 6.2.24.

13869 **5.9.2.2.2 max_frame_length**

13870 Holds the length of the LLC frame in bytes. See IEC 61334-4-32:1996, 5.1.4.

13871 In the case of the S-FSK profile, as specified in IEC 61334-5-1:2001, 4.2.2, the maximum value
13872 is 242, but lower values may be chosen due to performance considerations.

13873 **5.9.2.2.3 reply_status_list**

13874 Holds the MIB variable reply-status-list (variable 11) specified in IEC 61334-4-512:2001, 5.4.

13875 Lists the L-SAPs that have a not empty RDR (Reply Data on Request) buffer, which has not
13876 already been read. The length of a waiting L-SDU is specified in number of sub frames (different
13877 from zero). The variable is locally generated by the LLC sub layer.

```
13878           array     reply_status
13879
13880           reply_status ::= structure
13881           {
13882             L-SAP-selector:      unsigned,
13883             length-of-waiting-L-SDU:  unsigned
13884           }
13885
```

13886 length-of-waiting-LSDU in the case of the S-FSK profile is in number of sub-frames; valid values
13887 are 1 to 7.

13888 **5.10 Previous versions of interface classes for setting up the LLC layer for ISO/IEC
13889 8802-2**

13890 There are no previous versions to report.

13891 **5.11 Previous versions of interface classes for setting up and managing DLMS/COSEM
13892 narrowband OFDM PLC profile for PRIME networks**

13893 There are no previous versions to report.

13894 **5.12 Previous versions of interface classes for setting up and managing DLMS/COSEM
13895 narrowband OFDM PLC profile for G3-PLC networks**

13896 **5.12.1 Mapping of G3-PLC IB attributes to COSEM IC attributes (Original version)**

13897 In terms of IEEE 802.15.4:2006, a meter is Reduced Function Device (RFD) while a
13898 concentrator / Neighbourhood Network Access Point (NNAP) is a Full Function Device (FFD) /
13899 PAN coordinator. In terms of DLMS/COSEM the meter is the server and the concentrator /
13900 NNAP is the client (or an agent for a client).

13901 As COSEM models only the server and not the client, the G3 NB OFDM PLC setup classes
13902 concern only the RFD (Reduced Function Device) and not the PAN coordinator.

13903 Table 48 shows the mapping of G3 NB OFDM PLC PIB attributes to attributes of COSEM
13904 interface classes.

13905
13906**Table 48 – Mapping of G3-PLC IB attributes specified in ITU-T G.9903:2013 Amd. 1 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	0x02000101	PRIME NB OFDM PLC Physical layer parameters (class_id = 90, version = 0) (5.12.2)	90 / Att. 2
mac_Rx_data_packet_count	0x02000102		90 / Att. 3
mac_Tx_cmd_packet_count	0x02000201		90 / Att. 4
mac_Rx_cmd_packet_count	0x02000202		90 / Att. 5
mac_CSMA_fail_count	0x02000103		90 / Att. 6
mac_CSMA_no_ACK_count	0x02000104		90 / Att. 7
mac_bad_CRC_count	0x02000108		90 / Att. 8
mac_broadcast_count	0x02000106		90 / Att. 9
mac_multicast_count	0x02000107		90 / Att. 10
MAC setup PIB attributes – Read only & read-write variables ¹			
mac_short_address	0x01000112	G3 NB OFDM PLC MAC setup (class_id = 91, version = 0) (5.12.3)	91 / Att. 2
mac_coord_short_address	0x01000107		91 / Att. 3
mac_PAN_id	0x0100010F		91 / Att. 4
mac_max_orphan_timer	0x02000109		91 / Att. 5
mac_security_enabled	0x01000111		91 / Att. 6
mac_freq_notching	0x0000006D		91 / Att. 7
mac_TMR_TTL	0x02000113		91 / Att. 8
mac_max_frame_retries	0x0100010D		91 / Att. 9
mac_neighbour_table_entry_TTL	0x02000114		91 / Att. 10
mac_neighbour_table	0x0000006B		91 / Att. 11
6LoWPAN adaptation layer IB attributes – Read only & read-write variables ²			
adp_max_hops	0x10	G3 NB OFDM PLC 6LoWPAN adaptation layer setup (Class_id = 92, version = 0) (5.12.4)	92 / Att. 2
adp_weak_LQI_value	0x1B		92 / Att. 3
adp_tone_mask	0x0F		92 / Att. 4
adp_discovery_attempts_wait_time	0x06		92 / Att. 5
adp_routing_configuration	0x12 – 0x19, 0x1A, 0x1C, 0x26		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	0x25		92 / Att. 10
adp_broadcast_log_table	0x0B		92 / Att. 11
adp_group_table	0x0E		92 / Att. 12
adp_max_join_wait_time	0x21		92 / Att. 13
adp_path_discovery_time	0x22		92 / Att. 14
adp_use_new_GMK_time	0x23		92 / Att. 15
adp_exp_prec_GMK_time	0x24		92 / Att. 16

¹⁾ See 3GPP TS 36.321 V15.5.0 (2019-05) clauses 9.3.6.2.2 and 9.3.6.2.3²⁾ See 3GPP TS 36.321 V15.5.0 (2019-05) clause 9.4.1.1

NOTE Whereas in 3GPP TS 36.321 V15.5.0 (2019-05) the camel notation is used, in COSEM interface class specifications – and in this table – the underscore notation is used.

13907

13908 **5.12.2 G3 NB OFDM PLC MAC layer counters (class_id = 90, version = 0)**13909 **5.12.2.1 Overview**

13910 An instance of the “G3 NB OFDM PLC MAC layer counters” IC stores counters related to the
 13911 MAC layer exchanges. The objective of these counters is to provide statistical information for
 13912 management purposes.

13913 The attributes of instances of this IC shall be read only. They can be reset using the reset
 13914 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				

13915

13916 **5.12.2.2 Attribute description**13917 **5.12.2.2.1 logical_name**

13918 Identifies the “G3 NB OFDM PLC MAC layer counters” object instance. See 6.2.28.

13919 **5.12.2.2.2 mac_Tx_data_packet_count**

13920 PIB attribute 0x02000101: Statistic counter of successfully transmitted data packets (MSDUs).

13921 **5.12.2.2.3 mac_Rx_data_packet_count**

13922 PIB attribute 0x02000102: Statistic counter of successfully received data packets (MSDUs).

13923 **5.12.2.2.4 mac_Tx_cmd_packet_count**

13924 PIB attribute 0x02000201: Statistic counter of successfully transmitted command packets.

13925 **5.12.2.2.5 mac_Rx_cmd_packet_count**

13926 PIB attribute 0x02000202: Statistic counter of successfully received command packets.

13927 **5.12.2.2.6 mac_CSMA_fail_count**

13928 PIB attribute 0x02000103: Counts the number of times when CSMA backoffs exceed
13929 macMaxCSMABackoffs.

13930 **5.12.2.2.7 mac_CSMA_no_ACK_count**

13931 PIB attribute 0x02000104: Counts the number of times when an ACK is not received while
13932 transmitting a unicast data frame (The loss of ACK is attributed to collisions).

13933 **5.12.2.2.8 mac_bad_CRC_count**

13934 PIB attribute 0x02000108: Statistic counter of the number of frames received with bad CRC.

13935 **5.12.2.2.9 mac_broadcast_count**

13936 PIB attribute 0x02000106: Statistic counter of the number of broadcast frames sent.

13937 **5.12.2.2.10 mac_multicast_count**

13938 PIB attribute 0x02000107: Statistic counter of the number of multicast frames sent.

13939 NOTE When a counter reaches the maximum value (0xFFFFFFFF), it is automatically rolled-over.

13940 **5.12.2.3 Method description**

13941 **5.12.2.3.1 reset (data)**

13942 This method forces a reset of the object. By invoking this method, the value of all counters is
13943 set to 0.

13944 data ::= integer (0)

13945

13946 **5.12.3 G3 NB OFDM PLC MAC setup (class_id = 91, version = 0)**

13947 **5.12.3.1 Overview**

13948 An instance of the “G3 NB OFDM PLC MAC setup” IC holds the necessary parameters to set
13949 up and manage the G3 NB OFDM PLC IEEE 802.15.4:2006 MAC sub-layer.

13950 These attributes influence the functional behaviour of an implementation. Implementations may
13951 allow changes to the attributes during normal running, i.e. even after the device start-up
13952 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				

13953

13954 **5.12.3.2 Attribute description**

13955 **5.12.3.2.1 logical_name**

13956 Identifies the “G3 NB OFDM PLC MAC setup” object instance. See 6.2.28.

13957 **5.12.3.2.2 mac_short_address**

13958 PIB attribute 0x01000112: Device short address.

13959 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 13960 a short address, so that a device cannot be in the state where it is associated but does not have 13961 a short address.

13963 **5.12.3.2.3 mac_coord_short_address**

13964 PIB attribute 0x01000107: Coordinator short address.

13965 NOTE 1 The short address assigned to the coordinator through which the device is associated.

13966 **5.12.3.2.4 mac_PAN_id**

13967 PIB attribute 0x0100010F: PAN ID.

13968 NOTE 2 The 16-bit identifier of the PAN on which the device is operating. A value equal to 0xFFFF indicates that 13969 the device is not associated.

13970 **5.12.3.2.5 mac_max_orphan_timer**

13971 PIB attribute 0x02000109: The maximum number of seconds without communication with a 13972 particular device after which it is declared as an orphan.

13973 **5.12.3.2.6 mac_security_enabled**

13974 PIB attribute 0x01000111: Security enabled.

13975 boolean:
 13976 TRUE: security enabled,
 13977 FALSE: security disabled

13978 **5.12.3.2.7 mac_freq_notching**

13979 PIB attribute 0x0000006D: S-FSK 63 and 74 kHz frequency notching.

13980 boolean:
 13981 TRUE: notching enabled,
 13982 FALSE: notching disabled

13983 Default value is FALSE (disabled).

13984 **5.12.3.2.8 mac_TMR_TTL**

13985 PIB attribute 0x02000113: Maximum valid time of tone map parameters in the neighbour table
 13986 in seconds.

13987 **5.12.3.2.9 mac_max_frame_retries**

13988 PIB attribute 0x0100010D: Maximum number of retransmissions.

13989 **5.12.3.2.10 mac_neighbour_table_entry_TTL**

13990 PIB attribute 0x02000114: Maximum valid time for an entry in the neighbour table in seconds.

13991 **5.12.3.2.11 mac_neighbour_table**

13992 PIB attribute 0x0000006B: See ITU-T G.9903 Amd. 1:2013, 9.3.7.2 for CENELEC and FCC
 13993 bands.

13994 The neighbour table contains information about all the devices within the POS of the device.
 13995 One element of the table represents one PLC direct neighbour of the device.

```
13996                   array        mac_neighbour_table_element
13997
13998                   mac_neighbour_table_element::= structure
13999                   {
14000                    mac_short_address:           long-unsigned,
14001                    payload_modulation_scheme:    boolean,
14002                    tone_map:                      bit-string,
14003                    modulation:                    enum,
14004                    tx_gain:                      integer,
14005                    tx_res:                        boolean,
14006                    tx_coeff:                     array,
14007                    lqi:                            unsigned,
14008                    TMR_valid_time:                double-long-unsigned,
14009                    neighbour_valid_time:        double-long-unsigned
14010                   }
```

14011 NOTE 3 This table is actualized each time any frame is received from a neighbour device, and each time a Tone
 14012 Map Response is received.

mac_short_	The MAC Short Address of the node which this entry refers to.
address	

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	<p>The modulation type to use for communicating with the device.</p> <p>enum:</p> <ul style="list-style-type: none">(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
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. <ul style="list-style-type: none">– 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 macTMRTTL.
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.

14013

14014 **5.12.3.3 Method description**14015 **5.12.3.3.1 mac_get_neighbour_table_entry (data)**14016 This method is used to retrieve the mac neighbour table for one MAC short address. It may be
14017 used to perform topology monitoring by the client.

14018 The method invocation parameter contains a mac_short_address.

14019 data::= long-unsigned

14020 The response parameter includes the neighbour table for this mac_short_address.

14021 data::= array *mac_neighbour_table_element*

14022 where `mac_neighbour_table-element` is as defined in the `mac_neighbour_table` attribute of the
 14023 present IC.

14024 **5.12.4 G3 NB OFDM PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 0)**

14025 **5.12.4.1 Overview**

14026 An instance of the “G3 NB OFDM PLC 6LoWPAN adaptation layer setup” IC holds the
 14027 necessary parameters to set up and manage the G3 NB OFDM PLC 6LoWPAN Adaptation layer.

14028 These attributes influence the functional behaviour of an implementation. Implementations may
 14029 allow changes to their values during normal running, i.e. even after the device start-up sequence
 14030 has been executed.

14031

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_TTL (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	65535	5 000	x + 0x70	
15. adp_use_new_GMK_time (static)	long-unsigned	0	65535	3 600	x + 0x78	
16. adp_exp_prec_GMK_time (static)	long-unsigned	0	3 600	3 600	X + 0x80	
Specific methods	m/o					

14032

14033 **5.12.4.2 Attribute description**

14034 **5.12.4.2.1 logical_name**

14035 Identifies the “G3 NB OFDM 6LoWPAN adaptation layer setup” object instance. See 6.2.28.

14036 **5.12.4.2.2 adp_max_hops**

14037 PIB attribute 0x10: Defines the maximum number of hops to be used by the routing algorithm.

14038 **5.12.4.2.3 adp_weak_LQI_value**

14039 PIB attribute 0x1B: The weak link value defines the threshold below which a direct neighbour
 14040 is not taken into account during the commissioning procedure (compared to the LQI measured).

14041 **5.12.4.2.4 adp_tone_mask**

14042 PIB attribute 0x0F: Defines the Tone Mask to use during symbol formation. The length of the
 14043 bit-string is 70 bits.

14044 **5.12.4.2.5 adp_discovery_attempts_wait_time**

14045 PIB attribute 0x06: Allows programming the maximum wait time between invocations of two
 14046 consecutive network discovery primitives (in seconds).

14047 **5.12.4.2.6 adp_routing_configuration**

14048 The routing configuration element specifies all parameters linked to the routing mechanism
 14049 described in ITU-T G.9903 Amd. 1:2013. The elements are specified in 9.4.1.2 of that
 14050 Recommendation.

14051 NOTE 1 The Link cost calculation is provided in ITU-T G.9903 Amd. 1:2013 Annex B.

```
14052           array routing_configuration
14053
14054           routing_configuration ::= structure
14055           {
14056             adp_net_traversal_time:          long-unsigned,
14057             adp_routing_table_entry_TTL:    double-long-unsigned,
14058             adp_Kr:                         unsigned,
14059             adp_Km:                         unsigned,
14060             adp_Kc:                         unsigned,
14061             adp_Kq:                         unsigned,
14062             adp_Kh:                         unsigned,
14063             adp_Krt:                        unsigned,
14064             adp_RREQ_retries:              unsigned,
14065             adp_RREQ_RERR_wait:            long-unsigned,
14066             adp_blacklist_table_entry_TTL: long-unsigned
14067           }
```

14068

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_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

14069 **5.12.4.2.7 adp_broadcast_log_table_entry_TTL**14070 PIB attribute 0x02: Defines the time while an entry in the adpBroadcastLogTable remains active
14071 in the table (in seconds).14072 **5.12.4.2.8 adp_routing_table**14073 PIB attribute 0x0C: The routing table contains information about the different routes in which
14074 the device is implicated.14075 array routing_table
14076
14077 routing_table ::= structure

```

14078    {
14079        destination_address: long-unsigned,
14080        next_hop_address: long-unsigned,
14081        route_cost: long-unsigned,
14082        hop_count: unsigned,
14083        weak_link_count: unsigned,
14084        status: enum,
14085        valid_time: unsigned
14086    }
14087

```

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.

14088

5.12.4.2.9

14089

5.12.4.2.10 adp_context_information_table

14090

PIB attribute 0x07: Contains the context information associated to each CID extension field.

```

14091        array      context_information_table
14092        context_information_table::= structure
14093        {
14094            CID:          bit-string,
14095            context_length: unsigned,
14096            context:       octet-string,
14097            C:             boolean,
14098            valid_lifetime: long-unsigned
14099        }
14100

```

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
14101	
14102	5.12.4.2.11 adp_blacklist_table
14103	PIB attribute 0x25: Contains the list of the blacklisted neighbours.
14104	array blacklisted_neighbour_set
14105	blacklisted_neighbour_set ::= structure
14106	{
14107	blacklisted_neighbour_address: long-unsigned,
14108	valid_time: long-unsigned
14109	}
14110	
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.
14111	
14112	5.12.4.2.12 adp_broadcast_log_table
14113	PIB attribute 0x0B: Contains the broadcast log table.
14114	NOTE 5 This table provides a list of the broadcast packets recently received by this device.
14115	array broadcast_log_table
14116	broadcast_log_table ::= structure
14117	{
14118	source_address: long-unsigned,
14119	sequence_number: unsigned,
14120	time_to_live: long-unsigned
14121	}
14122	
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.

14123

14124

14125 **5.12.4.2.13 adp_group_table**

14126 PIB attribute 0x0E: Contains the group addresses to which the device belongs.

```
14127         array      group_table
14128         group_table ::= structure
14129         {
14130             group_address: long-unsigned
14131         }
14132
14133             group_address      Group address to which this node has been
14134             subscribed.
14135
```

14136 **5.12.4.2.14 adp_max_join_wait_time**

14137 PIB attribute 0x21: Network join timeout in seconds for LBD.

14138 **5.12.4.2.15 adp_path_discovery_time**

14139 PIB attribute 0x22: Timeout for path discovery in msec.

14140 **5.12.4.2.16 adp_use_new_GMK_time**

14141 PIB attribute 0x23: The wait time in seconds for a device to use new GMK after rekeying.

14142 **5.12.4.2.17 adp_exp_prec_GMK_time**

14143 PIB attribute 0x24: The time in seconds to keep PrecGMK after switching to a new GMK.

14144

14145

14146 **5.12.5 G3-PLC MAC setup (class_id = 91, version = 1)**

14147 **5.12.5.1 Overview**

14148 An instance of the “G3-PLC MAC setup” IC holds the necessary parameters to set up and
14149 manage the G3-PLC IEEE 802.15.4 MAC sub-layer.

14150 These attributes influence the functional behaviour of an implementation. Implementations may
14151 allow changes to the attributes during normal running, i.e. even after the device start-up
14152 sequence has been executed.

G3-PLC MAC setup		0...n	class_id = 91, version = 1			
Attributes		Data type	Min.	Max.	Def.	Short name
23. logical_name	(static)	octet-string				x
24. mac_short_address	(dyn.)	long-unsigned	0x0000	0xFFFF	0xFFFF	x + 0x08
25. mac_RC_coord	(dyn.)	long-unsigned	0x0000	0xFFFF	0xFFFF	x + 0x10
26. mac_PAN_id	(dyn.)	long-unsigned	0x0000	0xFFFF	0xFFFF	x + 0x18
27. mac_key_table	(dyn.)	array				x + 0x20
28. mac_frame_counter	(dyn.)	double-long-unsigned	0	4 294 967 295	0	x + 0x28
29. mac_tone_mask	(static)	bit-string			0x00000 0000FF FFFFFFF F	x + 0x30
30. mac_TMR_TTL	(static)	unsigned	0	255	2	x + 0x38
31. mac_max_frame_retries	(static)	unsigned	0	10	5	x + 0x40
32. mac_neighbour_table_entry_TTL	(static)	unsigned	0	255	255	x + 0x48
33. mac_neighbour_table	(dyn.)	array				x + 0x50
34. mac_high_priority_window_size	(static)	unsigned	1	7	7	x + 0x58
35. mac_CSMA_fairness_limit	(static)	unsigned	See below	255	25	x + 0x60
36. mac_beacon_randomization_window_length	(static)	unsigned	1	254	12	x + 0x68
37. mac_A	(static)	unsigned	3	20	8	x + 0x70
38. mac_K	(static)	unsigned	1	See below	5	x + 0x78
39. mac_min_CW_attempts	(static)	unsigned	0	255	10	x + 0x80
40. mac_cenelec_legacy_mode	(static)	unsigned	0	255	1	x + 0x88
41. mac_FCC_legacy_mode	(static)	unsigned	0	255	1	x + 0x90
42. mac_max_BE	(static)	unsigned	0	20	8	x + 0x98
43. mac_max_CSMA_backoffs	(static)	unsigned	0	255	50	x + 0xA0
44. mac_min_BE	(static)	unsigned	0	20	3	x + 0xA8
Specific methods		m/o				
2. mac_get_neighbour_table_entry(data)		o				x + 0xB0

14153

5.12.5.2 Attribute description

5.12.5.2.1 logical_name

Identifies the “G3-PLC MAC setup” object instance. See 6.2.28.

5.12.5.2.2 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.

5.12.5.2.3 mac_RC_coord

PIB attribute 0x010F: Route cost to coordinator, to be used in the beacon payload as RC_COORD.

14165 **5.12.5.2.4 mac_PAN_id**

14166 PIB attribute 0x0050: The 16-bit identifier of the PAN through which the device is operating. A
 14167 value equal to 0xFFFF indicates that the device is not associated.

14168 **5.12.5.2.5 mac_key_table**

14169 PIB attribute 0x0071: This attribute holds GMK keys required for MAC layer ciphering. The
 14170 attribute can hold up to two 16-bytes keys. The Key Identifier value must be different for each
 14171 key.

14172 For security reason, the key entries cannot be read, only written.

```
14173         array mac_GMK
14174         mac_GMK ::= structure
14175         {
14176             key_id:     unsigned,
14177             key:       octet-string
14178         }
```

14179 Where:

14180
 14181 key_id The Key Identifier used to refer to this key, can take the value 0 or 1.
 14182 key The AES-128 key used for ciphering the frames exchanged at MAC layer.

14184 **5.12.5.2.6 mac_frame_counter**

14185 PIB attribute 0x0077: The outgoing frame counter for this device, used when ciphering frames
 14186 at MAC layer.

14187 **5.12.5.2.7 mac_tone_mask**

14188 PIB attribute 0x0110: Defines the tone mask to use during symbol formation.

14189 **5.12.5.2.8 mac_TMR_TTL**

14190 PIB attribute 0x010D: Maximum time to live of tone map parameters entry in the neighbour table
 14191 in minutes.

14192 **5.12.5.2.9 mac_max_frame_retries**

14193 PIB attribute 0x0059: Maximum number of retransmissions.

14194 **5.12.5.2.10 mac_neighbour_table_entry_TTL**

14195 PIB attribute 0x010E: Maximum time to live for an entry in the neighbour table in minutes.

14196 **5.12.5.2.11 mac_neighbour_table**

14197 PIB attribute 0x010A: See ITU-T G.9903:2014, 9.3.7.2 for CENELEC and FCC bands.

14198 The neighbour table contains information about all the devices within the POS of the device.
 14199 One element of the table represents one PLC direct neighbour of the device.

```
14200         array neighbour_table
14201         neighbour_table ::= structure
14202         {
14203             short_address:           long-unsigned,
14204             payload_modulation_scheme: boolean,
14205             tone_map:                 bit-string,
```

```

14206     modulation:          enum,
14207     tx_gain:              integer,
14208     tx_res:               enum,
14209     tx_coeff:             bit-string,
14210     lqi:                  unsigned,
14211     phase_differential: integer,
14212     TMR_valid_time:       unsigned,
14213     neighbour_valid_time: unsigned
14214   }
14215

```

14216 NOTE 1 This table is actualized each time any frame is received from a neighbour device, and each time a
 14217 Tone Map Response is received.

14218 Where:

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

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: '0001 1101 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.

14219

5.12.5.2.12 mac_high_priority_window_size

PIB attribute 0x0100: The high priority contention window size in number of slots.

5.12.5.2.13 mac_CSMA_fairness_limitPIB 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.**5.12.5.2.14 mac_beacon_randomization_window_length**

PIB attribute 0x0111: Duration time in seconds for the beacon randomization.

5.12.5.2.15 mac_A

PIB attribute 0x0112: This parameter controls the adaptive CW linear decrease.

5.12.5.2.16 mac_K

PIB attribute 0x0113: Rate adaptation factor for channel access fairness limit. This attribute can take a value between 1 and macCSMAFairnessLimit.

5.12.5.2.17 mac_min_CW_attempts

PIB attribute 0x0114: Number of consecutive attempts while using minimum CW.

5.12.5.2.18 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 $I(i,j) = 0$.

5.12.5.2.19 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.

5.12.5.2.20 mac_max_BE

PIB attribute 0x0047: Maximum value of backoff exponent. It should always be greater than macMinBE.

5.12.5.2.21 mac_max_CSMA_backoffs

PIB attribute 0x004E: Maximum number of backoff attempts.

5.12.5.2.22 mac_min_BE

PIB attribute 0x004F: Minimum value of backoff exponent.

14262 **5.12.5.3 Method description**

14263 **5.12.5.3.1 mac_get_neighbour_table_entry (data)**

14264 This method is used to retrieve the mac neighbour table for one MAC short address. It may be
14265 used to perform topology monitoring by the client.

14266 The method invocation parameter contains a mac_short_address.

14267 `data ::= long-unsigned`

14268 The response parameter includes the neighbour table for this mac_short_address.

14269 `data ::= array neighbour_table`

14270 where `mac_neighbour_table` is as defined in the *mac_neighbour_table* attribute of the present
14271 IC.

14272

14273

14274

14275 **5.12.6 G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 1)**

14276 **5.12.6.1 Overview**

14277 An instance of the “G3-PLC 6LoWPAN adaptation layer setup” IC holds the necessary
14278 parameters to set up and manage the G3-PLC 6LoWPAN Adaptation layer.

14279 These attributes influence the functional behaviour of an implementation. Implementations may
14280 allow changes to their values during normal running, i.e. even after the device start-up sequence
14281 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

G3-PLC 6LoWPAN adaptation layer setup	0...n	class_id = 92, version = 1			
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				

14282

14283 **5.12.6.2 Attribute description**14284 **5.12.6.2.1 logical_name**

14285 Identifies the “G3-PLC OFDM 6LoWPAN adaptation layer setup” object instance. See 6.2.28.

14286 **5.12.6.2.2 adp_max_hops**

14287 PIB attribute 0x0F: Defines the maximum number of hops to be used by the routing algorithm.

14288 **5.12.6.2.3 adp_weak_LQI_value**14289 PIB attribute 0x1A: The weak link value defines the LQI value below which a link to a neighbour
14290 is considered as a weak link. A value of 52 represents an SNR of 3 dB.14291 **5.12.6.2.4 adp_security_level**14292 PIB attribute 0x00: The minimum security level to be used for incoming and outgoing adaptation
14293 frames. Only values 0 (no ciphering) and 5 (ciphering with 32 bits integrity code) are supported.14294 **5.12.6.2.5 adp_prefix_table**

14295 PIB attribute 0x01: Contains the list of prefixes defined on this PAN.

14296 **5.12.6.2.6 adp_routing_configuration**14297 The routing configuration element specifies all parameters linked to the routing mechanism
14298 described in ITU-T G.9903:2014. The elements are specified in 9.4.1.2 of that Recommendation.

14299 NOTE 1 The Link cost calculation is provided in ITU-T G.9903:2014 Annex B.

```

14300         array routing_configuration
14301         routing_configuration::= structure
14302         {
14303             adp_net_traversal_time:           unsigned,
14304             adp_routing_table_entry_TTL:     long-unsigned,
14305             adp_Kr:                          unsigned,
14306             adp_Km:                          unsigned,
14307             adp_Kc:                          unsigned,
14308             adp_Kq:                          unsigned,
14309             adp_Kh:                          unsigned,
14310             adp_Krt:                         unsigned,
14311             adp_RREQ_retries:               unsigned,
14312             adp_RREQ_RERR_wait:              unsigned,
14313             adp_blacklist_table_entry_TTL:   long-unsigned,
14314             adp_unicast_RREQ_gen_enable:    boolean,
14315             adp_RLC_Enabled:                 boolean,

```

```
14316     adp_add_rev_link_cost:           unsigned  
14317 }
```

Where

14319 adp_net_traversal_time PIB attribute 0x11: Maximum time that a packet is
14320 expected to take to reach any node from any node in
14321 seconds.
14322 Range : 0-255
14323 Default value : 20

14324	<code>adp_routing_table_entry_TTL</code>	PIB attribute 0x12: Maximum time-to-live of a routing table entry (in minutes).
14325		Range : 0-65 535
14326		Default value : 60
14327		

14328 adp_Kr PIB attribute 0x13: A weight factor for the Robust Mode to calculate link cost.
14329
14330 Range : 0-31
14331 Default value : 0

14332 adp_Km PIB attribute 0x14: A weight factor for modulation to calculate link cost.
14333 Range : 0-31
14334 Default value : 0
14335

14336 adp_Kc PIB attribute 0x15: A weight factor for number of active tones to calculate link cost.
14337 Range : 0-31
14338 Default value : 0
14339

14340 adp_Kq PIB attribute 0x16: A weight factor for LQI to calculate route cost.
14341 Range : 0-50
14342 Default value : 10 for CENELEC A band / 40 for FCC band.
14343
14344

14345 adp_Kh PIB attribute 0x17: A weight factor for hop to calculate link cost.
14346
14347 Range : 0-31
14348 Default value : 4 for CENELEC A band / 2 for FCC
14349 band.

14350 adp_Krt PIB attribute 0x1B: A weight factor for the number of active routes in the routing table to calculate link cost.
14351
14352 Range : 0-31
14353 Default value : 0

14354	adp_RREQ_retries	PIB attribute 0x18: The number of RREQ re-transmission in case of RREP reception time out.
14355		Range : 0-255
14356		Default value : 0
14357		

14358	adp_RREQ_RERR_wait	PIB attribute 0x19: The number of seconds to wait between two consecutive RREQ – RERR generations.
14359		
14360		Range : 0-255
14361		Default value : 30

14362	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
14363		
14364		
14365		
14366	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
14367		
14368		
14369		
14370		
14371	adp_RLC_enabled	PIB attribute 0x09: Enable the sending of RLCREQ frame by the device. Default value : FALSE
14372		
14373		
14374	adp_add_rev_ink_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
14375		
14376		
14377		

14378 **5.12.6.2.7 adp_broadcast_log_table_entry_TTL**

14379 PIB attribute 0x02: Maximum time to live of an adpBroadcastLogTable entry (in minutes).

14380 **5.12.6.2.8 adp_routing_table**

14381 PIB attribute 0x0C: Contains the routing table.

```
14382         array routing_table
14383
14384         routing_table ::= structure
14385
14386         {
14387             destination_address: long-unsigned,
14388             next_hop_address: long-unsigned,
14389             route_cost: long-unsigned,
14390             hop_count: unsigned,
14391             weak_link_count: unsigned,
14392             valid_time: long-unsigned
14393         }
```

14395 NOTE 2 This table is actualized each time a route is built or updated (triggered by data traffic) and each time the
14396 TTL timer expires.

14397 Where

14398 **destination_address** Address of the destination.

14399 **next_hop_address** Address of the next hop on the route towards the destination.

14400 **route_cost** Cumulative link cost along the route towards the destination.

14401 **hop_count** Number of hops of the selected route to the destination.
14402 Range: 0-14

14403 NOTE 3 Practically the maximum allowed value is limited by adp_max_hops.

14404	weak_link_count	Number of weak links to destination. Range: 0-14
14405		
14406		NOTE 4 Practically the maximum allowed value is limited by adp_max_hops.
14407	valid_time	Remaining time in minutes until when this entry in the routing table is considered valid.

14409 **5.12.6.2.9 adp_context_information_table**

14410 PIB attribute 0x07: Contains the context information associated to each CID extension field.

```
14411     array context_information_table
14412
14413         context_information_table ::= structure
14414
14415         {
14416             CID:          bit-string,
14417             context_length: unsigned,
14418             context:        octet-string,
14419             C:              boolean,
14420             valid_lifetime: long-unsigned
14421         }
```

14422 Where

14423	CID	Corresponds to the 4-bit context information used for source and destination addresses (SCI, DCI). Range: 0x00-0x0F
-------	------------	--

14426	context_length	Indicates the length of the carried context (up to 128-bit contexts may be carried). Range: 0-128
-------	-----------------------	--

14429	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
-------	----------------	---

14433	C	Indicates if the context is valid for use in compression.
14434		FALSE: Only decompression is allowed,
14435		TRUE: Compression and decompression are allowed
14436		
14437		
14438		
14439		A context may be used for decompression purposes only.
14440		Moreover, recommendations made in RFC 6775 should be
14441		followed to take into account the propagation of the context to
14442		all nodes of the PAN.

14443	valid_lifetime	Remaining time in minutes during which the context information table is considered valid. It is updated upon reception of the advertised context.
14444		
14445		
14446		
14447		Range: 0-65 535

14448 **5.12.6.2.10 adp_blacklist_table**

14449 PIB attribute 0x1E: Contains the list of the blacklisted neighbours.

```

14450     array blacklisted_neighbour_set
14451
14452     blacklisted_neighbour_set ::= structure
14453
14454     {
14455         blacklisted_neighbour_address: long-unsigned,
14456         valid_time: long-unsigned
14457     }

```

14458 Where

14459 blacklisted_neighbour_address The 16-bit address of the blacklisted neighbour.

14460 valid_time Remaining time in minutes until which this entry in the
14461 blacklisted neighbour table is considered valid.

14462 **5.12.6.2.11 adp_broadcast_log_table**

14463 PIB attribute 0x0B: Contains the broadcast log table.

14464 NOTE 5 This table provides a list of the broadcast packets recently received by this device.

```

14465     array broadcast_log_table
14466
14467     broadcast_log_table ::= structure
14468
14469     {
14470         source_address: long-unsigned,
14471         sequence_number: unsigned,
14472         valid_time: long-unsigned
14473     }

```

14475 Where

14476 source_address The 16-bit source address of a broadcast packet. This is the
14477 address of the broadcast initiator.

14478 sequence_number The sequence number contained in the BC0 header.

14479 valid_time Remaining time in minutes until when this entry in the broadcast
14480 log table is considered valid.

14481 **5.12.6.2.12 adp_group_table**

14482 PIB attribute 0x0E: Contains the group addresses to which the device belongs.

```

14483     array group_address
14484
14485     group_address ::= long-unsigned
14486
14487     group_address Group address to which this node has been subscribed.

```

14488 **5.12.6.2.13 adp_max_join_wait_time**

14489 PIB attribute 0x20: Network join timeout in seconds for LBD.

14490 **5.12.6.2.14 adp_path_discovery_time**

14491 PIB attribute 0x21: Timeout for path discovery in seconds.

14492 5.12.6.2.15 adp_active_key_index

14493 PIB attribute 0x22: Index of the active GMK to be used for data transmission

14494 5.12.6.2.16 adp_metric_type

14495 PIB attribute 0x03: Metric Type to be used for routing purposes

14496 5.12.6.2.17 adp_coord_short_address

14497 PIB attribute 0x08: Defines the short address of the coordinator.

14498 5.12.6.2.18 adp_disable_default_routing

14499 PIB attribute 0xF0: If TRUE, the default routing (LOADng) is disabled. If FALSE, the default
14500 routing (LOADng) is enabled.

14501 5.12.6.2.19 adp_device_type

14502 PIB attribute 0x10: Defines the type of the device connected to the modem:

14503 enum:

- 14504 (0) PAN device,
14505 (1) PAN coordinator,
14506 (2) Not Defined

14507

14508

**14509 5.13 Previous versions of interface classes for setting up and managing DLMS/COSEM
14510 HS-PLC ISO/IEC 12139-1 neighbourhood networks**

14511 There are no previous versions to report.

14512

14513 5.14 Previous versions of ZigBee® setup classes

14514 There are no previous versions to report.

14515

14516

14517 **6 Relation to OBIS**

14518 **6.1 General**

14519 This Clause 6 specifies the use of COSEM interface objects to model various data items.

14520 It also specifies the logical names of the objects. The naming system is based on OBIS, the
14521 Object Identification System: each logical name is an OBIS code.

14522 OBIS codes are specified in the following clauses:

- 14523 • 6.2 specifies the use and the logical names of abstract COSEM objects, i.e. objects not
14524 related to an energy type;
- 14525 • 6.3 specifies the use and logical names for electricity related COSEM objects;
- 14526 • the detailed OBIS code allocations are specified in IEC 62056-6-1:**2021**.

14527 NOTE The use and the logical names of COSEM objects related to other media / energy types are under
14528 consideration.

- 14529 • the detailed OBIS code allocations – for all media / energy types – are specified in IEC
14530 62056-6-1:2021.
- 14531 •

14532 Unless otherwise specified the use of value group B shall be:

- 14533 • if just one object is instantiated, the value in value group B shall be 0;
- 14534 • if more than one object is instantiated in the same physical device, the value group B shall
14535 number the measurement or communication channels as appropriate, from 1...64. This is
14536 indicated by the letter "b".

14537 Unless otherwise specified the use of value group E shall be:

- 14538 • if just one object is instantiated, value in value group E shall be 0;
- 14539 • if more than one object is instantiated in the same physical device, the value group E shall
14540 number the instantiations from zero to the maximum value needed. This is indicated by the
14541 letter "e". For the values allocated, see IEC 62056-6-1:**2021**.

14542 All codes, which are not explicitly listed, but which are outside the manufacturer, utility or
14543 consortia specific ranges are reserved for future use.

14544 **6.2 Abstract COSEM objects**

14545 **6.2.1 Use of value group C**

14546 Table 49 shows the use of value group C for abstract objects in the COSEM context. See also
14547 IEC 62056-6-1:**2021**, Table 5.

14548 **Table 49 – Use of value group C for abstract objects in the COSEM context**

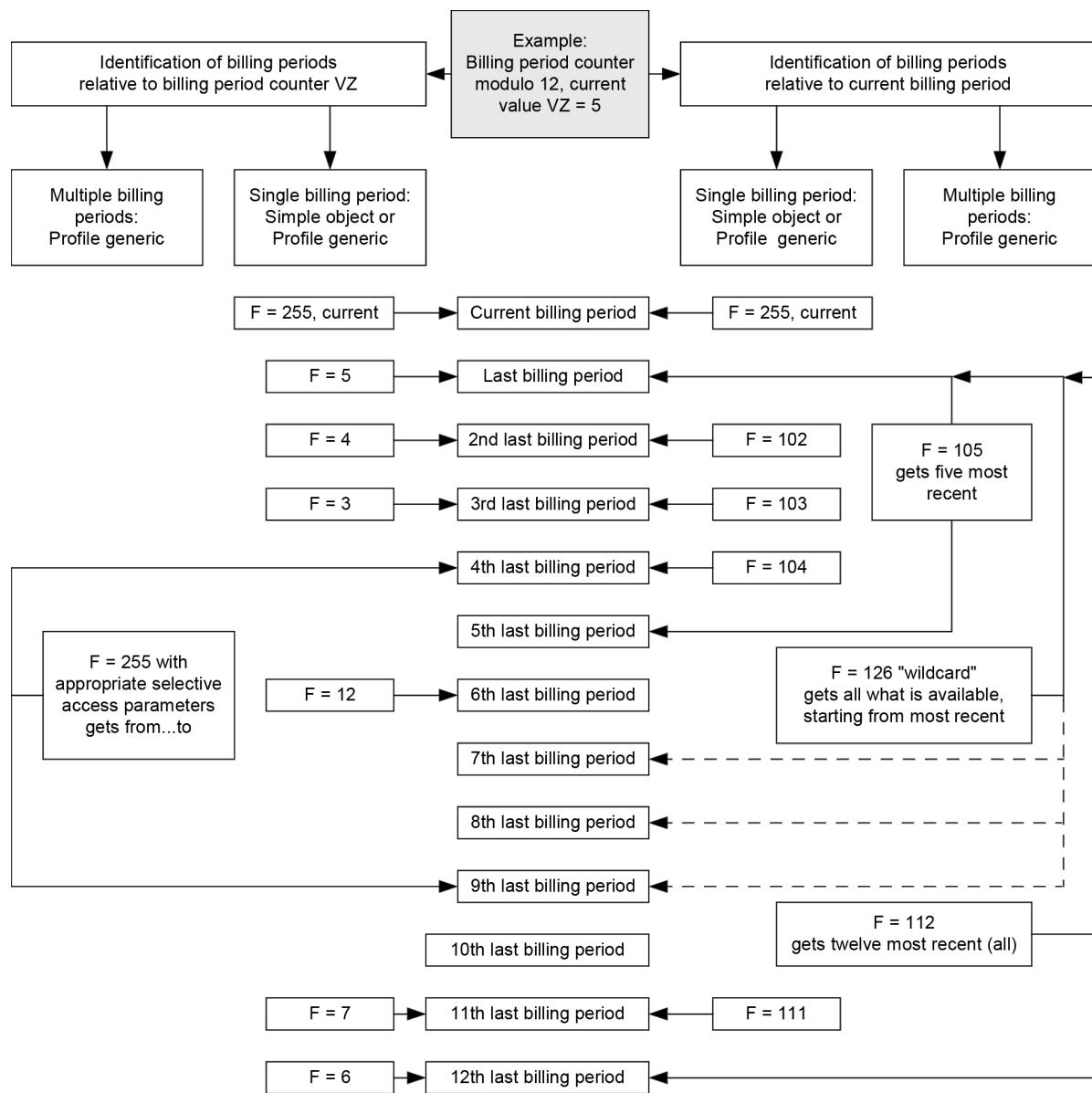
Value group C Abstract objects (A = 0)	
0	General purpose COSEM objects
1	Instances of IC "Clock"
2	Instances of IC "Modem configuration" and related IC-s
10	Instances of IC "Script table"

Value group C Abstract objects (A = 0)	
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"
18	Instances of IC "Array manager"
19	COSEM objects related to payment metering: "Account", "Credit", "Charge", "Token gateway"
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", "NTP setup", "LTE monitoring", "SCHC-LPWAN setup", "SCHC-LPWAN diagnostic", "SCHC-LoRaWAN setup", "SCHC-LoRaWAN diagnostic"
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)
32	Instances of IC "ISO/IEC 14908 identification", "ISO/IEC 14908 protocol setup", "ISO/IEC 14908 protocol setup", "ISO/IEC protocol status", "ISO/IEC 14908 diagnostic"
33	COSEM objects for data exchange using HS-PLC ISO/IEC 12139-1 networks
34	Instances of IC "Wi-SUN setup", "Wi-SUN diagnostic", "RPL diagnostic", "MPL diagnostic"
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" and "Data protection"
44	Instances of IC "Image transfer" and "Function control" objects
65	Instances of IC "Utility tables"
66	Instances of "Compact data"
128...199	Manufacturer specific COSEM related abstract objects
All other	Reserved

14549

14550 6.2.2 Data of historical billing periods

14551 COSEM provides three mechanisms to represent values of historical billing periods. This is
 14552 shown in Figure 31 and is described below:



14553

IEC

Figure 31 – 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...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...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...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...125 the two last, ...25 last values can be reached. F = 126 identifies an unspecified number of historical values;

- 14570 • when values of historical billing periods are represented by “Profile generic” objects, more
 14571 than one billing periods schemes may be used. The billing period scheme is identified by
 14572 the billing period counter object captured in the profile.

14573 **6.2.3 Billing period values / reset counter entries**

14574 These values are represented by instances of the IC “Data”.

14575 For billing period / reset counters and for number of available billing periods the data type shall
 14576 be *unsigned*, *long-unsigned* or *double-long-unsigned*. For time stamps of billing periods, the
 14577 data type shall be *double-long-unsigned* (in the case of UNIX time), *octet-string* or *date-time*
 14578 formatted as specified in 4.1.6.1.

14579 These objects may be related to energy type – see also 6.3.3 and IEC 62056-6-1:**2021**, Table
 14580 20 – and channels.

14581 When the values of historical periods are represented by “Profile generic” objects, the time
 14582 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: 2021 , Table 8.	1, Data ^a	0	b	0	1	e	255
^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.							

14583

14584 **6.2.4 Other abstract general purpose OBIS codes**

14585 Program entries shall be represented by instances of the IC “Data” with data type *unsigned*,
 14586 *long-unsigned* or *octet-string*.

14587 For identifying the firmware the following objects are available:

- Active firmware identifier objects hold the identifier of the currently active firmware;
- Active firmware version objects hold the version of the currently active firmware;

14590 NOTE *Firmware version* can be used to distinguish between different releases of a firmware identified by the
 14591 same *Firmware identifier*.

- Active firmware signature objects hold the digital signature of the currently active
 14593 firmware. The digital signature algorithm is not specified here.

14594 These three elements are inextricably linked to the currently active firmware.

14595 Firmware identifiers may be also energy type and channel related.

14596 Time entry values shall be represented by instances of IC “Data”, “Register” or “Extended
 14597 register” with the data type of the value attribute *octet-string*, formatted as *date-time* in 4.1.6.1.

14598 For detailed OBIS codes, see IEC 62056-6-1:2021, Table 8.

Abstract general purpose OBIS codes	IC	OBIS code					
		A	B	C	D	E	F
Program entries	1, Data ^a	0	b	0	2	e	255
Time entries		0	b	0	9	e	255
^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.							

14599

14600 6.2.5 Clock objects (class_id = 8)

14601 Instances of the IC “Clock” – see 4.5.1 – control the system clock of the physical device.

14602 “UNIX clock” objects are instances of the Interface class “Data”, with data type *double-long-unsigned*. They hold the number of seconds since 1970-01-01 00:00:00.

14604 NOTE: DLMS uses *double-long-unsigned* rather than the UNIX standard *double-long-signed* data type to extend the roll over date to 2106.

14606 “Microseconds clock” objects are instances of the IC “Data”, with data type *long64-unsigned*. They hold the number of microseconds since 1970-01-01 00:00:00.

14608 NOTE Previously, the name of these objects was “High resolution clock”.

14609 “Minutes clock” objects are instances of the IC “Data”, with data type *double-long-unsigned*. They hold the whole number of minutes since 1970-01-01 00:00:00 UTC.

14611 “Hours clock” objects are instances of the IC “Data”, with data type *double-long-unsigned*. They hold the whole number of the hours since 1970-01-01 00:00:00 UTC.

14613 “Days clock” objects are instances of the IC “Data”, with data type *long-unsigned*. They hold the whole number of the days since 1970-01-01 00:00:00 UTC.

14615 “Weeks clock” objects are instances of the IC “Data”, with data type *long-unsigned*. They hold the whole number of weeks since 1970-01-01 00:00:00 UTC.

14617 These objects are all inter-related in that it is anticipated that there is a single notion of time within the device.

Clock objects	IC	OBIS code					
		A	B	C	D	E	F
Clock	8, Clock	0	b	1	0	e	255
UNIX clock	1, Data ^a	0	b	1	1	e	255
Microseconds clock	1, Data ^a	0	b	1	2	e	255
Minutes clock	1, Data ^a	0	b	1	3	e	255
Hours clock	1, Data ^a	0	b	1	4	e	255
Days clock	1, Data ^a	0	b	1	5	e	255
Weeks clock	1, Data ^a	0	b	1	6	e	255

14619

14620 6.2.6 Modem configuration and related objects

14621 In this group, the following objects are available:

- Instances of the IC “Modem configuration” – see 4.7.4 – define and control the behaviour of the device regarding the communication through a modem;
- Instances of the IC “Auto connect” – see 4.7.6 – 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 4.7.5 – 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	b	2	0	0	255
Auto connect	29, Auto connect	0	b	2	1	0	255
Auto answer	28, Auto answer	0	b	2	2	0	255

14630

6.2.7 Script table objects (class_id = 9)

14632 Instances of the IC “Script table” – see 4.5.2 – control the behaviour of the device.

14633 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.
- Tarification* “Script table” objects define the entry point into tarification 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;
- Load profile control* “Script table” allow to change attributes of “Profile generic” objects e.g. to change the capture period and thus allow extended time control;
- M-Bus profile control* “Script table” allow to change attributes of M-Bus related “Profile generic” objects e.g. to change the capture period and thus allow extended time control;
- Function control* “Script table” objects allow making changes to “Function control” objects;
- Broadcast* “Script table” objects allow standardising utility wide the entry point into regularly needed functionality.

14657

Script table objects	IC	OBIS code					
		A	B	C	D	E	F
Global meter reset ^a Script table	9, Script table	0	b	10	0	0	255
MDI reset / End of billing period ^a Script table		0	b	10	0	1	255
Tarification Script table		0	b	10	0	100	255
Activate test mode ^a Script table		0	b	10	0	101	255
Activate normal mode ^a Script table		0	b	10	0	102	255
Set output signals Script table		0	b	10	0	103	255
Switch optical test output ^{b, c} Script table		0	b	10	0	104	255
Power quality measurement management Script table		0	b	10	0	105	255
Disconnect control Script table		0	b	10	0	106	255
Image activation Script table		0	b	10	0	107	255
Push Script table		0	b	10	0	108	255
Load profile control Script table		0	b	10	0	109	255
M-Bus profile control Script table		0	b	10	0	110	255
Function control Script table		0	b	10	0	111	255
Broadcast Script table		0	b	10	0	125	255
<p>^a The activation of these scripts is performed by calling the execute() method to the script identifier 1 of the corresponding script object.</p> <p>^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 62056-6-1:2021, Table 13. The default value of A is 1 (Electricity).</p> <p>EXAMPLE In the case of electricity meters, A = 1, default, execute (21) switches the test output to display the active power + of phase 1.</p> <p>^c The optical test output is also switched back to its default value when this script is activated.</p>							

14658

14659 6.2.8 Special days table objects (class_id = 11)

14660 Instances of the IC “Special days table” – see 4.5.4 – define and control the behaviour of the
 14661 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	b	11	0	e	255

14662

14663 6.2.9 Schedule objects (class_id = 10)

14664 Instances of the IC “Schedule” – see 4.5.3 – define and control the behaviour of the device in
 14665 a sequenced way.

Schedule objects	IC	OBIS code					
		A	B	C	D	E	F
Schedule	10, Schedule	0	b	12	0	e	255

14666

14667 **6.2.10 Activity calendar objects (class_id = 20)**

14668 Instances of the IC “Activity calendar” – see 4.5.5 – define and control the behaviour of the
14669 device in a calendar-based way.

Activity calendar objects	IC	OBIS code					
		A	B	C	D	E	F
Activity calendar	20, Activity calendar	0	b	13	0	e	255

14670

14671 **6.2.11 Register activation objects (class_id = 6)**

14672 Instances of the IC “Register activation” – see 4.3.5 – are used to handle different tariffication
14673 structures.

Register activation objects	IC	OBIS code					
		A	B	C	D	E	F
Register activation	6, Register activation	0	b	14	0	e	255

14674

14675 **6.2.12 Single action schedule objects (class_id = 22)**

14676 Instances of the IC “Single action schedule” – see 4.5.7 – control the behaviour of the device.
14677 Implementation specific instances should use values different from zero in value group D.

14678 Instances of Push “Single action schedule” objects activate scripts in Push “Script table” objects,
14679 which invoke the push method of the appropriate “Push setup” objects.

14680 Load profile control “Single action schedule” objects activate scripts in Load profile control
14681 “Script table” objects and thus allow extended time control.

14682 M-Bus profile control “Single action schedule” objects activate scripts in M-Bus profile control
14683 “Script table” objects and thus allow extended time control.

14684 Function control “Single action schedule” objects activate scripts in Function control “Script
14685 table” 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	b	15	0	0	255
Disconnect control Single action schedule		0	b	15	0	1	255
Image activation Single action schedule		0	b	15	0	2	255
Output control Single action schedule		0	b	15	0	3	255
Push Single action schedule		0	b	15	0	4	255
Load profile control Single action schedule		0	b	15	0	5	255
M-Bus profile control Single action schedule		0	b	15	0	6	255
Function control Single action schedule		0	b	15	0	7	255

14686

14687 **6.2.13 Register monitor objects (class_id = 21)**

14688 Instances of the IC “Register monitor” – see 4.5.6 – control the registerand alarm monitoring
 14689 function of the device. They define the value to be monitored, the set of thresholds to which the
 14690 value is compared, and the actions to be performed when a threshold is crossed.

14691 In general, the logical name(s) shown in the table below shall be used. See also 6.3.9 and
 14692 6.3.10.

14693 *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	b	16	0	e	255
Alarm monitor		0	b	16	1	0...9	255

14694

14695 **6.2.14 Parameter monitor objects (class_id = 65)**

14696 Instances of the IC “Parameter monitor” – see 5.4.1 – control the Parameter monitoring function
 14697 of the device. They define the list of parameters to be monitored and hold the identifier and the
 14698 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	b	16	2	e	255

14699

14700 **6.2.15 Limiter objects (class_id = 71)**

14701 Instances of the IC “Limiter” handle the monitoring of values in normal and emergency
 14702 conditions. See also 4.5.9.

Limiter objects	IC	OBIS code					
		A	B	C	D	E	F
Limiter	71, Limiter	0	b	17	0	e	255

14703

14704 **6.2.16 Array manager objects (class_id = 123)**

14705 Instances of the IC “Array manager” – see 4.4.11 – allow managing COSEM interface object
 14706 attributes of type *array*.

Array manager objects	IC	OBIS code					
		A	B	C	D	E	F
Array manager	123	0	b	18	0	e	255

14707

14708 **6.2.17 Payment metering related objects**

14709 Payment accounting can be applied to any commodity.

- 14710 An instance of the “Account” – see 4.6.2 – IC holds the summary information for a given contract
 14711 and lists the “Credit” and “Charge” objects used by that “Account”. If more than one “Account”
 14712 is for any reason required in a given context then field D should be other than 0.
- 14713 One or several instances of the “Credit” IC – see 4.6.3 – represent the different credit sources.
- 14714 One or several instances of the “Charge” IC – see 4.6.4 – represent the different charges
 14715 applicable.
- 14716 One or more instances of the “Token gateway” IC – see 4.6.5 – are available to enter tokens.
 14717 If only a single gateway is defined in a single “Account” then field E of the OBIS code shall be
 14718 zero. If more than one “Token gateway” object is for any reason required in a single “Account”
 14719 then field E should be other than 0.
- 14720 The “Account” is linked to its associated “Credit”, “Charge” and “Token gateway” objects by
 14721 use of the value group D and B field such that an “Account” with D=0 should be linked to a
 14722 “Token gateway” with D=40 and have a “Credit” objects with D=10 and “Charge” objects with
 14723 D=20. Whereas an “Account” with D=1 should have “token gateway” with D=41, “Credit” objects
 14724 with D=11 and “Charge” objects with D=21 etc. Multiple “Credit” and “Charge” objects are
 14725 identified using different values in the value group E field. See also Additional Notes there
 14726 describing the “Max credit_limit” and „Max vend limit” objects there.
- 14727 Instances of “Profile Generic” IC hold the history of the token credit and of the charge collections
 14728 with a “Parameter Monitor” interface class monitoring a value used to trigger capture.

14729

Payment metering related objects	IC	OBIS code					
		A	B	C	D	E	F
Account	111, Account	0	b	19	0..9	0	255
Credit	112, Credit	0	b	19	10..19	e	255
Charge	113, Charge	0	b	19	20..29	e	255
Token gateway	115, Token gateway	0	b	19	40...49	e	255
<i>Configurable limit objects</i>							
Max credit limit	01, Data	0	b	19	50...59	1	255
Max vend limit	01, Data	0	b	19	50...59	2	255

14730

14731 6.2.18 IEC local port setup objects (class_id = 19)

14732 These objects define and control the behaviour of local ports using the protocol specified in
 14733 IEC 62056-21:2002. See also 4.7.1.

IEC local port setup objects	IC	OBIS code					
		A	B	C	D	E	F
IEC optical port setup	19, IEC local port setup	0	b	20	0	0	255
IEC electrical port setup		0	b	20	0	1	255

14734

14735 6.2.19 Standard readout profile objects (class_id = 7)

14736 A set of objects is defined to carry the standard readout as it would appear with IEC 62056-
 14737 21:2002 (modes A to D). See also 4.3.6.

14738

Standard readout objects	IC	OBIS code					
		A	B	C	D	E	F
General local port readout	7, Profile generic	0	b	21	0	0	255
General display readout		0	b	21	0	1	255
Alternate display readout		0	b	21	0	2	255
Service display readout		0	b	21	0	3	255
List of configurable meter data		0	b	21	0	4	255
Additional readout profile 1		0	b	21	0	5	255
.....							
Additional readout profile n		0	b	21	0	N	255

14739

14740 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	b	21	0	e	255

14741

14742 **6.2.20 IEC HDLC setup objects (class_id = 23)**14743 Instances of the IC “IEC HDLC setup” – see 4.7.2 – hold the parameters of the HDLC based
14744 data link layer.

IEC HDLC setup objects	IC	OBIS code					
		A	B	C	D	E	F
IEC HDLC setup	23, IEC HDLC setup	0	b	22	0	0	255

14745

14746 **6.2.21 IEC twisted pair (1) setup objects (class_id = 24)**14747 An instance of the IC “IEC twisted pair (1) set up” IC – see 4.7.3 – stores the parameters
14748 necessary to manage a communication profile specified in IEC 62056-3-1:2013.

14749 An instance of the IC “MAC address set up” IC stores the Secondary Station Address ADS.

14750 An instance of the “Data” stores the Fatal Error register.

14751 Instances of the IC “Profile generic” IC instances allow the configuration of IEC 62056-3-1
14752 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 43, MAC address setup 1, Data 7, Profile generic	0	b	23	0	0	255
IEC twisted pair (1) MAC address setup		0	b	23	1	0	255
IEC twisted pair (1) Fatal Error register		0	b	23	2	0	255
IEC 62056-3-1 Short readout		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

14753

14754 For the parametrization of the IEC 62056-3-1 readout “Data” objects can be used.

Standard readout parametrization objects	IC	OBIS code					
		A	B	C	D	E	F
IEC 62056-3-1 readout parametrization	1, Data	0	b	23	3	e	255

14755

14756 **6.2.22 Objects related to data exchange over M-Bus**14757 The following objects are available to model and control data exchange using the M-Bus
14758 protocol specified in the EN 13757 series:

- 14759 • instances of the IC “M-Bus slave port setup” define and control the behaviour of M-Bus
14760 slave ports of a DLMS/COSEM device. See 4.8.2;
- 14761 • instances of the IC “M-Bus client” are used to configure DLMS/COSEM devices as M-Bus
14762 clients. There is one “M-Bus client” object for each M-Bus slave. Value group B identifies
14763 the M-Bus channels. See 4.8.3;
- 14764 • M-Bus value objects, instances of the IC “Extended register”, hold the values captured
14765 from M-Bus slave devices on the relevant channel. The link between the M-Bus client
14766 setup objects and the M-Bus value objects is provided by the channel number.
- 14767 • M-Bus “Profile generic” objects capture M-Bus value objects possibly along with other, not
14768 M-Bus specific objects;
- 14769 • M-Bus “Disconnect control” objects control disconnect devices of M-Bus devices (e.g. gas
14770 valves);
- 14771 • instances of the IC “Wireless mode Q” define and control the behaviour of the device
14772 regarding the communication parameters according to mode Q of EN 13757-5:2015. A
14773 node having more than one network address, i.e. a multi-homed node, will have multiple
14774 objects of these types. See 4.8.4;
- 14775 • M-Bus control log objects are instances of the IC “Profile generic”. They log the changes
14776 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 4.8.5;
- instances of the IC ”DLMS/COSEM server M-Bus port setup” are used in DLMS/COSEM servers hosted by M-Bus slave devices, using the DLMS/COSEM wired or wireless M-Bus communication profile. See 4.8.6;
- instances of the IC “M-Bus diagnostic” hold information related to the operation of the M-Bus network. See 4.8.7.

14785

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
DLMS/COSEM server M-Bus port setup	76, DLMS/COSEM server M-Bus port setup	0	b	24	8	e ^b	255
M-Bus diagnostic	77, M-Bus diagnostic	0	b	24	9	e ^b	255

^a “e” is equal to the index of the captured value in accordance to index of capture_definition_element in the *capture_definition* 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.

14786

6.2.23 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 4.9.1 – 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 4.9.2 – 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 4.9.3 – 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 4.9.4. – 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 4.9.5 – handle all information related to the setup of the PPP data link layer of the Internet based communication profiles;

- 14805 • Instances of the IC “GPRS modem setup” – see 4.7.7 – handle all information related to
14806 the setup of the GPRS modem;
- 14807 • Instances of the IC “SMTP setup” – see 4.9.6 – handle all information related to the setup
14808 of the SMTP service.

14809 NOTE The following objects have internet related OBIS codes, although they are not strictly related to use over the
14810 internet only.

- 14811 • Instances of the IC “GSM diagnostic” – see 5.6.9 – handle all diagnostic information
14812 related to the GSM/GPRS network.
- 14813 • Instances of the IC “Push setup” – see 4.4.8 – handle all information about the data to be
14814 pushed, the push destination and the method the data should be pushed;
- 14815 • Instances of the IC "NTP setup" – 4.9.7 – see handle all information related to the setup of
14816 the NTP time synchronisation service;
- 14817 • Instances of the IC “LTE monitoring” – see 4.7.9 – allow monitoring LTE modems.
- 14818 • Instances of the IC “SCHC-LPWAN setup” – see 4.16.2.1 – handle all information related
14819 to the setup of the SCHC-LPWAN layer;

14820 Instances of the IC “SCHC-LPWAN diagnostic” – see 4.16.2.2 – handle all diagnostic
 14821 information related to the SCHC-LPWAN network.

14822

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
<i>Reserved for FTP setup</i>							
Push setup	40, Push setup	0	b	25	9	0	255
NTP setup	100, NTP setup	0	b	25	10	0	255
LTE monitoring	151, LTE monitoring	0	b	25	11	0	255
SCHC-LPWAN setup	126, SCHC-LPWAN setup	0	b	25	12	0	255
SCHC-LPWAN diagnostic	127, SCHC-LPWAN diagnostic	0	b	25	13	0	255
SCHC-LoRaWAN setup	128, SCHC – LoRaWAN setup	0	b	25	14	0	255
SCHC-LoRaWAN diagnostic	129, SCHC – LoRaWAN diagnostic	0	b	25	15	0	255

14823

6.2.24 Objects to set up Push Setup (class_id = 40)

14824 Instances of the IC “Push setup” – see 4.4.8 – handle all information about the data to be
 14825 pushed, the push destination and the method by which the data should be pushed.
 14826

Push Setup	IC	OBIS code					
		A	B	C	D	E	F
Push setup	40, Push setup	0	b	25	9	0	255

14827

6.2.25 Objects for setting up data exchange using S-FSK PLC

14828 In this group, the following objects are available:

- 14829 • Instances of the IC “S-FSK Phy&MAC setup” – see 4.10.3 – handle all information related to setting up the PLC S-FSK lower layer profile specified in IEC 61334-5-1:2001.
- 14830 • Instances of the IC “S-FSK Active initiator” – see 4.10.4 – handle all information related to the active initiator in the PLC S-FSK lower layer profile specified in IEC 61334-5-1:2001.
- 14831 • Instances of the IC “S-FSK MAC synchronization timeouts” – see 4.10.5 – 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.
- 14832 • Instances of the IC “S-FSK MAC counters” – see 4.10.6 – 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.
- 14833 • Instances of the IC “IEC 61334-4-32 LLC setup” – see 4.10.7 – handle all information related to the LLC layer specified in IEC 61334-4-32:1996.

- 14842 • Instances of the IC “S-FSK Reporting system list” – see 4.10.8 – hold information on
 14843 reporting systems in the PLC S-FSK lower layer profile specified in IEC 61334-5-1:2001.

14844

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

14845

14846 6.2.26 Objects for setting up the ISO/IEC 8802-2 LLC layer

14847 In this group, the following objects are available:

- 14848 • Instances of the IC “ISO/IEC 8802-2 LLC Type 1 setup” – see 4.11.2 – handle all
 14849 information related to the LLC layer specified in IEC 62056-8-8:2020, Electricity **metering**
 14850 **data exchange - The DLMS/COSEM suite - Part 8-8: Communication profile for ISO/IEC**
 14851 **14908 series networks**
- 14852 • ISO/IEC 8802-2:1998 in Type 1 operation.
- 14853 • Instances of the IC “ISO/IEC 8802-2 LLC Type 2 setup” – see 4.11.3 – handle all
 14854 information related to the LLC layer specified in IEC 62056-8-8:2020, Electricity **metering**
 14855 **data exchange - The DLMS/COSEM suite - Part 8-8: Communication profile for ISO/IEC**
 14856 **14908 series networks**
- 14857 • ISO/IEC 8802-2:1998 in Type 2 operation.
- 14858 • Instances of the IC “ISO/IEC 8802-2 LLC Type 3 setup” – see 4.11.4 – handle all
 14859 information related to the LLC layer specified in IEC 62056-8-8:2020, Electricity **metering**
 14860 **data exchange - The DLMS/COSEM suite - Part 8-8: Communication profile for ISO/IEC**
 14861 **14908 series networks**
- 14862 • ISO/IEC 8802-2:1998 in Type 3 operation.

14863

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	b	27	0	0	255
ISO/IEC 8802-2 LLC Type 2 setup	58, ISO/IEC 8802-2 LLC Type 2 setup	0	b	27	1	0	255
ISO/IEC 8802-2 LLC Type 3 setup	59, ISO/IEC 8802-2 LLC Type 3 setup	0	b	27	2	0	255

14864

14865 6.2.27 Objects for data exchange using narrowband OFDM PLC for PRIME networks

14866 For setting up and managing data exchange using narrowband OFDM PLC for PRIME networks
 14867 one instance of each following classes shall be implemented for each interface:

- an instance of the 61334-4-32 LLC SSCS setup – see 4.12.3 – holds the addresses related to the CL_432 layer;
- an instance of the IC “PRIME NB OFDM PLC Physical layer counters” – see 4.12.5 – stores counters related to the physical layers exchanges;
- an instance of the IC “PRIME NB OFDM PLC MAC setup” – see 4.12.6 – 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 4.12.7 – 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 4.12.8 – 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 4.12.9 – 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 4.12.10;
- an instance of the IC “PRIME NB OFDM PLC Application identification” – see 4.12.11 – holds identification information related to administration and maintenance of PRIME NB OFDM PLC devices.

14887

Objects for data exchange using PRIME NB 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	b	28	0	0	255
PRIME NB OFDM PLC Physical layer counters	81, PRIME NB OFDM PLC Physical layer counters	0	b	28	1	0	255
PRIME NB OFDM PLC MAC setup	82, PRIME NB OFDM PLC MAC setup	0	b	28	2	0	255
PRIME NB OFDM PLC MAC functional parameters	83, PRIME NB OFDM PLC MAC functional parameters	0	b	28	3	0	255
PRIME NB OFDM PLC MAC counters	84, PRIME NB OFDM PLC MAC counters	0	b	28	4	0	255
PRIME NB OFDM PLC MAC network administration data	85, PRIME NB OFDM PLC MAC network administration data	0	b	28	5	0	255
PRIME NB OFDM PLC MAC address setup	43, MAC address setup	0	b	28	6	0	255
PRIME NB OFDM PLC Application identification	86, PRIME NB OFDM PLC Application identification	0	b	28	7	0	255

14888

6.2.28 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 4.13.3 – to store counters related to the MAC layer exchanges;
- an instance of the IC “G3-PLC MAC setup” – see 4.13.4 – 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 4.13.5 – 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

14898

6.2.29 ZigBee® setup objects

14900 The following objects are available for setting up and managing a ZigBee® network; see also
14901 4.15.

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

14902

6.2.30 Objects for setting up and managing data exchange using ISO/IEC 14908 PLC networks

14903 For setting up and managing data exchange using ISO/IEC 14908 PLC networks at least one
14904 instance of each of the following interface classes shall be implemented, see also 4.16

Objects to set up and manage ISO/IEC 14908 PLC netwrks	IC	OBIS identification					
		A	B	C	D	E	F
ISO/IEC 14908 identification	130, ISO/IEC 14908 identification	0	b	32	0	0	255
ISO/IEC 14908 protocol setup	131, ISO/IEC 14908 protocol setup	0	b	32	1	0	255
ISO/IEC 14908 protocol status	132, ISO/IEC 14908 protocol status	0	b	32	2	0	255
ISO/IEC 14908 diagnostic	133, ISO/IEC 14908 diagnostic	0	b	32	3	0	255

14907

6.2.31 Objects for data exchange using HS-PLC ISO/IEC 12139-1 ISO/EC 12139-1 networks

14910 For setting up and managing data exchange using HS-PLC ISO/IEC 12139-1 networks one
14911 instance of each following classes shall be implemented for each interface:

- 14912 • an instance of the IC “HS-PLC ISO/IEC 12139-1 MAC setup” – see 4.14.2 – holds the
14913 necessary parameters for setting up the MAC layer;
- 14914 • an instance of the IC “HS-PLC ISO/IEC 12139-1 CPAS setup” – see 4.14.3 – holds the
14915 necessary parameters for setting up the CPAS;
- 14916 • an instance of the IC “HS-PLC ISO/IEC 12139-1 IP SSAS setup” – see 4.14.4 – holds the
14917 necessary parameters for setting up the IP SSAS;
- 14918 • an instance of the IC “HS-PLC ISO/IEC 12139-1 HDLC SSAS setup” – see 4.14.5 – holds
14919 the necessary parameters for setting up the HDLC SSAS.

Objects for data exchange using HS-PLC ISO/IEC 12139-1	IC	OBIS code					
		A	B	C	D	E	F
HS-PLC ISO/IEC 12139-1 MAC setup	140, HS-PLC ISO/IEC 12139-1 MAC setup	0	b	33	0	0	255
HS-PLC ISO/IEC 12139-1 CPAS setup	141, HS-PLC ISO/IEC 12139-1 CPAS setup	0	b	33	1	0	255
HS-PLC ISO/IEC 12139-1 IP SSAS setup	142, HS-PLC ISO/IEC 12139-1 IP SSAS setup	0	b	33	2	0	255
HS-PLC ISO/IEC 12139-1 HDLC SSAS setup	143, HS-PLC ISO/IEC 12139-1 HDLC SSAS setup	0	b	33	3	0	255

14920

6.2.32 Objects for data exchange using Wi-SUN networks

For setting up and managing data exchange using Wi-SUN networks one instance of each following classes shall be implemented for each interface:

- An instance of the IC “IPv6 setup” – see 4.9.3 – 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;
- An instance of the IC “Wi-SUN setup” – see 4.18.1 – handle all information related to the setup of the SMTP service.
- An instance of the IC “Interface diagnostic” – see 4.18.2 – handle all diagnostic information related to the network.
- An instance of the IC “RPL diagnostic” – see 4.18.3 – handle all diagnostic information related to RPL.
- An instance of the IC “MPL diagnostic” – see 4.18.4 – handle all diagnostic information related to MPL.

14936

Objects to set up data exchange over the Wi-SUN FAN	IC	OBIS code					
		A	B	C	D	E	F
Wi-SUN setup	95, Wi-SUN setup	0	b	34	0	0	255
Wi-SUN diagnostic	96, Wi-SUN diagnostic	0	b	34	1	0	255
RPL diagnostic	97, RPL diagnostic	0	b	34	2	0	255
MPL diagnostic	98, MPL diagnostic	0	b	34	3	0	255

14937

6.2.33 Association objects (class_id = 12, 15)

A series of Association SN / LN objects – see 4.4.3, 4.4.4 – 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

14941

14942 **6.2.34 SAP assignment object (class_id = 17)**

14943 An instance of the IC “SAP assignment” – see 4.4.5 – holds information about the addresses
14944 (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

14945

14946 **6.2.35 COSEM logical device name object**

14947 Each COSEM logical device shall be identified by its Logical Device Name, unique worldwide.
14948 See 4.1.8.2. It is held by the *value* attribute of a “Data” object, with data type *octet-string* or
14949 *visible-string*. For short name referencing, the *base_name* of the object is fixed. See 4.1.3.

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.

14950

14951 **6.2.36 Information security related objects**

14952 Instances of the IC “Security setup” – see 4.4.7 – are used to set up the message security
14953 features. For each Association object, there is one Security setup object managing security
14954 within that AA. See 5.3.2 and 5.3.3. 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

14955

14956 Invocation counter objects hold the invocation counter element of the initialization vector. They
14957 are instances of the IC “Data”. The value in value group B identifies the communication channel.

14958 NOTE The same client may use different communication channels e.g. a remote port and a local port. The invocation
14959 counter on the different channels may be different.

14960 The value in value group E shall be the same as in the logical name of the corresponding
14961 “Security setup” object.

Invocation counter objects	IC	OBIS code					
		A	B	C	D	E	F
Invocation counter	1, Data ^a	0	b	43	1	e	255

NOTE In earlier versions of this standard, these objects were called Frame counter objects.

^a If the IC “Data” is not available, “Register” (with scaler = 0, unit = 255) may be used.

14962

14963 Instances of the IC “Data protection” – see 4.4.9 – are used to apply / remove protection on
14964 COSEM data, i.e. sets of attributes values, method invocation and return parameters. Value
14965 group E numbers the instances.

Data protection objects	IC	OBIS code					
		A	B	C	D	E	F
Data protection	30, Data protection	0	0	43	2	e	255

14966

6.2.37 Image transfer objects (class_id = 18)

Instances of the IC “Image transfer” – see 4.4.6 – 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

14969

6.2.38 Function control objects (class_id = 122)

Instances of the IC “Function control” – see 4.4.10 – allow enabling and disabling functions in the server.

Function control related objects	IC	OBIS code					
		A	B	C	D	E	F
Function control	122, Function control	0	0	44	1	e	255

14973

6.2.39 Communication port protection objects (class_id = 124)

Instances of the “Communication port protection” IC – see 4.4.12 – control the communication port protection mechanism.

Communication port protection objects	IC	OBIS code					
		A	B	C	D	E	F
Communication port protection objects	124	0	b	44	2	e	255

14977

6.2.40 Utility table objects (class_id = 26)

Instances of the IC “Utility tables” – see 4.3.7 – 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.

14987

Utility table objects	IC	OBIS code					
		A	B	C	D	E	F
Standard tables 0-127	26, Utility tables	0	b	65	0	e	255
Standard tables 128-255		0	b	65	1	e	255
...							
Standard tables 1920-2047		0	b	65	15	e	255
Manufacturer tables 0-127		0	b	65	16	e	255
Manufacturer tables 128-255		0	b	65	17	e	255
...							
Manufacturer tables 1920-2047		0	b	65	31	e	255
Std pending tables 0-127		0	b	65	32	e	255
Std pending tables 128-255		0	b	65	33	e	255
...							
Std pending tables 1920-2047		0	b	65	47	e	255
Mfg pending tables 0-127		0	b	65	48	e	255
Mfg pending tables 128-255		0	b	65	49	e	255
...							
Mfg pending tables 1920-2047		0	b	65	63	e	255

14988

6.2.41 Compact data objects (class_id = 62)

“Compact data” objects – see 4.3.10 – store data and metadata separated, thus they allow reducing overhead.

Compact data objects	IC	OBIS code					
		A	B	C	D	E	F
Compact data	62, Compact data	0	b	66	0	e	255

14992

6.2.42 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 4.3.8 – can be used. See also IEC 62056-6-1:2021, Table 8.

Device ID objects	IC	OBIS code					
		A	B	C	D	E	F
Device ID 1...10 object (manufacturing number)	1, Data ^a	0	b	96	1	0...9	255
Device ID-s object	7, Profile generic	0	b	96	1	255	255
Device ID-s object	61, Register table	0	b	96	1	255	255

^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

15002

15003 **6.2.43 Metering point ID objects**

15004 One object is available to store a media type independent metering point ID. It is held by the
 15005 *value* attribute of a “Data” object, with data type *double-long-unsigned*, *octet-string*, *visible-*
 15006 *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	b	96	1	10	255

^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

15007

15008 **6.2.44 Parameter changes and calibration objects**

15009 A set of simple COSEM objects describes the history of the configuration of the device. All
 15010 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 62056-6-1:2021, Table 8.	1, Data ^a	0	b	96	2	e	255

^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

15011

15012 **6.2.45 I/O control signal objects**

15013 A series of objects are available to define and control the status of I/O lines of the physical
 15014 metering equipment.

15015 The status is held by the *value* attribute of a “Data” object, with data type *octet-string* or *bit-*
 15016 *string*. Alternatively, the status is held by a “Status mapping” object, see 4.3.9, which holds both
 15017 the status word and the mapping of its bits to the reference table. If there are several I/O control
 15018 status objects used, it is allowed to combine them into an instance of the IC “Profile generic” or
 15019 “Register table”, using the OBIS code of the global state of I/O control signals object. See also
 15020 IEC 62056-6-1:2021, Table 8.

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	b	96	3	0...4	255
I/O control signal objects, contents mapped to a reference table	63, Status mapping	0	b	96	3	0...4	255
I/O control signal objects, global	7, Profile generic or 61, Register table	0	b	96	3	0	255

^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

15021

15022 **6.2.46 Disconnect control objects (class_id = 70)**

15023 Instances of the IC “Disconnect control” – see 4.5.8 – manage internal or external disconnect
 15024 units (e.g. electricity breaker, gas valve) in order to connect or disconnect – partly or entirely –
 15025 the premises of the consumer to / from the supply. See also 6.2.22.

Disconnect control objects	IC	OBIS code					
		A	B	C	D	E	F
Disconnect control	70, Disconnect control	0	b	96	3	10	255

15026

6.2.47 Arbitrator objects (class_id = 68)

Instances of the IC “Arbitrator” – see 4.5.12 – are used

Arbitrator Objects	IC	OBIS code					
		A	B	C	D	E	F
General-purpose Arbitrator	68, Arbitrator	0	b	96	3	20... 29	255

15029

6.2.48 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 4.3.9, 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 IEC 62056-6-1:2021, Table 8.

Internal control signals objects	IC	OBIS code					
		A	B	C	D	E	F
Internal control signals, contents manufacturer specific	1, Data ^a	0	b	96	4	0...4	255
Internal control signals, contents mapped to a reference table	63, Status mapping	0	b	96	4	0...4	255
Internal control signals, global	7, Profile generic or 61, Register table	0	b	96	4	0	255

^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

15039

6.2.49 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 4.3.9, 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 IEC 62056-6-1:2021, Table 8.

Internal operating status objects	IC	OBIS code					
		A	B	C	D	E	F
Internal operating status objects, contents manufacturer specific	1, Data ^a	0	b	96	5	0...4	255
Internal operating status objects, contents mapped to a reference table	63, Status mapping	0	b	96	5	0...4	255
Internal operating status objects, global	7, Profile generic or 61, Register table	0	b	96	5	0	255

^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

15049

15050 Internal operating status objects can also be related to an energy type. See 6.3.7.

15051 **6.2.50 Battery entries objects**15052 A series of objects are available for holding information relative to the battery of the device.
15053 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 62056-6-1:2021, Table 8.	1, Data, 3, Register or 4, Extended register	0	b	96	6	0...6	255

15054

15055 **6.2.51 Power failure monitoring objects**

15056 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 62056-6-1:2021, Table 8.	1, Data, 3, Register or 4, Extended register	0	b	96	7	0...21	255

15072

15073 These objects may be collected in a “Power failure event log” object. See IEC 62056-6-1:2021,
15074 Table 23.

15075 **6.2.52 Operating time objects**

15076 A series of objects are available for holding the cumulated operating time and the various tariff
 15077 registers of the device. These objects are instances of the IC “Data”, “Register” or “Extended
 15078 register”. The data type shall be *unsigned*, *long-unsigned* or *double-long-unsigned* with
 15079 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 62056-6-1:2021, Table 8.	1, Data, 3, Register or 4, Extended register	0	b	96	8	0...63	255

15080

15081 **6.2.53 Environment related parameters objects**

15082 A series of objects are available to store environmental related parameters. They are held by
 15083 the *value* attribute of instances of the IC “Register” or “Extended register”, with appropriate data
 15084 types.

Environment related parameters objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2021, Table 8.	3, Register or 4, Extended register	0	b	96	9	0...2	255

15085

15086 **6.2.54 Status register objects**

15087 A series of objects are available to hold statuses that can be captured in load profiles. See also
 15088 see IEC 62056-6-1:2021, Table 8.

Status register objects	IC	OBIS code					
		A	B	C	D	E	F
Status register, contents manufacturer specific	1, Data ^a	0	b	96	10	1...10	255
Status register, contents mapped to reference table	63, Status mapping	0	b	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.

15089

15090 The status register is held by the *value* attribute of a “Data” object, with data type *bit-string*,
 15091 *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-unsigned* or *octet-string*. It carries
 15092 binary information from a bitmap. Its contents is not specified.

15093 Alternatively, the status register may be held by the *status_word* attribute of a “Status mapping”
 15094 object, see 4.3.9. The *mapping_table* attribute holds mapping information between the bits of
 15095 the status word and entries of a reference table.

15096 **6.2.55 Event code objects**

15097 In the meter or in its environment, various events may be generated. A series of objects are
 15098 available to hold an identifier of a most recent event (event code). Different instances of event
 15099 code objects may be captured in different instances of event logs; see 6.2.67.

15100 NOTE The definition of event identifiers is out of the Scope of this document.

Event code objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2021, Table 8.	1, Data, 3, Register, or 4, Extended register	0	b	96	11	0...99	255

15101

15102 Events may also set flags in error registers and alarm registers. See also 6.2.65.

15103 **6.2.56 Communication port log parameter objects**15104 A series of objects are available to hold various communication log parameters. They are
15105 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 62056-6-1:2021, Table 8.	1, Data, 3, Register, or 4, Extended register	0	b	96	12	0...6	255

15106

15107 **6.2.57 Consumer message objects**15108 A series of objects are available to store information sent to the energy end-user. The
15109 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 62056-6-1:2021, Table 8.	1, Data, 3, Register, or 4, Extended register	0	b	96	13	0, 1	255

15110

15111 **6.2.58 Currently active tariff objects**15112 A series of objects are available to hold the identifier of the currently active tariff. They carry
15113 the same information as the *active_mask* attribute of the corresponding “Register activation”
15114 object.

Currently active tariff objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2021, Table 8.	1, Data, 3, Register, or 4, Extended register	0	b	96	14	0...15	255

15115

15116 **6.2.59 Event counter objects**15117 A series of objects are available to count events. The number of the events is held by the value
15118 attribute.

Event counter objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2021, Table 8.	1, Data, 3, Register, or 4, Extended register	0	b	96	15	0...99	255

15119

15120 **6.2.60 Profile entry digital signature objects**

15121 Instances of “Data”, “Register” or “Extended register” objects hold digital signatures of “Profile
 15122 generic” object buffer entries. If the *capture_object* attribute of a “Profile generic” object
 15123 contains a reference to a “Profile entry digital signature” object, then the digital signature is
 15124 calculated and captured together with the other attribute values.

15125 The security context is determined by the “Security setup” object which is visible in the
 15126 same AA (*object_list*).

15127 NOTE The digital signature may be generated when the entry is captured or “on the fly”, when an entry is accessed.

Profile entry digital signature objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2021, Table 8.	1, Data, 3, Register, or 4, Extended register	0	b	96	16	0...9	255

15128

15129 **6.2.61 Meter tamper event related objects**

15130 A series of objects are available to register characteristics of various meter tamper events.
 15131 These objects are instances of the IC “Data”, “Register” or “Extended register”. The data type
 15132 shall be *unsigned*, *long-unsigned* or *double-long-unsigned* with appropriate scaler and unit. For
 15133 time stamps, the data type shall be *double-long-unsigned* (in the case of UNIX time), *octet-string*
 15134 or *date-time* formatted as specified in 4.1.6.1.

- 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.

15144 The method of detecting the various tampers is out of the Scope of this Technical Specification.

Meter tamper event related objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2021, Table 8.	1, Data, 3, Register, or 4, Extended register	0	b	96	20	e	255

15145

15146 **6.2.62 Profile entry counter objects**

15147 A series of objects are available to keep track of the total number of entries captured in a
 15148 “Profile generic” object. These objects are instances of the IC “Data”, “Register” or “Extended
 15149 register”. The data type shall be *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*
 15150 with appropriate scaler and unit. If the *capture_objects* attribute of a “Profile generic”
 15151 object contains a reference to a “Profile entry counter” object, then the counter is monotonically
 15152 increased by one and captured together with the other attribute values when the *capture* method
 15153 is invoked or when auto capture takes place.

15154 If a “Profile entry counter” object reaches its maximum value, it will wrap-around and the count
 15155 will re-start at zero.

Profile entry counter objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2021, Table 8.	1, Data, 3, Register, or 4, Extended register	0	b	96	17	0... 127	255

15156

15157 6.2.63 Meter tamper event related objects

15158 A series of objects are available to register characteristics of various meter tamper events.
 15159 These objects are instances of the IC “Data”, Register” or “Extended register”. The data type
 15160 shall be *unsigned*, *long-unsigned* or *double-long-unsigned* with appropriate scaler and unit. For
 15161 time stamps, the data type shall be *double-long-unsigned* (in the case of UNIX time), *octet-
 15162 string* or *date-time* formatted as specified in 4.1.6.1.

- 15163 • Meter open events are related to cases when the meter case is open;
- 15164 • Terminal cover open events are related to cases when a terminal cover is removed (open);
- 15165 • Tilt events are related to cases when the meter is not in its normal operation position;
- 15166 • Strong DC magnetic field events are related to cases when the presence of a strong DC
 15167 magnetic field is detected;
- 15168 • Metrology tamper events are related to cases when an anomaly in the operation of the
 15169 metrology is detected due to a perceived tamper;
- 15170 • Communication tamper events are related to cases when an anomaly in the operation of
 15171 the communication interfaces is detected due to a perceived tamper.

15172 The method of detecting the various tampers is out of the Scope of this document.

Meter tamper event related objects	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2021, Table 8.	1, Data, 3, Register, or 4, Extended register	0	b	96	20	e	255

15173

15174 6.2.64 Error register objects

15175 A series of objects are used to communicate error indications of the device. The different error
 15176 registers are held by the *value* attribute of “Data” objects, with data type or *bit-string*, *octet-
 15177 string*, *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*.

15178 The individual bits of the error register may be set and cleared by a pre-defined selection of
 15179 events – see 6.2.55. Depending on the type of the error, some errors may clear themselves
 15180 when the reason setting the error flag disappears.

15181 If more than one of those objects is used, it is allowed to combine them into one instance of the
 15182 IC “Profile generic”. In this case, the captured objects are the *value* attributes “Data” objects,
 15183 the capture period is 1 to have just actual values, the sort method is FIFO, the profile entries
 15184 are limited to 1. Alternatively, an instance of the IC “Register table” can be used.

15185 Error register objects can also be related to an energy type and to a channel. See IEC 62056-
 15186 6-1:2021, 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.							

15187

15188 6.2.65 Alarm register, Alarm filter and Alarm descriptor objects

15189 A number of objects are available to hold alarm registers. The different alarm registers are held
 15190 by the value attribute of "Data" objects, with data type *bit-string*, *octet-string*, *unsigned*, *long-*
 15191 *unsigned*, *double-long-unsigned* or *long64-unsigned*. When selected events occur, they set the
 15192 corresponding flag and the device may raise an alarm. Depending on the type of alarm, some
 15193 alarms may clear themselves when the reason setting the alarm flag disappears.

15194 If more than one of those objects is used, it is also allowed to combine them into one instance
 15195 of the IC "Profile generic". In this case, the captured objects are the *value* attributes of "Data"
 15196 objects, the capture period is 1 to have just actual values, the sort method is FIFO, and the
 15197 profile entries are limited to 1. Alternatively, an instance of the IC "Register table" can be used.

15198 *Alarm filter* objects are available to define if an event is to be handled as an alarm when it
 15199 appears. The different alarm filters are held by the value attribute of "Data" objects, with data
 15200 type *bit-string*, *octet-string*, *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*.
 15201 The bit mask has the same structure as the corresponding alarm register object. If a bit in the
 15202 alarm filter is set, then the corresponding alarm is enabled, otherwise it is disabled. *Alarm filter*
 15203 objects act on *Alarm register* and *Alarm descriptor* objects the same way.

15204 *Alarm descriptor* objects are available to persistently hold the occurrence of alarms. The
 15205 different alarm descriptors are of the same type as the corresponding *Alarm register*. When a
 15206 selected event occurs, the corresponding flag is set in the *Alarm register* as well as in the *Alarm*
 15207 *descriptor* objects. An alarm descriptor flag remains set even if the corresponding alarm
 15208 condition has disappeared. Alarm descriptor flags do not reset themselves; they can be reset
 15209 by writing the value attribute only.

15210 NOTE The alarm conditions, the structure of the *Alarm register* / *Alarm filter* / *Alarm descriptor* objects are subject
 15211 to a project specific companion specification.

15212 15213 Alarm register, Alarm filter and Alarm descriptor objects	15214 15215 IC	OBIS code					
		A	B	C	D	E	F
15216 15217 Alarm register objects 1...10	1, Data ^a	0	b	97	98	0...9	255
15218 15219 Alarm register profile object	7, Profile generic	0	b	97	98	255	255
15220 15221 Alarm register table object	61, Register table	0	b	97	98	255	255
15222 15223 Alarm filter objects 1...10	1, Data ^a	0	b	97	98	10...19	255
15224 15225 Alarm descriptor objects 1...10	1, Data ^a	0	b	97	98	20...29	255

^a If the IC "Dtaata" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

15212

15213 6.2.66 General list objects

15214 Instances of the IC "Profile generic" are used to model lists of any kind of data, for example
 15215 measurement values, constants, statuses, events. They are modelled by "Profile generic"
 15216 objects. One standard object per billing period scheme is defined.

15217 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 62056-6-1:2021, 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 62056-6-1:2021, Clause A.3.							

15218

15219 6.2.67 Event log objects (class_id 7)

15220 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
 15221 IEC 62056-6-1:2021, 6.5 and 7.5.4.
 15222

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 62056-6-1:2021, 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.							

15223

15224 6.2.68 Inactive objects

15225 Inactive objects are objects, which are present in the meter, but which do not have an assigned
 15226 functionality. Inactive instances of any IC may be present. See also IEC 62056-6-1:2021, 5.3.2.

Inactive objects	IC	OBIS code					
		A	B	C	D	E	F
Inactive objects	Any	0	b	127	0	e	255

15227

15228 6.3 Electricity related COSEM objects

15229 6.3.1 Value group D definitions

15230 The different ways of processing measurement values as defined by value group D – see
 15231 IEC 62056-6-1:2021, 7.2.1 – are modelled as shown in Table 50.

15232 **Table 50 – 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".
Under/Over limit thresholds	Instances of IC "Register" or "Extended register".
Over/Under limit occurrence counters	Instances of IC "Register" or "Extended register".
Under/Over limit durations	Instances of IC "Register" or "Extended register".
Over/Under limit magnitudes	Instances of IC "Register" or "Extended register".

15233

15234 6.3.2 Electricity ID numbers

15235 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
 15236 of those is used, it is allowed to combine them into a "Profile generic" object. In this case, the
 15237 captured objects are *value* attributes of electricity ID "Data" objects, the capture period is 1 to
 15238 have just actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively,
 15239 a "Register table" object can be used. See also IEC 62056-6-1:**2021**, Table 20.

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.

15241

15242 6.3.3 Billing period values / reset counter entries

15243 These values are represented by instances of the IC "Data".

15244 For billing period / reset counters and for number of available billing periods the data type shall
 15245 be *unsigned*, *long-unsigned* or *double-long-unsigned*. For time stamps of billing periods, the
 15246 data type shall be *double-long-unsigned* (in the case of UNIX time), *octet-string* or *date-time*
 15247 formatted as specified in 4.1.6.1.

15248 These objects may be related to channels.

15249 When the values of historical periods are represented by "Profile generic" objects, the time
 15250 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: 2021 , Table 20.	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.

15251

15252 6.3.4 Other electricity related general purpose objects

15253 Program entries shall be represented by instances of the IC "Data" with data type *unsigned*,
15254 *long-unsigned*, *octet-string* or *visible-string*. For "Meter connection diagram ID" objects data
15255 type *enumerated* can be used as well. Program entries can also be related to a channel.

15256 Output pulse constant, reading factor, CT/VT ratio, nominal value, input pulse constant,
15257 transformer and line loss coefficient values shall be represented by instances of the IC "Data",
15258 "Register" or "Extended register". For the *value* attribute, only simple data types are allowed.

15259 Measurement period, recording interval and billing period duration values shall be represented
15260 by instances of IC "Data", "Register" or "Extended register" with the data type of the *value*
15261 attribute *unsigned*, *long-unsigned* or *double-long-unsigned*. The default unit is the second.

15262 Time entry values shall be represented by instances of IC "Data", "Register" or "Extended
15263 register" with the data type of the *value* attribute *octet-string*, formatted as *date-time* in 4.1.6.1.
15264 The data types *unsigned*, *integer*, *long-unsigned* or *double-long-unsigned* can also be used
15265 where appropriate.

15266 The *Clock synchronization method* shall be represented by an instance of an IC "Data" with
15267 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

15268

15269 For the detailed OBIS codes, see IEC 62056-6-1:**2021**, Table 20.

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, Data 3, Register 4, Extended Register	1	b	0	3	e	255
Reading factor and CT/VT ratio		1	b	0	4	e	255
Nominal values	3, Register 4, Extended Register	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, Data 3, Register 4, Extended Register	1	b	0	8	e	255
Time entries		1	b	0	9	e	255
Transformer and line loss coefficients	3, Register 4, Extended Register	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.

15270

15271 6.3.5 Measurement algorithm

15272 These values are represented by instances of the IC “Data”, with data type `enum`.

15273

Measurement algorithm objects	IC	OBIS code					
		A	B	C	D	E	F
Measuring algorithm for active power	1, Data ^a	1	b	0	11	1	255
Measurement algorithm for active energy		1	b	0	11	2	255
Measurement algorithm for reactive power		1	b	0	11	3	255
Measurement algorithm for reactive energy		1	b	0	11	4	255
Measurement algorithm for apparent power		1	b	0	11	5	255
Measurement algorithm for apparent energy		1	b	0	11	6	255
Measurement algorithm for power factor calculation		1	b	0	11	7	255

^a In cases where the IC “Data” is not available, “Register” (with scaler = 0, unit = 255) may be used.

15274

15275 The enumerated values are specified in Table 51:

Table 51 – 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.

15278 **6.3.6 Metering point ID (electricity related)**

15279 A series of objects are available to hold electricity related metering point IDs. They are held by
 15280 the *value* attribute of “Data” objects, with data type *unsigned*, *long-unsigned*, *double-long-unsigned*,
 15281 *octet-string* or *visible-string*. If more than one of those is used, it is allowed to combine
 15282 them into one instance of the IC “Profile generic”. In this case, the captured objects are the
 15283 *value* attributes of the electricity related metering point ID “Data” objects, the capture period is
 15284 1 to have just actual values, the sort method is FIFO, the profile entries are limited to 1.
 15285 Alternatively, an instance of the IC “Register table” can be used. For detailed OBIS codes, see
 15286 IEC 62056-6-1:2021, Table 20.

Metering point ID objects	IC	OBIS code					
		A	B	C	D	E	F
Metering point ID 1...10 (electricity related)	1, Data ^a	1	b	96	1	0...9	255
Metering point ID-s object	7, Profile generic	1	b	96	1	255	255
Metering point ID-s object	61, Register table	1	b	96	1	255	255

^a If the IC “Data” is not available, “Register” (with scaler = 0, unit = 255) may be used.

15287

15288 **6.3.7 Electricity related status objects**

15289 A number of electricity related objects are available to hold information about the internal
 15290 operating status, the starting of the meter and the status of voltage and current circuits.

15291 The status is held by the value attribute of a “Data” object, with data type *bit-string*, *unsigned*,
 15292 *long-unsigned*, *double-long-unsigned*, *long64-unsigned* or *octet-string*.

15293 Alternatively, the status is held by a “Status mapping” object, which holds both the status word
 15294 and the mapping of its bits to the reference table.

15295 If there are several electricity related internal operating status objects used, it is allowed to
 15296 combine them into an instance of the IC “Profile generic” or “Register table”, using the OBIS
 15297 code of the global internal operating status. For detailed OBIS codes, see IEC 62056-6-1:2021,
 15298 Table 20.

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	b	96	5	0...5	255
Internal operating status signals, electricity related, contents mapped to reference table	63, Status mapping	1	b	96	5	0...5	255
Electricity related status data, contents manufacturer specific	1, Data ^a	1	b	96	10	0...3	255
Electricity related status data, contents mapped to reference table	63, Status mapping	1	b	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.

15299

15300 **6.3.8 List objects – Electricity (class_id = 7)**

15301 These COSEM objects are used to model lists of any kind of data, for example measurement
 15302 values, constants, statuses, events. They are modelled by “Profile generic” objects.

15303 One standard object per billing period scheme is defined. See also IEC 62056-6-1:2021, 7.5.3.

List objects – Electricity	IC	OBIS code					
		A	B	C	D	E	F
For names and OBIS codes see IEC 62056-6-1:2021, Table 22.	7, Profile generic	1	b	98	d	e	255 ^a
^a F = 255 means a wildcard here. See IEC 62056-6-1:2021, Clause A.3.							

15304

15305 6.3.9 Threshold values

15306 A number of objects are available for representing thresholds for instantaneous quantities. The
15307 thresholds may be “under limit”, “over limit”, “missing” and “time thresholds”. Time thresholds
15308 are used to detect “under limit”, “over limit” and “missing” conditions.

15309 Objects are also available to represent the number of occurrences when these thresholds are
15310 exceeded, the duration of such events and the magnitude of the quantity during such events.

15311 These values are represented by instances of IC “Data”, “Register” or “Extended register”.

15312 All these quantities may be related to tariffs.

15313 As defined in IEC 62056-6-1:2021, 7.4.2, value group F may be used to identify multiple
15314 thresholds.

15315 For OBIS codes, see Table 52 below and IEC 62056-6-1:2021, Table 14.

15316 **Table 52 – 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	b	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				11, 12, 15, 31, 32, 35, 51, 52, 55, 71, 72, 75, 90...92			

15317

15318 For monitoring the supply voltage, a more sophisticated functionality is also available, that
15319 allows counting the number of occurrences classified by the duration of the event and the depth
15320 of the voltage dip. For OBIS codes, see IEC 62056-6-1:2021, Table 8.

15321 **6.3.10 Register monitor objects (class_id = 21)**

15322 Further to 6.2.13, the following definitions apply:

- 15323 • For monitoring thresholds of instantaneous values, the logical name of the “Register
15324 monitor” object may be the OBIS identifier of the threshold;
- 15325 • For monitoring current average and last average values, the logical name of the “Register
15326 monitor” object may be the OBIS identifier of the demand value monitored.

15327 See Table 53.

15328 **Table 53 – Register monitor objects, electricity**

Register monitor objects	IC	OBIS code					
		A	B	C	D	E	F
Instantaneous values, under limit / over limit / missing	21, Register monitor	1	b	c1	31, 35, 39	0-63	0-99, 255
		1	b	c2		0-120, 124- 127	
Current average and last average values	21, Register monitor	1	b	c1	4, 5, 14, 15, 24, 25	0-63	0-120, 124- 127
		1	b	c2		0-120, 124- 127	

c1 = 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.
 c2 = 11, 12, 15, 31, 32, 35, 51, 52, 55, 71, 72, 75, 90-92, 100-107.

15329

15330 For the use of value group D, see IEC 62056-6-1:2021, Table 14.

15331 For the use of value group E, see IEC 62056-6-1:2021, Table 15 and Table 16.

15332 For the use of value group F, see IEC 62056-6-1:2021, 7.4.2.

15333

15334 **6.4 Coding of OBIS identifications**

15335 To identify different instances of the same IC, their logical_name shall be different. In COSEM,
 15336 the logical_name is taken from the OBIS definition (see 6.2, 6.3 and IEC 62056-6-1:2021).

15337 OBIS codes are used within the COSEM environment as an *octet-string* [6]. Each octet contains
 15338 the unsigned value of the corresponding OBIS value group, coded without tags.

15339 If a data item is identified by less than six value groups, all unused value groups shall be filled
 15340 with 255.

15341 Octet 1 contains the binary coded value of A (A = 0, 1, 2 ...9) in the four rightmost bits. The four
 15342 leftmost bits contain the information on the identification system. The four leftmost bits set to
 15343 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 62056-6-1:2021.	0 0 0 0			
Reserved for future use	0 0 0 1 ... 1 1 1 1			

15344

15345 Within all value groups, the usage of a certain selection is fully defined; others are reserved for
 15346 future use. If in the value groups B to F a value belonging to the manufacturer specific range
 15347 (see IEC 62056-6-1:2021, 4.2) is used, then the whole OBIS code shall be considered as
 15348 manufacturer specific, and the value of the other groups does not necessarily carry a meaning
 15349 defined neither by Clause 5 of this standard nor by IEC 62056-6-1:2021.

15350
15351
15352
15353

Annex A (informative)

Additional information on Auto answer and Auto connect ICs

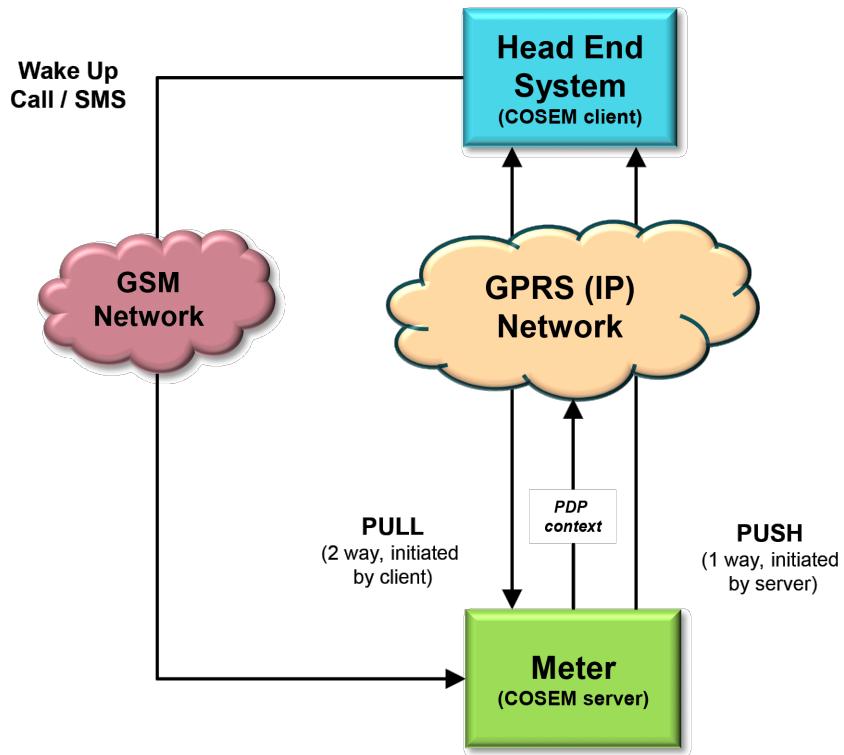
15354 NOTE This information is related to the "Auto answer" (class_id = 28, version = 2, see 4.7.5) and "Auto connect" (class_id = 29, version = 2, see 4.7.6) interface classes.

15356 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.

15359 Devices may connect to the network in regular intervals or on special events either to send unsolicited data or just to become accessible.

15361 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.

15365 Figure A.1 below shows an example for a GSM/GPRS communication network. Please note that 15366 the dashed lines represent the network services, the solid lines refer to possible application 15367 layer services.



15368
15369

Figure A.1 – Network connectivity example for a GSM/GPRS network

15370 The basic network connectivity in the case of a mobile network (GPRS or equivalent service) is
15371 modelled by the "Auto connect" IC. Depending on the mode the connection can be 'always on',
15372 'always on in a time window' or 'only on after a wake-up'. If the device is connected to the
15373 network it has the PDP context attached and it is accessible from by the HES via its IP address.
15374 If necessary the current IP address of the server (meter) can be sent to the client (HES) using
15375 the DataNotification xDLMS service.

15376 The wake-up process is modelled using an instance of the “Auto answer” IC which provides
15377 additional security (check calling number) compared to today's solution.

15378 Also note that the “Auto answer” class is fully decoupled from the xDLMS application layer
15379 services. The main reason is to have a clear separation between the communication layers as
15380 well as to avoid creating an unsecured backdoor to execute application layer services with
15381 almost no protection. The execution of xDLMS services via SMS should be handled by sending
15382 ciphered xDLMS APDUs in a pre-established AA.

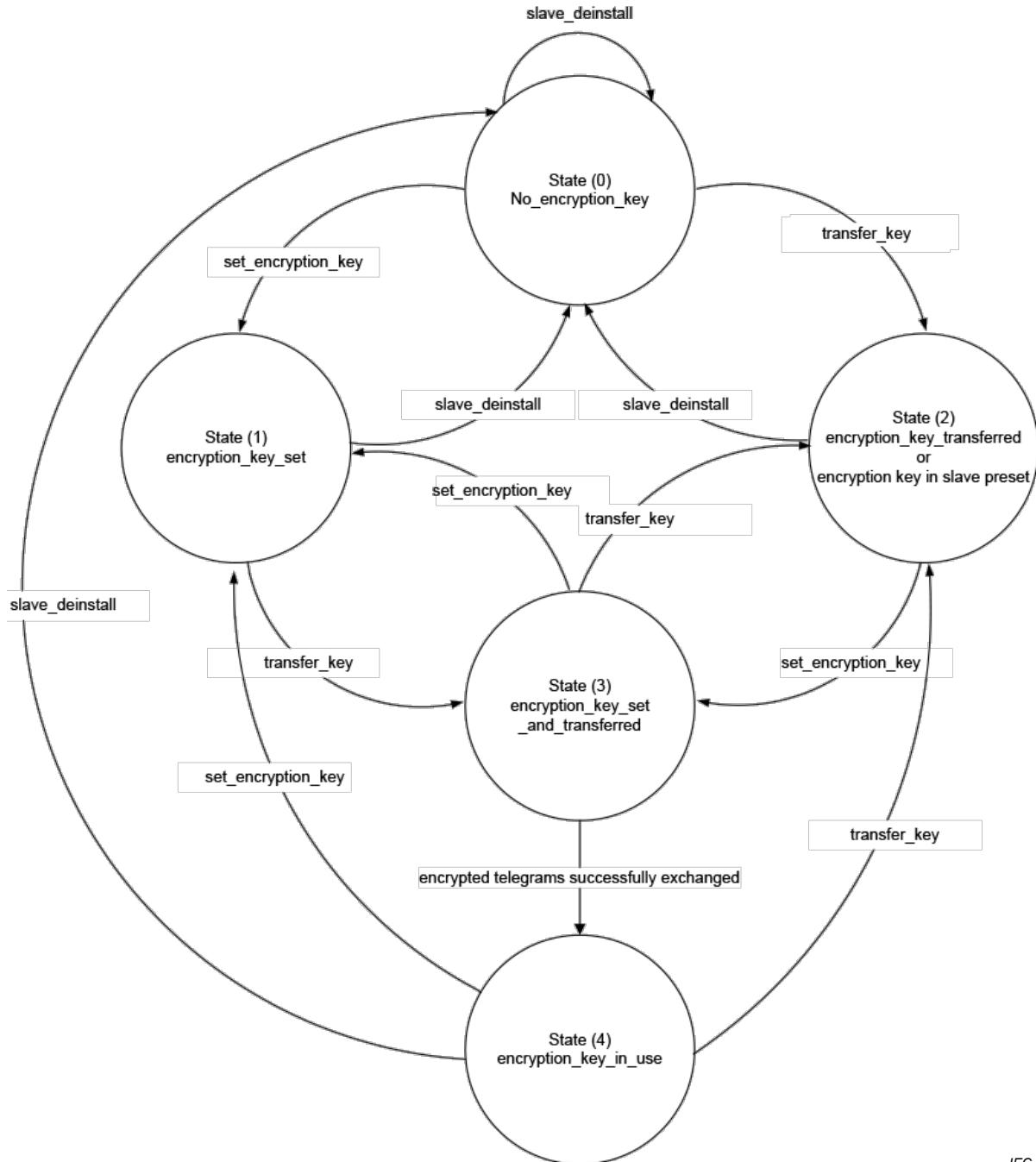
15383

15384
15385
15386
15387

Annex B (informative)

Additional information to M-Bus client (class_id = 72, version 1)

15388 State transitions of the *encryption_key_status* attribute for different use cases are shown in
15389 Figure B.1.



15390

IEC

15391

Figure B.1 – Encryption key status diagram

15392 State transitions of the `encryption_key_status` attribute for different use cases are given below.
 15393 At the time of installation of the slave four cases are possible:

- 15394 a) Encryption key is preset in the slave and cannot be changed, see Table B.1;
 15395 b) Encryption key is preset in the slave and new key is set after installation, see Table B.2;
 15396 c) Encryption key is not preset in the slave, but can be set, see Table B.3 and Table B.4;
 15397 d) The slave is used without encryption. In this case, the `encryption_key_status` stays in
 15398 state (0).

15399 **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)	<code>set_encryption_key</code>
2	(3)	encrypted telegrams successfully exchanged
3	(4)	–

15400

15401 **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)	<code>set_encryption_key</code>
2	(3)	<code>transfer_key</code>
3	(2)	<code>set_encryption_key</code>
4	(3)	encrypted telegrams successfully exchanged
5	(4)	–

15402

15403 **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)	<code>set_encryption_key</code>
2a	(1)	<code>transfer_key</code>
3a	(3)	encrypted telegrams successfully exchanged
4a	(4)	–

15404

15405 **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)	<code>transfer_key</code>
2b	(2)	<code>set_encryption_key</code>
3b	(3)	encrypted telegrams successfully exchanged
4b	(4)	–

15406

15407
15408
15409
15410

Annex C (informative)

Additional information on IPv6 setup class (class_id = 48, version = 0)

15411

C.1 General

15412 In most regards, IPv6 is a conservative extension of IPv4. Most transport and application-layer
 15413 protocols need little or no change to operate over IPv6; exceptions are application protocols
 15414 that embed internet-layer addresses, such as FTP or NTPv3.

15415 IPv6 specifies a new packet format, designed to minimize packet-header processing. Since the
 15416 headers of IPv4 packets and IPv6 packets are significantly different, the two protocols are not
 15417 interoperable.

15418

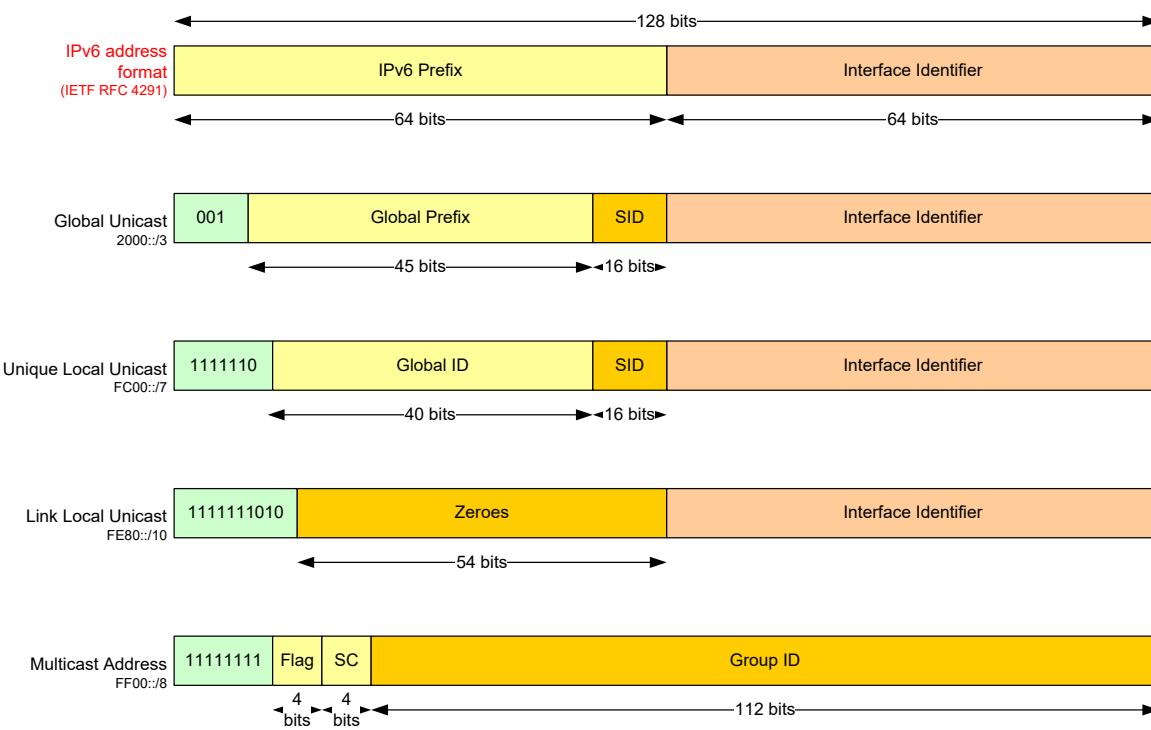
C.2 IPv6 addressing

15419 The most important feature of IPv6 is a much larger address space than that of IPv4: addresses
 15420 in IPv6 are 128 bits long, compared to 32-bit addresses in IPv4. Furthermore, compared to IPv4,
 15421 IPv6 supports multi-addressing on one physical interface (global, unique or link local IPv6
 15422 addresses).

15423 IPv6 addresses are typically composed of two logical parts: a 64-bit (sub-)network prefix used
 15424 for routing, and a 64-bit host part used to identify a host within the network.

15425 The formats allowed for an IPv6 address are shown in Figure C.1.

15426 (see <http://www.iana.org/assignments/ipv6-address-space/>). Note that to facilitate the IPv6 address
 15427 writing, a specific notation defined in RFC 4291 has been specified by IETF (e.g. FF00::/8).



15428

Figure C.1 – IPv6 address formats

15429

15430

15431

15432

15433 Where:

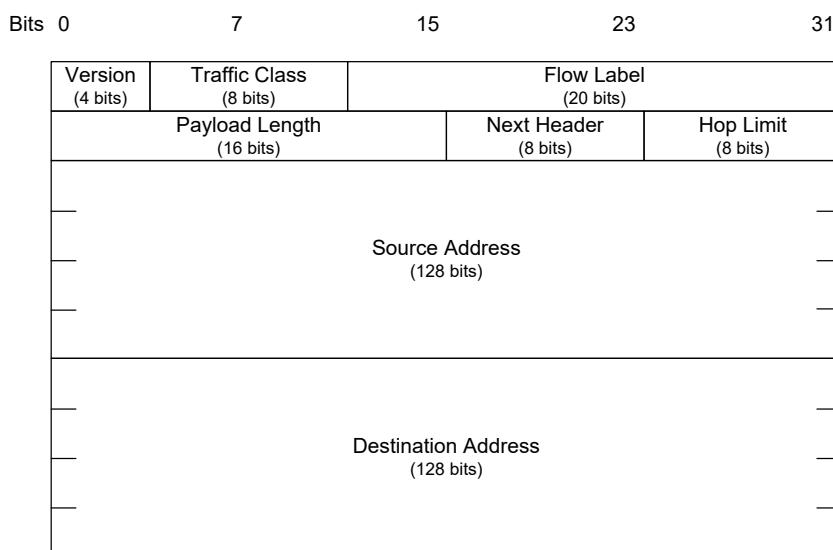
- *Global Unicast* is a routable address in the whole internet network and is composed as follows:
 - Global prefix assigned by IANA (see <http://www.iana.org/assignments/ipv6-unicast-address-assignments/>);
 - Subnet ID (SID) allocated by the network administrator; and
 - Interface Identifier either 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.

15450 It is important to note that there is no broadcast address defined in IPv6.

15451 For more information concerning IPv6 addressing, see RFC 4291.

15452 **C.3 IPv6 header format**

15453 As defined in RFC 2460, Clause 3, the header of one IPv6 packet is composed as shown in
 15454 Figure C.2:



15455

15456 **Figure C.2 – IPv6 header format**

15457

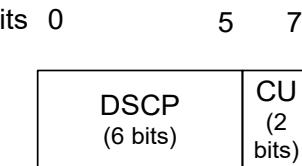
15458

15459 Where:

- 15460 • *Version* specifies the version of the protocol. For IPv6, the value is fixed and equals to 6;
- 15461 • *Traffic class* is used by originating nodes and/or forwarding routers to identify and
- 15462 distinguish between different classes or priorities of IPv6 packets (see RFC 2474, Clause
- 15463 3). Note that the traffic class value is only a notion of prioritization used to distribute the
- 15464 IPv6 frame and do not secure the transmission (the philosophy of the IP network is to do
- 15465 its best).

15466 Figure C.3 shows the content of the traffic class parameter:

15467



15468

Figure C.3 – Traffic class parameter format

15469 Where:

- 15470 – DSCP – Differentiated services code point contains the prioritization of the IPv6 packet in the network (see RFC 2474 for details);
- 15471 – CU – Currently unused;
- 15472 • *Flow label* may be used by a source to label sequences of packets for which it requests
- 15473 special handling by the IPv6 routers, such as non-default quality of service or "real-time"
- 15474 service. Currently, no fully definition of this field is given by IETF and may be seen as
- 15475 reserved for future used;
- 15476 • *Payload length* indicates the size of the upper layer payload carried by the IPv6 frame;
- 15477 • *Next header* identifies the type of header immediately following the IPv6 header of the
- 15478 current frame. It may indicate an upper applicative header (ICMP, UDP, TCP, etc.) or
- 15479 extensions;
- 15480 • *Hop limit* defines the time of life of the IPv6 packet in term of number of hop. The usage is
- 15481 similar to the one defined for IPv4 (decremented by 1 by each node that forwards the
- 15482 packet, the packet is discarded if Hop Limit is decremented to zero);
- 15483 • *Source and destination addresses* indicate the originator of the IPv6 packet and the
- 15484 intended recipient.

15485 Table C.1 summarizes the IPv6 headers managed / not managed by the IPv6 objects.

15488

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

15489

15490 **C.4 IPv6 header extensions**

15491 **C.4.1 Overview**

15492 Contrary to IPv4, IPv6 provides an extensible header by adding optional elements one by one.
 15493 Currently, the following list of optional headers is defined by **RFC 2460**, see Table C.2.

15494 Table C.2 summarizes the options managed / not managed by the IPv6 setup objects.

15495

Table C.2 – Optional IPv6 header extensions vs. IPv6 IC

Extensions	IPv6 setup IC	See subclause
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.		

15496

15497 **C.4.2 Hop-by-Hop options**

15498 The Hop-by-Hop options field must be examined by all devices on the path. Four elements
 15499 currently defined may compose this header (see RFC 2460, 4.3):

- 15500 • A padding form Pad1 which introduces one byte of padding (see RFC 2460, 4.2);
- 15501 • A padding form PadN which introduces more than 2 bytes of padding (see RFC 2460RFC
 15502 2460, 4.2);
- 15503 • A Jumbogram field to indicate the length of the IPv6 datagram in case of jumbo payload
 15504 (higher than the maximum size of length field of IPv6 header) (see RFC 2460 and
 15505 RFC 2675); and
- 15506 • A router alert: The option indicates that the contents of the datagram may be interesting to
 15507 the router (see RFC 2711).

15508 These optional elements are directly managed by the IPv6 protocol stack. Therefore, they are
 15509 not managed by the COSEM IPv6 setup class.

15510 **C.4.3 Destination options**

15511 The Destination options field must be examined only by the target device of the IP datagram.
 15512 Four elements are currently defined by IETF (see RFC 2460, 4.6):

- 15513 • A padding form Pad1 which introduces one byte of padding (see RFC 2460, 4.2);
- 15514 • A padding form PadN which introduces more than 2 bytes of padding (see RFC 2460, 4.2);
- 15515 • Mobility parameters (linked to new IP wireless networks – UMTS, LTE, etc.)
 15516 (see RFC 3775); and
- 15517 • Tunnelling options which permit to communicate between two IPv6 networks separated by
 15518 an IPv4 network (6To4 mechanism) (see RFC 2460).

15519 NOTE The Mobility parameters and the Tunnelling options have been defined separately from RFC 2460RFC 2460
 15520 and are not mentioned in this document. See appropriate RFCs for more details.

15521 These optional elements are directly managed by the IPv6 protocol stack. These optional fields
 15522 are not managed by the COSEM IPv6 setup class.

15523 **C.4.4 Routing options**

15524 Routing options element is used to specify some routing parameters (see **RFC** 2460, 4.4).
 15525 Currently, only one type is defined by IETF (type 2), which is used in case of mobility IPv6.

15526 This notion is out of scope of the COSEM IPv6 setup class.

15527 For information, due to an important security issue (see RFC 5095), the type 0 previously
15528 defined has been deprecated and is forbidden. Furthermore, the type 1 has been temporarily
15529 defined for experimental Nimrod and is obsolete since 1996.

15530 **C.4.5 Fragment options**

15531 Fragment options field is used in the case of fragmentation of the applicative payload if its size
15532 is higher than the MTU of the network (see RFC 2460, 4.5).

15533 This element is directly managed by the IPv6 protocol stack. These optional fields are not
15534 managed by the COSEM IPv6 setup class.

15535 **C.4.6 Security options**

15536 Contrary to IPv4, IPv6 protocol layer includes natively the IPsec protocol. These extensions
15537 are fully defined in the RFC 4291 *IP Version 6 Addressing Architecture*

15538 RFC 4302 and RFC 4303.

15539 The security options are composed of two extensible headers:

- 15540 • Authentication Header (AH): Contains information used to verify the authenticity of most
15541 parts of the packet (see RFC 4291 *IP Version 6 Addressing Architecture*)
- 15542 • RFC 4302, Clause 2);
- 15543 • Encapsulating Security Payload (ESP): Carries encrypted data for secure communication
15544 (see RFC 4303, Clause 2).

15545 Due to the complexity of the IPsec protocol, the configuration of IPsec is out of scope of this
15546 document and must be treated separately.

15547

15548
15549
15550

Annex D (informative)

15551 **Overview of the narrow-band OFDM PLC technology for PRIME networks**

15552 For the specification of the PRIME narrow-band OFDM PLC setup classes, see 4.12.

15553 NOTE This technology is supported by the PRIME Alliance, <http://www.prime-alliance.org>.

15554 ITU-T G.9904:2012 specifies a physical layer, a medium access control layer and convergence
15555 layers for cost-effective narrowband (<200 kbps) data transmission over electrical power lines,
15556 intended for use in smart metering and smart grid applications. It is based on Orthogonal
15557 Frequency Division Multiplexing (OFDM).

15558 The specification currently describes the following:

- 15559 • a low-cost PHY capable of achieving rates of encoded 128 kbps;
- 15560 • a Master-Slave MAC optimised for the power line environment;
- 15561 • a convergence layer for the LLC layer specified in IEC 61334-4-32;
- 15562 • a convergence layer for IPv4;
- 15563 • a convergence layer for IPv6;

15564 The DLMS/COSEM communication profiles for narrow-band OFDM PLC PRIME
15565 neighbourhood networks are specified in draft IEC 62056-8-4,13/1749/CDV.

15566

15567
15568
15569
15570 **Annex E**
15571 **(informative)**

15572 **Overview of the narrow-band OFDM PLC technology for G3-PLC networks**

15573 For the specification of the G3 narrow-band OFDM PLC setup classes, see 4.13.

15574 NOTE This specification is supported by the G3-PLC Alliance, <http://www.G3-PLC.com>.

15575 ITU-T G.9903:2014 specifies the physical, MAC and 6LoWPAN Adaptation layers of the G3-PLC
15576 technology while IEC 62056-8-5, *Electricity metering data exchange –The DLMS/COSEM suite –*

15577 *Part 8-5: Narrow-band OFDM G3-PLC communication profile*

15578 *for neighbourhood networks*

15579 ITU-T E.212 (05.2008), *Series E: Overall network operation, telephone service, service
operation and human factors – International operation – Maritime mobile service and public
land mobile service – The international identification plan for public networks and subscriptions*

15580 ITU-T G.9901:2014 deals with frequency bandplan allocation and associated transmission level
15581 limitations.

15582 Power line communication has been used for many decades, but a variety of new services and
15583 applications require more reliability and higher data rates. However, the power line channel is
15584 very hostile. Channel characteristics and parameters vary with frequency, location, time and
15585 the type of equipment connected to it. The lower frequency regions from 10 kHz to 200 kHz are
15586 especially susceptible to interference. Furthermore, the power line is a very frequency selective
15587 channel. Besides background noise, it is subject to impulsive noise often occurring at 50/60 Hz
15588 and group delays up to several hundred microseconds.

15589 G3-PLC uses advanced modulation and channel coding techniques, which enables efficient use
15590 of the limited bandwidth of the CENELEC bands and facilitates communication over the power
15591 line channel. This combination enables a very robust communication in the presence of narrow-
15592 band interference, impulsive noise, and frequency selective attenuation. The specification
15593 addresses the following main objectives:

- 15594 • provide robust communication on extremely harsh power line channels;
- 15595 • provide a minimum of 20 kbps effective data rate in the normal mode of operation;
- 15596 • ability to notch selected frequencies, to allow the cohabitation with other Narrow-band
15597 PLC communication technologies (e.g. IEC 61334-5-1:2001 S-FSK) or to be compliant
15598 with specific regulatory requirements;
- 15599 • dynamic tone adaptation capability to select frequencies on the channel that do not have
15600 major interference, thereby ensuring a robust communication;
- 15601 • access control, authentication, confidentiality and integrity to ensure high level of security.

15602 To this end, the G3-PLC protocol stack aggregates several layers and sub-layers that form the
15603 G3-PLC profile:

- 15604 • a robust high-performance PHY layer based on OFDM and adapted to the narrow-band
15605 PLC environment;
- 15606 • a MAC layer of the IEEE 802.15.4:2006 type (extended), well suited to low data rates;
- 15607 • IPv6, the new generation of IP (Internet Protocol), which widely opens the range of
15608 potential applications and services; and
- 15609 • to allow good IPv6 and MAC interoperability, an Adaptation sublayer taken from the
15610 Internet world (IETF) and called 6LoWPAN (RFC 4944 extended and RFC 6282). The

- 15611 adaptation sub layer also embeds the LOADng routing algorithm to allow multi-hop mesh
15612 connectivity.
- 15613 For more information about the extensions of IEEE 802.15.4:2006, RFC 4944 and RFC 6282
15614 (AKA 6LoWPAN) standards, and LOADng, see ITU-T G.9903:2014.
- 15615 The DLMS/COSEM narrow-band G3-PLC communication profile is specified in IEC 62056-8-5:2017.
- 15616

15617
15618
15619
15620
15621

Annex F 
(informative)

**Significant technical changes with respect
to IEC 62056-6-2, Edition 3.0:2017**

15622 This Edition 4.0 of IEC 62056-6-2 includes the following significant technical changes with
15623 respect to IEC 62056-6-2, Ed.3.0:2017:

Item	Clause	Description
1.	4.4.8.2	Class updated – 40 – Push set up
2.	4.4.12	New class added – 124 – Communication port protection
3.	4.5.10	Class updated – 65 – Parameter monitor
4.	4.7.8	Class updated – 47 – GSM diagnostic
5.	4.7.9	Calss updated – 151 – LTE monitoring
6.	4.13.5	Class updated – 92 – G3-PLC 6LoWPAN
7.	4.16	New interface classes for LPWAN.
8.	4.17	New interface classes for LoRaWAN
9.	4.18	New interface classes for WiSUN.
10.	4.19	New interface classes for ISO/IEC 14908 PLC networks.

15624
15625

15626

Bibliography

- 15627 ANSI/TIA-4957.200 *Layer 2 Standard Specification for the Smart Utility Network*
- 15628 IEC 61334-6:2000, *Distribution automation using distribution line carrier systems – Part 6: A-XDR encoding rule*
- 15629
- 15630 IEC TR 62051:1999, *Electricity metering – Glossary of terms*
- 15631 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*
- 15632
- 15633
- 15634 IEC 62056-4-7:2015, *Electricity metering data exchange – The DLMS/COSEM suite – Part 4-7: DLMS/COSEM transport layer for IP networks*
- 15635
- 15636 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*
- 15637
- 15638 IEC 62056-9-7:2013, *Electricity metering data exchange – The DLMS/COSEM suite – Part 9-7: Communication profile for TCP-UDP/IP networks*
- 15639
- 15640 ISO/IEC 10646:2014, *Information technology – Universal Coded Character Set (UCS)*
- 15641 draft IEC 62056-8-4: 21XX, 13/1749/CDV, *Electricity metering data exchange – The DLMS/COSEM suite – Part 8-4: Communication profiles for narrow-band OFDM PLC PRIME neighbourhood networks*
- 15642
- 15643
- 15644 IEC 62056-8-5, *Electricity metering data exchange –The DLMS/COSEM suite –*
- 15645 *Part 8-5: Narrow-band OFDM G3-PLC communication profile*
- 15646 *for neighbourhood networks*
- 15647 ITU-T E.212 (05.2008), *Series E: Overall network operation, telephone service, service operation and human factors – International operation – Maritime mobile service and public land mobile service – The international identification plan for public networks and subscriptions*
- 15648
- 15649
- 15650 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*
- 15651
- 15652
- 15653 ITU Recommendation X.217:1995, *Information technology – Open Systems Interconnection – Service definition for the association control service element*
- 15654
- 15655 ITU Recommendation X.227:1995, *Information technology – Open Systems Interconnection Connection-oriented protocol for the association control service element: Protocol specification*
- 15656
- 15657 IETF STD 5, *Internet Protocol*, 1981. (Also IETF RFC 0791, RFC 0792, RFC 0919, RFC 0922, RFC 0950, RFC 1112)
- 15658
- 15659 IETF STD 51, *The Point-to-Point Protocol (PPP)*, 1994. (Also RFC 1661, RFC 1662)
- 15660 IETF STD 51 / RFC 1661, *The Point-to-Point Protocol (PPP)* (Also: IETF STD 0051), 1994, Updated by: RFC 2153, Obsoletes: RFC 1548
- 15661

- 15662 IETF STD 51 / RFC 1662, *PPP in HDLC-like Framing*, (Also: IETF STD 0051), 1994, Obsoletes:
15663 RFC 1549
- 15664 3GPP TS 36.214 V13.0.0 (2015-12), *Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer; Measurements*
- 15666 The following RFCs are available online from the Internet Engineering Task Force (IETF):
15667 <http://www.ietf.org/rfc/std-index.txt>, <http://www.ietf.org/rfc/>
- 15668 **RFC 768 User Datagram Protocol**

- 15669 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.
- 15671 RFC 793, *Transmission Control Protocol (Also IETF STD 0007)*, 1981, Updated by: RFC 3168
- 15672 RFC 940, *Toward an Internet Standard Scheme for Subnetting*, 1985
- 15673 RFC 950, *Internet Standard Subnetting Procedure*, 1985
- 15674 RFC 1144, *Compressing TCP/IP Headers for Low-Speed Serial Links*, 1990
- 15675 **RFC 1213 Management Information Base for Network Management of TCP/IP-based internets: MIB-II**
- 15677 RFC 1332, *The PPP Internet Protocol Control Protocol (IPCP)*, 1992, Updated by: RFC 3241.
15678 Obsoletes: RFC 1172
- 15679 RFC 1570, *PPP LCP Extensions*, 1994
- 15680 RFC 1994, *PPP Challenge Handshake Authentication Protocol (CHAP)*, 1996. Obsoletes: RFC
15681 1334
- 15682 RFC 2433, *PPP CHAP Extension*, 1998
- 15683 RFC 2460, *Internet Protocol, Version 6 (IPv6)*, 1998
- 15684 RFC 2473, *Generic Packet Tunneling in IPv6*, 1998
- 15685 RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, 1998
15686
- 15687 RFC 2507, *IP Header Compression*, 1999
- 15688 RFC 2508, *Compressing IP/UDP/RTP Headers for Low-Speed Serial Links*, 1999
- 15689 RFC 2675, *IPv6 Jumbograms*, 1999
- 15690 RFC 2711, *IPv6 Router Alert Option*, 1999
- 15691 RFC 2759, *Microsoft PPP CHAP Extensions, Version 2*, 2000
- 15692 RFC 2986, *PKCS #10 v1.7: Certification Request Syntax Standard*

- 15693 RFC 3095, *RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed*, 2001
- 15694
- 15695 RFC 3241, *Robust Header Compression (ROHC) over PPP*, 2002. Updates: RFC1332
- 15696 **RFC 3315 Dynamic Host Configuration Protocol for IPv6 (DHCPv6)**

- 15697 RFC 3513, *Internet Protocol Version 6 (IPv6) Addressing Architecture*, 2003
- 15698 RFC 3544, *IP Header Compression over PPP*, 2003
- 15699 RFC 3748, *Extensible Authentication Protocol (EAP)*, 2004
- 15700 RFC 3775, *Mobility Support in IPv6*, 2004
- 15701 RFC 4291 *IP Version 6 Addressing Architecture*
- 15702 RFC 4302, *IP Authentication Header*, 2005
- 15703 RFC 4303, *IP Encapsulating Security Payload (ESP)*, 2005
- 15704 **RFC 4443 Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification**
- 15705
- 15706 **RFC 4944 Transmission of IPv6 Packets over IEEE 802.15.4 Networks**
- 15707 RFC 4861, *Neighbor Discovery for IP version 6 (IPv6)*, 2007
- 15708 RFC 4944, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, 2007
- 15709 RFC 5095, *Deprecation of Type 0 Routing Headers in IPv6*, 2007
- 15710 **RFC 5216 The EAP-TLS Authentication Protocol**

- 15711 RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, 2008
- 15712
- 15713 RFC 5905, *Network Time Protocol Version 4: Protocol and Algorithms Specification*, 2010
- 15714 **RFC 6206 The Trickle Algorithm**
- 15715 RFC 6282, *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*
15716 [online]. Edited by J. Hui, Ed. September 2011
- 15717 **RFC 6550 RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks**
- 15718 RFC 6775, *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*, 2012
- 15719
- 15720 **RFC 7731 Multicast Protocol for Low-Power and Lossy Networks (MPL)**
- 15721 **RFC 7774 Multicast Protocol for Low-Power and Lossy Networks (MPL) Parameter Configuration Option for DHCPv6**
15722

- 15723 Point-to-Point (PPP) Protocol Field Assignments. *Online database*. Available from:
15724 <http://www.iana.org/assignments/ppp-numbers/ppp-numbers.xhtml>
- 15725
- 15726 DLMS UA 1000-1, the “Blue Book” Ed. 12.1:2015, *COSEM interface classes and OBIS*
15727 *identification system*
- 15728 DLMS UA 1000-2, the “Green Book” Ed. 8.1:2015, *DLMS/COSEM Architecture and Protocols*
- 15729 DLMS UA 1001-1, the “Yellow Book”, Ed. 5.0:2015, *DLMS/COSEM Conformance test and*
15730 *certification process*
- 15731 DLMS UA 1002, the “White Book”, Ed. 1.0:2003, *COSEM Glossary of terms*
- 15732 3GPP TS 24.008 V13.7.0 (2016-10), *Technical Specification Digital cellular*
15733 *telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System*
15734 *(UMTS); LTE; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3*
- 15735 3GPP TS 24.301 V13.4.0 (2016-01), *Technical Specification Group Core Network and*
15736 *Terminals; Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3*
- 15737 3GPP TS 24.301 V13.11.0 (2018-01), *Technical Specification Group Core Network and*
15738 *Terminals; Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3*
- 15739 3GPP TS 27.007 V16.1.0 (2019-06) *Technical Specification Group Core Network and*
15740 *Terminals; AT command set for User Equipment (UE)*
- 15741 3GPP TS 36.101 V15.4.0 (2019-01) *Technical Specification Group Radio Access Network;*
15742 *Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission*
15743 *and reception*
- 15744 3GPP TS 36.133 V13.11.0 (2018-04) *Technical Specification LTE; Evolved Universal*
15745 *Terrestrial Radio Access (E-UTRA); Requirements for support of radio resource management*
- 15746 3GPP TS 36.133 V14.4.0 (2017-07) *Technical Specification LTE; Evolved Universal*
15747 *Terrestrial Radio Access (E-UTRA); Requirements for support of radio resource management*
- 15748 3GPP TS 36.213 V15.5.0 (2019-05) *Technical Specification Group Radio Access Network;*
15749 *Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures*
- 15750 3GPP TS 36.304 V13.0.0 (2015-12), *Technical Specification Group Radio Access Network;*
15751 *Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) procedures in idle*
15752 *mode*
- 15753 3GPP TS 36.304 V13.8.0 (2018-01) *Technical Specification Group Radio Access Network;*
15754 *Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) procedures in idle*
15755 *mode*
- 15756 3GPP TS 36.321 V15.5.0 (2019-05) *Technical Specification Group Radio Access Network;*
15757 *Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol*
15758 *specification*

- 15759 3GPP TS 36.331 V15.5.1 (2019-05) *Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification*
- 15762 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*
- 15765 NOTE This Recommendation is referenced in version 0 of the G3-PLC setup classes.
- 15766 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*
- 15769 NOTE This Recommendation is referenced in version 1 of the G3-PLC setup classes.
- 15770 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*
- 15773 EN 13757-1:2014, *Communication system for meters – Part 1: Data exchange*
- 15774 EN 13757-2:2004, *Communication system for and remote reading of meters – Part 2: Physical and link layer*
- 15776 EN 13757-3:2004, *Communication systems for and remote reading of meters – Part 3: Dedicated application layer*
- 15778 NOTE This standard is referenced in the “M-Bus client setup” interface class version 0.
- 15779 EN 13757-3:2013, *Communication systems for and remote reading of meters – Part 3: Dedicated application layer*
- 15781 NOTE This standard is referenced in the M-Bus client setup interface class version 1.
- 15782 EN 13757-4:2013, *Communication system for and remote reading of meters – Part 4: Wireless meter (Radio meter reading for operation in SRD bands)*
- 15784 EN 13757-5:2015, *Communication systems for meters – Part 5: Wireless M-Bus relaying*
- 15785 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)*
- 15789 NOTE This standard is also available as ISO/IEC/IEEE 8802-15-4:2010.
- 15790 ETSI GSM 05.08:1996, *Digital cellular telecommunications system (Phase 2+); Radio subsystem link control*
- 15792 ANSI C12.19:1997, IEEE 1377:1997, *Utility industry end device data tables*
- 15793 ZigBee® 053474 ZigBee® Specification. *The specification can be downloaded free of charge from <http://www.zigbee.org/Standards/ZigBeeSmartEnergy/Specification.aspx>*
- 15795 [FANSPEC] Wi-SUN Alliance: *Field Area Network Working Group (FANWG):Technical Profile Specification:Field Area Network:Version 1v26.*

15797 [PHYSPEC] *Wi-SUN Alliance: PHY Working Group (PHYWG) Wi-SUN PHY Specification*
15798 *Revision 1V02*

15799 *LoRaWAN 1.0.3 LoRaWAN® Specification v1.0.3*

15800 [*https://lora-alliance.org/resource-hub/lorawanr-specification-v103*](https://lora-alliance.org/resource-hub/lorawanr-specification-v103)

15801 The following RFCs are available online from the Internet Engineering Task Force (IETF):
15802 <http://www.ietf.org/rfc/std-index.txt>, <http://www.ietf.org/rfc/>

15803

15804

15805

15806

Index

15807

- 61334-4-32 LLC SSCS setup 298
Abstract object 505
Access rights 46
Access selector 390
Account 23, 190, 192, 486, 494
account_mode_and_status 194
actions 186
activate_account 201
Active initiator 19
Active power 519
Activity calendar 163, 164, 486, 492
Actor, Arbitrator 185
Advanced power failure monitoring 509
AES-GCM-128 109
AES-GCM-256 109
Agreed_key 133
Alarm descriptor 115, 514
Alarm filter 514
Alarm monitor 115, 493
Alarm register 115, 514
Algorithm 518, 519
All configured 280
All Physical 280
Apparent power 519
Application context 46
Arbitrator 28, 508
Array manager 486
Association 486, 503
Association LN 86, 91, 398, 405, 409
Association object 296
Association SN 34, 86, 87, 384, 388, 391
Association view 46
Attribute 32, 34, 37, 98, 345, 349, 351, 355, 357, 362, 366, 368
Authenticated request 108, 134
Authenticated response 108, 134
Authentication key 110
Authentication mechanism 46
Auto answer 229, 437
Auto connect 233, 441
Auto dial 439
Base node 19, 296
base_name 34, 87, 388, 392
Battery 509
Beacon slot 20
Billing period 487, 491, 516
Billing period counter 488
Billing period, end of 492
Block demand 61
Broadcast 490
CAD 21
Calendar 164
Calling line identification 231
Capture period 506, 513, 516, 520
capture_tim 238, 447
Captured object 516, 520
Certificate 128
Certificate Signing Request 112
Certificate, export 113
Certificate, import 112
Challenge 404

Charge	23, 190, 212, 486, 494	Credit, selected/invoked.....	202
class_id.....	92	CT/VT ratio.....	518
Client user identification	86	Cumulative value	515
client_SAP	94	Current and last average values.....	516
Clock.....	47, 154, 486, 489	Current association.....	46
<i>Clock synchronization method</i>	517	current_average_value	61
collect.....	23	Currently active tariff.....	511
Commodity.....	23, 26	Data..	47, 54, 75, 167, 506, 509, 513, 517, 518, 521
Communication port log parameter	511	Data format	37, 39
Communication tamper event	512, 513	Data protection, COSEM	504
Compact data	486	Data security, access.....	47
Configuration	246, 507	Data security, transport.....	47
Conformance block.....	95	Data type.....	37
Consumer message	511	data_index	69, 126, 420
consumption_based_collection	191	Date and time format	39
consumption_based_credit	190	Daylight saving.....	154, 162
Contracted value	516	Demand register	47, 60, 65, 516
Convergence layer, PRIME NB OFDM PLC	534	Demotion.....	21
COSEM logical device name.....	45, 46	Device ID	506
COSEM objects, Abstract.....	485	Device start-up.....	300
COSEM objects, Electricity.....	515	device_serial_number.....	227
COSEM physical device	273, 295	Digital signature, profile entry	512
Credit	190, 202, 486, 494	Digitally signed request.....	108, 134
credit mode	23	Digitally signed response	108
Credit states	202	Disconnect control	169, 490, 492, 507
credit token	27	Disconnected state	20
Credit, emergency	24	DLMS version	95
Credit, enabled	202	ECDH P-384	109
Credit, exhausted	202	ECDSA P-256	109
Credit, in use	202	Electrical port	494
Credit, priority	202	Electricity breaker	507
Credit, selectable.....	202	Emergency activation time	173

Emergency credit.....	24	G3-PLC 6LoWPAN adaptation layer setup	310, 318, 478
Emergency duration	173	G3-PLC MAC layer counters	309, 311, 501
Emergency profile.....	173	G3-PLC MAC setup	310, 312, 472, 501
Emergency profile, Limiter.....	174	Gas valve	507
Emergency threshold.....	173	<i>get_protected_attributes</i>	128, 135
emergency_credit	190	Global broadcast encryption key	110
Enabled.....	24	Global unicast encryption key	110
Encrypted request	108, 134	global_broadcast_encryption_key	133
Encrypted response.....	108, 134	global_unicast_encryption_key.....	133
Encryption key	97, 246, 409, 449	GPRS modem setup ...	235, 238, 447, 486, 498
Enterprise Resource Planning.....	24	GSM diagnostic ..	236, 442, 445, 486, 498, 499, 503
Environmental related parameters	510	Harmonics	519, 521
Ephemeral Unified Model	111	Hayes standard	229
equipment_id	227	HDLC setup	486, 495
Error register.....	495, 513, 514	HDLC setup class	223, 431
Event code.....	510	Head End System	24
Event counter	511	High resolution clock	489
Event log.....	515	HLS secret	97, 409
execute	159	HS-PLC ISO/IEC 12139-1 neighbourhood network.....	326
Extended register	59, 65, 75, 167, 179, 496, 509, 510, 515, 516, 517, 521	HS-PLC ISO/IEC 12139-1 networks	486
Fatal error register	227	I/O control signal	507
Firmware.....	488	ID numbers, Electricity	516
Floating point format.....	42	Identification number.....	449
Frequency notching, G3-PLC.....	535	Identification system	523
Friendly credit	24	Identified_key	133
Full Function Device	309	IEC 61334-4-32 LLC setup	499
Function control	144, 486, 490, 492, 505	IEC local port setup.....	486
Fundamental.....	519	IEC twisted pair (1) Fatal Error register	496
Fundamental component.....	519	IEC twisted pair (1) MAC address setup....	496
G3-PLC	535	IEC twisted pair (1) setup ...	225, 433, 495, 496
G3-PLC 6LoWPAN adaptation layer	501	IEEE address	22

IHD	22	IPv6 header	529
Image	17, 106	IPv6 setup	486, 497, 503
<i>Image activation</i>	490, 492	ISO/IEC 8802-2 LLC layer setup	486
Image transfer	86, 99, 100, 486, 505	ISO/IEC 8802-2 LLC Type 1 setup	290, 500
<u>Image, activation</u>	107	ISO/IEC 8802-2 LLC Type 2 setup	291, 500
Image, <u>completeness</u>	106	ISO/IEC 8802-2 LLC Type 3 setup	293, 500
<u>Image, initiate transfer</u>	106	Key agreement	109
ImageBlock	18	Key information	138
ImageBlockNumber	18	key_info	133
<u>ImageBlocks, transfer</u>	106	last_average_value	61
ImageBlockSize	18, 103	Limiter	173, 486, 493
ImageSize	18	Link key	22
In Home Display	22	List objects – Electricity	520
Inactive objects	515	List objects – General	515
Information security	47	LLC layer	288, 459
Information, measured	33	LLC sub-layer	273
Information, static	33	LLS secret	90, 395
Initiator	18, 280	Load limiting	24
insert_entry	149	<i>Load profile control</i>	490, 492
Install code	22	Local port setup	221, 429, 494
Instantaneous value	516, 521, 522	local_disconnect	171
Instantiation	32, 33, 54, 55, 167	local_reconnect	171
Interface class	32, 47, 50	Logical device	45, 46, 87, 91, 98, 99, 388, 391, 396, 405, 409
Interface object	32	Logical Device Name	34, 273, 295, 486, 504
Internal control signals	508	Logical name	33
Internal operating status	508, 520	Logical name referencing	91, 405, 409
Internet	497	logical_name	32
Invocation counter	504	LTE monitoring	486, 498
invoke_credit	210	MAC address	22, 227
<i>invoke_protected_method</i>	129, 136	MAC address setup	486, 497, 501
IPv4 setup	257, 486, 497	MAC sub-layer	273
IPv6 addressing	528	Managed payment mode	25

Management Logical Device	45, 47, 295	Modem	228, 229, 439, 490
manual_disconnect.....	171	Modem configuration	228, 486, 489
manual_reconnect.....	171	NB OFDM PLC for G3-PLC networks.....	486
Manufacturer ID	449	NB OFDM PLC for PRIME networks	486
manufacturer_id.....	227	NB OFDM PLC profile for PRIME networks	295
Master key	110, 128	NEW	280
master_key	134	New system	19
Max credit limit.....	494	New system title	19
Max vend limit.....	494	NO-BODY	280
Maximum and minimum values	516	Node	20
M-Bus	486	Nonce	138
M-Bus client.....	449, 461, 463, 467, 496, 526	Normal mode	491
M-Bus control log object	496	Normal threshold.....	173
M-Bus diagnostics	497	Notation.....	34
M-Bus disconnect control object	496	NTP protocol	271
M-Bus master port setup	248, 497	NTP setup	486, 498
M-Bus port setup	449	number_of_periods	60
<i>M-Bus profile control</i>	490, 492	OBIS.....	33, 523
M-Bus profile generic object.....	449, 496	Object Identification System	485
M-Bus slave	449, 496	<i>object_list</i>	46, 93, 94, 97, 98, 403
M-Bus slave port setup.....	242, 496	Occurrence counter	516
M-Bus value object	449	One-Pass Diffie-Hellman	138
Measurement period.....	517	Operating time	510
Medium access control layer.....	534	Optical port.....	494
Meter open event.....	512, 513	Optical test output	491
Meter reset.....	491	originator_system_title	133
Meter tamper event	512, 513	Orthogonal Frequency Division Multiplexing	534
Metering point ID	507, 520	other_information	133
Method.....	32, 37	Output control	492
metrological_	181	Output pulse constant	517
Metrology tamper event.....	512, 513	Output signals	491
MIB variables.....	273	Packet switched network	233

PAN coordinator	309	PRIME NB OFDM PLC MAC setup	300
Parameter changes	507	PRIME NB OFDM PLC Physical layer counters	299
Parameter monitor	175, 427, 486, 493	<i>priority</i>	202
Parameter monitor log	176, 427	Process value	55
Password	90, 395	Profile	377
Payment metering	190	Profile entries	506, 513, 516, 520
payment_event_based_collection	191	Profile generic 47, 66, 487, 488, 506, 513, 516, 520	
Period	60	Program entries	517
Permission	184	Promotion	21
Personal Area Network	331	Proprietary	32
Physical device	45, 98, 504	Protection parameters	127
Physical layer, PRIME NB OFDM PLC	534	<i>protection_buffer</i>	128, 129, 131
PLC OFDM Type 1 Application identification	501	PSTN modem configuration	435
PLC OFDM Type 1 MAC counters	501	Public client	47
PLC OFDM Type 1 MAC functional parameters	501	Publish/subscribe	114
PLC OFDM Type 1 MAC network administration data	501	Push	114, 490, 492
PLC OFDM Type 1 MAC setup	501	Push setup	115, 486, 498, 499
PLC OFDM Type 1 physical layer counters	501	Push window	116
Post-payment	25	random delay	116, 416
Power factor	519	Reactive power	519
Power factor, displacement	519	Reading factor	518
Power factor, true	519	Readout	72, 486
Power failure	162, 509	recipient_system_title	133
Power failure duration	509	Reduced Function Device	309
PPP setup	265, 486, 497	Register 33, 55, 59, 65, 75, 167, 509, 515, 516, 517, 521	
Preferred readout	72	Register activation	65, 486, 492
Primary address	449	Register monitor	167, 179, 486, 493, 522
PRIME NB OFDM PLC Applications identification	308	Register table	73
PRIME NB OFDM PLC MAC counters	304	<i>register_assignment</i>	65
PRIME NB OFDM PLC MAC functional parameters	302	Registered system	19
		Registration	20

remote_disconnect	171, 172	server_SAP.....	94
remote_reconnect.....	171, 173	Service node	20
remove_entries	150	Service Specific Convergence Sublayer.....	298
repayable	25	set_amount_to_value.....	210
repayment mode.....	25	set_protected_attributes	129, 135
Repetition phase.....	284	S-FSK Active initiator	499
Reporting system.....	19	S-FSK MAC counters.....	499
Reporting system list	289	S-FSK MAC synchronization timeouts.....	499
Reserved base_names.....	34	S-FSK Phy&MAC setup	499
Reserved credit	26	S-FSK Physical layer	273
reserved_credit	190	S-FSK PLC setup.....	486
reset.....	182	S-FSK Reporting system list.....	500
reset_account	201	Shared line	230
retrieve_entries	148	Short name	34
SAP.....	98, 99	Single action schedule	168, 486, 492
SAP assignment ...	34, 45, 47, 86, 98, 486, 504	Sliding demand	61
Schedule.....	159, 163, 164, 486, 491	SMTP setup	270, 498, 503
Script.....	158, 162	Social credit	26
Script table.....	34, 158, 164, 167, 486, 490	Sort method	506, 513, 516, 520
Script, null.....	158	Special days	160
Secret	90, 97, 395, 404	Special days table	163, 164, 486, 491
Security mechanisms	47	Standard readout profile	494
Security setup	86, 504	Static Unified Model	138
Security suite	127	Status mapping	76
Selectable	26	Status register	510
Selected/Invoked	26	Status value	55
Selective access	37, 68, 70, 72, 78, 88, 93, 131, 403	Strong DC magnetic field event	512, 513
Selective access parameters.....	37	Subnetwork	20
Sensor manager	179	Sub-slot	19
Sensor, pressure	182	Switch node	296
Serial number, sensor	180	Switch state.....	20
Server model	44	Synchronization	162

Tamper	512, 513	Unit.....	56
Tarification	65, 491, 492	UNIX clock	489
TCP-UDP setup	256, 486, 497	update_amount	210
Temporary debt	26	update_entry	149
Terminal cover open event.....	512, 513	UTC.....	154, 156
Terminal node.....	296	Utility tables.....	72, 486, 505
Terminal state	20	V.44 compression	109
Test mode.....	491	Value	33
Threshold value	173, 521	Value group C	486
Threshold, missing	521	Value group D	515
Threshold, over limit	521	Value group F	521
Threshold, under limit	521	Vend.....	27
Tilt event	512, 513	Version.....	92, 523
Time changes	162	Version, previous	377
Time entries	517	Wake-up.....	524
Time integral	516	Wake-up request.....	232
Time setting backward	162	Weighting	184
Time setting forward	162	Wireless Mode Q	486, 496
Time synchronization	162	Wireless Mode Q channel.....	248
time_based_collection	191	Wrapped_key	133
time_based_credit	190	xDLMS context.....	46
Timeslot	19	year_of_manufacture	227
Token	26	ZigBee®	21, 22, 120, 331, 486, 502
Token carrier	27	ZigBee® 053474	542
Token Carrier Interface.....	218	ZigBee® 2007	333
Token gateway	218, 486, 494	ZigBee® client.....	22
token_credit	190	ZigBee® cluster	22
Top-up	27	ZigBee® Communications hub	332
transaction_id	133	ZigBee® coordinator	22, 331
Transmission phase	284	ZigBee® mirror.....	22
Twisted pair	225, 433	ZigBee® network control	337, 502
Twisted pair setup	486	ZigBee® PAN creation.....	331

ZigBee® Pro	22, 333	ZigBee® SAS startup.....	333, 502
ZigBee® router	23	ZigBee® server.....	23, 331
ZigBee® SAS APS fragmentation	337, 502	ZigBee® Trust Center.....	23, 331
ZigBee® SAS join.....	335, 502	ZigBee® tunnel setup	343, 502
