

TECHNICAL REPORT
Companion Specification
for Energy Metering

**COSEM Interface Classes
and
OBIS Object Identification
System**

DLMS User Association



| | | | |
|-----------------------|------------|-------------------------|-------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 1/492 |
|-----------------------|------------|-------------------------|-------|

CONTENTS

| | |
|--|----|
| Foreword..... | 16 |
| Introduction..... | 20 |
| 1 Scope | 21 |
| 2 Referenced documents (see also the Bibliography) | 22 |
| 3 Terms, definitions and abbreviations | 25 |
| 3.1 Terms and definitions related to the Image transfer process (see 4.4.6) | 25 |
| 3.1 Terms and definitions related to the Image transfer process (see 4.4.6) | 25 |
| 3.2 Terms and definitions related to the S-FSK PLC setup classes (see 4.10) | 26 |
| 3.3 Terms and definitions related to the PRIME NB OFDM PLC setup ICs (see 4.12)..... | 27 |
| 3.4 Terms and definitions related to ZigBee® (see 4.14) | 28 |
| 3.5 Terms and definitions related to Payment metering interface classes (see 4.6)..... | 29 |
| 3.6 Terms and definitions related to the Arbitrator IC (see 4.5.12)..... | 33 |
| 3.7 Abbreviations | 34 |
| 4 The COSEM interface classes | 39 |
| 4.1 Basic principles | 39 |
| 4.1.1 General..... | 39 |
| 4.1.2 Referencing methods | 40 |
| 4.1.3 Reserved base_names for special COSEM objects | 40 |
| 4.1.4 Class description notation | 40 |
| 4.1.5 Common data types | 43 |
| 4.1.6 Data formats | 44 |
| 4.1.6.1 Date and time formats | 44 |
| 4.1.6.2 Floating point number formats | 46 |
| 4.1.7 The COSEM server model | 48 |
| 4.1.8 The COSEM logical device | 48 |
| 4.1.8.1 General..... | 48 |
| 4.1.8.2 COSEM logical device name (LDN)..... | 48 |
| 4.1.8.3 The “association view” of the logical device..... | 49 |
| 4.1.8.4 Mandatory contents of a COSEM logical device..... | 49 |
| 4.1.8.5 Management logical device..... | 49 |
| 4.1.9 Information security | 50 |
| 4.2 Overview of the COSEM interface classes..... | 51 |
| 4.3 Interface classes for parameters and measurement data | 56 |
| 4.3.1 Data (class_id = 1, version = 0)..... | 56 |
| 4.3.2 Register (class_id = 3, version = 0)..... | 57 |
| 4.3.3 Extended register (class_id = 4, version = 0) | 60 |
| 4.3.4 Demand register (class_id = 5, version = 0) | 61 |
| 4.3.5 Register activation (class_id = 6, version = 0) | 64 |
| 4.3.6 Profile generic (class_id = 7, version = 1) | 66 |
| 4.3.7 Utility tables (class_id = 26, version = 0)..... | 71 |
| 4.3.8 Register table (class_id = 61, version = 0) | 72 |

| | | |
|-------------------------------|---|-----|
| 4.3.9 | Status mapping (class_id = 63, version = 0) | 74 |
| 4.3.10 | Compact data | 75 |
| 4.3.10.1 | Compact data (class_id = 62, version = 0) | 75 |
| 4.3.10.2 | Examples for using compact data | 78 |
| 4.3.10.2.1 | Daily billing data | 78 |
| 4.3.10.2.2 | Diagnostic and Alarm data | 79 |
| 4.3.10.2.3 | Logbook reading | 80 |
| 4.4 | Interface classes for access control and management | 82 |
| 4.4.1 | Overview | 82 |
| 4.4.2 | Client user identification | 82 |
| 4.4.3 | Association SN class (class_id = 12, version 4) | 82 |
| 4.4.4 | Association LN class (class_id = 15, version 3) | 87 |
| 4.4.5 | SAP assignment (class_id = 17, version = 0) | 93 |
| 4.4.6 | Image transfer | 94 |
| 4.4.6.1 | General | 94 |
| 4.4.6.2 | The steps of the image transfer process | 94 |
| 4.4.6.3 | Image transfer (class_id = 18, version = 0) | 95 |
| 4.4.6.4 | Example for a standard Image transfer process | 97 |
| 4.4.7 | Security setup (class_id = 64, version = 1) | 102 |
| <i>Specific methods</i> | 102 | |
| 4.4.8 | Push interface classes and objects | 108 |
| 4.4.8.1 | Overview | 108 |
| 4.4.8.2 | Push setup (class_id = 40, version = 0) | 110 |
| 4.4.9 | COSEM data protection | 115 |
| 4.4.9.1 | Overview | 115 |
| 4.4.9.2 | Data protection (class_id = 30, version = 0) | 118 |
| 4.5 | Interface classes for time- and event bound control | 132 |
| 4.5.1 | Clock (class_id = 8, version = 0) | 132 |
| 4.5.2 | Script table (class_id = 9, version = 0) | 135 |
| 4.5.3 | Schedule (class_id = 10, version = 0) | 136 |
| 4.5.4 | Special days table (class_id = 11, version = 0) | 139 |
| 4.5.5 | Activity calendar (class_id = 20, version = 0) | 140 |
| 4.5.6 | Register monitor (class_id = 21, version = 0) | 142 |
| 4.5.7 | Single action schedule (class_id = 22, version = 0) | 144 |
| 4.5.8 | Disconnect control (class_id = 70, version = 0) | 145 |
| 4.5.9 | Limiter (class_id = 71, version = 0) | 147 |
| 4.5.10 | Parameter monitor (class_id = 65, version = 0) | 150 |
| 4.5.11 | Sensor manager interface class | 151 |
| 4.5.11.1 | General | 151 |
| 4.5.11.2 | Sensor manager (class_id = 67, version = 0) | 152 |
| 4.5.11.3 | Example for absolute pressure sensor | 154 |
| 4.5.12 | Arbitrator | 155 |
| 4.5.12.1 | Overview | 155 |
| 4.5.12.2 | Arbitrator (class_id = 68, version = 0) | 156 |
| 4.5.13 | Modelling examples: tariffication and billing | 159 |

| | | |
|-------------|---|-----|
| 4.6 | Payment metering related interface classes | 161 |
| 4.6.1 | Overview of the COSEM accounting model | 161 |
| 4.6.2 | Account (class_id = 111, version = 0) | 163 |
| 4.6.3 | Credit interface class | 172 |
| 4.6.3.1 | General | 172 |
| 4.6.3.2 | Credit states | 173 |
| 4.6.3.3 | Current credit status flags | 175 |
| 4.6.3.4 | Credit (class_id = 112, version = 0) | 177 |
| 4.6.3.5 | Additional notes..... | 181 |
| 4.6.4 | Charge (class_id = 113, version = 0) | 183 |
| 4.6.5 | Token gateway (class_id = 115, version = 0) | 189 |
| 4.7 | Interface classes for setting up data exchange via local ports and modems | 191 |
| 4.7.1 | IEC local port setup (class_id = 19, version = 1)..... | 191 |
| 4.7.2 | IEC HDLC setup (class_id = 23, version = 1) | 192 |
| 4.7.3 | IEC twisted pair (1) setup (class_id = 24, version = 1) | 194 |
| 4.7.3.1 | Overview | 194 |
| 4.7.3.2 | IEC twisted pair (1) setup (class_id = 24, version = 1) | 195 |
| 4.7.3.3 | MAC address..... | 196 |
| 4.7.3.4 | Fatal error register..... | 196 |
| 4.7.4 | Modem configuration (class_id = 27, version = 1) | 197 |
| 4.7.5 | Auto answer (class_id = 28, version = 2) | 198 |
| 4.7.6 | Auto connect (class_id = 29, version = 2) | 201 |
| 4.7.7 | GPRS modem setup (class_id = 45, version = 0) | 203 |
| 4.7.8 | GSM diagnostic (class_id = 47, version = 0)..... | 204 |
| 4.8 | Interface classes for setting up data exchange via M-Bus | 206 |
| 4.8.1 | Overview | 206 |
| 4.8.2 | M-Bus slave port setup (class_id = 25, version = 0) | 206 |
| 4.8.3 | M-Bus client (class_id = 72, version = 1) | 207 |
| 4.8.4 | Wireless Mode Q channel (class_id = 73, version = 1)..... | 212 |
| 4.8.5 | M-Bus master port setup (class_id = 74, version = 0)..... | 212 |
| 4.8.6 | DLMS/COSEM server M-Bus port setup (class_id = 76, version = 0) | 213 |
| 4.8.7 | M-Bus diagnostic (class_id = 77, version = 0) | 215 |
| 4.9 | Interface classes for setting up data exchange over the Internet | 218 |
| 4.9.1 | TCP-UDP setup (class_id = 41, version = 0) | 218 |
| 4.9.2 | IPv4 setup (class_id = 42, version = 0) | 219 |
| 4.9.3 | IPv6 setup (class_id = 48, version = 0) | 222 |
| 4.9.4 | MAC address setup (class_id = 43, version = 0) | 224 |
| 4.9.5 | PPP setup (class_id = 44, version = 0) | 225 |
| 4.9.6 | SMTP setup (class_id = 46, version = 0) | 229 |
| 4.10 | Interface classes for setting up data exchange using S-FSK PLC | 230 |
| 4.10.1 | General..... | 230 |
| 4.10.2 | Overview..... | 230 |
| 4.10.3 | S-FSK Phy&MAC set-up (class_id = 50, version = 1) | 232 |
| 4.10.4 | S-FSK Active initiator (class_id = 51, version = 0) | 237 |
| 4.10.5 | S-FSK MAC synchronization timeouts (class_id = 52, version = 0)..... | 238 |

| | | |
|---------|---|-----|
| 4.10.6 | S-FSK MAC counters (class_id = 53, version = 0)..... | 240 |
| 4.10.7 | IEC 61334-4-32 LLC setup (class_id = 55, version = 1) | 244 |
| 4.10.8 | S-FSK Reporting system list (class_id = 56, version = 0) | 245 |
| 4.11 | Interface classes for setting up the LLC layer for ISO/IEC 8802-2..... | 246 |
| 4.11.1 | General..... | 246 |
| 4.11.2 | ISO/IEC 8802-2 LLC Type 1 setup (class_id = 57, version = 0) | 246 |
| 4.11.3 | ISO/IEC 8802-2 LLC Type 2 setup (class_id = 58, version = 0) | 246 |
| 4.11.4 | ISO/IEC 8802-2 LLC Type 3 setup (class_id = 59, version = 0) | 248 |
| 4.12 | Interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for PRIME networks | 250 |
| 4.12.1 | Overview..... | 250 |
| 4.12.2 | Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes..... | 251 |
| 4.12.3 | 61334-4-32 LLC SSCS setup (class_id = 80, version = 0) | 253 |
| 4.12.4 | PRIME NB OFDM PLC Physical layer parameters..... | 253 |
| 4.12.5 | PRIME NB OFDM PLC Physical layer counters (class_id = 81, version = 0)..... | 253 |
| 4.12.6 | PRIME NB OFDM PLC MAC setup (class_id = 82, version = 0) | 254 |
| 4.12.7 | PRIME NB OFDM PLC MAC functional parameters (class_id = 83 version = 0)..... | 256 |
| 4.12.8 | PRIME NB OFDM PLC MAC counters (class_id = 84, version = 0) | 257 |
| 4.12.9 | PRIME NB OFDM PLC MAC network administration data (class_id = 85, version = 0) | 258 |
| 4.12.10 | PRIME NB OFDM PLC MAC address setup (class_id = 43, version = 0) | 261 |
| 4.12.11 | PRIME NB OFDM PLC Application identification (class_id = 86, version = 0)..... | 262 |
| 4.13 | Interface classes for setting up and managing the DLMS/COSEM narrowband OFDM PLC profile for G3-PLC networks..... | 263 |
| 4.13.1 | Overview..... | 263 |
| 4.13.2 | Mapping of G3-PLC PIB attributes to COSEM IC attributes | 263 |
| 4.13.3 | G3-PLC MAC layer counters (class_id = 90, version = 1) | 265 |
| 4.13.4 | G3-PLC MAC setup (class_id = 91, version = 1) | 266 |
| 4.13.5 | G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 1) | 272 |
| 4.14 | ZigBee® setup classes | 278 |
| 4.14.1 | Overview..... | 278 |
| 4.14.2 | ZigBee® SAS startup (class_id = 101, version = 0)..... | 279 |
| 4.14.3 | ZigBee® SAS join (class_id = 102, version = 0) | 281 |
| 4.14.4 | ZigBee® SAS APS fragmentation (class_id = 103, version = 0) | 282 |
| 4.14.5 | ZigBee® network control (class_id = 104, version = 0) | 283 |
| 4.14.6 | ZigBee® tunnel setup (class_id = 105, version = 0) | 290 |
| 5 | Maintenance of the interface classes..... | 291 |
| 5.1 | New versions of interface classes | 291 |
| 5.2 | New interface classes | 291 |
| 5.3 | Removal of interface classes | 291 |
| 5.4 | Previous versions of interface classes..... | 291 |

| | | |
|--------|---|-----|
| 5.4.1 | General | 291 |
| 5.4.2 | Profile generic (class_id = 7, version = 0) | 292 |
| 5.4.3 | Association SN (class_id = 12, version = 0) | 295 |
| 5.4.4 | Association SN (class_id = 12, version = 1) | 297 |
| 5.4.5 | Association SN (class_id = 12, version = 2) | 299 |
| 5.4.6 | Association SN (class_id = 12, version = 3) | 302 |
| 5.4.7 | Association LN (class_id = 15, version = 0) | 307 |
| 5.4.8 | Association LN (class_id = 15, version = 1) | 312 |
| 5.4.9 | Association LN (class_id = 15, version = 2) | 317 |
| 5.4.10 | Security setup (class_id = 64, version = 0) | 322 |
| 5.4.11 | IEC local port setup (class_id = 19, version = 0) | 324 |
| 5.4.12 | IEC HDLC setup, (class_id = 23, version = 0) | 325 |
| 5.4.13 | IEC twisted pair (1) setup (class_id = 24, version = 0) | 327 |
| 5.4.14 | PSTN modem configuration (class_id = 27, version = 0) | 328 |
| 5.4.15 | Auto answer (class_id = 28, version = 0) | 330 |
| 5.4.16 | PSTN auto dial (class_id = 29, version = 0) | 331 |
| 5.4.17 | Auto connect (class_id = 29, version = 1) | 332 |
| 5.4.18 | S-FSK Phy&MAC setup (class_id = 50, version = 0) | 333 |
| 5.4.19 | S-FSK IEC 61334-4-32 LLC setup (class_id = 55, version = 0) | 337 |
| 5.4.20 | M-Bus client (class_id = 72, version = 0) | 338 |
| 5.4.21 | G3 NB OFDM PLC MAC layer counters (class_id = 90, version = 0) | 342 |
| 5.4.22 | G3 NB OFDM PLC MAC setup (class_id = 91, version = 0) | 344 |
| 5.4.23 | G3 NB OFDM PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 0) | 347 |
| 6 | Relation to OBIS | 352 |
| 6.1 | General | 352 |
| 6.2 | Abstract COSEM objects | 352 |
| 6.2.1 | Use of value group C | 352 |
| 6.2.2 | Data of historical billing periods | 354 |
| 6.2.3 | Billing period values / reset counter entries | 355 |
| 6.2.4 | Other abstract general purpose OBIS codes | 355 |
| 6.2.5 | Clock objects (class_id = 8) | 356 |
| 6.2.6 | Modem configuration and related objects | 356 |
| 6.2.7 | Script table objects (class_id = 9) | 356 |
| 6.2.8 | Special days table objects (class_id = 11) | 357 |
| 6.2.9 | Schedule objects (class_id = 10) | 357 |
| 6.2.10 | Activity calendar objects (class_id = 20) | 357 |
| 6.2.11 | Register activation objects (class_id = 6) | 358 |
| 6.2.12 | Single action schedule objects (class_id = 22) | 358 |
| 6.2.13 | Register monitor objects (class_id = 21) | 358 |
| 6.2.14 | Parameter monitor objects (class_id = 65) | 358 |
| 6.2.15 | Limiter objects (class_id = 71) | 359 |
| 6.2.16 | Payment metering related objects | 359 |
| 6.2.17 | IEC local port setup objects (class_id = 19) | 359 |
| 6.2.18 | Standard readout profile objects (class_id = 7) | 360 |

| | | | |
|-----------------------|------------|-------------------------|-------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 6/492 |
|-----------------------|------------|-------------------------|-------|

| | | |
|--------|--|-----|
| 6.2.19 | IEC HDLC setup objects (class_id = 23)..... | 360 |
| 6.2.20 | IEC twisted pair (1) setup objects (class_id = 24)..... | 360 |
| 6.2.21 | Objects related to data exchange over M-Bus | 361 |
| 6.2.22 | Objects to set up data exchange over the Internet | 362 |
| 6.2.23 | Objects for setting up data exchange using S-FSK PLC | 363 |
| 6.2.24 | Objects for setting up the ISO/IEC 8802-2 LLC layer | 363 |
| 6.2.25 | Objects for data exchange using narrowband OFDM PLC for PRIME networks..... | 364 |
| 6.2.26 | Objects for data exchange using narrow-band OFDM PLC for G3-PLC networks | 365 |
| 6.2.27 | ZigBee® setup objects | 365 |
| 6.2.28 | Association objects (class_id = 12, 15)..... | 365 |
| 6.2.29 | SAP assignment object (class_id = 17)..... | 365 |
| 6.2.30 | COSEM logical device name object..... | 366 |
| 6.2.31 | Information security related objects | 366 |
| 6.2.32 | Image transfer objects (class_id = 18) | 366 |
| 6.2.33 | Utility table objects (class_id = 26)..... | 367 |
| 6.2.34 | Compact data objects (class_id = 62) | 367 |
| 6.2.35 | Device ID objects..... | 367 |
| 6.2.36 | Metering point ID objects..... | 368 |
| 6.2.37 | Parameter changes and calibration objects | 368 |
| 6.2.38 | I/O control signal objects | 368 |
| 6.2.39 | Disconnect control objects (class_id = 70)..... | 369 |
| 6.2.40 | Arbitrator objects (class_id = 68) | 369 |
| 6.2.41 | Status of internal control signals objects..... | 369 |
| 6.2.42 | Internal operating status objects | 369 |
| 6.2.43 | Battery entries objects..... | 370 |
| 6.2.44 | Power failure monitoring objects | 370 |
| 6.2.45 | Operating time objects | 371 |
| 6.2.46 | Environment related parameters objects | 371 |
| 6.2.47 | Status register objects | 371 |
| 6.2.48 | Event code objects | 371 |
| 6.2.49 | Communication port log parameter objects | 372 |
| 6.2.50 | Consumer message objects..... | 372 |
| 6.2.51 | Currently active tariff objects | 372 |
| 6.2.52 | Event counter objects | 372 |
| 6.2.53 | Profile entry digital signature objects | 372 |
| 6.2.54 | Meter tamper event related objects..... | 373 |
| 6.2.55 | Error register objects | 373 |
| 6.2.56 | Alarm register, Alarm filter and Alarm descriptor objects | 374 |
| 6.2.57 | General list objects | 374 |
| 6.2.58 | Event log objects | 375 |
| 6.2.59 | Inactive objects | 375 |
| 6.3 | Electricity related COSEM objects | 376 |
| 6.3.1 | Value group D definitions | 376 |
| 6.3.2 | Electricity ID numbers | 376 |

| | | |
|---------|--|-----|
| 6.3.3 | Billing period values / reset counter entries | 376 |
| 6.3.4 | Other electricity related general purpose objects..... | 377 |
| 6.3.5 | Measurement algorithm..... | 378 |
| 6.3.6 | Metering point ID (electricity related)..... | 379 |
| 6.3.7 | Electricity related status objects | 379 |
| 6.3.8 | List objects – Electricity (class_id = 7) | 380 |
| 6.3.9 | Threshold values | 380 |
| 6.3.10 | Register monitor objects (class_id = 21) | 381 |
| 6.4 | Gas related COSEM objects..... | 382 |
| 6.4.1 | General..... | 382 |
| 6.4.2 | Gas ID numbers | 382 |
| 6.4.3 | Billing period values / reset counter entries | 382 |
| 6.4.4 | Other gas related general purpose objects | 382 |
| 6.4.5 | Gas related internal operating status objects | 385 |
| 6.4.6 | Measured values | 385 |
| 6.4.6.1 | Indexes and index differences | 385 |
| 6.4.6.2 | Flow rate..... | 386 |
| 6.4.6.3 | Process values | 386 |
| 6.4.7 | Conversion related factors and coefficients | 386 |
| 6.4.8 | Calculation methods | 387 |
| 6.4.9 | Natural gas analysis | 387 |
| 6.4.10 | List objects – Gas (class_id = 7) | 388 |
| 6.5 | Coding of OBIS identifications | 388 |
| 7 | COSEM Object Identification System (OBIS) | 389 |
| 7.1 | Scope | 389 |
| 7.2 | OBIS code structure..... | 389 |
| 7.2.1 | Value groups and their use | 389 |
| 7.2.2 | Manufacturer specific codes..... | 390 |
| 7.2.3 | Reserved ranges | 390 |
| 7.2.4 | Summary of rules for manufacturer, utility, consortia and country specific codes..... | 390 |
| 7.2.5 | Standard object codes..... | 391 |
| 7.3 | Value group definitions – overview | 391 |
| 7.3.1 | Value group A..... | 391 |
| 7.3.2 | Value group B..... | 391 |
| 7.3.3 | Value group C..... | 392 |
| 7.3.3.1 | General..... | 392 |
| 7.3.3.2 | Abstract objects..... | 392 |
| 7.3.4 | Value group D..... | 393 |
| 7.3.4.1 | General..... | 393 |
| 7.3.4.2 | Consortia specific identifiers | 393 |
| 7.3.4.3 | Country specific identifiers | 393 |
| 7.3.4.4 | Identification of general and service entry objects | 395 |
| 7.3.5 | Value group E | 395 |
| 7.3.6 | Value group F | 396 |

| | | | |
|-----|---------|--|-----|
| | 7.3.6.1 | General..... | 396 |
| | 7.3.6.2 | Identification of billing periods | 396 |
| 7.4 | | Abstract objects (Value group A = 0) | 396 |
| | 7.4.1 | General and service entry objects – Abstract | 396 |
| | 7.4.2 | Error registers, alarm registers / filters / descriptor objects – Abstract | 400 |
| | 7.4.3 | List objects – Abstract | 400 |
| | 7.4.4 | Register table objects – Abstract..... | 401 |
| | 7.4.5 | Data profile objects – Abstract | 401 |
| 7.5 | | Electricity (Value group A = 1) | 402 |
| | 7.5.1 | Value group C codes – Electricity | 402 |
| | 7.5.2 | Value group D codes – Electricity | 404 |
| | 7.5.2.1 | Processing of measurement values | 404 |
| | 7.5.2.2 | Use of value group D for identification of other objects | 406 |
| | 7.5.3 | Value group E codes – Electricity..... | 406 |
| | 7.5.3.1 | General..... | 406 |
| | 7.5.3.2 | Tariff rates | 406 |
| | 7.5.3.3 | Harmonics | 407 |
| | 7.5.3.4 | Phase angles..... | 408 |
| | 7.5.3.5 | Transformer and line loss quantities | 408 |
| | 7.5.3.6 | UNIPEDE voltage dips | 411 |
| | 7.5.3.7 | Use of value group E for the identification of other objects..... | 411 |
| | 7.5.4 | Value group F codes – Electricity..... | 411 |
| | 7.5.4.1 | Billing periods | 411 |
| | 7.5.4.2 | Multiple thresholds | 412 |
| | 7.5.5 | OBIS codes – Electricity..... | 413 |
| | 7.5.5.1 | General and service entry objects – Electricity | 413 |
| | 7.5.5.2 | Error register objects – Electricity | 416 |
| | 7.5.5.3 | List objects – Electricity | 416 |
| | 7.5.5.4 | Data profile objects – Electricity | 416 |
| | 7.5.5.5 | Register table objects – Electricity | 417 |
| 7.6 | | Heat Cost Allocators (Value group A = 4)..... | 418 |
| | 7.6.1 | General..... | 418 |
| | 7.6.2 | Value group C codes – HCA | 418 |
| | 7.6.3 | Value group D codes – HCA | 419 |
| | 7.6.4 | OBIS codes – HCA..... | 420 |
| | 7.6.4.1 | General and service entry objects – HCA..... | 420 |
| | 7.6.4.2 | Error register objects – HCA | 421 |
| | 7.6.4.3 | Data profile objects – HCA..... | 421 |
| | 7.6.4.4 | OBIS codes for HCA related objects (examples)..... | 422 |
| 7.7 | | Thermal energy (Value group A = 5 or A = 6)..... | 423 |
| | 7.7.1 | General..... | 423 |
| | 7.7.2 | Value group C codes – Thermal energy | 423 |
| | 7.7.3 | Value group D codes – Thermal energy | 424 |
| | 7.7.4 | OBIS codes – Thermal energy | 425 |

| | | |
|-----------|---|-----|
| 7.7.4.1 | General and service entry objects – Thermal energy | 425 |
| 7.7.4.2 | Error register objects – Thermal energy..... | 427 |
| 7.7.4.3 | Data profile objects – Thermal energy | 427 |
| 7.7.4.4 | OBIS codes for Thermal energy related objects (examples) | 428 |
| 7.8 | Gas (Value group A = 7) | 429 |
| 7.8.1 | General introduction to gas measurement | 429 |
| 7.8.1.1 | Overview | 429 |
| 7.8.1.2 | Typical gas metering installations | 429 |
| 7.8.1.2.1 | Residential application..... | 429 |
| 7.8.1.2.2 | Industrial application | 430 |
| 7.8.1.2.3 | Gas transport application..... | 430 |
| 7.8.1.3 | Gas volume conversion..... | 431 |
| 7.8.1.3.1 | General | 431 |
| 7.8.1.3.2 | Step 1: Error correction (optional)..... | 432 |
| 7.8.1.3.3 | Step 2: Volume conversion to base conditions..... | 432 |
| 7.8.1.3.4 | Step 3: Energy conversion..... | 432 |
| 7.8.1.3.5 | Model of data flow for volume conversion and energy calculation | 432 |
| 7.8.1.4 | Data logging | 434 |
| 7.8.1.4.1 | General | 434 |
| 7.8.1.4.2 | Time bound processing | 435 |
| 7.8.1.4.3 | Gas day..... | 435 |
| 7.8.1.4.4 | Data profiles..... | 435 |
| 7.8.2 | Value group C codes – Gas | 436 |
| 7.8.3 | Value group D codes – Gas | 438 |
| 7.8.3.1 | General..... | 438 |
| 7.8.3.2 | Gas indexes and index differences | 438 |
| 7.8.3.3 | Flow rate..... | 442 |
| 7.8.3.4 | Process values | 445 |
| 7.8.3.5 | Conversion related factors and coefficients | 448 |
| 7.8.3.6 | Natural gas analysis values | 448 |
| 7.8.4 | Value group E codes – Gas | 450 |
| 7.8.4.1 | General..... | 450 |
| 7.8.4.2 | Indexes and index differences – Tariff rates | 450 |
| 7.8.4.3 | Flow rate..... | 450 |
| 7.8.4.4 | Process values | 450 |
| 7.8.4.5 | Conversion related factors and coefficients – Averages..... | 450 |
| 7.8.4.6 | Calculation methods..... | 451 |
| 7.8.4.7 | Natural gas analysis values – Averages | 452 |
| 7.8.5 | Value group F codes – Gas..... | 453 |
| 7.8.6 | OBIS codes – Gas..... | 453 |
| 7.8.6.1 | General and service entry objects – Gas | 453 |
| 7.8.6.2 | Error register objects – Gas | 460 |
| 7.8.6.3 | List objects – Gas..... | 460 |
| 7.8.6.4 | Data profile objects – Gas | 461 |

| | | |
|---|---|-----|
| 7.9 | Water (Value group A = 8 and A = 9) | 462 |
| 7.9.1 | General..... | 462 |
| 7.9.2 | Value group C codes – Water | 462 |
| 7.9.3 | Value group D codes – Water | 462 |
| 7.9.4 | OBIS codes – Water..... | 463 |
| 7.9.4.1 | General and service entry objects – Water | 463 |
| 7.9.4.2 | Error register objects – Water | 464 |
| 7.9.4.3 | Data profile objects – Water..... | 464 |
| 7.9.4.4 | OBIS codes for water related objects (examples)..... | 464 |
| 7.10 | Other media (Value group A= 15)..... | 465 |
| 7.10.1 | General..... | 465 |
| 7.10.2 | Value group C codes – Other media..... | 465 |
| 7.10.3 | Value group D codes – Other media..... | 465 |
| 7.10.4 | Value group E codes – Other media | 465 |
| 7.10.5 | Value group F codes – Other media | 465 |
| 7.11 | Code presentation..... | 466 |
| 7.11.1 | Reduced ID codes (e.g. for IEC 62056-21)..... | 466 |
| 7.11.2 | Display | 466 |
| 7.11.3 | Special handling of value group F | 466 |
| 7.11.4 | COSEM | 467 |
| Annex A (Informative) | Additional information on Auto answer and Auto connect ICs | 468 |
| Annex B (Informative) | Additional information to M-Bus client (class_id = 72, version 1)..... | 470 |
| Annex C (Informative) | Additional information on IPv6 setup class (class_id = 48, version = 0) | 472 |
| C.1 | Introduction | 472 |
| C.2 | IPv6 addressing | 472 |
| C.3 | IPv6 header format | 473 |
| C.4 | IPv6 header extensions | 474 |
| C.4.1 | Overview..... | 474 |
| C.4.2 | Hop-by-Hop options | 475 |
| C.4.3 | Destination options | 475 |
| C.4.4 | Routing options..... | 475 |
| C.4.5 | Fragment options | 476 |
| C.4.6 | Security options | 476 |
| Annex D (Informative) | Overview of the narrow-band OFDM PLC technology for PRIME networks | 477 |
| Annex E (informative) | Overview of the narrow-band OFDM PLC technology for G3-PLC networks | 478 |
| Annex F (informative) | Bibliography | 479 |
| Index | 480 | |
| Figure 1 – The three steps approach of DLMS/COSEM: Modelling – Messaging – Transporting | 21 | |
| Figure 2 – The meaning of the definitions concerning the Image | 25 | |
| Figure 3 – An interface class and its instances | 39 | |
| Figure 4 – The COSEM server model | 48 | |

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 11/492 |
|-----------------------|------------|-------------------------|--------|

| | |
|--|-----|
| Figure 5 – Combined metering device | 48 |
| Figure 6 – Overview of the interface classes – Part 1 | 51 |
| Figure 7 – Overview of the interface classes – Part 2 | 52 |
| Figure 8 – The time attributes when measuring sliding demand..... | 61 |
| Figure 9 – The attributes in the case of block demand | 61 |
| Figure 10 – The attributes in the case of sliding demand (number of periods = 3)..... | 62 |
| Figure 11 – Image transfer process flow chart..... | 100 |
| Figure 12 – COSEM model of push operation | 109 |
| Figure 13 – Push windows and delays..... | 110 |
| Figure 14 – COSEM model of data protection | 116 |
| Figure 15 – Example: Read <i>protection_buffer</i> attribute | 118 |
| Figure 16 – The generalized time concept | 132 |
| Figure 17 – State diagram of the Disconnect control IC | 145 |
| Figure 18 – Definition of upper and lower thresholds..... | 154 |
| Figure 19 – COSEM tariffication model (example)..... | 159 |
| Figure 20 – COSEM billing model (example)..... | 160 |
| Figure 21 – Outline Account model | 162 |
| Figure 22 – Diagram of attribute relationships | 163 |
| Figure 23 – Credit States when priority >0 | 173 |
| Figure 24 – Operation of <i>current_credit_status</i> flags | 176 |
| Figure 25 – Interaction of <i>current_credit_amount</i> and <i>available_credit</i> with Token “Credit” and Emergency “Credit” | 182 |
| Figure 26 – Object model of DLMS/COSEM servers | 230 |
| Figure 27 – Object model of DLMS/COSEM servers | 250 |
| Figure 28 – Example of a ZigBee® network | 279 |
| Figure 29 – Data of historical billing periods – example with module 12, VZ = 5 | 354 |
| Figure 30 – Quadrant definitions for active and reactive power..... | 403 |
| Figure 31 – Model of the line and the transformer for calculation of loss quantities | 408 |
| Figure 32 – Residential gas metering installation | 429 |
| Figure 33 – Industrial gas metering installation (single stream)..... | 430 |
| Figure 34 – City gate or border crossing installation (multi stream)..... | 431 |
| Figure 35 – Data flow of volume conversion and energy calculation..... | 433 |
| Figure 36 – Reduced ID code presentation..... | 466 |
| Figure A. 1 – Network connectivity example for a GSM/GPRS network..... | 468 |
| Figure B. 1 – Encryption key status diagram..... | 470 |
| Figure C. 1 – IPv6 address formats..... | 472 |
| Figure C. 2 – IPv6 header format | 473 |
| Figure C. 3 – Traffic class parameter format..... | 474 |
| Table 1 – Reserved base_names for SN referencing | 40 |

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 12/492 |
|-----------------------|------------|-------------------------|--------|

| | |
|---|------------|
| Table 2 – Common data types | 43 |
| Table 3 – List of interface classes by class_id | 53 |
| Table 4 – Enumerated values for physical units | 58 |
| Table 5 – Examples for scaler_unit | 60 |
| Table 6 – Daily billing data | 78 |
| Table 7 – Attributes of the “Compact data” object..... | 78 |
| Table 8 – A-XDR encoding of the data (SEQUENCE OF Get-Data-Result)..... | 79 |
| Table 9 – Diagnostic and Alarm data | 79 |
| Table 10 – Attributes of the “Compact data” object..... | 80 |
| Table 11 – Encoding the data read from the buffer attribute of a “Profile generic” object | 80 |
| Table 12 – Logbook data..... | 80 |
| Table 13 – Attributes of the “Compact data” object..... | 81 |
| Table 14 – Attributes of the “Compact data” object..... | 81 |
| Table 15 – A-XDR encoding of the data read from the <i>buffer</i> attribute..... | 81 |
| Table 16 – Encoding of selective access parameters with data_index..... | 114 |
| Table 17 – Key information required to establish data protection keys | 126 |
| Table 18 – Protection parameters of <i>protection_parameters_get</i> attribute..... | 127 |
| Table 19 – Protection parameters of <i>protection_parameters_set</i> attribute | 128 |
| Table 20 – Protection parameters of <i>get_protected_attributes</i> method | 129 |
| Table 21 – Protection parameters of <i>set_protected_attributes</i> method | 130 |
| Table 22 – Protection parameters of <i>invoke_protected_method</i> method | 131 |
| Table 23 – Schedule..... | 136 |
| Table 24 – Special days table | 136 |
| Table 25 – Disconnect control IC – states and state transitions | 146 |
| Table 26 – Explicit presentation of threshold value arrays..... | 154 |
| Table 27 – Explicit presentation of action_sets | 155 |
| Table 28 – Credit states | 173 |
| Table 29 – Credit state transitions..... | 174 |
| Table 30 – ADS address elements..... | 196 |
| Table 31 – Fatal error register | 196 |
| Table 32 – Mapping IEC 61334-4-512:2001 MIB variables to COSEM IC attributes / methods | 231 |
| Table 33 – MAC addresses in the S-FSK profile | 236 |
| Table 34 – Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes | 251 |
| Table 35 – Mapping of G3-PLC IB attributes to COSEM IC attributes | 263 |
| Table 36 – Use of ZigBee® setup COSEM interface classes | 279 |
| Table 37 – Use of value group C for abstract objects in the COSEM context | 353 |
| Table 38 – Representation of various values by appropriate ICs | 376 |
| Table 39 – Measuring algorithms – enumerated values | 378 |
| Table 40 – Threshold objects, electricity | 381 |
| Table 41 – Register monitor objects, electricity | 381 |

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 13/492 |
|-----------------------|------------|-------------------------|--------|

| | |
|--|-----|
| Table 42 – Digital / Analogue output configurations – enumerated values | 383 |
| Table 43 – Indexes and index differences..... | 385 |
| Table 44 – Flow rate | 386 |
| Table 45 – Process values | 386 |
| Table 46 – Conversion related factors and coefficients | 387 |
| Table 47 – Calculation methods | 387 |
| Table 48 – Natural gas analysis | 388 |
| Table 49 – OBIS code structure and use of value groups | 389 |
| Table 50 – Rules for manufacturer, utility, consortia and country specific codes | 390 |
| Table 51 – Value group A codes | 391 |
| Table 52 – Value group B codes | 392 |
| Table 53 – Value group C codes – Abstract objects | 392 |
| Table 54 – Value group D codes – Consortia specific identifiers..... | 393 |
| Table 55 – Value group D codes – Country specific identifiers..... | 394 |
| Table 56 – OBIS codes for general and service entry objects | 396 |
| Table 57 – OBIS codes for error registers, alarm registers and alarm filters – Abstract..... | 400 |
| Table 58 – OBIS codes for list objects – Abstract..... | 401 |
| Table 59 – OBIS codes for Register table objects – Abstract | 401 |
| Table 60 – OBIS codes for data profile objects – Abstract | 401 |
| Table 61 – Value group C codes – Electricity | 402 |
| Table 62 – Value group D codes – Electricity | 404 |
| Table 63 – Value group E codes – Electricity – Tariff rates | 407 |
| Table 64 – Value group E codes – Electricity – Harmonics | 407 |
| Table 65 – Value group E codes – Electricity – Extended phase angle measurement | 408 |
| Table 66 – Value group E codes – Electricity – Transformer and line losses..... | 409 |
| Table 67 – Value group E codes – Electricity – UNIPEDE voltage dips | 411 |
| Table 68 – OBIS codes for general and service entry objects – Electricity | 413 |
| Table 69 – OBIS codes for error register objects – Electricity | 416 |
| Table 70 – OBIS codes for list objects – Electricity..... | 416 |
| Table 71 – OBIS codes for data profile objects – Electricity | 417 |
| Table 72 – OBIS codes for Register table objects – Electricity | 417 |
| Table 73 – Value group C codes – HCA | 418 |
| Table 74 – Value group D codes – HCA | 419 |
| Table 75 – OBIS codes for general and service entry objects – HCA | 420 |
| Table 76 – OBIS codes for error register objects – HCA | 421 |
| Table 77 – OBIS codes for data profile objects – HCA | 421 |
| Table 78 – OBIS codes for HCA related objects (examples) | 422 |
| Table 79 – Value group C codes – Thermal energy | 423 |
| Table 80 – Value group D codes – Thermal energy | 424 |
| Table 81 – OBIS codes for general and service entry objects – Thermal energy..... | 425 |

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 14/492 |
|-----------------------|------------|-------------------------|--------|

| | |
|--|-----|
| Table 82 – OBIS codes for error register objects – Thermal energy..... | 427 |
| Table 83 – OBIS codes for data profile objects – Thermal energy | 427 |
| Table 84 – OBIS codes for Thermal energy related objects (examples) | 428 |
| Table 85 – OBIS codes of the main objects in the gas conversion process data flow | 434 |
| Table 86 – Value group C codes – Gas | 436 |
| Table 87 – Value group D codes – Gas – Indexes and index differences | 439 |
| Table 88 – Value group D codes – Gas – Flow rate | 443 |
| Table 89 – Value group D codes – Gas – Process values..... | 445 |
| Table 90 – Value group D codes – Gas – Conversion related factors and coefficients..... | 448 |
| Table 91 – Value group D codes – Gas – Natural gas analysis values | 449 |
| Table 92 – Value group E codes – Gas – Indexes and index differences – Tariff rates | 450 |
| Table 93 – Value group E codes – Gas – Conversion related factors and coefficients..... | 451 |
| Table 94 – Value group E codes – Gas – Calculation methods..... | 452 |
| Table 95 – Value group E codes – Gas – Natural gas analysis values – Averages | 452 |
| Table 96 – OBIS codes for general and service entry objects – Gas | 453 |
| Table 97 – OBIS codes for error register objects – Gas | 460 |
| Table 98 – OBIS codes for list objects – Gas..... | 460 |
| Table 99 – OBIS codes for data profile objects – Gas | 461 |
| Table 100 – Value group C codes – Water | 462 |
| Table 101 – Value group D codes – Water | 462 |
| Table 102 – OBIS codes for general and service entry objects – Water | 463 |
| Table 103 – OBIS codes for error register objects – Water | 464 |
| Table 104 – OBIS codes for data profile objects – Water | 464 |
| Table 105 – OBIS codes for water related objects (examples)..... | 464 |
| Table 106 – Value group C codes – Other media..... | 465 |
| Table 107 – Example of display code replacement | 466 |
| Table 108 – Value group F – Billing periods | 467 |
| Table B. 1 – Encryption key is preset in the slave and cannot be changed..... | 471 |
| Table B. 2 – Encryption key is preset in the slave and new key is set after installation | 471 |
| Table B. 3 – Encryption key is not preset in the slave, but can be set, case a)..... | 471 |
| Table B. 4 – Encryption key is not preset in the slave, but can be set, case b)..... | 471 |
| Table C. 1 – IPv6 header vs. IPv6 IC | 474 |
| Table C. 2 – Optional IPv6 header extensions vs. IPv6 IC | 475 |

Foreword

Copyright

© Copyright 1997-2014 DLMS User Association.

This document is confidential. It may not be copied, nor handed over to persons outside the standardization environment.

The copyright is enforced by national and international law. The "Berne Convention for the Protection of Literary and Artistic Works", which is signed by 166 countries world-wide, and other treaties apply.

Intellectual Property Rights

The DLMS User Association (DLMS UA) draws attention to the fact that it is claimed that compliance with this document may involve the use of a patent concerning the Image transfer procedure.

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

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

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

The DLMS UA maintains on-line databases of patents relevant to their standards. Users are encouraged to consult the databases for the most up to date information concerning patents.

Acknowledgement

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

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

Status of standardization

The contents of this edition is the basis of:

- IEC 62056-6-1 Ed. 3:—, Electricity Metering Data Exchange – The DLMS/COSEM suite – Part 6-1: Object identification system (OBIS); and
- IEC 62056-6-2 Ed. 3:—, Electricity Metering Data Exchange – The DLMS/COSEM suite – Part 6-2: COSEM interface classes.

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 16/492 |
|-----------------------|------------|-------------------------|--------|

Revision history

| Version | Date | Author | Comment |
|-------------------------------|---------------------------------|---------|--|
| Release 1 | 01 April 1998 | DLMS UA | Initial version |
| First Edition | 12 Nov. 1998 | DLMS UA | Considering comments received after R1 |
| Second Edition | 03 May 1999 | DLMS UA | Major rework, classes and associations added |
| Third Edition | 29 Feb. 2000 | DLMS UA | Major rework, adapted to CDVs of IEC TC13, OBIS added |
| Fourth Edition | 25 March 2001 | DLMS UA | Considering comments to CDVs by IEC National Committees |
| Fifth Edition | 05 March 2002 | DLMS UA | Content adapted to IEC International Standards |
| Sixth Edition | 16th August 2004 | DLMS UA | Content adapted to draft IEC 62056-61 Edition 2 ¹ draft IEC 62056-62 (CD versions) and EN 13757-1:2002 |
| Seventh Edition | 12 th September 2005 | DLMS UA | Content adapted to 13/1341/CDV, draft IEC 62056-61 Edition 2, 13/1342/CDV, draft IEC 62056-62 Edition 2 and comments received on these drafts, as well as on EN 13757-1:2002 |
| Eighth Edition | 14 th September 2007 | DLMS UA | Document restructured, editorial errors in Ed. 7.0 corrected. For details, see list of main changes in Ed. 7.0. |
| Ninth Edition | 9 th February 2009 | DLMS UA | New elements for smart metering and advanced gas volume and energy conversion added. |
| Tenth Edition | 26 th August 2010 | DLMS UA | Brought in line with Green Book 9. New interface classes and new versions of interface classes added. New OBIS codes added. See the detailed list of changes below. |
| Eleventh Edition | 26th August 2013 | DLMS UA | New interface classes and new versions of existing interface classes added. New OBIS codes added. See the detailed list of changes below. |
| Twelfth Edition | 10 th September 2014 | DLMS UA | New interface classes and new versions of existing interface classes added. New OBIS codes added. See the detailed list of changes below. |
| Twelfth Edition Corrigendum 1 | 21 st December 2015 | DLMS UA | Technical and editorial corrections |
| Edition 12.1 | 21 st December 2015 | DLMS UA | Consolidated edition including the Corrigendum 1. |

List of main technical changes in Edition 12.0 (compared to Edition 11.0)

| Item | Clause | Description |
|------|--------|--|
| 1. | 2 | List of standards and RFCs referenced updated. |
| 2. | 3 | All definitions brought here. |
| 3. | 3.5 | Terms and definitions related to Payment metering interface classes added. |
| 4. | 3.6 | Terms and definitions related to the Arbitrator IC added. |
| 5. | 4.1.9 | Information security: text amended to include COSEM data protection |
| 6. | 4.2 | Overview of the COSEM interface classes: Figure 6 and Figure 7 have been updated. Table 3 has been added, listing the interface classes by class_id, complete with cross references. |
| 7. | 4.3.9 | Specification of Status mapping IC <i>mapping_table</i> attribute amended (editorial). |
| 8. | 4.3.10 | “Compact data” IC (class_id: 62) added. |
| 9. | 4.4.3 | Association SN class (class_id = 12, version 4): New version added, specifying new access rights. |
| 10. | 4.4.4 | Association LN class (class_id = 15, version 3): New version added, specifying new access rights. Specification of the <i>application_context_name</i> and <i>authentication_mechanism_name</i> attributes have been amended (editorial). These changes have also been implemented in the earlier version of this IC. |
| 11. | 4.4.7 | Security setup (class_id = 64, version = 1): New version added, supporting new security suites. |
| 12. | 4.4.8 | “Push setup” IC specification: In Table 16 editorial corrections have been made. This table is also referenced now from the new “Compact data” and the “Data protection” ICs. |
| 13. | 4.4.9 | COSEM data protection (class_id = 30): New interface class added |
| 14. | 4.5.3 | “Schedule (class_id = 10, version = 0): limitation on entries – index (9999) removed. |
| 15. | 4.5.11 | In “Sensor manager” IC, <i>processed_value_thresholds</i> attribute, text amended to remove ambiguity. |
| 16. | 4.5.12 | New interface class “Arbitrator” (class_id = 68) added. |
| 17. | 4.5.13 | Examples for modelling tariffication and billing in COSEM added. |
| 18. | 4.6 | Payment metering related interface classes: New ICs added |
| 19. | 4.8 | Two new M-B us related ICs have been added: - DLMS/COSEM server M-Bus port setup (class_id = 76), see 4.8.6; - M-Bus diagnostic (class_id = 77, version = 0), see 4.8.7. |
| 20. | 4.13 | Interface classes for setting up and managing the DLMS/COSEM narrowband OFDM PLC profile for G3-PLC networks: New version 1 of the ICs added to be in line with ITU-T G.9903:2014. |
| 21. | 5.4 | Previous versions of interface classes: Previous versions of ICs moved here. |
| 22. | 6.2.1 | Use of value group C: New allocations for payment metering ICs, “Data protection” IC and “Compact data” IC added. |
| 23. | 6.2.16 | Payment metering related objects: OBIS codes added |
| 24. | 6.2.20 | IEC twisted pair (1) setup objects (class_id = 24): New OBIS codes for Euridis readout profiles added. |
| 25. | 6.2.21 | OBIS codes for “DLMS/COSEM server M-Bus port setup” and “M-Bus diagnostic” ICs added. |
| 26. | 6.2.31 | Information security related objects - Instead of the term <i>Frame counter</i> the term <i>Invocation counter</i> is used now. - OBIS codes for “Data protection” IC added. |
| 27. | 6.2.34 | Compact data objects: OBIS codes added |
| 28. | 6.2.40 | Arbitrator objects: OBIS codes added |

| Item | Clause | Description |
|------|----------------------|--|
| 29. | 6.2.53 | Profile entry digital signature objects: OBIS codes added. |
| 30. | 7.3.1 | Value group A: - Instead of <i>Cooling and Heat</i> the term <i>Thermal energy</i> is used (throughout the document). - Value F corrected to 15. |
| 31. | 7.3.4.4 | A subclause Identification of general and service entry objects has been added. |
| 32. | 7.3.5 | Use of value group E to identify general and service entry objects has been added. |
| 33. | 7.4.1 Table 56 | OBIS code for "Arbitrator" objects added. |
| 34. | 7.4.5 | Data profile objects – Abstract: Profiles objects used in relation to payment metering have been added. |
| 35. | 7.6.4.1 Table 75 | OBIS code for Time stamp (local time) of the most recent billing period and Note b) has been added. |
| 36. | 7.7 | Thermal energy (Value group A = 5 or A = 6): Text has been renewed (earlier, it was Heat/Cooling) to be in line with EN 1434-1 and EN 1434-2. |
| 37. | 7.7.4.1 Table 81 | OBIS code for Time stamp (local time) of the most recent billing period and Note i) has been added. |
| 38. | 7.9.4.1 Table 102 | OBIS code for Time stamp (local time) of the most recent billing period and Note a) has been added. |

List of main technical changes in Edition 12.1 (compared to Edition 12.0)

| Item | Clause | Description |
|------|-----------|---|
| 1. | Fore-word | Intellectual Property Rights Statement concerning the Image Transfer mechanism added. |
| 2. | 4.1.6.1 | Add a Note at the end of the "Date" specification: "For countries not using the Gregorian calendar, Month 1 is the starting month of the calendar and the range of dayOfMonth may be different." |
| 3. | 4.4.7 | <i>key_agreement (data)</i> attribute specification. At the end of the second paragraph below Note 6 formula for <i>key_data</i> added. |
| 4. | 4.4.9.2 | <i>invoke_protected_method (data)</i> method A sentence added to cover the case when there are no method invocation return parameters. |
| 5. | 4.13.1 | A NOTE stating that version 0 of the G3-PLC setup classes is deprecated has been added. |
| 6. | 6.2.5 | A High resolution clock object has been added. |
| 7. | 6.2.20 | IEC 62056-3-1 readout parametrization objects have been added. |
| 8. | 6.3.1 | In Table 38, D = 14, 24 and D = 15, 25 have been added. |
| 9. | 7.5.2.1 | In Table 62, D = 56, Current average 4 for harmonics measurement allocated. |
| 10. | 7.5.5.1 | In Table 68, Measurement period 4, for harmonics measurement 1.b.0.8.8.VZ has been added. |

Introduction

Object modelling and data identification

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

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

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

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

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

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

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

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

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 20/492 |
|-----------------------|------------|-------------------------|--------|

1 Scope

The DLMS/COSEM specification specifies a data model and communication protocols for data exchange with metering equipment. It follows a three-step approach as illustrated in Figure 1.

Step 1, Modelling: This covers the data model of metering equipment as well as rules for data identification. The data model provides a view of the functionality of the meter, as it is available at its interface(s). It uses generic building blocks to model this functionality. The model does not cover internal, implementation-specific issues.

Step 2, Messaging: This covers the communication services and protocols for mapping the elements of the data model to application protocol data units (APDU).

Step 3, Transporting: This covers the services and protocols for the transportation of the messages through the communication channel.

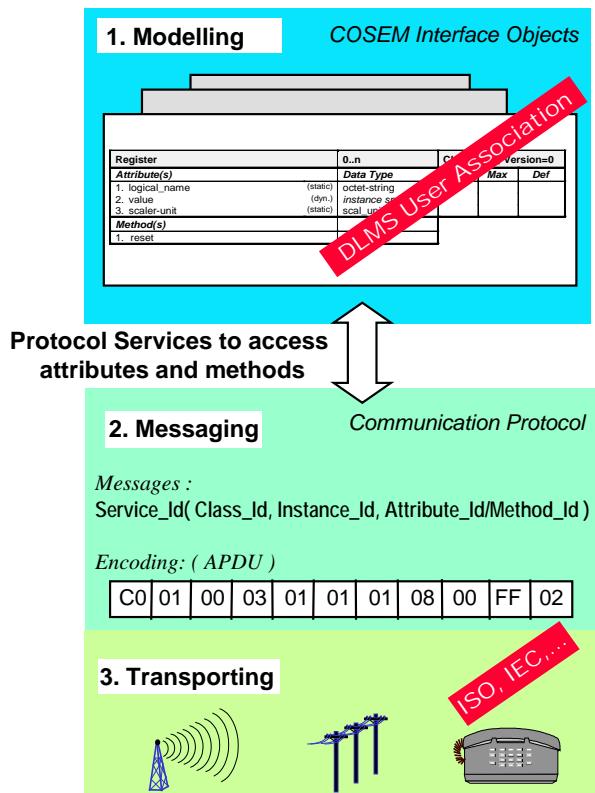


Figure 1 – The three steps approach of DLMS/COSEM: Modelling – Messaging – Transporting

Step 1 is specified in this document. It specifies the COSEM interface classes (ICs), the OBIS object identification system, and the use of interface objects for modelling the various functions of the metering equipment.

Step 2 and 3 are specified in the Green Book, DLMS UA 1000-2. It specifies communication profiles for various communication media and the protocol layers of these communication profiles. The top layer in any profile is the DLMS/COSEM application layer. It provides services to establish a logical connection between the client and the server(s). It also provides the xDLMS messaging services to access attributes and methods of the COSEM interface objects. The lower, communication profile specific protocol layers transport the information.

Rules for conformance testing are specified in the "Yellow Book", DLMS UA 1001-1 "DLMS/COSEM Conformance Test Process".

Terms are explained in the "White book" DLMS UA 1002, "COSEM Glossary of Terms".

2 Referenced documents (see also the Bibliography)

| Reference | Title |
|---|---|
| DLMS UA 1000-2 Ed. 8.1:2015 | <i>DLMS/COSEM Architecture and Protocols, the "Green Book"</i> |
| DLMS UA 1001-1, Ed. 5.0:2015 | <i>DLMS/COSEM Conformance test and certification process, the "Yellow Book"</i> |
| DLMS UA 1002, Ed. 1.0:2003 | <i>COSEM Glossary of terms, the "White Book"</i> |
| IEC/TR 61000-2-8:2002 | <i>Electromagnetic compatibility (EMC) – Part 2-8: Environment - Voltage dips and short interruptions on public electric power supply systems with statistical measurement results</i> |
| IEC 61334-4-32:1996 | <i>Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 32: Data link layer – Logical link control (LLC)</i> |
| IEC 61334-4-41:1996 | <i>Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 41: Application protocols – Distribution line message specification</i> |
| IEC 61334-4-511:2000 | <i>Distribution automation using distribution line carrier systems – Part 4-511: Data communication protocols – Systems management – CIASE protocol</i> |
| IEC 61334-4-512:2001 | <i>Distribution automation using distribution line carrier systems – Part 4-512: Data communication protocols – System management using profile 61334-5-1 – Management Information Base (MIB)</i> |
| IEC 61334-5-1:2001 | <i>Distribution automation using distribution line carrier systems – Part 5-1: Lower layer profiles – The spread frequency shift keying (S-FSK) profile</i> |
| IEC/TR 62051:1999 | <i>Electricity metering – Glossary of terms</i> |
| IEC/TR 62051-1:2004 | <i>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</i> |
| IEC 62053-23:2003 | <i>Electricity metering equipment (a.c.) – Particular requirements – Part 23: Static meters for reactive energy (classes 2 and 3)</i> |
| IEC/TR 62055-21:2005 | <i>Technical report Electricity metering - Payment systems- Part 21: Framework for standardization. 2005-8</i> |
| IEC 62056-21:2002 | <i>Electricity metering – Data exchange for meter reading, tariff and load control – Part 21: Direct local data exchange</i> |
| IEC 62056-31:1999 | <i>Electricity metering – Data exchange for meter reading, tariff and load control – Part 31: Using local area networks on twisted pair with carrier signalling</i> |
| IEC 62056-3-1:2013 | <i>Electricity metering data exchange – The DLMS/COSEM suite – Part 3-1: Use of local area networks on twisted pair with carrier signalling</i> |
| IEC 62056-46:2007 | <i>Electricity metering – Data exchange for meter reading, tariff and load control – Part 46: Data link layer using HDLC protocol</i> |
| IEC 62056-4-7:2015 | <i>Electricity metering – Data exchange for meter reading, tariff and load control – Part 47: COSEM transport layers for IPv4 networks</i> |
| IEC 62056-5-3 | <i>Electricity metering data exchange – The DLMS/COSEM suite – Part 5-3: DLMS/COSEM application layer</i> |
| IEC 62056-6-1 | <i>Electricity metering data exchange – The DLMS/COSEM suite – Part 6-1: Object identification system (OBIS)</i> |
| IEC 62056-6-2 | <i>Electricity metering data exchange – The DLMS/COSEM suite Part 6-2: COSEM interface classes</i> |
| Future IEC 62056-7-3:— | <i>Electricity metering data exchange – The DLMS/COSEM suite – Part 7-3: Wired and wireless M-Bus communication profiles for local and neighbourhood networks</i> |
| IEEE 802.15.4: 2006 also available as ISO/IEC/IEEE 8802-15-4 Ed 1.0 | <i>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) – September 2006.</i> |
| ISO/IEC 8802-2:1998 | <i>IEEE Standard for Information technology – Telecommunications and information</i> |

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 22/492 |
|-----------------------|------------|-------------------------|--------|

| Reference | Title |
|--|--|
| | <i>exchange between systems – Local and metropolitan area networks – Specific requirements – Part 2: Logical Link Control</i> |
| ISO/IEC 10646:2012 | <i>Information technology – Universal Coded Character Set (UCS)</i> |
| ISO/IEC/IEEE 60559:2011 | <i>Information technology – Microprocessor Systems – Floating-Point arithmetic</i> |
| ISO 4217 | <i>Codes for the representation of currencies and funds</i> |
| ITU-T G.9901:2012 | <i>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</i> |
| ITU-T G.9903 Amd. 1:2013 | <i>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</i> |
| ITU-T G.9901:2014 | <i>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</i> |
| ITU-T G.9903:2014 | <i>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</i> |
| ITU-T G.9904:2012 | <i>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</i> |
| EN 834:1994 | <i>Heat cost allocators for the determination of the consumption of room heating radiators – Appliances with electrical energy supply</i> |
| EN 1434-1:2015 | <i>Heat meters – Part 1: General requirements</i> |
| EN 1434-2:2015 | <i>Heat meters – Part 2: Constructional requirements</i> |
| EN 13757-1:2014 | <i>Communication system for meters – Part 1: Data exchange</i> |
| EN 13757-2:2004 | <i>Communication systems for meters and remote reading of meters – Part 2: Physical and link layer</i> |
| EN 13757-3:2004 | <i>Communication systems for and remote reading of meters – Part 3: Dedicated application layer</i> NOTE This standard is referenced in the “M-Bus client setup” interface class version 0. |
| EN 13757-3:2013 | <i>Communication systems for and remote reading of meters – Part 3: Dedicated application layer</i> NOTE This standard is referenced in the M-Bus client setup interface class version 1. |
| EN 13757-4:2013 | <i>Communication system for and remote reading of meters – Part 4: Wireless meter (Radio meter reading for operation in SRD bands)</i> |
| EN 13757-5:2015 | <i>Communication systems for meters – Part 5: Wireless M-Bus relaying</i> |
| CLC/TS 52056-8-4:2015 | <i>ELECTRICITY METERING DATA EXCHANGE – THE DLMS/COSEM SUITE – Part 8-4: The narrowband OFDM PLC profile for PRIME networks</i> |
| CLC/TS 52056-8-5:2015 | <i>ELECTRICITY METERING DATA EXCHANGE – THE DLMS/COSEM SUITE – Part 8-5: The narrowband OFDM PLC profile for G3-PLC networks</i> |
| ETSI GSM 05.08 | <i>Digital cellular telecommunications system (Phase 2+); Radio subsystem link control</i> |
| ANSI C12.19:1997 | <i>IEEE 1377:1997, Utility industry end device data tables</i> |
| ZigBee® 053474 | <i>ZigBee® Specification. The specification can be downloaded free of charge from http://www.zigbee.org/Standards/ZigBeeSmartEnergy/Specification.aspx</i> |
| <i>The following RFCs are available online from the Internet Engineering Task Force (IETF):</i> | |

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 23/492 |
|-----------------------|------------|-------------------------|--------|

| Reference | Title |
|---|--|
| http://www.ietf.org/rfc/std-index.txt , http://www.ietf.org/rfc/ | |
| IETF STD 5 | <i>Internet Protocol, 1981. (Also IETF RFC 0791, RFC 0792, RFC 0919, RFC 0922, RFC 0950, RFC 1112)</i> |
| IETF STD 51 | <i>The Point-to-Point Protocol (PPP), 1994. (Also RFC 1661, RFC 1662)</i> |
| RFC 791 | <i>Internet Protocol (Also: IETF STD 0005), 1981</i> |
| RFC 1332 | <i>The PPP Internet Protocol Control Protocol (IPCP), 1992, Updated by: RFC 3241. Obsoletes: RFC 1172</i> |
| RFC 1570 | <i>PPP LCP Extensions, 1994</i> |
| IETF STD 51 / RFC 1661 | <i>The Point-to-Point Protocol (PPP) (Also: IETF STD 0051), 1994, Updated by: RFC 2153, Obsoletes: RFC 1548</i> |
| IETF STD 51 / RFC 1662 | <i>PPP in HDLC-like Framing, (Also: IETF STD 0051), 1994, Obsoletes: RFC 1549</i> |
| RFC 1994 | <i>PPP Challenge Handshake Authentication Protocol (CHAP), 1996. Obsoletes: RFC 2213</i> |
| RFC 2433 | <i>PPP CHAP Extension, 1998</i> |
| RFC 2474 | <i>Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, 1998</i> |
| RFC 2507 | <i>IP Header Compression, 1999</i> |
| RFC 2759 | <i>Microsoft PPP CHAP Extensions, Version 2, 2000</i> |
| RFC 2986 | <i>PKCS #10 v1.7: Certification Request Syntax Standard</i> |
| RFC 3241 | <i>Robust Header Compression (ROHC) over PPP, 2002. Updates: RFC1332</i> |
| RFC 3513 | <i>Internet Protocol Version 6 (IPv6) Addressing Architecture, 2003</i> |
| RFC 3544 | <i>IP Header Compression over PPP, 2003</i> |
| RFC 3748 | <i>Extensible Authentication Protocol (EAP), 2004</i> |
| RFC 4861 | <i>Neighbor Discovery for IP version 6 (IPv6), 2007</i> |
| RFC 4944 | <i>Internet Engineering Task Force (IETF). RFC 4944: Transmission of IPv6 Packets over IEEE 802.15.4 Networks [online]. Edited by G. Montenegro, N. Kushalnagar and D. Culler. September 2007</i> |
| RFC 6282 | <i>Internet Engineering Task Force (IETF). RFC 6282: Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks [online]. Edited by J. Hui, Ed. September 2011</i> |
| RFC 6775 | <i>Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs), 2012. http://www.ietf.org/rfc/rfc6775.txt</i> |
| | <i>Point-to-Point (PPP) Protocol Field Assignments. Online database. Available from: http://www.iana.org/assignments/ppp-numbers/ppp-numbers.xhtml</i> |

3 Terms, definitions and abbreviations

3.1 Terms and definitions related to the Image transfer process (see 4.4.6)

3.1 Terms and definitions related to the Image transfer process (see 4.4.6)

3.1.1

Image

binary data of a specified size

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

3.1.2

ImageSize

size of the whole Image to be transferred

Note 1 to entry: ImageSize is expressed in octets.

3.1.3

ImageBlock

part of the Image of size ImageBlockSize

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

3.1.4

ImageBlockSize

size of ImageBlock expressed in octets

3.1.5

ImageBlockNumber

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

The meaning of the definitions above is illustrated in Figure 2.

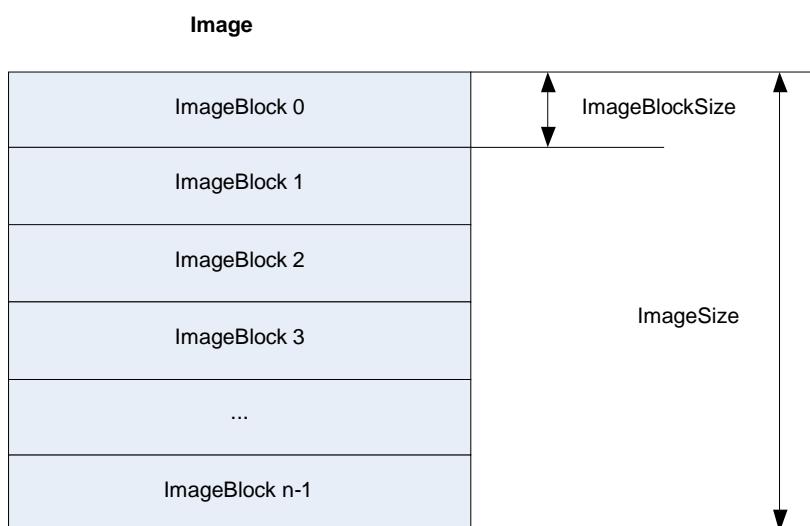


Figure 2 – The meaning of the definitions concerning the Image

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

3.2.1

initiator

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

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

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

3.2.2

active initiator

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

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

3.2.3

new system

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

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

3.2.4

new system title

system-title of a new system

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

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

3.2.5

registered system

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

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

3.2.6

reporting system

server system which issues a DiscoverReport

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

3.2.7

sub-slot

the 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

the time needed to transmit a physical frame

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

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 26/492 |
|-----------------------|------------|-------------------------|--------|

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]

3.3.2

beacon slot

the location of the beacon PDU within a frame

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

3.3.3

node

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

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

3.3.4

registration

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

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

3.3.5

service node

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

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

3.3.6

subnetwork

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

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

Definitions related to the MAC layer

3.3.7

disconnected state (of a service node)

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

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

3.3.8

terminal state (of a service node)

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

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

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 27/492 |
|-----------------------|------------|-------------------------|--------|

3.3.9**switch state (of a service node)**

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

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

3.3.10**promotion**

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

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

3.3.11**demotion**

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

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

3.4 Terms and definitions related to ZigBee® (see 4.14)

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

3.4.1**CAD**

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

3.4.2**IHD**

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

3.4.3**install code**

a Hashed (via MMO) Pre-Configured Linked Key (PCLK) that is provided to a Trust Center via out-of-band communications. A new device wishing to join the network would need to send this install code to the Trust Center, which would allow the Trust Center to execute the joining process, using this install code as part of the security information

3.4.4**link key ***

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

3.4.5**MAC address/IEEE address**

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

3.4.6**ZigBee®**

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

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 28/492 |
|-----------------------|------------|-------------------------|--------|

3.4.7**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

3.4.8**ZigBee® coordinator ***

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

3.4.9**ZigBee® cluster**

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

3.4.10**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

3.4.11**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 multicasting, many-to-one routing and high security with Symmetric-Key Key Exchange (SKKE), while ZigBee® (stack profile 1) offers a smaller footprint in RAM and flash. Both offer full mesh networking and work with all ZigBee® application profiles

3.4.12**ZigBee® router ***

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

3.4.13**ZigBee® server**

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

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

3.4.14**ZigBee® Trust Center ***

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

3.5 Terms and definitions related to Payment metering interface classes (see 4.6)**3.5.1****account**

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

3.5.2**available**

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

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 29/492 |
|-----------------------|------------|-------------------------|--------|

3.5.3**charge**

a representation of a financial liability on an account

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

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

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

3.5.4**credit mode**

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

3.5.5**collect**

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

3.5.6**commodity**

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

3.5.7**enabled**

when used in the context of "Credit" or "Charge" types; means that the "Credit" or "Charge" type appears in the *credit_reference_list* or *charge_reference_list* respectively of the "Account" object

3.5.8**emergency credit**

an amount of credit administered in a payment metering system working in prepayment mode, representing a short term loan to the consumer

Note1 to entry: This is a feature of some payment metering systems in which the consumer is able to obtain a limited amount of credit as a short-term loan, often mediated locally by the prepayment unit itself. The word "emergency" indicates urgent need rather than disaster.

3.5.9**Enterprise Resource Planning (ERP) system****Back Office System**

computer system carrying out the business processing of an organisation (such as an energy supplier), as distinct from the communications system. See also Head End System.

3.5.10**friendly credit**

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

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

3.5.11**Head End System (HES)**

a computer system, connected by a communications network to a population of intelligent devices, whose job is the control and coordination of information flows to and from those devices, typically on behalf of a separate ERP ("back office") system

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 30/492 |
|-----------------------|------------|-------------------------|--------|

3.5.12**Home Area Network (HAN)**

a communications network constructed with the principal aim of connecting devices in one premises

3.5.13**in use**

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

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

3.5.14**load limiting**

a mode of operation of some payment metering systems (not necessarily in prepayment mode) in which the consumer is able to draw on a supply provided they do not exceed a configured level of demand

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

3.5.15**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

3.5.16**manual entry**

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

3.5.17**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.

3.5.18**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.

3.5.19**prepayment mode**

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

3.5.20**post-payment**

the 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.

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 31/492 |
|-----------------------|------------|-------------------------|--------|

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.

3.5.21

remote communications

the 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

3.5.22

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

3.5.23

reserved credit

an 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.

3.5.24

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.

3.5.25

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).

3.5.26

service

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

3.5.27

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 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*.

3.5.28

temporary debt

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

Note1 to entry: This temporary debt amount is accumulated in *amount_to_clear* in the "Account" object.

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 32/492 |
|-----------------------|------------|-------------------------|--------|

3.5.29**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”.

3.5.30**token carrier**

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

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

3.5.31**token carrier interface**

interface between the token carrier and the payment metering installation

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

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

3.5.32**top-up****credit token**

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

3.5.33**transient device communications**

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

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

3.5.34**vend**

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

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

3.6 Terms and definitions related to the Arbitrator IC (see 4.5.12)**3.6.1****action**

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

3.6.2**actor**

an entity requesting an action

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 33/492 |
|-----------------------|------------|-------------------------|--------|

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

3.6.3

arbitrator

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

3.7 Abbreviations

| Abbreviation | Explanation |
|--------------|--|
| 6LoWPAN | IPv6 over Low-Power Wireless Personal Area Network |
| AA | Application Association |
| AARE | A-Associate Response – an APDU of the ACSE |
| AARQ | A-Associate Request – an APDU of the ACSE |
| ACSE | Association Control Service Element |
| ADP | Primary Station Address |
| ADS | Secondary Station Address |
| AGA | American Gas Association |
| AGA 8 | Method for calculation of compressibility (Gas metering) |
| AGC | Automatic Gain Control |
| AL | Application layer |
| AP | Application process |
| APDU | Application Protocol Data Unit |
| APS | Application Support Sublayer (ZigBee® term) |
| ASE | Application Service Element |
| A-XDR | Adapted Extended Data Representation (IEC 61334-6) |
| base_name | The short_name corresponding to the first attribute ("logical_name") of a COSEM object |
| BCD | Binary Coded Decimal |
| BER | Bit Error Rate |
| CBCP | CallBack Control Protocol (PPP) |
| CC | Current Credit (S-FSK PLC profile) |
| CENELEC | European Committee for Electrotechnical Standardization |
| CHAP | Challenge Handshake Authentication Protocol |
| CIASE | Configuration Initiation Application Service Element (S-FSK PLC profile) |
| class_id | Interface class identification code |
| CLI | Calling Line Identity |
| COSEM | Companion Specification for Energy Metering |
| COSEM object | An instance of a COSEM interface class |
| CRC | Cyclic Redundancy Check |
| CSD | Circuit Switched Data |
| CSMA | Carrier Sense Multiple Access |
| CtoS | Client to Server challenge |
| CU | Currently Unused |
| DC | Delta credit (S-FSK PLC profile) |
| DHCP | Dynamic Host Control Protocol |

| Abbreviation | Explanation |
|--------------|---|
| DIB | Data Information Block (M-Bus) |
| DIF | Data Information Field (M-Bus) |
| DL | Data Link |
| DLMS | Device Language Message Specification |
| DLMS UA | DLMS User Association |
| DNS | Domain Name Server |
| DSCP | Differentiated Services Code Point |
| DSSID | Direct Switch ID |
| EAP | Extensible Authentication Protocol |
| EDGE | Enhanced Data rates for GSM Evolution |
| 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 |
| FCC | Federal Communications Commission |
| FFD | Full-Function Device |
| FIFO | First-In-First-Out |
| FTP | File Transfer Protocol |
| GCM | Galois/Counter Mode, an algorithm for authenticated encryption with associated data |
| GMT | Greenwich Mean Time. Replaced by Coordinated Universal Time (UTC). |
| GPRS | General Packet Radio Service |
| GPS | Global Positioning System |
| GSM | Global System for Mobile Communications |
| HAN | Home Area Network |
| HART | Highway Addressable Remote Transducer see http://www.hartcomm.org/ (in relation with the Sensor manager interface class) |
| HDLC | High-level Data Link Control |
| HES | Head End System |
| HHT | Hand Held Terminal |
| HLS | High Level Security Authentication |
| HSDPA | High-Speed Downlink Packet Access |
| IANA | Internet Assigned Numbers Authority |
| IB | Information Base |
| IC | Interface Class (COSEM) |
| IC | Initial credit (S-FSK PLC profile) |
| IEC | International Electrotechnical Commission |
| IEEE | Institute of Electrical and Electronics Engineers |
| IETF | Internet Engineering Task Force |
| IPCP | Internet Protocol Control Protocol |
| IPv4 | Internet Protocol version 4 |
| IPv6 | Internet Protocol version 6 |

| Abbreviation | Explanation |
|--------------|---|
| ISO | International Organization for Standardization |
| ISP | Internet Service Provider |
| IT | Information Technology |
| ITU-T | International Telecommunication Union – Telecommunication |
| KEK | Key Encryption Key |
| LA | Local Area |
| LAC | Local Area Code |
| LAN | Local Area Network |
| LCID | Local Connection Identifier |
| LCP | Link Control Protocol |
| LDN | Logical Device Name |
| LLC | Logical Link Control (sublayer) |
| LLS | Low Level Security |
| LN | Logical Name |
| LNID | Local Node Identifier |
| LOADng | 6LoWPAN Ad Hoc On-Demand Distance Vector Routing Next Generation (LOADng) |
| LQI | Link Quality Indicator (ZigBee ® term) |
| LSB | Least Significant Bit |
| LSID | Local Switch Identifier |
| LTE | Long Term Evolution (Wireless communication) |
| m | mandatory |
| MAC | Medium Access Control |
| M-Bus | Meter Bus |
| MD5 | Message Digest Algorithm 5 |
| MIB | Management Information Base (S-FSK PLC profile) |
| MID | Measuring Instruments Directive 2004/22/EC of the European Parliament and of the Council |
| MMO | Matyas-Meyer-Oseas hash (ZigBee ® term) |
| MPAN | (UK term) Meter Point Access Number – reference of the location of the Electricity meter on the electricity distribution network. |
| MPDU | MAC Protocol Data Unit |
| MSB | Most Significant Bit |
| MSDU | MAC Service Data Unit |
| MT | Mobile Termination |
| NB | Narrow-band |
| ND | Neighbour Discovery |
| NTP | Network Time Protocol |
| o | optional |
| OBIS | Object Identification System |
| OFDM | Orthogonal Frequency Division Multiplexing |
| OTA | Over the Air – Refers to Firmware Upgrade using ZigBee ® |
| PAN | Personal Area Network (Term used in relation to G3-PLC ¹) and ZigBee ® |

| Abbreviation | Explanation |
|--------------|---|
| Pad | Padding |
| PAP | Password Authentication Protocol |
| PCLK | Pre-Configured Link Key (ZigBee® term) |
| PDU | Protocol Data Unit |
| PhL, PHY | Physical Layer |
| PIB | PLC Information Base |
| PIN | Personal Identity Number |
| PLC | Power Line Carrier |
| PLMN | Public Land Mobile Network |
| PNPDU | Promotion Needed PDU |
| POS | Personal Operating Space (ZigBee ®) |
| POS | Point Of Sale (Payment metering) |
| PPDU | Physical Protocol Data Unit |
| PPP | Point-to-Point Protocol |
| PSTN | Public Switched Telephone Network |
| REJ PDU | Reject Protocol Data Unit |
| RFC | Request for Comments; a document published by the Internet Engineering Task Force |
| RFD | Reduced Function Device |
| ROHC | Robust Header Compression |
| RREP | Route Reply |
| RREQ | Route Request |
| RRER | Route Error |
| RSSI | Receive Signal Strength Indication (ZigBee® term) |
| SAP | Service Access Point |
| SAS | Startup Attribute Set (ZigBee® term) |
| SCP | Shared Contention Period |
| SE | Smart Energy |
| SEP | Smart Energy Profile (ZigBee® term) |
| S-FSK | Spread – Frequency Shift Keying |
| SGERG88 | Method for calculation of compressibility (Gas metering) |
| SHA | Secure Hash Algorithm |
| SID | Switch identifier |
| SMS | Short Message Service |
| SMTP | Simple Mail Transfer Protocol |
| SN | Short Name |
| SNA | Subnetwork Address |
| SSCS | Service Specific Convergence Layer |
| StoC | Server to Client Challenge |
| TAB | In the case of the EURIDIS profiles without DLMS and without DLMS/COSEM: data code. In the case of profiles using DLMS or DLMS/COSEM: value at which the equipment is programmed for Discovery |

| Abbreviation | Explanation |
|---|--|
| TABi | List of TAB field |
| TCC | Transmission Control Code (IPv4) |
| TCP | Transmission Control Protocol |
| TFTP | Trivial File Transfer Protocol |
| TOU | Time of use |
| TTL | Time To Live |
| UDP | User Datagram Protocol |
| UMTS | Universal Mobile Telecommunications System |
| UNC | Unconfigured (S-FSK PLC profile) |
| UTC | Coordinated Universal Time |
| VIB | Value Information Block (M-Bus) |
| VIF | Value Information Field (M-Bus) |
| VZ | Billing period counter (Form <i>Vorwertzähler</i> in German, see DIN 43863-3) |
| wake-up | trigger the meter to connect to the communication network to be available to a client (e.g. HES) |
| WAN | Wide Area Network |
| WM-Bus | Wireless M-Bus |
| ZTC | ZigBee® Trust Center |
| ¹⁾ In the case of the G3-PLC technology, PAN may be defined as PLC Area Network. | |

4 The COSEM interface classes

4.1 Basic principles

4.1.1 General

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

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

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

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

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

Figure 3 illustrates these terms by means of an example:

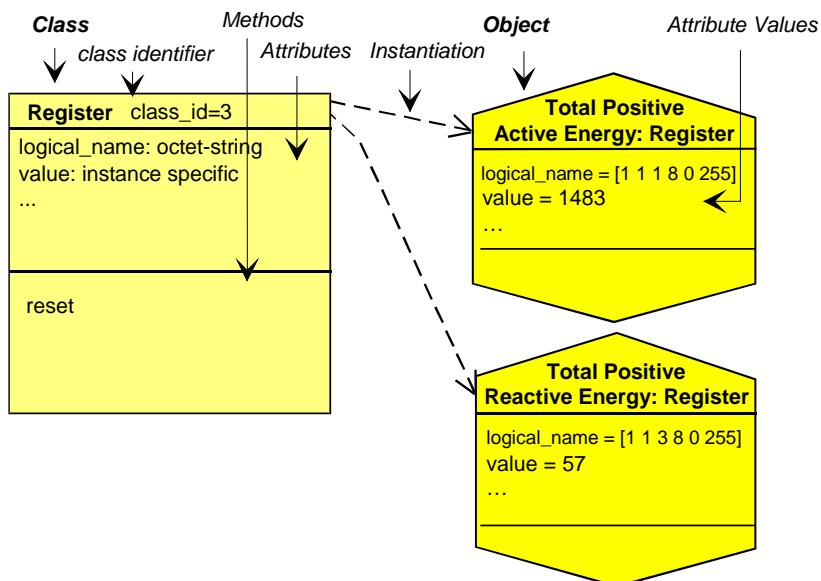


Figure 3 – An interface class and its instances

The IC “Register” is formed by combining the features necessary to model the behaviour of a generic register (containing measured or static information) as seen from the client (data collection system, hand held terminal). The contents of the register are identified by the attribute *logical_name*. The *logical_name* contains an OBIS identifier (see Clause 7). The actual (dynamic) content of the register is carried by its *value* attribute.

Defining a specific meter means defining several specific objects. In the example of Figure 3, the meter contains two registers; i.e. two specific instances of the IC “Register” are instantiated. Through the instantiation, one COSEM object becomes a “total, positive, active energy register” whereas the other becomes a “total, positive, reactive energy register”.

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

4.1.2 Referencing methods

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

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

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

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

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

Using short names (SN referencing): This kind of referencing is intended for use in simple devices. In this case, each attribute and method of a COSEM object is identified with a 13-bit integer. The syntax for the short name is the same as the syntax of the name of a DLMS named variable. See IEC 61334-4-41:1996 and Clause 9.5 of DLMS UA 1000-2 Ed. 8.1:201.

4.1.3 Reserved base_names for special COSEM objects

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

Table 1 – Reserved base_names for SN referencing

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

4.1.4 Class description notation

This subclause describes the notation used to define the ICs.

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

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

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

NOTE 1 Interface classes names are mentioned in quotation marks.

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

value The IC shall be instantiated exactly "value" times.

Min...max. The IC shall be instantiated at least "min." times and at most "max." times. If min. is zero (0) then the IC is optional, otherwise (min. > 0) "min." instantiations of the IC are mandatory.

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

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

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

Version Identification code of the version of the IC. The version of each object is retrieved together with the class_id and the logical name by reading the *object_list* attribute of an "Association LN" / "Association SN" object.

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

Attributes Specifies the attributes that belong to the IC.

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

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

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

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

logical_name octet-string It is always the first attribute of an IC. It identifies the instantiation (COSEM object) of this IC. The value of the *logical_name* conforms to OBIS; see clauses 6 and 7.

| | |
|---------------------------|---|
| Data type | Defines the data type of an attribute; see 4.1.5. |
| Min. | Specifies if the attribute has a minimum value. X The attribute has a minimum value. <empty> The attribute has no minimum value. |
| Max. | Defines if the attribute has a maximum value. X The attribute has a maximum value. <empty> The attribute has no maximum value. |
| Def. | Specifies if the attribute has a default value. This is the value of the attribute after reset. X The attribute has a default value. <empty> The default value is not defined by the IC specification. |
| Short name | When Short Name (SN) referencing is used, each attribute and method of object instances has to be mapped to short names. The base_name x of each object instance is the DLMS named variable the logical name attribute is mapped to. It is selected in the implementation phase. The IC definition specifies the offsets for the other attributes and for the methods. |
| Specific method(s) | Provides a list of the specific methods that belong to the object. Method Name () The method has to be described in the subsection "Method description". NOTE 4 Method names use the underscore notation. When mentioned in the text they are in <i>italic</i> . Example: <i>add_object</i> . |
| m/o | Defines if the method is mandatory or optional. <i>M (mandatory)</i> The method is mandatory. <i>O (optional)</i> The method is optional. |

Attribute description

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

Method description

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

NOTE Services for accessing attributes or methods by the protocol are specified in Clause 9 of DLMS UA 1000-2 Ed. 8.1:201.

Selective access

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

4.1.5 Common data types

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

Table 2 – Common data types

| Type description | Tag ^a | Definition | Value range |
|-----------------------|------------------|--|---|
| -- simple data types | | | |
| null-data | [0] | | |
| boolean | [3] | boolean | TRUE or FALSE |
| bit-string | [4] | An ordered sequence of boolean values | |
| double-long | [5] | Integer32 | -2 147 483 648... 2 147 483 647 |
| double-long-unsigned | [6] | Unsigned32 | 0...4 294 967 295 |
| | [7] | Tag of the “floating-point” type in IEC 61334-4-41:1996, not usable in DLMS/COSEM. See tags [23] and [24] | |
| octet-string | [9] | An ordered sequence of octets (8 bit bytes) | |
| visible-string | [10] | An ordered sequence of ASCII characters | |
| | [11] | Tag of the “time” type in IEC 61334-4-41:1996, not usable in DLMS/COSEM. See tag [27] | |
| utf8-string | [12] | An ordered sequence of characters encoded as UTF-8 | |
| bcd | [13] | binary coded decimal | |
| integer | [15] | Integer8 | -128...127 |
| long | [16] | Integer16 | -32 768...32 767 |
| unsigned | [17] | Unsigned8 | 0...255 |
| long-unsigned | [18] | Unsigned16 | 0...65 535 |
| long64 | [20] | Integer64 | - 2 ⁶³ ...2 ⁶³ -1 |
| long64-unsigned | [21] | Unsigned64 | 0...2 ⁶⁴ -1 |
| enum | [22] | The elements of the enumeration type are defined in the <i>Attribute description</i> or <i>Method description</i> section of a COSEM IC specification. | 0...255 |
| float32 | [23] | OCTET STRING (SIZE(4)) | For formatting, see 4.1.6.2. |
| 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)) | |
| -- complex data types | | | |
| array | [1] | The elements of the array are defined in the <i>Attribute</i> or <i>Method description</i> section of a COSEM IC specification. | |
| Structure | [2] | The elements of the structure are defined in the <i>Attribute</i> or <i>Method description</i> section of a COSEM IC specification. | |
| Compact array | [19] | Provides an alternative, compact encoding of complex data. | |
| -- CHOICE | | For some COSEM interface objects attributes, the data type may be chosen at instantiation, in the implementation phase of the COSEM server. The server always shall send back the data type and the value of each attribute, so that together with the logical name an unambiguous interpretation is ensured. The list of possible data types is defined in the “Attribute description” section of a COSEM IC specification. | |

^a The tags are as defined in Clause 9.5 of DLMS UA 1000-2 Ed. 8.1:201.

4.1.6 Data formats

4.1.6.1 Date and time formats

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

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

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

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

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

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

The elements *dayOfMonth* and *dayOfWeek* shall be interpreted together:

- if *last dayOfMonth* is specified (0xFE) and *dayOfWeek* is wildcard, this specifies the last calendar day of the month;
- if *last dayOfMonth* is specified (0xFE) and an explicit *dayOfWeek* is specified (for example 7, Sunday)

| | |
|---|---|
| <p>then it is the last occurrence of the weekday specified in the month, i.e. the last Sunday;</p> <ul style="list-style-type: none"> - if the year is not specified (0xFFFF), and dayOfMonth and dayOfWeek are both explicitly specified, this shall be interpreted as the dayOfWeek on, or following dayOfMonth; - if the year and month are specified, and both the dayOfMonth and dayOfWeek are explicitly specified but the values are not consistent it shall be considered as an error. | |
| <p>Examples:</p> <ol style="list-style-type: none"> 1) year = 0xFFFF, month = 0x FF, dayOfMonth = 0xFE, dayofWeek = 0xFF: last day of the month in every year and month; 2) year = 0xFFFF, month = 0x FF, dayOfMonth = 0xFE, dayofWeek = 0x07: last Sunday in every year and month; 3) year = 0xFFFF, month = 0x03, dayOfMonth = 0xFE, dayofWeek = 0x07: last Sunday in March in every year; 4) year = 0xFFFF, month = 0x03, dayOfMonth = 0x01, dayofWeek = 0x07: first Sunday in March in every year; 5) year = 0xFFFF, month = 0x03, dayOfMonth = 0x16, dayofWeek = 0x05: fourth Friday in March in every year; 6) year = 0xFFFF, month = 0x0A, dayOfMonth = 0x16, dayofWeek = 0x07: fourth Sunday in October in every year; 7) year = 0x07DE, month = 0x08, dayOfMonth = 0x13, (2014.08.13, Wednesday) dayofWeek = 0x02 (Tuesday): error, as the dayOfMonth and dayOfWeek in the given year and month do not match. | |
| <i>Time</i> | <pre>OCTET STRING (SIZE(4)) { hour, minute, second, hundredths } Where: hour: interpreted as unsigned range 0...23, 0xFF, minute: interpreted as unsigned range 0...59, 0xFF, second: interpreted as unsigned range 0...59, 0xFF, hundredths: interpreted as unsigned range 0...99, 0xFF For hour, minute, second and hundredths: 0xFF = not specified. For repetitive times the unused parts shall be set to "not specified".</pre> |
| <i>Date-time</i> | <pre>OCTET STRING (SIZE(12)) { year highbyte, year lowbyte, month, day of month, day of week, hour, minute, second, hundredths of second, deviation highbyte, deviation lowbyte, clock status }</pre> <p>The elements of <i>date</i> and <i>time</i> are encoded as defined above. Some may be set to "not specified" as defined above.</p> <p>In addition:</p> |

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

4.1.6.2 Floating point number formats

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

The single format is:

| | | | |
|-----|-----|------------|-----------|
| 1 | 8 | 23 | ...widths |
| s | e | F | |
| msb | lsb | msb lsb | ...order |

Where:

- s is the sign bit;
- e is the exponent; it is 8 bits wide and the exponent bias is +127;
- f is the fraction, it is 23 bits.

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

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

The double format is:

| | | | |
|-----|-----|------------|-----------|
| 1 | 11 | 52 | ...widths |
| s | e | F | |
| msb | lsb | msb lsb | ...order |

Where:

- s is the sign bit;
 - e is the exponent; it is 11 bits wide and the exponent bias is +1 023;
 - f is the fraction, it is 52 bits.

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

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

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

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

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

| | | |
|--|---|---|
| Bit 31 Sign bit 0 0: + 1: - | Bits 30-23 Exponent field: 01111111 Decimal value of exponent field and exponent: 127-127 = 0 | Bits 22-0 Significand 1.000000000000000000000000000000 Decimal value of the significand: 1.0000000 |
|--|---|---|

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

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

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

The encoding, including the tag of the data type is (all values are hexadecimal):
18 3F E0 00 00 00 00 00 00.

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

| | | |
|--|--|--|
| Bit 31 Sign bit 0 0: + 1: - | Bits 30-23 Exponent field: 10001110 Decimal value of exponent field and exponent: 142-127 = 15 | Bits 22-0 Significand 1.1100100110100000000000 Decimal value of the significand: 1.8937988 |
|--|--|--|

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

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

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

4.1.7 The COSEM server model

The COSEM server is structured into three hierarchical levels as shown in Figure 4:

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

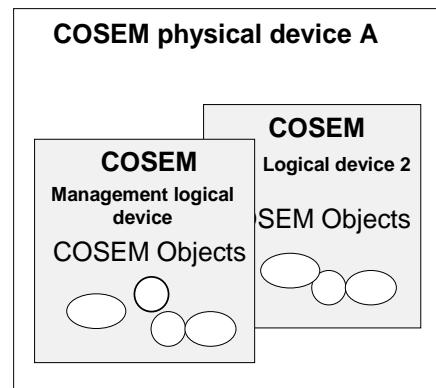


Figure 4 – The COSEM server model

The example in Figure 5 shows how a combined metering device can be structured using the COSEM server model.

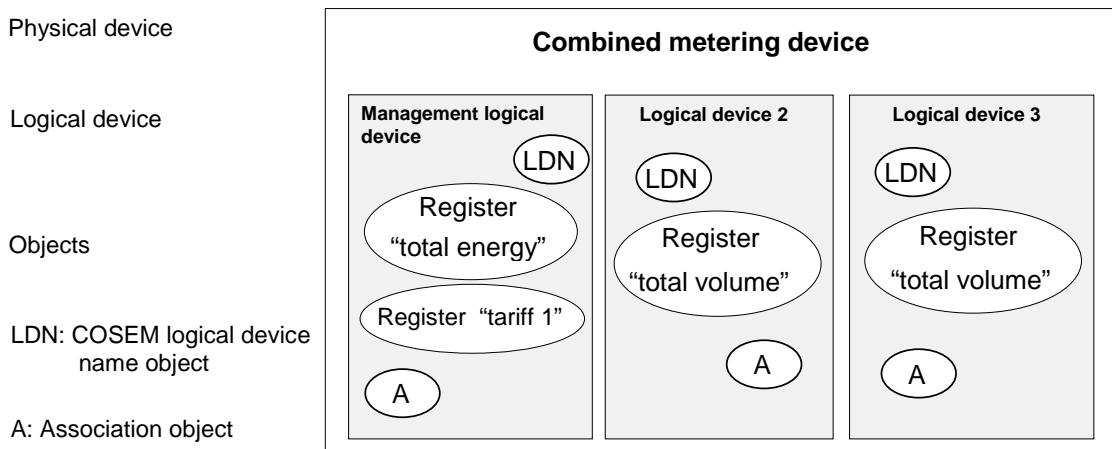


Figure 5 – Combined metering device

4.1.8 The COSEM logical device

4.1.8.1 General

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

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

4.1.8.2 COSEM logical device name (LDN)

The COSEM logical device can be identified by its unique COSEM LDN. This name can be retrieved from an instance of IC "SAP assignment" (see 4.4.5), or from a COSEM object "COSEM logical device name" (see 6.2.30).

The LDN is defined as an octet-string of up to 16 octets. The first three octets shall carry the manufacturer identifier¹. The manufacturer shall ensure that the LDN, starting with the three octets identifying the manufacturer and followed by up to 13 octets, is unique.

4.1.8.3 The “association view” of the logical device

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

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

AAs are modelled by special COSEM objects:

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

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

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

4.1.8.4 Mandatory contents of a COSEM logical device

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

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

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

4.1.8.5 Management logical device

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

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

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

¹ Administered by the DLMS User Association, in cooperation with the FLAG Association.

4.1.9 Information security

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

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

For a description of these security mechanisms, see DLMS UA 1000-2 Ed. 8.1:201.

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

- “Association SN”, see 4.4.3;
- “Association LN”, see 4.4.4;
- “Security setup”, see 4.4.7; and
- “Data protection”, see 4.4.9.

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 50/492 |
|-----------------------|------------|-------------------------|--------|

4.2 Overview of the COSEM interface classes

The ICs defined currently and the relations between them are shown in [Figure 6](#) and [Figure 7](#).

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

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

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

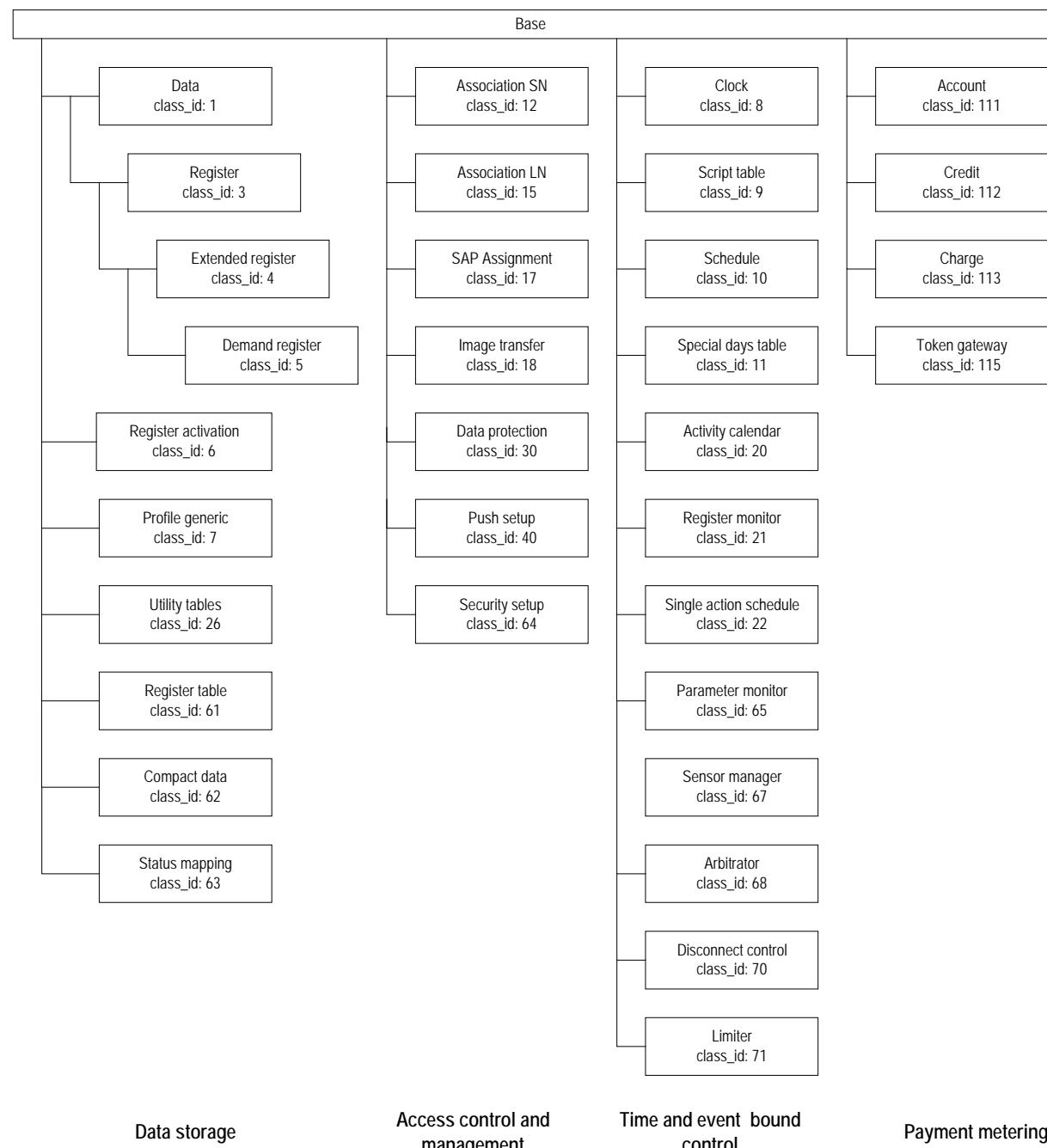
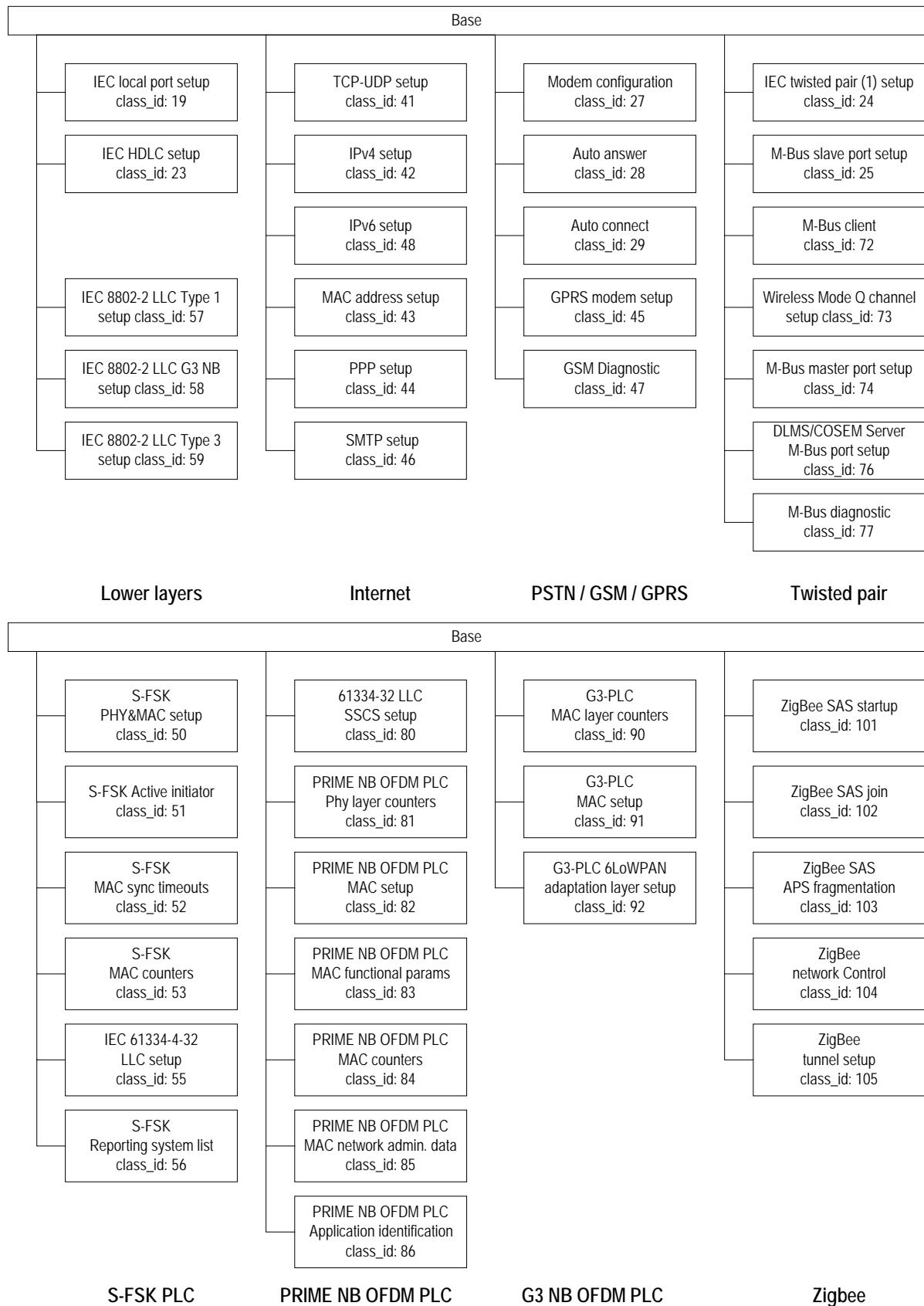


Figure 6 – Overview of the interface classes – Part 1

**Figure 7 – Overview of the interface classes – Part 2**

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 52/492 |
|-----------------------|------------|-------------------------|--------|

Table 3 lists the interface classes by class_id.

Table 3 – List of interface classes by class_id

| Interface class name | class_id | version(s) | Clause |
|---|----------|------------|----------------|
| Data | 1 | 0 | 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 5.4.2 |
| 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 | 5.4.6 |
| | | 2 | 5.4.5 |
| | | 1 | 5.4.4 |
| | | 0 | 5.4.3 |
| Association LN | 15 | 3 | 4.4.4 |
| | | 2 | 5.4.9 |
| | | 1 | 5.4.8 |
| | | 0 | 5.4.7 |
| SAP Assignment | 17 | 0 | 4.4.5 |
| Image transfer | 18 | 0 | 4.4.6 |
| IEC local port setup | 19 | 1 | 4.7.1 |
| | | 0 | 5.4.11 |
| 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 | 4.7.2 |
| | | 0 | 5.4.12 |
| IEC twisted pair (1) setup | 24 | 1 | 4.7.3 |
| | | 0 | 5.4.13 |
| M-BUS slave port setup | 25 | 0 | 4.8.2 |
| Utility tables | 26 | 0 | 4.3.7 |
| Modem configuration PSTN modem configuration | 27 | 1 | 4.7.4 |
| | | 0 | 5.4.14 |
| Auto answer | 28 | 2 | 4.7.5 |
| | | 1 | – |
| | | 0 | 5.4.15 |
| Auto connect PSTN Auto dial | 29 | 2 | 4.7.6 |
| | | 1 | 5.4.17 |
| | | 0 | 5.4.16 |

| Interface class name | class_id | version(s) | Clause |
|---|----------|------------|------------------|
| Data protection | 30 | 0 | 4.4.9 |
| Push setup | 40 | 0 | 4.4.8.2 |
| 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 | 0 | 4.7.8 |
| IPv6 setup | 48 | 0 | 4.9.3 |
| S-FSK Phy&MAC setup | 50 | 1 0 | 4.10.3 5.4.18 |
| 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 | 55 | 1 0 | 4.10.7 5.4.19 |
| 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 |
| ISO/IEC 8802-2 LLC Type 3 setup | 59 | 0 | 4.11.4 |
| Register table | 61 | 0 | 4.3.8 |
| Compact data | 62 | 0 | 4.3.10 |
| Status mapping | 63 | 0 | 4.3.9 |
| Security setup | 64 | 1 0 | 4.4.7 5.4.10 |
| Parameter monitor | 65 | 0 | 4.5.10 |
| 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 5.4.20 |
| 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 |

| Interface class name | class_id | version(s) | Clause |
|---|-----------------|-------------------|---------------|
| 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 | 4.13.3 |
| G3 NB OFDM PLC MAC layer counters | | 0 | 5.4.21 |
| G3-PLC MAC setup | 91 | 1 | 4.13.4 |
| G3 NB OFDM PLC MAC setup | | 0 | 5.4.22 |
| G3-PLC 6LoWPAN adaptation layer setup | 92 | 1 | 4.13.5 |
| G3 NB OFDM PLC 6LoWPAN adaptation layer setup | | 0 | 5.4.23 |
| ZigBee® SAS startup | 101 | 0 | 4.14.2 |
| ZigBee® SAS join | 102 | 0 | 4.14.3 |
| ZigBee® SAS APS fragmentation | 103 | 0 | 4.14.4 |
| ZigBee® network control | 104 | 0 | 4.14.5 |
| ZigBee® tunnel setup | 105 | 0 | 4.14.6 |
| 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 |

4.3 Interface classes for parameters and measurement data

4.3.1 Data (class_id = 1, version = 0)

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

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

Attribute description

| | |
|---------------------|--|
| logical_name | Identifies the “Data” object instance. See clauses 6 and 7. |
| value | <p>Contains the data.</p> <p>CHOICE</p> <pre>{ -- simple data types null-data [0], boolean [3], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], utf8-string [12], bcd [13], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], enum [22], float32 [23], float64 [24], date-time [25], date [26], time [27], -- complex data types array [1], structure [2], compact-array [19] }</pre> |

4.3.2 Register (class_id = 3, version = 0)

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

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

Attribute description

| | |
|---------------------|---|
| logical_name | Identifies the “Register” object instance. See clauses 6 and 7. |
| value | <p>Contains the current process or status value.</p> <p>CHOICE</p> <pre>{ -- simple data types null-data [0], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], utf8-string [12], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], enum [22], float32 [23], float64 [24], }</pre> <p>The data type of the value depends on the instantiation defined by <i>logical_name</i> and possibly on the choice of the manufacturer. It has to be chosen so that, together with the <i>logical_name</i>, an unambiguous interpretation of the value is possible.</p> <p>When, instead of a “Data” object, a “Register” object is used, (with the <i>scaler_unit</i> attribute not used or with <i>scaler</i> = 0, <i>unit</i> = 255) then the data types allowed for the <i>value</i> attribute of the “Data” IC are allowed.</p> |
| scaler_unit | <p>Provides information on the unit and the scaler of the value.</p> <p>scal_unit_type ::= structure</p> <pre>{ scaler, unit }</pre> <p>scaler: integer</p> <p>This is the exponent (to the base of 10) of the multiplication factor.</p> <p>REMARK If the value is not numerical, then the scaler shall be set to 0.</p> <p>unit: enum</p> <p>Enumeration defining the physical unit; for details see Table 4 below.</p> |

Method description

reset (data) Forces a reset of the object. By invoking this method, the value is set to the default value. The default value is an instance specific constant.
 data ::= integer (0)

Table 4 – Enumerated values for physical units

| unit ::= enum | Unit | Quantity | Unit name | SI definition (comment) |
|---------------|-----------------------|---|----------------------|---------------------------------|
| (1) | a | time | year | |
| (2) | mo | time | month | |
| (3) | wk | time | week | $7*24*60*60\text{ s}$ |
| (4) | d | time | day | $24*60*60\text{ s}$ |
| (5) | h | time | hour | $60*60\text{ s}$ |
| (6) | min. | time | min | 60 s |
| (7) | s | time (<i>t</i>) | second | s |
| (8) | ° | (phase) angle | degree | $\text{rad} * 180/\pi$ |
| (9) | °C | temperature (<i>T</i>) | degree-celsius | K-273.15 |
| (10) | currency | (local) currency | | |
| (11) | m | length (<i>l</i>) | metre | m |
| (12) | m/s | speed (<i>v</i>) | metre per second | m/s |
| (13) | m^3 | volume (<i>V</i>) <i>r_V</i> , meter constant or pulse value (volume) | cubic metre | m^3 |
| (14) | m^3 | corrected volume | cubic metre | m^3 |
| (15) | m^3/h | volume flux | cubic metre per hour | $\text{m}^3/(60*60\text{s})$ |
| (16) | m^3/h | corrected volume flux | 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 | | $\text{m}^3/(24*60*60\text{s})$ |
| (19) | l | volume | litre | 10^{-3} m^3 |
| (20) | kg | mass (<i>m</i>) | kilogram | |
| (21) | N | force (<i>F</i>) | newton | |
| (22) | Nm | energy | newton meter | J = Nm = Ws |
| (23) | Pa | pressure (<i>p</i>) | pascal | N/m^2 |
| (24) | bar | pressure (<i>p</i>) | bar | $10^5\text{ N}/\text{m}^2$ |
| (25) | J | energy | joule | J = Nm = Ws |
| (26) | J/h | thermal power | joule per hour | $\text{J}/(60*60\text{s})$ |
| (27) | W | active power (<i>P</i>) | watt | $\text{W} = \text{J}/\text{s}$ |
| (28) | VA | apparent power (<i>S</i>) | volt-ampere | |
| (29) | var | reactive power (<i>Q</i>) | var | |
| (30) | Wh | active energy <i>r_W</i> , active energy meter constant or pulse value | watt-hour | $\text{W}*(60*60\text{s})$ |
| (31) | VAh | apparent energy <i>r_A</i> , apparent energy meter constant or pulse value | volt-ampere-hour | $\text{VA}*(60*60\text{s})$ |
| (32) | varh | reactive energy <i>r_B</i> , reactive energy meter constant or pulse value | var-hour | $\text{var}*(60*60\text{s})$ |
| (33) | A | current (<i>I</i>) | ampere | A |
| (34) | C | electrical charge (<i>Q</i>) | coulomb | C = As |

| unit ::= enum | Unit | Quantity | Unit name | SI definition (comment) |
|----------------------|----------------------|---|----------------------|--|
| (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 | $\Omega = V/A$ |
| (39) | $\Omega m^2/m$ | resistivity (ρ) | | Ωm |
| (40) | Wb | magnetic flux (ϕ) | weber | Wb = 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 | H = Wb/A |
| (44) | Hz | frequency (f, ω) | hertz | 1/s |
| (45) | 1/(Wh) | R_W , active energy meter constant or pulse value | | |
| (46) | 1/(varh) | R_B , reactive energy meter constant or pulse value | | |
| (47) | 1/(VAh) | R_S , apparent energy meter constant or pulse value | | |
| (48) | $V^2 h$ | volt-squared hour, r_{U2h} , volt-squared hour meter constant or pulse value | volt-squared-hours | $V^2(60*60s)$ |
| (49) | $A^2 h$ | ampere-squared hour, r_{I2h} , ampere-squared hour meter constant or pulse value | ampere-squared-hours | $A^2(60*60s)$ |
| (50) | kg/s | mass flux | kilogram per second | kg/s |
| (51) | S, mho | conductance | siemens | 1/ Ω |
| (52) | K | temperature (T) | kelvin | |
| (53) | 1/(V ² h) | R_{U2h} , volt-squared hour meter constant or pulse value | | |
| (54) | 1/(A ² h) | R_{I2h} , ampere-squared hour meter constant or pulse value | | |
| (55) | 1/m ³ | R_V , meter constant or pulse value (volume) | | |
| (56) | | percentage | % | |
| (57) | Ah | ampere-hours | Ampere-hour | |
| ... | | | | |
| (60) | Wh/m ³ | energy per volume | $3,6*10^3 J/m^3$ | |
| (61) | J/m ³ | calorific value, wobbe | | |
| (62) | Mol % | molar fraction of gas composition | mole percent | (Basic gas composition unit) |
| (63) | g/m ³ | mass density, quantity of material | | (Gas analysis, accompanying elements) |
| (64) | Pa s | dynamic viscosity | pascal second | (Characteristic of gas stream) |
| (65) | J/kg | Specific energy NOTE The amount of energy per unit of mass of a substance | Joule / kilogram | $m^2 \cdot kg \cdot s^{-2} / kg$ $= m^2 \cdot s^{-2}$ |
| | | | | |
| (70) | dBm | Signal strength, dB milliwatt (e.g. of GSM radio systems) | | |
| (71) | db μ V | Signal strength, dB microvolt | | |
| (72) | dB | Logarithmic unit that expresses the ratio between two values of a physical quantity | | |
| ... | | | | |
| (253) | | reserved | | |

| unit ::= enum | Unit | Quantity | Unit name | SI definition (comment) |
|---------------|-------|--------------------------|-----------|-------------------------|
| (254) | other | other unit | | |
| (255) | count | no unit, unitless, count | | |

Some examples are shown in Table 5 below.

Table 5 – 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 |

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

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

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

Attribute description

| | |
|----------------------|--|
| logical_name | Identifies the “Extended register” object instance. See Clause 6.3.1. |
| value | See the specification of the IC “Register” in 4.3.2. |
| scaler_unit | See the specification of the IC “Register” in 4.3.2. |
| status | Provides “Extended register” specific status information. The meaning of the elements of the status shall be provided for each object instance. |
| CHOICE | The data type and the encoding depend on the instantiation and possibly on the choice of the manufacturer. For the interpretation, extra information from the manufacturer may be necessary. |
| { | |
| -- simple data types | |
| null-data | [0], |
| bit-string | [4], |
| double-long-unsigned | [6], |
| octet-string | [9], |
| visible-string | [10], |
| utf8-string | [12], |
| unsigned | [17], |
| long-unsigned | [18], |

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 60/492 |
|-----------------------|------------|-------------------------|--------|

```
        long64-unsigned      [21]
    }
```

Def. Depending on the status type definition.

| | |
|---------------------|--|
| capture_time | Provides an “Extended register” specific date and time information showing when the value of the attribute <i>value</i> has been captured. octet-string, formatted as specified in 4.1.6.1 for <i>date-time</i> . |
|---------------------|--|

Method description

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

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

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

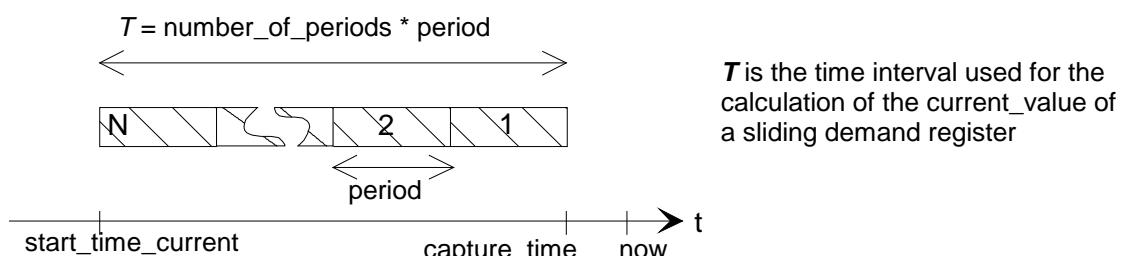


Figure 8 – The time attributes when measuring sliding demand

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

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

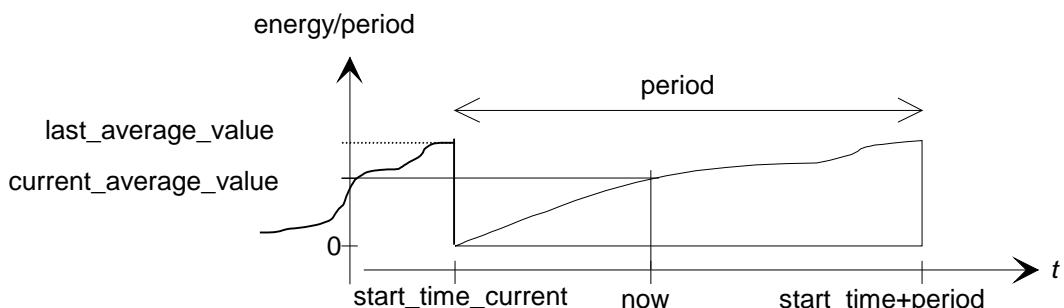


Figure 9 – The attributes in the case of block demand

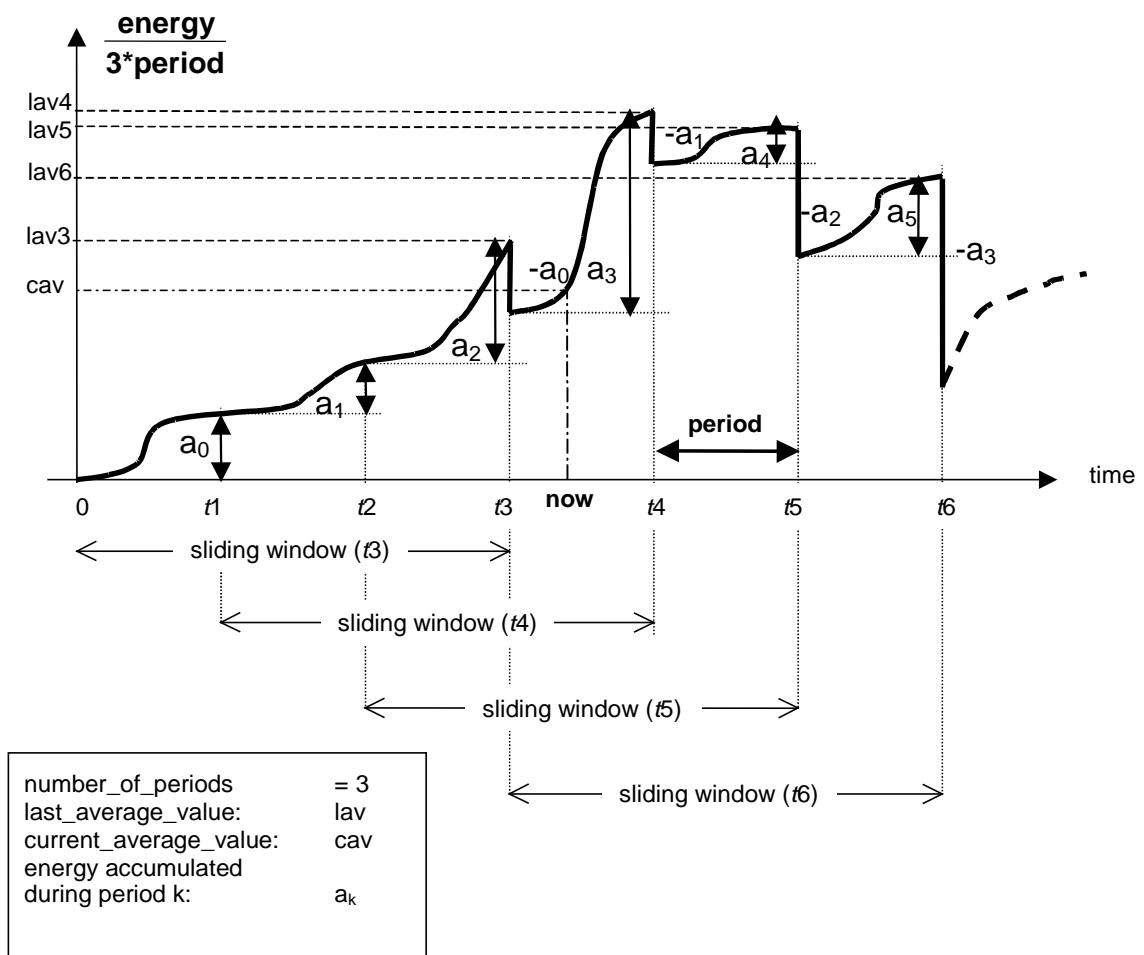


Figure 10 – The attributes in the case of sliding demand (number of periods = 3)

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

Attribute description

| | |
|------------------------------|---|
| logical_name | Identifies the “Demand register” object instance. See 6.3.1 and 6.4.6. |
| current_average_value | <p>Provides the current value (running demand) of the energy accumulated since <i>start_time</i>, divided by <i>number_of_periods*period</i>.</p> <p>The data type of the value depends on the instantiation defined by <i>logical_name</i> and possibly on the choice of the manufacturer. The type has to be chosen so that – together with the <i>logical_name</i> – unambiguous interpretation of the value is possible.</p> <p>If a quantity other than energy is measured, other calculation methods may apply (for example for calculating average values of voltage or current).</p> <p>CHOICE For data types, see “Register” in 4.3.2, value attribute.</p> |
| last_average_value | <p>Provides the value of the energy accumulated (over the last <i>number_of_periods*period</i>) divided by <i>number_of_periods*period</i>. The energy of the current (not terminated) period is not considered by the calculation.</p> <p>If a quantity other than energy is measured, other calculation methods may apply (for example for calculating average values of voltage or current).</p> <p>CHOICE For data types, see “Register” IC, value attribute.</p> |
| scaler_unit | See the specification of IC “Register” in 4.3.2. |
| status | <p>Provides “Demand register” specific status information. The data type and the encoding depend on the instantiation and possibly on the choice of the manufacturer. For the interpretation, extra information from the manufacturer may be necessary.</p> <p>CHOICE For data types, see “Extended register” IC in 4.3.3, status attribute.</p> <p>Def. Depending on the status type definition.</p> |
| capture_time | <p>Provides the date and time when the <i>last_average_value</i> has been calculated.</p> <p>octet-string, formatted as specified in 4.1.6.1 for <i>date-time</i>.</p> |
| start_time_current | <p>Provides the date and time when the measurement of the <i>current_average_value</i> has been started.</p> <p>octet-string, formatted as specified in 4.1.6.1 for <i>date-time</i>.</p> |
| period | <p>Period is the interval between two successive updates of the <i>last_average_value</i>. (<i>number_of_periods*period</i> is the denominator for the calculation of the demand).</p> <p>double-long-unsigned Measuring period in seconds</p> <p>The behaviour of the meter after writing a new value to this attribute shall be specified by the manufacturer.</p> |

| | |
|--------------------------|--|
| number_of_periods | The number of periods used to calculate the <i>last_average_value</i> . number_of_periods >= 1 <ul style="list-style-type: none"> - <i>number_of_periods</i> > 1 indicates that the <i>last_average_value</i> represents "sliding demand", - <i>number_of_periods</i> = 1 indicates that the <i>last_average_value</i> represents "block demand". The behaviour of the meter after writing a new value to this attribute shall be specified by the manufacturer. |
|--------------------------|--|

Method description

| | |
|---------------------------|--|
| reset (data) | This method forces a reset of the object. Activating this method provokes the following actions: <ul style="list-style-type: none"> - the current period is terminated; - the <i>current_average_value</i> and the <i>last_average_value</i> are set to their default values; - the <i>capture_time</i> and the <i>start_time_current</i> are set to the time of the execution of <i>reset (data)</i>. data ::= integer (0) |
| next_period (data) | This method is used to trigger the regular termination (and restart) of a period. Closes (terminates) the current measuring period. Updates <i>capture_time</i> and <i>start_time</i> and copies <i>current_average_value</i> to <i>last_average_value</i> , sets <i>current_average_value</i> to its default value. Starts the next measuring period. REMARK The old <i>last_average_value</i> (and <i>capture_time</i>) can be read during the time "period". The old <i>current_average_value</i> is not available any more at the interface. data ::= integer (0) |

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

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

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

Attribute description

| | |
|----------------------------|---|
| logical_name | Identifies the “Register activation” object instance. See 6.2.11. |
| register_assignment | <p>Specifies an ordered list of COSEM objects assigned to the “Register activation” object. The list may contain different kinds of COSEM objects, for example instances of “Register”, “Extended register” or “Demand register”.</p> <p>array object_definition</p> <pre>object_definition ::= structure { class_id: long-unsigned, logical_name: octet-string }</pre> |
| mask_list | <p>Specifies a list of register activation masks. Each entry (mask) is identified by its mask_name and contains an array of indices referring to the registers assigned to the mask (the first object in register_assignment is referenced by index 1, the second object by index 2,...).</p> <p>array register_act_mask</p> <pre>register_act_mask ::= structure { mask_name: octet-string, index_list: index_array } mask_name has to be uniquely defined within the object index_array ::= array unsigned</pre> |
| active_mask | <p>Defines the currently active mask. The mask is defined by its mask_name (see mask_list).</p> <p>The <i>active_mask</i> defines the registers currently enabled; all other registers listed in the <i>register_assignment</i> are disabled.</p> |

Method description

| | |
|--------------------------------|---|
| add_register (data) | Adds one more registers to the attribute <i>register_assignment</i> . The new register is added at the end of the array; i.e. the newly added register has the highest index. The indices of the existing registers are not modified. |
| | <p>data ::= structure</p> <pre>{ class_id: long-unsigned, logical_name: octet-string }</pre> |
| add_mask (data) | Adds another mask to the attribute <i>mask_list</i> . If there exists already a mask with the same name, the existing mask will be overwritten by the new mask. |
| | <p>data ::= register_act_mask (see above)</p> |
| delete_mask (data) | Deletes a mask from the attribute <i>mask_list</i> . The mask is defined by its mask name. |
| | <p>data ::= octet-string (mask_name)</p> |

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

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

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

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

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

The size of profile data is determined by three parameters:

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

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

Attribute description

| | |
|---------------------|--|
| logical_name | Identifies the “Profile generic” object instance. For examples, see 6.2.18, 6.2.20, 0, 6.2.35, 6.2.38, 6.2.41, 6.2.52, 6.2.55 6.2.56, 6.3.2 etc. |
| buffer | <p>Contains a sequence of entries. Each entry contains values of the captured objects.</p> <p>compact-array or array entry</p> <pre>entry ::= structure { CHOICE { -- simple data types null-data [0], boolean [3], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], utf8-string [12], bcd [13], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], enum [22], float32 [23], float64 [24], date-time [25], date [26], time [27], -- complex data types array [1], structure [2], compact-array [19] } }</pre> |

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

Default: The *buffer* is empty after reset.

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

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

selective access (see 4.1.4) to the attribute *buffer* may be available (optional). The selective access parameters are as defined below.

| | |
|------------------------|--|
| capture_objects | <p>Specifies the list of capture objects that are assigned to the object instance. Upon a call of the <i>capture</i> (data) method or automatically in defined intervals, the selected attributes are copied into the <i>buffer</i> of the profile.</p> <p>array capture_object_definition</p> <pre>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.</pre> |
| capture_period | <p>>= 1: Automatic capturing assumed. Specifies the capturing period in seconds.</p> <p>0: No automatic capturing; capturing is triggered externally or capture events occur asynchronously.</p> |
| sort_method | <p>If the profile is unsorted, it works as a “first in first out” buffer (it is hence sorted by capturing, and not necessarily by the time maintained in the “Clock” object). If the <i>buffer</i> is full, the next call to <i>capture</i> () will push out the first (oldest) entry of the <i>buffer</i> to make space for the new entry.</p> <p>If the profile is sorted, a call to <i>capture</i> () will store the new entry at the appropriate position in the buffer, moving all following entries and probably losing the least interesting entry. If the new entry would enter the <i>buffer</i> after the last entry and if the <i>buffer</i> is already full, the new entry will not be retained at all.</p> <p>enum: (1) fifo (first in first out), (2) lifo (last in first out), (3) largest, (4) smallest, (5) nearest_to_zero, (6) farest_from_zero</p> <p>Def. fifo</p> |
| sort_object | <p>If the profile is sorted, this attribute specifies the register or clock that the ordering is based upon.</p> <p>capture_object_definition See above.</p> <p>Def. no object to sort by (only possible with sort_method fifo or lifo)</p> <p>NOTE 1 If the sort_method is FIFO or LIFO, then all elements of the capture_object_definition specifying the sort object can be zero.</p> |

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

Parameters for selective access to the buffer attribute

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

range_descriptor ::= structure

```
{
    restricting_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

    CHOICE
    {
        -- simple data types
        double-long          [5],
        double-long-unsigned [6],
        octet-string         [9],
        visible-string       [10],
        utf8-string          [12],
        integer              [15],
        long                 [16],
        unsigned             [17],
        long-unsigned         [18],
        long64               [20],
        long64-unsigned       [21],
        float32              [23],
        float64              [24],
        date-time            [25],
        date                 [26],
        time                 [27]
    }
    to_value:                                         Newest or largest entry to retrieve
    {see above}

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

```

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

```

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

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

Method description

| | |
|-----------------------|---|
| reset (data) | Clears the <i>buffer</i> . It has no valid entries afterwards; <i>entries_in_use</i> is zero after this call. This call does not trigger any additional operations on the capture objects. Specifically, it does not reset any attributes captured. data ::= integer (0) |
| capture (data) | Copies the values of the objects to capture into the <i>buffer</i> by reading each capture object. Depending on the <i>sort_method</i> and the actual state of the <i>buffer</i> this produces a new entry or a replacement for the less significant entry. As long as not all entries are already used, the <i>entries_in_use</i> attribute will be incremented. This call does not trigger any additional operations within the capture objects such as <i>capture ()</i> or <i>reset ()</i> . Note, that if more than one attribute of an object need to be captured, they have to be defined one by one on the list of <i>capture_objects</i> . If the <i>attribute_index</i> = 0, all attributes are captured. data ::= integer (0) |

Behaviour of the object after modification of certain attributes

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

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

Restrictions

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

Profile used to define a subset of preferred readout values

By setting *profile_entries* to 1, a “Profile generic” object can be used to define a set of preferred readout values. See also 6.2.18. Setting *capture_period* to 1 ensures that the values are updated every second.

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

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

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

Attribute description

| | |
|---------------------|--|
| logical_name | Identifies the “Utility tables” object instance. See 6.2.33. |
| table_ID | Table number. This table number is as specified in the ANSI standard and may be either a standard table or a manufacturer’s table. |
| length | Number of octets in table buffer. |
| buffer | Contents of the table. <i>Selective access</i> (see 4.1.4) to the attribute <i>buffer</i> may be available (optional). The selective access parameters are defined below. |

Parameters for selective access to the buffer attribute

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

```

offset_selector ::= structure
{
  Offset: double-long-
         unsigned          Offset in octets to the start of access area, relative to the start of
                           the table.
  Count: long-unsigned        Number of octets requested or transferred
}
index_selector ::= structure
{
  Index: array long-
         unsigned          Sequence of indices to identify elements within the table's
                           hierarchy.
  Count: long-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.
}

```

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

This IC allows to group homogenous entries, identical attributes of multiple objects, which are all instances of the same IC, and in their *logical_name* (OBIS code) the value in value groups A to D and F is identical. The possible values in value group E are defined in Clause 7 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 65 or the “UNIPEDE voltage dip quantities” table, see Table 67.

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

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

Attribute description

| | |
|--------------------------|---|
| logical_name | Identifies the “Register table” object instance. When the format of the <i>logical_name</i> is A.B.C.D.255.F; the values A to D and F define the common part of the logical name of the objects, the attributes of which are captured. Only one attribute of the objects concerned can be captured (for example the <i>value</i> attribute). When the format of the <i>logical_name</i> is A.B.98.10.x.255, several instances of the “Register table” IC can be used to capture different attributes of the objects concerned. The value group E numbers the instances. See 7.4.4. |
| table_cell_values | Holds the value of the attributes captured, as they would be returned to a GET or Read .request to the individual attributes. compact array or array table_cell_entry table_cell_entry ::= CHOICE { -- simple data types null-data [0], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], utf8-string [12], bcd [13], integer [15], long [16], unsigned [17], long-unsigned [18], |

```

long64          [20],
long64-unsigned [21],
float32         [23],
float64         [24],
-- complex data types
structure        [2]
}

```

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

table_cell_definition Specifies the list of attributes captured in the register table.

structure

```

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

where:

- class_id defines the common class_id of the objects the attributes of which are captured;
- logical_name contains the common logical name of the objects, with E = 255 (wildcard);
- group_E_values contain the list of cell identifiers, of type *unsigned*, as defined in the respective table of Clause 7;
- attribute_index is a pointer to the attribute within the object. attribute_index 0 refers to all public attributes.

If the *logical_name* of the “Register table” object is in the format A.B.C.D.255.F and the defined attribute of all objects identified in the respective table in Clause 7 are captured, then attribute 3 may not be accessible. In this case:

- the class_id shall be 1 “Data”, 3 “Register” or 4 “Extended register”;
- the *logical_name* of the objects to be captured is defined by the *logical_name* of the “Register table” object and the respective table in Clause 7.
- the attribute index shall be 2 (value).

scaler_unit See the description of IC “Register”.

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

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

Method description

reset (data) Clears the *table_cell_values*. It has no effect on the attributes captured.
data ::= integer (0)

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

Behaviour of the object after modification of the `table_cell_definition` attribute

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

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

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

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

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

Attribute description

logical_name Identifies the “Status mapping” object instances. See 6.2.38, 6.2.40, 6.2.47 and 6.3.7.

status_word Contains the current value of the status word.

CHOICE

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

The size of the `status_word` is $n \times 8$ bits, the maximum size is 65 536 bits.

NOTE 1 Manufacturers may choose any of the types listed above. However, the status word is always interpreted as a bit-string.

mapping_table Contains the mapping of the `status_word` to the positions in the reference table.

structure

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

Where:

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

4.3.10 Compact data

4.3.10.1 Compact data (class_id = 62, version = 0)

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

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

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

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

The client can reconstruct the data in the uncompacted form – i.e. including the COSEM attribute descriptor, the data type and the data values – using the *capture_objects*, *template_id* 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 | | | | x + 0x38 |
| 2. capture (data) | o | | | | x + 0x40 |

Attribute description

logical_name Identifies the “Compact data” object instance. See 6.2.34.

| | |
|------------------------|---|
| compact_buffer | Contains the values of the attributes captured as an <i>octet-string</i> . When the data captured is of type <i>octet-string</i> , <i>bit-string</i> , <i>visible-string</i> , <i>utf8-string</i> or <i>array</i> the length is also included here. |
| capture_objects | <p>Specifies the list of COSEM object attributes that are assigned to the “Compact data” object instance.</p> <p>The <i>template_id</i> attribute shall be the first element in the <i>capture_objects</i> array.</p> <p>Upon an explicit or implicit invocation of the <i>capture (data)</i> method the values of the selected attributes are captured into the <i>compact_buffer</i>.</p> <pre>array capture_object_definition capture_object_definition ::= structure { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer, data_index: long-unsigned }</pre> |

Where:

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

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

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

The encoding is specified in Table 16.

| | |
|--------------------|--|
| template_id | Contains the identifier of the template. It shall uniquely identify the instance of the “Compact data” IC and the <i>template_description</i> . |
|--------------------|--|

| | |
|-----------------------------|---|
| template_description | <p>Provides the data type of each attribute captured. It is an <i>octet-string</i> generated automatically by the server upon the programming of the <i>capture_objects</i> and it has the following structure:</p> <ul style="list-style-type: none"> – the first octet is 0x02 (the tag of a structure); – this is followed by the number of elements in the structure – the same as the number of elements in the <i>capture_objects</i> array – encoded as a variable length integer; – this is followed by the data type of each attribute, in the same order as in the <i>capture_object</i> array: <ul style="list-style-type: none"> – 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 <i>bit-string</i> [4], <i>octet-string</i> [9], <i>visible-string</i> [10], <i>utf8-string</i> [12] the length of the string is part of the data held in the <i>compact_buffer</i>; – in the case of an <i>array</i> [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 <i>compact_buffer</i>; – in the case of a <i>structure</i> [2], the data type is represented by a single octet 0x02, followed by the number of elements inside the structure followed by the tag of each element of the structure. |
| capture_method | <p>Defines the way the <i>compact_buffer</i> is updated.</p> <p>enum:</p> <ul style="list-style-type: none"> (0) Capture upon invoking the <i>capture</i> (data) method. This may occur remotely or locally (explicit capturing), (1) Capture upon reading the <i>compact_buffer</i> attribute (implicit capturing). |
| Method description | |
| reset (data) | <p>Clears the <i>compact_buffer</i>. After invoking this method the <i>compact_buffer</i> holds an octet-string of 0 length, until a new capture takes place.</p> <p>This call does not trigger any additional operations of the capture objects. Specifically, it does not reset any captured attributes.</p> <p>data ::= integer(0)</p> |
| capture (data) | <p>Copies the values of the attributes into the <i>compact_buffer</i> by reading each capture object.</p> <p>This call does not trigger any additional operations within the capture objects such as <i>capture</i> () or <i>reset</i> ().</p> <p>data ::= integer(0)</p> |

Behaviour of the object after modification of certain attributes:

Any modification of the *capture_objects* resets the *compact_buffer* and automatically updates the *template_description*.

Restrictions

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

4.3.10.2 Examples for using compact data

4.3.10.2.1 Daily billing data

Table 6 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 6 – 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 |

Table 7 shows the attributes of the “Compact data” object.

Table 7 – Attributes of the “Compact data” object

| | |
|---|--|
| capture_objects (array) | For the elements of the array, see columns 2, 3, 4 & 5 of Table 6. |
| template_id (unsigned) | 0 |
| template_description (octet-string) | -- For the data types, see column 7 of Table 6. 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 6. 00 51EE5305 29 18 0061FFD3 000D33D2 0012D687 0023CACE 06414243444546 1ED2 |

For comparison, the A-XDR encoding of the same data as if they were accessed using a GET-WITH-LIST service is shown in Table 8. Only the encoding of the result (SEQUENCE OF Get-Data-Result) is shown.

Table 8 – 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. | | |

4.3.10.2.2 Diagnostic and Alarm data

Table 9 shows the diagnostic and alarm data that are captured – together with the mandatory *template_id* – to the *compact_buffer* attribute of a “Compact data” object.

Table 9 – 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” |

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

Table 10 – Attributes of the “Compact data” object

| | |
|--|---|
| capture_objects (array) | For the elements of the array, see column 2, 3, 4 & 5 of Table 9. |
| template_id (unsigned) | 1 |
| template_description (octet-string) | -- For the data types, see column 7 of Table 9. 02 08 11 12 12 12 12 09 12 09 |
| compact_buffer (octet-string) 29 bytes | -- For the values, see column 8 of Table 9. 01 4200 4108 4308 02FB 084142434445464748 05FC 084445464748494A4B |

For comparison, the A-XDR encoding of the data as if they were read from the buffer attribute of a “Profile generic” object is shown in Table 11 (only the Data is shown).

Table 11 – 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 |

4.3.10.2.3 Logbook reading

In this example, the data to be compacted is the *buffer* attribute of a Logbook held by a “Profile generic” object capturing 2 elements, as shown in Table 12.

Table 12 – 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:
- 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.

Table 13 shows the data to be captured by the “Compact data” object.

Table 13 – 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 | 7 | 0-0:99.98.0.255 | 2 | 0 | dyn. ¹ | array of structure | 2 |

¹ The size is dynamic and depends on the number of entries captured.
² See Table 12.

Table 14 shows the attributes of the “Compact data” object.

Table 14 – Attributes of the “Compact data” object

| | |
|--|--|
| capture_objects (array) | For the elements of the array, see column 2, 3, 4 & 5 of Table 13. The capture object has only 2 elements: the Template_id and the <i>buffer</i> attribute of the Logbook. |
| template_id (unsigned) | 2 |
| template-description (octet-string) | -- For the data types, see column 7 of Table 13. 02 02 11 01 02 02 06 11 Meaning: 02 02 - a structure of 2 elements 11 - first element is an unsigned 01 02 02 - second element is an array of structure with two elements in the structure 06 first one is a double-long-unsigned 11 second one is an unsigned |
| compact_buffer (octet-string) | -- For the values, see column 8 of Table 13. 02 -- value of the template-id 32 -- number of the elements in the array and $50*5 = 250$ bytes (for 50 elements in the log book) 252 bytes in total |

For comparison, the A-XDR encoding of the same data when read from the *buffer* attribute of a “Profile generic” object is shown in Table 15.

Table 15 – 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 |

4.4 Interface classes for access control and management

4.4.1 Overview

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

- the “Association SN” class – see 4.4.3 – and the “Association LN” class – see 4.4.4 – model AAs. Their instances, the Association objects provide the list of objects accessible in each AA, manage and control access rights to their attributes and methods. They also manage the authentication of the communicating partners;
- the “SAP Assignment” class – see 4.4.5 – models the logical structure of the server;
- the “Image transfer” class – see 4.4.6 – models the firmware update process;
- the “Security setup” class – see 4.4.7 – models the elements of the security context. “Security setup” objects are referenced from the “Association” objects and allow configuring security suites and security policies and managing security material;
- the “Push setup” class – see 4.4.8 – models the push operation of the server;
- the “Data protection” class – see 4.4.9 – specifies the necessary elements to apply cryptographic protection to COSEM object attribute values as well as to method invocation and return parameters.

4.4.2 Client user identification

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

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

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

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

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

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

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

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

4.4.3 Association SN class (class_id = 12, version 4)

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

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

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 82/492 |
|-----------------------|------------|-------------------------|--------|

| 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 | | | | |
| 4. reserved from previous versions | o | | | | |
| 5. change_secret (data) | o | | | | |
| 6. reserved from previous versions | o | | | | |
| 7. reserved from previous versions | | | | | |
| 8. reply_to_HLS_authentication (data) | o | | | | |
| 9. add_user (data) | o | | | | |
| 10. remove_user (data) | o | | | | |

Attribute description

logical_name Identifies the “Association SN” object instance. See 6.2.28.

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

```
objlist_type ::= array          objlist_element
objlist_element ::= structure
{
    base_name:      long,
    class_id:       long-unsigned,
    version:        unsigned,
    logical_name:   octet-string
}
```

selective access (see 4.1.4) to the attribute object_list may be available. The access selector values and their parameters are as defined below.

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

access_rights_type ::= array access_rights_element

```
access_rights_element ::= structure
{
    base_name:          long,
    attribute_access:   attribute_access_descriptor,
    method_access:      method_access_descriptor
}
```

attribute_access_descriptor ::= array attribute_access_item

```
attribute_access_item ::= structure
{
    attribute_id:       integer,
    access_mode:        enum,
```

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

| | |
|---|----------------------------------|
| { | Bit attribute access_mode |
| | (0) read-access, |
| | (1) write-access, |
| | (2) authenticated request, |
| | (3) encrypted request, |
| | (4) digitally signed request, |
| | (5) authenticated response, |
| | (6) encrypted response, |
| | (7) digitally signed response |

```
}
```

access_selectors: CHOICE

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

}

method_access_descriptor ::= array method_access_item

```
method_access_item ::= structure
{
    method_id:          integer,
    access_mode:        enum
```

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

```

{
    Bit   method access_mode
    {
        (0)  access,
        (1)  not-used,
        (2)  authenticated request,
        (3)  encrypted request,
        (4)  digitally signed request,
        (5)  authenticated response,
        (6)  encrypted response,
        (7)  digitally signed response
    }
}

```

selective access (see 4.1.4) to the attribute *access_rights_list* may be available (optional). The access selector values and their parameters are as defined below.

| | |
|-----------------------|--|
| security_setup | References the “Security setup” object by its <i>logical_name</i> . The referenced object manages security for a given “Association SN” object instance. |
| user_list | <p>Contains the list of users allowed to use the AA managed by the given instance of the “Association SN” IC.</p> <p>array user_list_entry</p> <pre> user_list_entry ::= structure { user_id: unsigned, user_name: visible-string } </pre> <p>Where:</p> <ul style="list-style-type: none"> - <i>user_id</i> is the identifier of the user (this value is carried by the calling-AE-invocation-id field of the AARQ); - <i>user_name</i> is the name of the user. <p>If the <i>user_list</i> attribute is empty – i.e. it is an array of 0 elements – any user can use the AA, i.e. the calling-AE-invocation-id field of the AARQ is ignored.</p> <p>If the <i>user_list</i> attribute is not empty then only the users in the list can establish the AA, i.e. the calling-AE-invocation-id field of the AARQ shall be present and its value shall match one of <i>user_ids</i> in the <i>user_list</i> or else, the AA is not established.</p> |
| current_user | <p>Holds the identifier of the current user.</p> <p><i>current_user</i> ::= <i>user_list_entry</i> (see above)</p> <p>If the <i>user_list</i> is empty, then <i>current_user</i> shall be a structure {<i>user_id</i>: unsigned 0, <i>user_name</i>: visible string of 0 elements}</p> |

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

| Access selector value | Parameter | Available with attribute | Comment |
|-----------------------|---|--------------------------|---|
| 1 | class_id: long-unsigned | 2 | Delivers the subset of the object_list for a specific class_id. For the response: data ::= objlist_type |
| 2 | structure { class_id: long-unsigned, logical_name: octet-string } | 2 | Delivers the entry of the object_list for a specific class_id and 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 |

Method description

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

data ::= array attribute_identification

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

where attribute_index is a pointer (i.e. offset) to the attribute within the object.
attribute_index 0 delivers all attributes; attribute_index 1 delivers the first attribute (i.e. *logical_name*), etc.).

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

NOTE 1 If at least one attribute has no read access right under the current association, then a *read_by_logicalname ()* to attribute index 0 reveals the error message "scope-of-access-violated", see DLMS UA 1000-2 Ed. 8.1:201, 9.5, page 211/310.

change_secret (data) Changes the LLS or HLS secret (for example password).
data ::= octet-string new secret

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

NOTE 3 In the case of HLS with GMAC, the (HLS_) secret is held by the "Security setup" object referenced in attribute

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

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

If the authentication is accepted, then the response (Read.confirm primitive) contains Result == OK and the result of the secret processing by the server of

the client's challenge to the server, f(CtoS) in the data service parameter of the Read.response service.

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

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

| | |
|-------------------------------|--|
| add_user (data) | Adds a user to the user_list. data ::= user_list_entry (see above) |
| remove_user (data) | Removes a user from the user_list. data ::= user_list_entry (see above) |

4.4.4 Association LN class (class_id = 15, version 3)

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

| Association LN | 0...MaxNofAss. | 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 |

Attribute description

| logical_name | Identifies the “Association LN” object instance. See 6.2.28. | | | | | | | | | | | | | | | | | | | |
|---------------------|---|-----------------------|-----|-----------------------|-----|--------------|-----|---------------|-----|------------------------|-----|--------------------|-----|---------------------------|-----|-------------------------|-----|---------------------|-----|---------------------------|
| object_list | Contains the list of visible COSEM objects with their class_id, version, <i>logical_name</i> and the access rights to their attributes and methods within the given AA. object_list_type ::= array object_list_element object_list_element ::= structure { class_id: long-unsigned, version: unsigned, logical_name: octet-string, access_rights: access_right } access_right ::= structure { attribute_access: attribute_access_descriptor, method_access: method_access_descriptor } attribute_access_descriptor ::= array attribute_access_item attribute_access_item ::= structure { attribute_id: integer, access_mode: enum | | | | | | | | | | | | | | | | | | | |
| | When the enum value is interpreted as an unsigned8, the meaning of each bit is as shown below: | | | | | | | | | | | | | | | | | | | |
| | <table border="1"> <thead> <tr> <th style="background-color: #ffffcc;">{</th> <th style="background-color: #ffffcc;">Bit</th> <th style="background-color: #ffffcc;">attribute access_mode</th> </tr> </thead> <tbody> <tr> <td style="background-color: #ffffcc;">(0)</td> <td style="background-color: #ffffcc;">read-access,</td> </tr> <tr> <td style="background-color: #ffffcc;">(1)</td> <td style="background-color: #ffffcc;">write-access,</td> </tr> <tr> <td style="background-color: #ffffcc;">(2)</td> <td style="background-color: #ffffcc;">authenticated request,</td> </tr> <tr> <td style="background-color: #ffffcc;">(3)</td> <td style="background-color: #ffffcc;">encrypted request,</td> </tr> <tr> <td style="background-color: #ffffcc;">(4)</td> <td style="background-color: #ffffcc;">digitally signed request,</td> </tr> <tr> <td style="background-color: #ffffcc;">(5)</td> <td style="background-color: #ffffcc;">authenticated response,</td> </tr> <tr> <td style="background-color: #ffffcc;">(6)</td> <td style="background-color: #ffffcc;">encrypted response,</td> </tr> <tr> <td style="background-color: #ffffcc;">(7)</td> <td style="background-color: #ffffcc;">digitally signed response</td> </tr> </tbody> </table> | { | Bit | attribute access_mode | (0) | read-access, | (1) | write-access, | (2) | authenticated request, | (3) | encrypted request, | (4) | digitally signed request, | (5) | authenticated response, | (6) | encrypted response, | (7) | digitally signed response |
| { | Bit | attribute access_mode | | | | | | | | | | | | | | | | | | |
| (0) | read-access, | | | | | | | | | | | | | | | | | | | |
| (1) | write-access, | | | | | | | | | | | | | | | | | | | |
| (2) | authenticated request, | | | | | | | | | | | | | | | | | | | |
| (3) | encrypted request, | | | | | | | | | | | | | | | | | | | |
| (4) | digitally signed request, | | | | | | | | | | | | | | | | | | | |
| (5) | authenticated response, | | | | | | | | | | | | | | | | | | | |
| (6) | encrypted response, | | | | | | | | | | | | | | | | | | | |
| (7) | digitally signed response | | | | | | | | | | | | | | | | | | | |
| | access_selectors: CHOICE { null-data [0], array integer [1] } method_access_descriptor ::= array method_access_item | | | | | | | | | | | | | | | | | | | |

```
method_access_item ::= structure
{
    method_id:      integer,
    access_mode:    enum
```

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

```
{
    Bit   method access_mode
    (0)  access,
    (1)  not-used,
    (2)  authenticated request,
    (3)  encrypted request,
    (4)  digitally signed request,
    (5)  authenticated response,
    (6)  encrypted response,
    (7)  digitally signed response
}
```

Where:

- the attribute_access_descriptor and the method_access_descriptor elements always contain all implemented attributes or methods;
- access_selectors contain a list of the supported selector values.

selective access (see 4.1.4) to the attribute *object_list* may be available (optional). The selective access parameters are as defined below.

associated_partners_id

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

associated_partners_type ::= structure

```
{
    client_SAP:      integer,
    server_SAP:      long-unsigned
}
```

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

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

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

| | |
|---------------------------------|---|
| application_context_name | In the COSEM environment, it is intended that an application context pre-exists and is referenced by its name during the establishment of an AA. This attribute contains the name of the application context for that AA. |
| | <pre>context_name_type ::= CHOICE { context_name_structure [2], octet-string [9] }</pre> |

The application context name is specified as OBJECT IDENTIFIER in DLMS UA 1000-2 Ed. 8.1:201 9.4.2.2.2.

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

context_name_structure ::= structure

```
{
    joint_iso_ctt_element: unsigned,
    country_element: unsigned,
    country_name_element: long-unsigned,
    identified_organization_element: unsigned,
    DLMS_UA_element: unsigned,
    application_context_element: unsigned,
    context_id_element: unsigned
}
```

Example 1: In the case of context_id(1) the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 01 11 01 (all values are hexadecimal).

When the context_name_type is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See DLMS UA 1000-2 Ed. 8.1:201 11.4.

Example 2: In the case of context_id(1) the A-XDR encoding is: 09 07 60 85 74 05 08 01 01 (all values are hexadecimal).

| | |
|---------------------------|---|
| xDLMS_context_info | Contains all the necessary information on the xDLMS context for the given AA. xDLMS_context_type ::= structure |
| | <pre>{ conformance: bit-string, max_receive_pdu_size: long-unsigned, max_send_pdu_size: long-unsigned, dlms_version_number: unsigned, quality_of_service: integer, cyphering_info: octet-string }</pre> <p>Where:</p> <ul style="list-style-type: none"> - the conformance element contains the xDLMS conformance block supported by the server. The length of the bit-string is 24 bits; - the max_receive_pdu_size element contains the maximum length for an xDLMS APDU, expressed in bytes that the client may send. This is the same as the server-max-receive-pdu-size parameter of the xDLMS initiateResponse APDU; - the max_send_pdu_size element, in an active AA contains the maximum length for an xDLMS APDU, expressed in bytes that the server may send. This is the same as the client-max-receive-pdu-size parameter of the xDLMS initiateRequest APDU; |

- the dlms_version_number element contains the DLMS version number supported by the server;
- the quality_of_service element is not used;
- the cyphering_info element – in an active AA – contains the dedicated key parameter of the xDLMS initiateRequest APDU. See DLMS UA 1000-2 Ed. 8.1:201 Clause 9.5.

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

```
mechanism_name_type ::= CHOICE
{
    mechanism_name_structure [2],
    octet-string [9]
}
```

The authentication mechanism name is specified as an OBJECT IDENTIFIER in DLMS UA 1000-2 Ed. 8.1:201 9.4.2.2.3.

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

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

Example 3: In the case of mechanism_id(1) the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 02 11 01 (all values are hexadecimal):

When the mechanism_name_type is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See DLMS UA 1000-2 Ed. 8.1:201 11.4.

EXAMPLE 4: In the case of mechanism_id(1) the A-XDR encoding is: 09 07 60 85 74 05 08 02 01 (all values are hexadecimal).

No mechanism_name is required when no authentication is used.

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

NOTE 2 In the case of HLS with GMAC, the (HLS_)secret is held by the “Security setup” object referenced in attribute 9, security_setup_reference.

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

enum: (0) non-associated,
 (1) association-pending,
 (2) associated

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

| | |
|---------------------|--|
| user_list | <p>Contains the list of users allowed to use the AA managed by the given instance of the "Association LN" IC.</p> <p>array user_list_entry</p> <pre>user_list_entry ::= structure { user_id: unsigned, user_name: visible-string }</pre> <p>Where:</p> <ul style="list-style-type: none"> - user_id is the identifier of the user (this value is carried by the calling-AE-invocation-id field of the AARQ); - user_name is the name of the user. <p>If the <i>user_list</i> attribute is empty – i.e. it is an array of 0 elements – any user can use the AA, i.e. the calling-AE-invocation-id field of the AARQ is ignored.</p> <p>If the <i>user_list</i> attribute is not empty then only the users in the list can establish the AA, i.e. the calling-AE-invocation-id field of the AARQ shall be present and its value shall match one of user_ids in the <i>user_list</i> or else the AA is not established.</p> |
| current_user | <p>Holds the identifier of the current user.</p> <p>current_user ::= user_list_entry (see above)</p> <p>If the <i>user_list</i> is empty, then <i>current_user</i> shall be a structure {user_id: unsigned 0, user_name: visible string of 0 elements}</p> |

Parameters for selective access to the *object_list* attribute

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

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

Method description

| | |
|---|---|
| reply_to_HLS_authentication (data) | The remote invocation of this method delivers to the server the result of the secret processing by the client of the server's challenge to the client, f(StoC), as the <i>data</i> service parameter of the ACTION.request primitive invoked. data ::= octet-string client's response to the challenge If the authentication is accepted, then the response (ACTION.confirm primitive) contains Result == OK and the result of the secret processing by the server of the client's challenge to the server, f(CtoS) in the <i>data</i> service parameter of the response service. data ::= octet-string server's response to the challenge If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back. |
| change_HLS_secret (data) | Changes the HLS secret (for example encryption key). data ::= octet-string new HLS secret The structure of the "new secret" depends on the security mechanism implemented. The "new secret" may contain additional check bits and it may be encrypted. |
| add_object (data) | Adds the referenced object to the <i>object_list</i> . data ::= object_list_element (see above) |
| remove_object (data) | Removes the referenced object from the <i>object_list</i> . data ::= object_list_element (see above) |
| add_user (data) | Adds a user to the <i>user_list</i> . data ::= user_list_entry (see above) |
| remove_user (data) | Removes a user from the <i>user_list</i> . data ::= user_list_entry (see above) |

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

This IC allows modelling the logical structure of physical devices, by providing information on the assignment of the logical devices to their SAP-s. See DLMS UA 1000-2 Ed. 8.1:201 Clause 10.

| 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 | | | | | |

Attribute description

| | |
|---------------------|---|
| logical_name | Identifies the "SAP assignment" objects instance. See 6.2.29. |
|---------------------|---|

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

```

asslist_type ::= array      asslist_element

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

```

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

Method description

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

4.4.6 Image transfer

4.4.6.1 General

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

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

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

4.4.6.2 The steps of the image transfer process

The Image transfer usually takes place in several steps:

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

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 94/492 |
|-----------------------|------------|-------------------------|--------|

For an example with more detailed explanations, see 4.4.6.4.

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

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

Attribute description

| | |
|---|--|
| logical_name | Identifies the "Image transfer" object instance. See 6.2.32. |
| image_block_size | Holds the ImageBlockSize, expressed in octets, which can be handled by the server. ImageBlockSize shall not exceed the ServerMaxReceivePduSize negotiated. NOTE 1 <i>image_block_size</i> is a property of the server. |
| image_transferred_blocks_status | Provides information about the transfer status of each ImageBlock. Each bit in the bit-string provides information about one individual ImageBlock: 0 = Not transferred, 1 = Transferred NOTE 2 The size of the attribute may be dynamic, i.e. it may grow upon reception of new ImageBlocks. |
| image_first_not_transferred_block_number | Provides the ImageBlockNumber of the first ImageBlock not transferred. Once the Image is complete, the value returned should be equal to or above the number of blocks calculated from the Image size and the ImageBlockSize. |
| image_transfer_enabled | Controls the enabling of the Image transfer process. boolean: FALSE = Disabled, TRUE = Enabled The image transfer methods can be invoked successfully only if the value of this attribute is TRUE. Setting the value of this attribute to FALSE disables all methods (invoking them fails). |

| | |
|-------------------------------|---|
| image_transfer_status | Holds the status of the Image transfer process. enum: (0) Image transfer not initiated, (1) Image transfer initiated, (2) Image verification initiated, (3) Image verification successful, (4) Image verification failed, (5) Image activation initiated, (6) Image activation successful, (7) Image activation failed |
| image_to_activate_info | Provides information on the Image(s) ready for activation. It is generated as the result of the Image verification process. The client may check this information before activating the Images. array image_to_activate_info_element |

image_to_activate_info_element ::= structure
{
 image_to_activate_size: double-long-unsigned,
 image_to_activate_identification: octet-string,
 image_to_activate_signature: octet-string
}

Where:

- image_to_activate_size is the size of the Image to be activated, expressed in octets;
- image_to_activate_identification is the identification of the Image to be activated, and may contain information like manufacturer, device type, version information, etc.;
- image_to_activate_signature is the signature of the Image to be activated.

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

Method description

| | |
|---------------------------------------|--|
| image_transfer_initiate (data) | Initializes the Image transfer process. data ::= structure { image_identifier: octet-string, image_size: double-long-unsigned } Where: - image_identifier identifies the Image to be transferred; - image_size holds the ImageSize, expressed in octets. |
| | NOTE 4 The <i>image_identifier</i> identifies the Image to be transferred (container) but it is not necessarily linked to its content, i.e. the Images which will be activated. That information can be retrieved from the <i>image_to_activate</i> attribute after verification of the Image transferred. After a successful invocation of the method the <i>image_transfer_status</i> attribute is set to (1) and the <i>image_first_not_transferred_block_number</i> is set to 0. Any subsequent invocation of the method resets the whole Image transfer process and all ImageBlocks need to be transferred again. |

| | |
|------------------------------------|---|
| image_block_transfer (data) | <p>Transfers one block of the Image to the server.</p> <p>data ::= structure</p> <pre>{ image_block_number: double-long-unsigned, image_block_value: octet-string }</pre> <p>After a successful invocation of the method the corresponding bit in the <i>image_transferred_blocks_status</i> attribute is set to 1 and the <i>image_first_not_transferred_block_number</i> attribute is updated.</p> |
| image_verify (data) | <p>Verifies the integrity of the Image before activation.</p> <p>data ::= integer (0)</p> <p>The result of the invocation of this method may be success, temporary_failure or other_reason. If it is not success, then the result of the verification can be found by retrieving the value of the <i>image_transfer_status</i> attribute.</p> <p>In the case of success, the <i>image_to_activate_info</i> attribute holds the information about the images to activate.</p> <p>NOTE 5 xDLMS services Action-Result / Data-Access-Result codes are specified in DLMS UA 1000-2 Ed. 8.1:201 9.5.</p> |
| image_activate (data) | <p>Activates the Image.</p> <p>data ::= integer (0)</p> <p>If the Image transferred has not been verified before, then this is done as part of the Image activation.</p> <p>The result of the invocation of this method may be success, temporary_failure or other-reason. If it is not success, then the result of the activation can be learned by retrieving the value of the <i>image_transfer_status</i> attribute.</p> <p>NOTE 6 xDLMS services Action-Result / Data-Access-Result codes are specified in DLMS UA 1000-2 Ed. 8.1:201 9.5.</p> |

4.4.6.4 Example for a standard Image transfer process

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

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

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

Step 1 (Optional): Get ImageBlockSize

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

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

Step 2: Client initiates Image transfer

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

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

Step 3: Client transfers ImageBlocks

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

Step 4: Client checks completeness of the Image

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

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

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

NOTE 1 The two mechanisms can be freely combined.

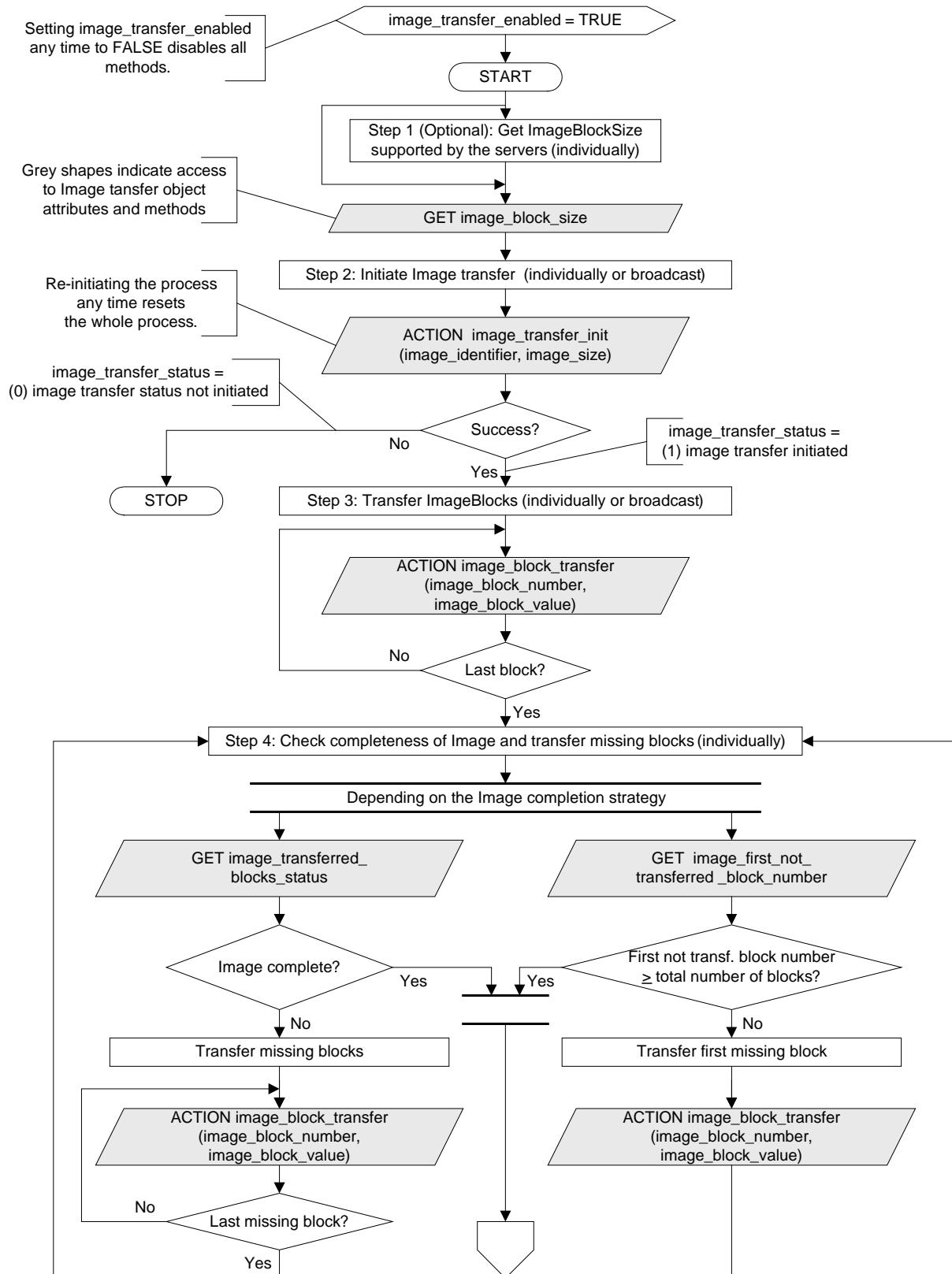
Step 5: Server verifies the Image

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

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

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

| | | | |
|-----------------------|------------|-------------------------|--------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 98/492 |
|-----------------------|------------|-------------------------|--------|



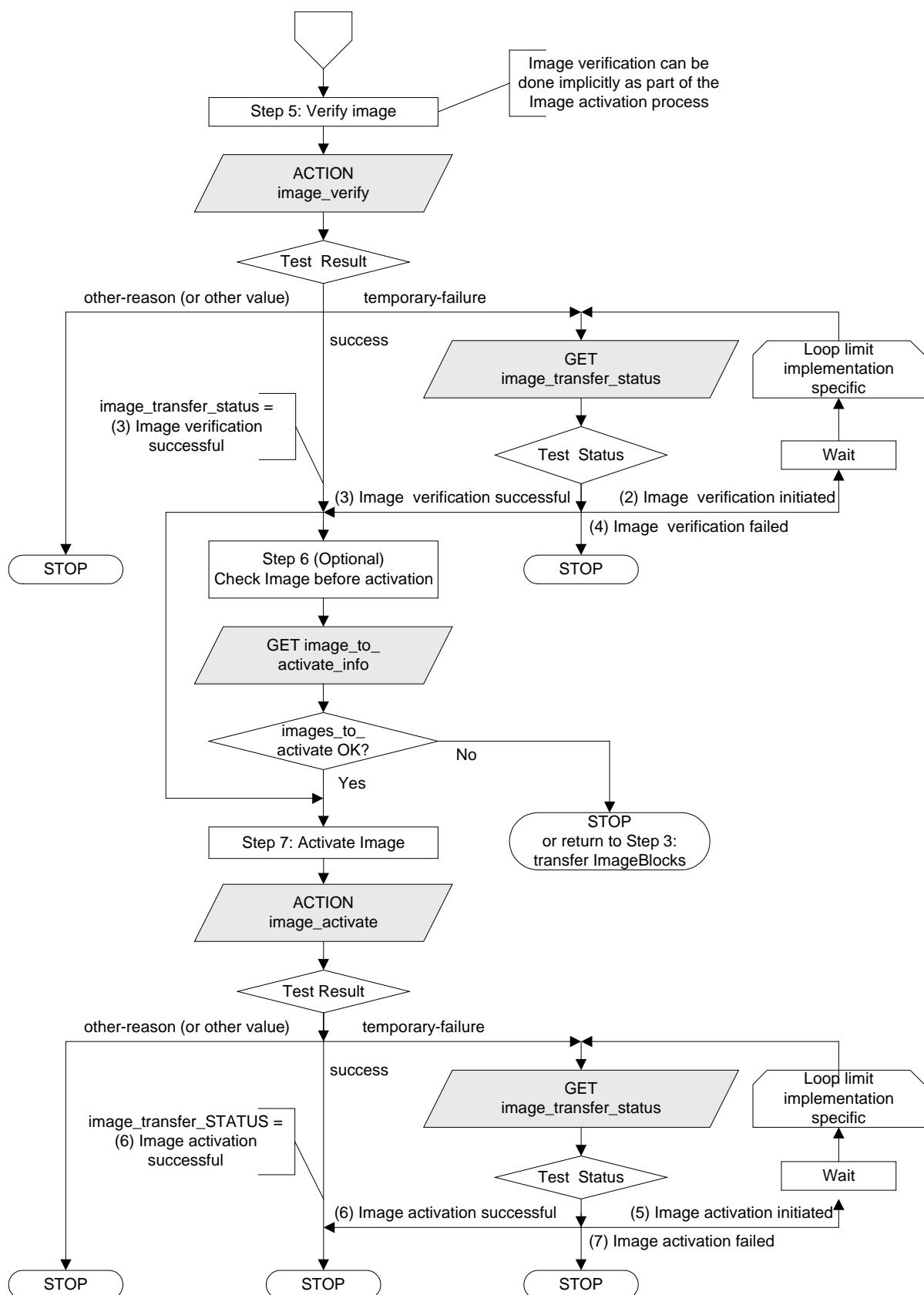


Figure 11 – Image transfer process flow chart

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

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

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

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

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

Step 7: Server activates the Image(s)

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

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

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

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

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 101/492 |
|-----------------------|------------|-------------------------|---------|

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

Instances of the “Security setup” IC contain the necessary information on the security suite in use and the security policy applicable between a client and a server identified by their respective system title. They also provide methods to increase the level of security and to manage 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 |

Attribute description

| | |
|------------------------|---|
| logical_name | Identifies the “Security setup” object instance. See 6.2.31. |
| security_policy | Enforces authentication and/or encryption and/or digital signature using the security algorithms available within the security suite. It applies independently for requests and responses. |

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

| Bit | Security policy |
|-----|----------------------------|
| 0 | unused, shall be set to 0, |
| 1 | unused, shall be set to 0, |
| 2 | authenticated request, |
| 3 | encrypted request, |
| 4 | digitally signed request, |
| 5 | authenticated response, |
| 6 | encrypted response, |
| 7 | digitally signed response |

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

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

| | |
|----------------------------|---|
| security_suite | Specifies the security algorithms available. enum: (0) AES-GCM-128 authenticated encryption and AES-128 key wrap, (1) AES-GCM-128 authenticated encryption, ECDSA P-256 digital signature, ECDH P-256 key agreement, SHA-256 hash, V.44 compression and AES-128 key wrap, (2) AES-GCM-256 authenticated encryption, ECDSA P-384 digital signature, ECDH P-384 key agreement, SHA-384 hash, V.44 compression and AES-256 key wrap (3) ... (15) reserved |
| client_system_title | Carries the (current) client system title: <ul style="list-style-type: none">- in the S-FSK PLC environment, the active initiator sends its system title using the CIASE protocol; NOTE 2 It is also held by the active_initiator attribute of the S-FSK Active initiator object; see 4.10.4.- during confirmed or unconfirmed AA establishment, it is carried by the calling-AP-title field of the AARQ APDU; If a client system title has already been sent during a registration process, like in the case of the S-FSK PLC profile, the client system title carried the AARQ APDU should be the same. Otherwise, the AA shall be rejected and appropriate diagnostic information should be sent.- in a pre-established AA, it can be written by the client using an unsecured service. |
| server_system_title | Carries the server system title. <ul style="list-style-type: none">- in the S-FSK PLC environment, the server sends its system title during the discover process, using the CIASE protocol;- during confirmed AA establishment, it is carried by the responding-AP-title field of the AARE APDU. <p>This attribute shall be read only.</p> |
| certificates | Carries X.509 v3 certificates available and stored in the server. array certificate_info certificate_info ::= structure { certificate_entity: enum: (0) server, (1) client, (2) certification authority, (3) other certificate_type: enum: (0) digital signature, (1) key agreement, (2) TLS, (3) other serial_number: octet-string, |

```

    issuer:          octet-string,
    subject:        octet-string,
    subject_alt_name: octet-string
}

```

For the elements serial_number, issuer, subject, subject_alt_name see DLMS UA 1000-2 Ed. 8.1:201 9.2.6.4.

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

For the server:

- Digital Signature Certificate,
- Static Key Agreement Certificate,
- TLS Certificate.

For each client and third party:

- Digital Signature Certificate,
- Static Key Agreement Certificate,
- TLS Certificate.

For Certification Authorities:

- Certification Authority Certificate.

Methods description

| | |
|-------------------------------------|--|
| security_activate (data) | Activates and strengthens the security policy: data ::= enum, as specified at the <i>security_policy</i> attribute. |
|-------------------------------------|--|

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

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

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

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 104/492 |
|-----------------------|------------|-------------------------|---------|

key_transfer (data) Used to transfer one or more symmetric keys.

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

data ::= array key_transfer_data

```
key_transfer_data ::= structure
{
    key_id:      enum:
                    (0) global unicast encryption key,
                    (1) global broadcast encryption key,
                    (2) authentication key,
                    (3) master key (KEK)
```

 key_wrapped: octet-string

}

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

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

The KEK is the master key.

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

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

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

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

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

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

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 105/492 |
|-----------------------|------------|-------------------------|---------|

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

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

```
data ::= array key_agreement_data
```

```
key_agreement_data ::= structure
{
    key_id: enum:
        (0) global unicast encryption key,
        (1) global broadcast encryption key,
        (2) authentication key,
        (3) master key (KEK)
    key_data: octet-string
}
```

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

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

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

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

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

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

generate_key_pair (data) Generates an asymmetric key pair as required by the security suite. The *data* parameter identifies the usage of the key pair to be generated.

```
data ::= enum:
    (0) digital signature key pair,
    (1) key agreement key pair,
    (2) TLS key pair
```

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

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

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 106/492 |
|-----------------------|------------|-------------------------|---------|

Model C(0e, 2s, ECC CDH) scheme.

| | |
|--|--|
| generate_certificate_request (data) | When this method is invoked, the server sends the Certificate Signing Request (CSR) data that is necessary for a CA to generate a certificate for a server public key. The data parameter identifies the key pair for which the certificate will be requested. data ::= enum: (0) digital signature key pair, (1) key agreement key pair, (2) TLS key pair |
|--|--|

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

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

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

response data ::= octet-string,
formatted as specified in PKCS #10 (RFC 2986).

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

| | |
|----------------------------------|--|
| import_certificate (data) | Imports an X.509 v3 certificate of a public key. |
|----------------------------------|--|

data ::= octet-string

Data is formatted as DER encoded X.509 v3 (RFC 5280).

| | |
|----------------------------------|--|
| export_certificate (data) | Exports an X.509 v3 certificate in the server. |
|----------------------------------|--|

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

```
data ::= certificate_identification
certificate_identification ::= structure
{
    certificate_identification_type: enum:
        (0) certificate_identification_entity,
        (1) certificate_identification_serial

    certification_identification_options: CHOICE
    {
        certificate_identification_by_entity,
        certificate_identification_by_serial
    }
}
```

| | |
|--------------------------------------|--|
| export_certificate (data) | <pre>certificate_identification_by_entity ::= structure { certificate_entity: enum: (0) server, (1) client, (2) certification authority, (3) other certificate_type: enum: (0) digital signature, (1) key agreement, (2) TLS system_title: octet-string }</pre> |
| | <pre>certificate_identification_by_serial ::= structure { serial_number: octet-string, issuer: octet-string }</pre> |
| | <p>response data ::= octet-string response data octet-string is formatted as X.509 v3 DER format.</p> |
| | <p>If no matching certificate is found, the response shall contain an appropriate error code.</p> |
| remove_certificate (data) | <p>Removes X.509 v3 certificate in the server.</p> <p>data ::= certificate_identification</p> <p>See export_certificate for certificate_identification.</p> |
| | <p>NOTE 12 The certificates of the server for digital signature, key agreement and TLS cannot be removed from the server. When update of these certificates is needed the new key pair is generated, new certificate request is generated and new certificate is imported.</p> |

4.4.8 Push interface classes and objects

4.4.8.1 Overview

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

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

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 108/492 |
|-----------------------|------------|-------------------------|---------|

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

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

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

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

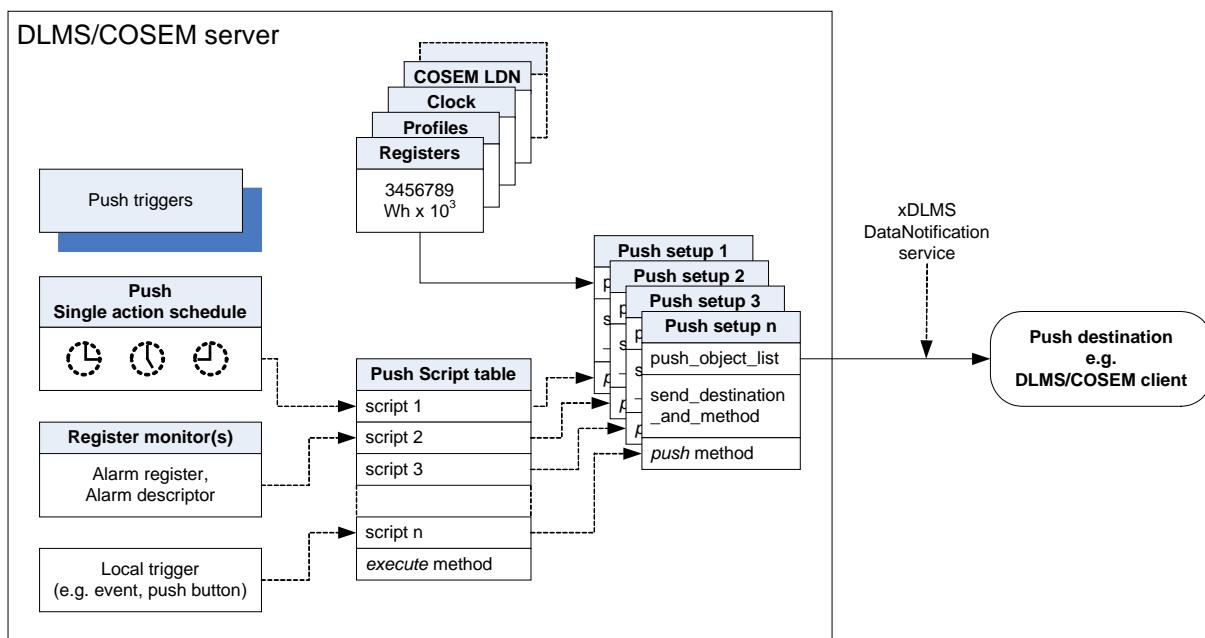


Figure 12 – COSEM model of push operation

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

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

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

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

The alarms are held by *Alarm register* or *Alarm descriptor* objects, see 6.2.56.

NOTE 1 The structure of the *Alarm descriptor* as well as the alarm conditions needs to be defined in a project specific companion specification.

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

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

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

NOTE 2 Details are subject to project specific companion specifications.

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

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

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

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

The push takes place upon invoking the *push* method, triggered by a Push “Single action schedule” object, by an alarm “Register monitor” object, by a dedicated internal event or externally. After the push operation has been triggered, it is executed according to the settings made in the given “Push setup” object. Depending on the communication window settings, the push is executed immediately or as soon as a communication window becomes active, after a random delay. If the push was not successful, retries are made. Push windows, delays and retries are show in Figure 13.

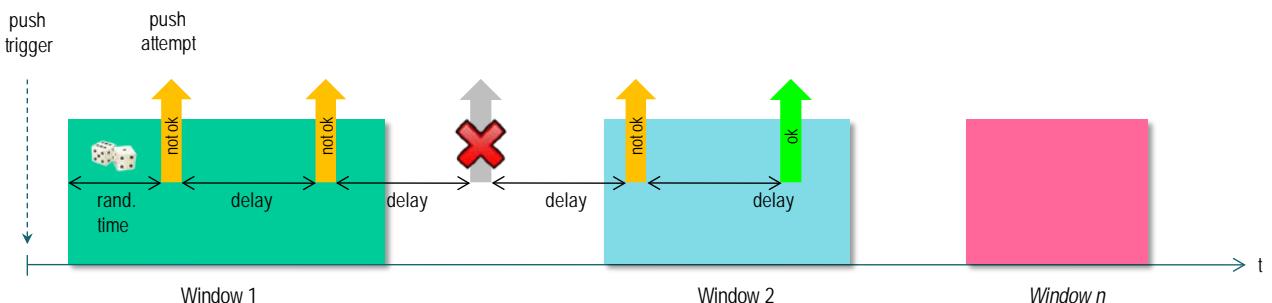


Figure 13 – Push windows and delays

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

Attribute description

logical_name Identifies the “Push setup” object instance. See 6.2.22.

push_object_list Defines the list of attributes to be pushed.

Upon invocation of the *push* (data) method the items are sent to the destination defined in the *send_destination_and_method* attribute.

array object_definition

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

Where:

- attribute_index is a pointer to the attribute within the object, identified by class_id and logical_name: attribute_index 1 refers to the 1st attribute (i.e. the logical_name), attribute_index 2 to the 2nd attribute etc.; attribute_index 0 refers to all public attributes;
- data_index is a pointer selecting one or several specific elements of an attribute with a complex data type (structure or array).

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

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

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

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

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

The encoding is specified in Table 16.

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

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

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

send_destination_and_method

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

send_destination_and_method ::= structure

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

Where:

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

transport_service_type ::= enum:

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

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

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

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

message_type ::= enum:

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

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

| | |
|-------------------------------------|---|
| communication_window | <p>Defines the time points when the communication window(s) for the push become(s) active (<i>start_time</i>) and inactive (<i>end_time</i>). See Figure 13.</p> <p>array window_element</p> <pre>window_element ::= structure { start_time: octet-string, end_time: octet-string }</pre> <p><i>start_time</i> and <i>end_time</i> are formatted as specified in clause 4.1.6.1 for <i>date-time</i> including wildcards.</p> <p>If the end of a communication window is reached an already started push operation is completed.</p> <p>If no communication windows are defined (array [0]) the push operation is always possible.</p> |
| randomisation_start_interval | <p>To avoid simultaneous push operations by a lot of devices at exactly the same point in time, a randomisation interval – in seconds – can be defined. This means that the push operation is not started immediately after the invocation of the <i>push</i> method but randomly delayed within the interval.</p> <p>The <i>randomisation_start_interval</i> attribute defines the maximum value which can be obtained from a randomizing algorithm. The resulting value is used to delay the first push operation. It is not used anymore in case of retries.</p> <p>If no communication window is active when invoking the <i>push</i> method, the delay starts at the beginning of the next communication window. If the <i>push</i> method is invoked during an active communication window, the push operation is still delayed.</p> <p>The <i>randomisation_start_interval</i> is only active for the first push attempt. If no <i>communication_window</i> is defined, the <i>randomisation_start_interval</i> is active for every push attempt.</p> <p>If the <i>randomisation_start_interval</i> is set to 0 no delay is active.</p> <p>If the randomisation time is longer than the first communication window, the first push attempt will be made during any later window.</p> |
| number_of_retries | <p>Defines the maximum number of retries in the case of unsuccessful or skipped push attempts. After a successful push operation no further push attempts are made until the push operation is triggered again. A push is treated as successful if a lower layer transmission confirmation has been received.</p> <p>The conditions of detecting an unsuccessful push attempt may depend on the communication profile used and on the implementation and it is therefore out of the Scope of this Technical Specification.</p> |
| repetition_delay | <p>The time delay, expressed in seconds until the next push attempt is started after an unsuccessful push.</p> <p>NOTE 4 The repetition delay itself is not influenced by the communication window. But a retry only can be made if a communication window is active at that time. Otherwise it is handled like an unsuccessful push attempt.</p> <p>NOTE 5 The push data is not stored in an intermediate buffer. In the case of retries, the current values of the attributes may change with every push attempt.</p> |

Method description

- push (data)** Activates the push process leading to the elaboration and the sending of the push data taking into account the values of the attributes defined in the given instance of this IC.
- data ::= integer (0)
- The push data returned shall be a structure. The number of elements in this structure shall be the same as number of elements in the array of the *push_object_list* attribute.

Table 16 – 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. |
| 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. |

4.4.9 COSEM data protection

4.4.9.1 Overview

Instances of this IC allow applying cryptographic protection on COSEM data i.e. on attribute values and method invocation and return parameters. This is achieved by accessing attributes and/or methods of other COSEM objects indirectly through instances of the “Data protection” interface class that provide the necessary mechanisms and parameters to apply / verify / remove protection on COSEM data.

NOTE 1 “Accessing” includes reading / writing / capturing / pushing COSEM object attributes or invoking methods.

NOTE 2 When attributes and methods of COSEM objects are accessed directly, protection can be provided by protecting the xDLMS APDUs as stipulated by the relevant security policy and the access rights.

NOTE 3 For definitions and abbreviations related to cryptographic security see DLMS UA 1000-2 Ed. 8.1:201 Clause 3.

Protection on COSEM data is aligned with and complements protection on xDLMS APDUs as defined in DLMS UA 1000-2 Ed. 8.1:201 Clause 9.

The use cases for COSEM data protection include, but are not limited to:

- retrieving a pre-defined set of protected attribute values;
- storing a pre-defined set of protected attribute values in “Profile generic” objects for later retrieval;
- pushing a pre-defined set of protected attribute values;
- reading or writing selected attributes of other COSEM objects with protection;
- invoking a method of another COSEM object with protected method invocation and return parameters.

Protection may comprise any combination of authentication, encryption and digital signature and can be applied in a layered manner. The parties applying and removing the protection are the DLMS/COSEM server and another identified party, which may be a DLMS/COSEM client or a third party.

Applying data protection between a DLMS/COSEM server and a third party allows keeping critical / sensitive data confidential towards the client through which the third party accesses the server. Signing COSEM data by a third party supports non-repudiation.

For end-to-end protection between third parties and servers, see also DLMS UA 1000-2 Ed. 8.1:201 4.7 and 9.2.2.5.

The protection parameters are always controlled by the client with some elements filled in by the server as appropriate.

The security suite is determined by the “Security setup” object referenced from the current “Association SN” / “Association LN” object.

Figure 14 shows the COSEM model of data protection and the relationship of a “Data protection” object with other COSEM objects.

For accessing attributes of other COSEM objects with protected data, there are two mechanisms available:

- reading or writing the *protection_buffer* attribute. The *protection_buffer* can be also captured in “Profile generic” objects or pushed using “Push setup” objects;
- invoking the *get_protected_attributes* / *set_protected_attributes* method.

For accessing a method of another COSEM object with protected data, the *invoke_protected_method* method is available.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 115/492 |
|-----------------------|------------|-------------------------|---------|

APDUs carrying service invocations to access attributes and methods of “Data protection” objects are protected as stipulated by access rights to these attributes and methods, and by “Security setup” *security_suite* and *security_policy*.

The master key and Certificates – as required by the security suite – are held by “Security setup”.

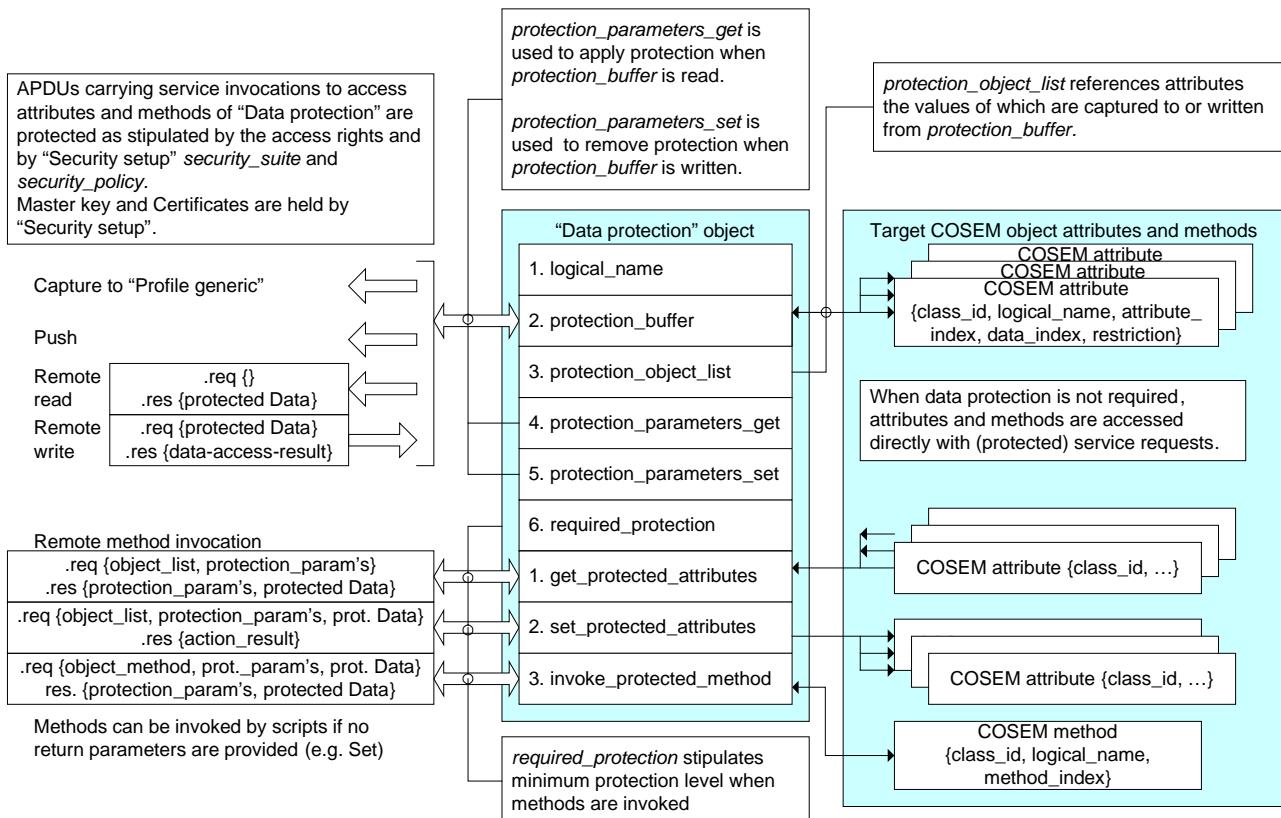


Figure 14 – COSEM model of data protection

Protection on COSEM data is applied and removed in the various cases as follows:

- 1) When the *protection_buffer* attribute is read / captured in a “Profile generic” object / pushed:
 - attributes determined by *protection_object_list* are captured;
 - protection according to *protection_parameters_get* is applied on the set of attributes and the result is put to *protection_buffer*;
 - the value of *protection_buffer* is returned / captured in the “Profile generic” *buffer* / pushed using “Push setup” objects.
- 2) When the *protection_buffer* is written:
 - protected Data are written to *protection_buffer*; and
 - protection according to *protection_parameters_set* is removed and the resulting attribute values are written to the attributes specified by *protection_object_list*.

- 3) When the *get_protected_attributes* method is invoked:
 - attributes determined by the *object_list* element of *get_protected_attributes_request* are captured;
 - protection according to the *required_protection* attribute and response *protection_parameters* is applied;
 - the protected attribute values are returned.
- 4) When the *set_protected_attributes* method is invoked:
 - protection on *protected_attributes* is verified and removed using the *protection_parameters* that must meet *required_protection*;
 - the resulting attribute values are put in the attributes specified by the *object_list* element;
- 5) When the *invoke_protected_method* method is invoked:
 - protection from protected method invocation parameters is removed using the *protection_parameters* in the request that must meet *required_protection*;
 - the method specified by the *object_method* element of *invoke_protected_method_request* is invoked with this method invocation parameter;
 - on the return parameters, the protection using the response *protection_parameters* that must meet *required_protection* is applied;
 - the protected method return parameters are returned.

Figure 15 shows, as an example, how protected Data in *protection_buffer* is constructed from the attributes determined by *protection_object_list* according to the *protection_parameters_get*. See also DLMS UA 1000-2 Ed. 8.1:201 Figure 81.

When the *protection_buffer* attribute is read the following steps are performed:

- 1) prerequisites: *protection_object_list*, *protection_parameters_get*, master key, key agreement and digital signature certificates as needed;
- 2) capture COSEM object attributes determined by *protection_object_list* and create Data, a structure containing the individual Data of the attributes captured;
- 3) protect Data according to *protection_parameters_get*.

NOTE 4 In the example shown in Figure 15 two layers of protection are applied:

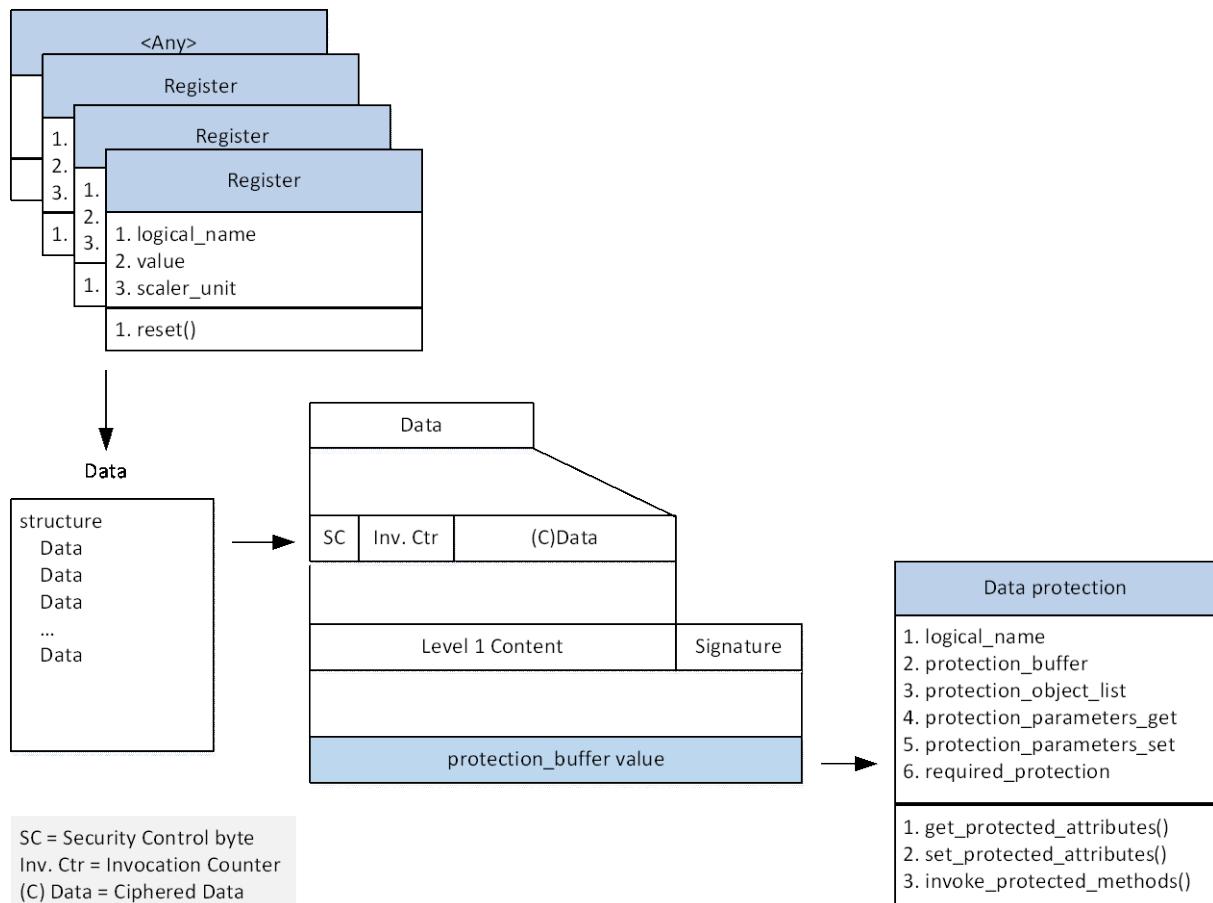
- the first layer is a combination of compression / encryption / authentication as determined by the Security control byte SC, resulting (C)Data,
- the second layer is digital signature applied to (C)Data.

- 4) put the protected data, of data type octet-string, into *protection_buffer*,
- 5) return the value of *protection_buffer*.

It may be necessary to read also *protection_parameters_get* to obtain the protection parameters to verify / remove protection by the recipient.

The invocation counter used when protection is applied / removed is related to the key used. When the protection is applied the corresponding invocation counter is incremented. When the key is changed the invocation counter shall be reset to 0.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 117/492 |
|-----------------------|------------|-------------------------|---------|

Figure 15 – Example: Read *protection_buffer* attribute

4.4.9.2 Data protection (class_id = 30, version = 0)

| 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 | | | | x + 0x30 |
| 2. set_protected_attributes(data) | | m | | | | x + 0x38 |
| 3. invoke_protected_method(data) | | m | | | | x + 0x40 |

Attribute description

| | |
|-------------------------------|--|
| logical_name | Identifies the “Data protection” object instance. See 6.2.31. |
| protection_buffer | <p>Contains the protected Data.</p> <p>When read, the attributes determined by <i>protection_object_list</i> are captured then protection is applied according to <i>protection_parameters_get</i>.</p> <p>When written, the protected Data is put into the <i>protection_buffer</i>, then the protection is verified / removed according to <i>protection_parameters_set</i> and the attributes determined by <i>protection_object_list</i> are set.</p> |
| protection_object_list | <p>Defines the list of attributes to be captured to <i>protection_buffer</i> when it is read or the list of attributes to be set when <i>protection_buffer</i> is written.</p> <p>Two, mutually exclusive selective access mechanisms are available:</p> <ul style="list-style-type: none"> – relative selective access, i.e. entries defined relative to current date or entry are returned: this mechanism is controlled by the <i>data_index</i> element; or – absolute selective access, i.e. entries in an explicitly defined date range or entry range are returned: this mechanism is controlled by the <i>restriction</i> element. |

array object_definition

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

Where:

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

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

protection_object_list
(continued)

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

The encoding is specified in Table 16.

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

restriction_element ::= structure

```
{
    restriction_type: enum:
        (0) none,
        (1) restriction_by_date,
        (2) restriction_by_entry
    restriction_value: CHOICE
    {
        null-data,                      // no restrictions apply
        restriction_by_date,
        restriction_by_entry
    }
}
```

restriction_by_date ::= structure

```
{
    from_date: octet-string,
    to_date: octet-string
}
```

restriction_by_entry ::= structure

```
{
    from_entry: double-long-unsigned,
    to_entry: double-long-unsigned
}
```

- 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;
- restriction_element is composed of restriction_type and restriction_value:
 - for restriction by date range the restriction_type element holds (1) restriction by date and the restriction_value element holds restriction_by_date structure;
 - for restriction by entries the restriction_type element holds (2) restriction by entry and the restriction_value element holds restriction_by_entry structure;
 - 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.

protection_parameters_get Contains all necessary parameters to specify the protection applied when the *protection_buffer* is read.

array protection_parameters_element

```
protection_parameters_element ::= structure
{
    protection_type: enum:
        (0) authentication,
        (1) encryption,
        (2) authentication and encryption,
        (3) digital signature
    protection_options: structure
    {
        transaction_id:          octet-string,
        originator_system_title: octet-string,
        recipient_system_title:  octet-string,
        other_information:       octet-string,
        key_info:                key_info_element
    }
}
```

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.

key_info_element ::= structure

```
{
    key_info_type: enum:
        (0) identified_key,
            -- used with identified_key_info_options
        (1) wrapped_key,
            -- used with wrapped_key_info_options
        (2) agreed_key
            -- used with agreed_key_info_options
}
```

```

protection_
parameters_get          key_info_options: CHOICE
(continued)                {
                           identified_key_info_options,
                           wrapped_key_info_options,
                           agreed_key_info_options
                         }
}
identified_key_info_options ::= enum:
                           (0) global_unicast_encryption_key,
                           (1) global_broadcast_encryption_key

wrapped_key_info_options ::= structure
{
  kek_id:                 enum:
                           (0) master_key,
  key_ciphered_data:      octet-string
}
agreed_key_info_options ::= structure
{
  key_parameters:         octet-string,
  key_ciphered_data:      octet-string
}

```

This attribute is first written by the client. The server may need to fill in some additional elements, in which case this attribute has to be read back by the client – and if needed, forwarded to the third party – so that they can use these parameters to verify / remove protection.

For the use of the various elements see Table 17 and Table 18.

| | |
|-----------------------------------|---|
| protection_ parameters_set | Contains all necessary parameters to verify / remove the protection applied when the <i>protection_buffer</i> is written. |
|-----------------------------------|---|

| |
|-------------------------------------|
| array protection_parameters_element |
|-------------------------------------|

protection_parameters_element: see the *protection_parameters_get* attribute.

This attribute is written by the client and it is used by the server to verify and remove protection.

For the use of the various elements see Table 17 and Table 19.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 122/492 |
|-----------------------|------------|-------------------------|---------|

| | |
|---|---|
| required_protection | Stipulates the required protection on the attribute values / invocation and return parameters of methods accessed through the "Data protection" object. |
| enum: | |
| When the enum value is interpreted as an unsigned, the meaning of each bit is as shown below: | |
| Bit | Required protection |
| 0 | unused, shall be set to 0, |
| 1 | unused, shall be set to 0, |
| 2 | authenticated request, |
| 3 | encrypted request, |
| 4 | digitally signed request, |
| 5 | authenticated response, |
| 6 | encrypted response, |
| 7 | digitally signed response |

Method description

| | |
|--|---|
| get_protected_attributes (data) | Gets the value of the attributes specified by the object_list element, with the protection according to the protection_parameters in get_protected_attributes_response applied. |
|--|---|

Method invocation parameters:

```

data ::= get_protected_attributes_request
get_protected_attributes_request ::= structure
{
    object_list:          array object_definition,
    protection_parameters: array protection_parameters_element,
}

```

Return parameters:

```

data ::= get_protected_attributes_response
get_protected_attributes_response ::= structure
{
    protection_parameters: array protection_parameters_element,
    protected_attributes:   octet-string
}

```

Where:

- object_list: see the *protection_object_list* attribute;
 - protection_parameters ::= array protection_parameters_element.
-

| | |
|--|---|
| get_protected_attributes (data) | protection_parameters_element: see the <i>protection_parameters_get</i> attribute. |
| (continued) | The protection_parameters in get_protected_attributes_response are elaborated by copying the protection_parameters from get_protected_attributes_request except that the server may need to fill in some elements. The remote party uses the protection_parameters in get_protected_attributes_response to verify / remove protection on the protected_attributes returned. |

For the use of the various elements see Table 17 and Table 20

| | |
|--|--|
| set_protected_attributes (data) | Sets the value of the attributes specified by the object_list element, after verifying and removing the protection on the protected_attributes element according to the protection_parameters element. |
|--|--|

Method invocation parameters:

```
data ::= set_protected_attributes_request
set_protected_attributes_request ::= structure
{
    object_list:          array object_definition,
    protection_parameters: array protection_parameters_element,
    protected_attributes: octet-string
}
```

Where:

- object_list: see the *protection_object_list* attribute;
- protection_parameters ::= array protection_parameters_element.

protection_parameters_element: see the *protection_parameters_get* attribute.

For the use of the various elements see Table 17 and Table 21.

| | |
|---------------------------------------|--|
| invoke_protected_method (data) | Invokes the method specified by the object_method element, after verifying and removing the protection on protected_method_invocation_parameters according to the protection_parameters in invoke_protected_method_request. After invoking the method specified by the object_method element, the protection according to the protection_parameters in invoke_protected_method_response – that must meet <i>required_protection</i> – is applied on the return parameters, and invoke_protected_method_response composed of protection_parameters and protected_method_return_parameters is returned. |
|---------------------------------------|--|

invoke_protected_method (data)
(continued)

Method invocation parameters:

```
data ::= invoke_protected_method_request
```

```
invoke_protected_method_request ::= structure
{
    object_method:          object_method_definition,
    protection_parameters: array protection_parameters_element,
    protected_method_invocation_parameters: octet-string
}
object_method_definition ::= structure
{
    class_id:      long-unsigned,
    logical_name: octet-string,
    method_index: integer,
}
```

Return parameters:

```
data ::= invoke_protected_method_response
```

```
invoke_protected_method_response ::= structure
{
    protection_parameters: array protection_parameters_element,
    protected_method_return_parameters: octet-string
}
```

protection_parameters_element: see the specification of the protection_parameters_get attribute.

If the method invoked does not provide return parameters then invoke_protected_method_response shall be a structure, with protection_parameters an array of zero elements and protected_method_return_parameters an octet-string of zero length.

The server uses the protection_parameters in invoke_protected_method_request – that must meet *required_protection* – to verify and remove protection from protected_method_invocation_parameters.

The protection_parameters in invoke_protected_method_response are elaborated by copying the protection_parameters from invoke_protected_method_request to invoke_protected_method_response except that the server may need to fill in some elements.

The protection_parameters in invoke_protected_method_response – that must meet *required_protection* – are applied then to protect data in protected_method_return_parameters.

For the use of the various elements see Table 17 and Table 22.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 125/492 |
|-----------------------|------------|-------------------------|---------|

Table 17 – 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 Qu, 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, $Nonce_U$. See DLMS UA 1000-2 Ed. 8.1:201 9.2.3.4.6.4 and 9.2.3.4.6.5.</p> |
| Legend: M: Mandatory (part of a structure) S: Selectable (part of a CHOICE) | | |

Table 18 – 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 | |
| (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 |
| 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. | | | |
| Legend: M: Mandatory (part of a structure) S: Selectable (part of a CHOICE) | | | |

Table 19 – Protection parameters of *protection_parameters_set* 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 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 | |

Legend:

M: Mandatory (part of a structure)

S: Selectable (part of a CHOICE)

Table 20 – 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.

Legend:

M: Mandatory (part of a structure)

S: Selectable (part of a CHOICE)

Table 21 – 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. | | |
| Legend: M: Mandatory (part of a structure) S: Selectable (part of a CHOICE) | | |

Table 22 – 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.

Legend:

M: Mandatory (part of a structure)

S: Selectable (part of a CHOICE)

4.5 Interface classes for time- and event bound control

4.5.1 Clock (class_id = 8, version = 0)

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

The *date* information includes the elements year, month, day of month and day of week. The *time* information includes the elements hour, minutes, seconds, hundredths of seconds, and the deviation of the local time from UTC. The daylight saving time function modifies the deviation of local time to UTC depending on the attributes; Figure 16. The start and end point of that function is normally set once. An internal algorithm calculates the real switch point depending on these settings.

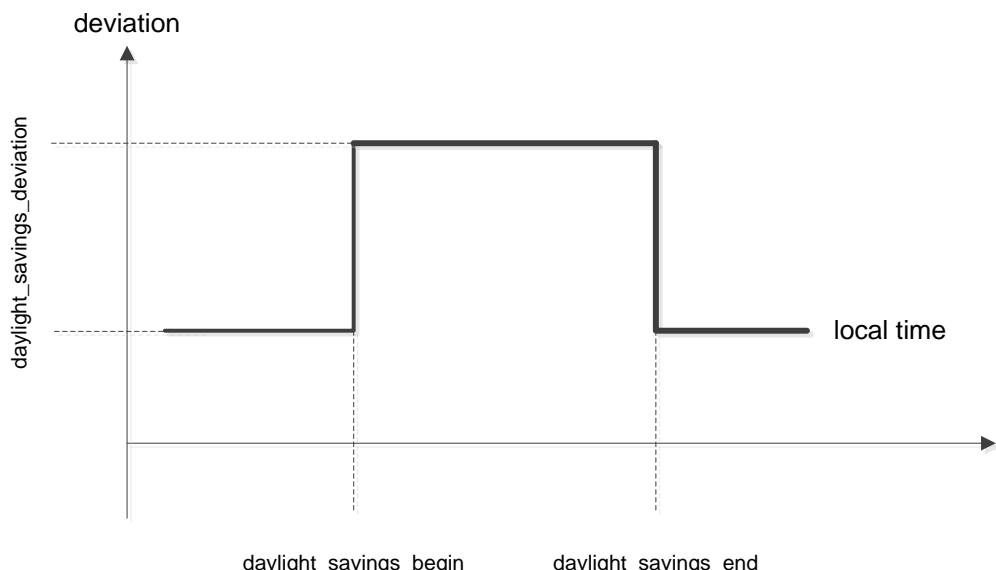


Figure 16 – The generalized time concept

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

Attribute description

| | |
|-------------------------------|---|
| logical_name | Identifies the “Clock” object instance. See 6.2.5. |
| time | <p>Contains the meter’s local date and time, its deviation to UTC and the status.</p> <p>octet-string, formatted as specified in 4.1.6.1 for <i>date-time</i>.</p> <p>When this attribute is set, all the fields in the octet-string representing <i>date-time</i> shall be evaluated and the local date and time in the meter shall be set according to the rules defined in 4.1.6.1.</p> <p>Only specified fields of the <i>date-time</i> are changed.</p> <p>EXAMPLE For setting the <i>date</i> without changing the <i>time</i>, all time-relevant octets in the octet string representing <i>date-time</i> shall be set to “not specified”.</p> |
| time_zone | The deviation of local, normal time to UTC in minutes. The value depends on the geographical location of the meter. |
| status | <p>The <i>clock_status</i> maintained by the meter. The <i>clock_status</i> element indicated in the <i>time</i> attribute is equal to the value of this attribute.</p> <p>unsigned, formatted as specified in 4.1.6.1 for <i>clock_status</i></p> |
| daylight_savings_begin | <p>Defines the local switch date and time when the local time starts to deviate from the normal time.</p> <p>For generic definitions, wildcards are allowed.</p> <p>octet-string, formatted as specified in 4.1.6.1 for <i>date-time</i>.</p> |
| daylight_savings_end | <p>Defines the local switch date and time when the local time ends to deviate from the local normal time.</p> <p>octet-string, formatted as specified in 4.1.6.1 for <i>date-time</i>.</p> |

| | |
|-----------------------------------|--|
| daylight_savings_deviation | Contains the number of minutes by which the deviation in generalized time shall be corrected at daylight savings begin. integer: Deviation in the range of ± 120 min |
| daylight_savings_enabled | boolean: TRUE = DST enabled, FALSE = DST disabled NOTE This is a parameter to enable and disable the daylight savings time feature. The current status is indicated in the status attribute. |
| clock_base | Defines where the basic timing information comes from. enum: (0) not defined, (1) internal crystal, (2) mains frequency 50 Hz, (3) mains frequency 60 Hz, (4) GPS (global positioning system), (5) radio controlled |

Method description

| | |
|--|--|
| adjust_to_quarter (data) | Sets the meter's time to the nearest (+/-) quarter of an hour value (*:00, *:15, *:30, *:45). data ::= integer (0) |
| adjust_to_measuring_period (data) | Sets the meter's time to the nearest (+/-) starting point of a measuring period. data ::= integer (0) |
| adjust_to_minute (data) | Sets the meter's time to the nearest minute. If second_counter < 30 s, second_counter is set to 0. If second_counter \geq 30 s, second_counter is set to 0, and minute_counter and all depending clock values are incremented if necessary. data ::= integer (0) |
| adjust_to_preset_time (data) | This method is used in conjunction with the preset_adjusting_time method. If the meter's time lies between validity_interval_start and validity_interval_end, then time is set to preset_time. data ::= integer (0) |
| preset_adjusting_time (data) | Presets the time to a new value (preset_time) and defines a validity_interval within which the new time can be activated. data ::= structure { preset_time: octet-string, validity_interval_start: octet-string, validity_interval_end: octet-string } all octet-strings formatted as specified in 4.1.6.1 for <i>date-time</i> . |
| shift_time (data) | Shifts the time by n (-900 \leq n \leq 900) s. data ::= long |

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

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

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

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

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

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

Attribute description

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

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

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

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

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

Where:

- the service_id element defines which action to be applied to the referenced object:
 - (1) write attribute,
 - (2) execute specific method
- the index element defines (with service_id 1) which attribute of the selected

object is affected; or (with service_id 2) which specific method is to be executed. The first attribute (logical_name) has index 1, the first specific method has index 1 as well.

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

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

Method description

| | |
|---------------------------|---|
| execute (data) | Executes the script specified in parameter data. data ::= long-unsigned If data matches one of the script_identifiers in the script table, then the corresponding action_specification is executed. |
|---------------------------|---|

4.5.3 Schedule (class_id = 10, version = 0)

This IC, together with the IC "Special days", allows modelling time- and date-driven activities within a device. Table 23 provides an overview and show the interactions between the two ICs.

Table 23 – Schedule

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

Table 24 – Special days table

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

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

Attribute description

| | |
|---------------------|---|
| logical_name | Identifies the "Schedule" object instance. See 6.2.9. |
| entries | Specifies the scripts to be executed at given times. There is only one script that can be executed per entry. array schedule_table_entry |
| | <pre>schedule_table_entry ::= structure { index: long-unsigned, enable: boolean, script_logical_name: octet-string, script_selector: long-unsigned, switch_time: octet-string, validity_window: long-unsigned, exec_weekdays: bit-string, exec_specdays: bit-string, begin_date: octet-string, end_date: octet-string }</pre> |

Where:

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

Method description

| | |
|----------------------------------|--|
| enable/disable (data) | Sets the disabled bit of range A entries to true and then enables the entries of range B. data ::= structure { firstIndexA: long-unsigned, lastIndexA: long-unsigned, firstIndexB: long-unsigned, lastIndexB: long-unsigned } |
| firstIndexA | first index of the range that is disabled, |
| lastIndexA | last index of the range that is disabled, |
| firstIndexB | first index of the range that is enabled, |
| lastIndexB | last index of the range that is enabled, |

| | | |
|--------------------------|---|--|
| | firstIndexA/B < lastIndexA/B: | all entries of the range A/B are disabled / enabled, |
| | firstIndexA/B = lastIndexA/B: | one entry is disabled/enabled, |
| | firstIndexA/B > lastIndexA/B: | nothing disabled/enabled, |
| | firstIndexA/B and lastIndexA/B > 9999: | no entry is disabled/enabled |
| insert (data) | Inserts a new entry in the table. If the index of the entry exists already, the existing entry is overwritten by the new entry. entry:schedule_table_entry data ::= corresponding to entry | |
| delete (data) | Deletes a range of entries in the table. data ::= structure { firstIndex: long-unsigned, lastIndex: long-unsigned } firstIndex: first index of the range that is deleted, lastIndex: last index of the range that is deleted, firstIndex < lastIndex: all entries of the range A/B are deleted, firstIndex = lastIndex: one entry is deleted, firstIndex > lastIndex: nothing deleted | |

Remarks concerning "inconsistencies" in the table entries:

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

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

Recovery after power failure

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

Handling of time changes

There are four different "actions" of time changes:

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

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

Time setting forward

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

Time setting backward

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

Time synchronization

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

Daylight saving

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

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

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

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

Attribute description

logical_name Identifies the "Special days table" object instance. See 6.2.8.

entries Specifies a special day identifier for a given date. The date may have wildcards for repeating special days like Christmas.

```
array      spec_day_entry
spec_day_entry ::= structure
{
    index:          long-unsigned,
    specialday_date: octet-string,
    day_id:         unsigned
}
```

Where:

- specialday_date formatting follows the rules set in 4.1.6.1 for *date*;
- the range of the day_id shall match the length of the bit-string

 exec_specdays in the related object of IC "Schedule".

Method description

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

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

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

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

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

The "Activity calendar" object defines the activation of certain scripts, which can perform different activities inside the logical device. The interface to the IC "Script table" is the same as for the IC "Schedule" (see 4.5.3).

If an instance of the IC "Special days table" (see 4.5.4) is available, relevant entries there take precedence over the "Activity calendar" object driven selection of a day profile. The day profile referenced in the "Special days table" activates the day_schedule of the day_profile_table in the "Activity calendar" object by referencing through the day_id.

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

Attribute description

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

| | | | | | |
|--|---|-------|--------------|--|--|
| logical_name | Identifies the “Activity calendar” object instance. See 6.2.10. | | | | |
| calendar_name | Typically contains an identifier, which is descriptive to the set of scripts activated by the object. | | | | |
| season_profile | <p>Contains a list of seasons defined by their starting date and a specific week_profile to be executed. The list is sorted according to season_start (in increasing order);</p> <table> <tr> <td>array</td> <td>season</td> </tr> <tr> <td colspan="2"> <pre>season ::= structure { season_profile_name: octet-string, season_start: octet-string, week_name: octet-string }</pre> </td> </tr> </table> <p>Where:</p> <ul style="list-style-type: none"> - season_profile_name is a user defined name identifying the current season_profile; - season_start defines the starting time of the season, formatted as specified in 4.1.6.1 for <i>date-time</i>. In season_start, wildcards are allowed. If all fields are wildcards, the season will never start. When using wildcards, special care has to be taken to avoid conflicting parametrization, i.e. that the season_start of two different seasons is the same; <p style="margin-left: 20px;">NOTE The current season is terminated by the season_start of the next season.</p> <ul style="list-style-type: none"> - week_name defines the week_profile active in this season. | array | season | <pre>season ::= structure { season_profile_name: octet-string, season_start: octet-string, week_name: octet-string }</pre> | |
| array | season | | | | |
| <pre>season ::= structure { season_profile_name: octet-string, season_start: octet-string, week_name: octet-string }</pre> | | | | | |
| week_profile_table | <p>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.</p> <table> <tr> <td>array</td> <td>week_profile</td> </tr> <tr> <td colspan="2"> <pre>week_profile ::= structure { week_profile_name: octet-string, monday: day_id, tuesday: day_id, wednesday: day_id, thursday: day_id, friday: day_id, saturday: day_id, sunday: day_id } day_id: unsigned</pre> </td> </tr> </table> | array | week_profile | <pre>week_profile ::= structure { week_profile_name: octet-string, monday: day_id, tuesday: day_id, wednesday: day_id, thursday: day_id, friday: day_id, saturday: day_id, sunday: day_id } day_id: unsigned</pre> | |
| array | week_profile | | | | |
| <pre>week_profile ::= structure { week_profile_name: octet-string, monday: day_id, tuesday: day_id, wednesday: day_id, thursday: day_id, friday: day_id, saturday: day_id, sunday: day_id } day_id: unsigned</pre> | | | | | |

| | |
|--|--|
| week_profile_table (continued) | <p>Where:</p> <ul style="list-style-type: none"> - week_profile_name is a user defined name identifying the current week_profile; - Monday defines the day_profile valid on Monday; - ... - Sunday defines the day_profile valid on Sunday. |
| day_profile_table | <p>Contains an array of day_profiles, identified by their day_id. For each day_profile, a list of scheduled actions is defined by a script to be executed and the corresponding activation time (start_time). The list is sorted according to start_time.</p> <pre>array day_profile</pre> <pre>day_profile ::= structure { day_id: unsigned, day_schedule: array day_profile_action } day_profile_action ::= structure { start_time: octet-string, script_logical_name: octet-string, script_selector: long-unsigned }</pre> |
| activate_passive_calendar_time | <p>Where:</p> <ul style="list-style-type: none"> - day_id is a user defined identifier, identifying the current day_profile; - start_time: defines the time when the script is to be executed (no wildcards); the format follows the rules set in 4.1.6.1 for <i>time</i>; - script_logical_name: defines the <i>logical_name</i> of the "Script table" object; - script_selector: defines the script_identifier of the script to be executed. <p>Defines the time when the object itself calls the specific method <i>activate_passive_calendar</i>. A definition with "not specified" notation in all fields of the attribute will deactivate this automatism. Partial "not specified" notation in just some fields of date and time are not allowed.</p> <p>octet-string, formatted as specified in 4.1.6.1 for <i>date-time</i>.</p> |

Method description

| | |
|--|---|
| activate_passive_calendar(data) | This method copies all attributes called ..._passive to the corresponding attributes called ..._active. |
|--|---|

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

This IC allows modelling the function of monitoring of values modelled by "Data", "Register", "Extended register" or "Demand register" objects. It allows specifying thresholds, the value monitored, and a set of scripts (see 4.5.2) that are executed when the value monitored crosses a threshold.

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

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 142/492 |
|-----------------------|------------|-------------------------|---------|

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

Attribute description

| | |
|------------------------|--|
| logical_name | Identifies the “Register monitor” object instance. See 6.2.13 and 6.3.10. |
| thresholds | <p>Provides the threshold values to which the attribute of the referenced register is compared.</p> <p>array threshold</p> <p>threshold: The threshold is of the same type as the monitored attribute of the referenced object.</p> |
| monitored_value | <p>Defines which attribute of an object is to be monitored. Only values with simple data types are allowed.</p> <pre>value_definition ::= structure { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer }</pre> |
| actions | <p>Defines the scripts to be executed when the monitored attribute of the referenced object crosses the corresponding threshold. The attribute <i>actions</i> has exactly the same number of elements as the attribute <i>thresholds</i>. The ordering of the action_items corresponds to the ordering of the thresholds (see above).</p> <p>array action_set</p> <pre>action_set ::= structure { action_up: action_item, action_down: action_item }</pre> <p>Where:</p> <ul style="list-style-type: none"> - action_up defines the action when the attribute value of the monitored register crosses the threshold in the upwards direction; - action_down defines the action when the attribute value of the monitored register crosses the threshold in the downwards direction. <pre>action_item ::= structure { script_logical_name: octet-string, script_selector: long-unsigned }</pre> |

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

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

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

Attribute description

| | |
|------------------------|--|
| logical_name | Identifies the “Single action schedule” object instance. See 6.2.12. |
| executed_script | Contains the logical name of the “Script table” object and the script selector of the script to be executed. script ::= structure { script_logical_name: octet-string, script_selector: long-unsigned } Script_logical_name and script_selector define the script to be executed. |
| type | enum: (1) size of execution_time = 1; wildcard in <i>date</i> allowed, (2) size of execution_time = <i>n</i> ; all <i>time</i> values are the same, wildcards in <i>date</i> not allowed, (3) size of execution_time = <i>n</i> ; all <i>time</i> values are the same, wildcards in <i>date</i> are allowed, (4) size of execution_time = <i>n</i> ; <i>time</i> values may be different, wildcards in <i>date</i> not allowed, (5) size of execution_time = <i>n</i> ; <i>time</i> values may be different, wildcards in <i>date</i> are allowed |
| execution_time | Specifies the time and the date when the script is executed. array execution_time_date execution_time_date ::= structure { time: octet-string, date: octet-string } The two octet-strings contain <i>time</i> and <i>date</i> , in this order; <i>time</i> and <i>date</i> are formatted as specified in 4.1.6.1. Hundredths of seconds shall be zero. |

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

Instances of the “Disconnect control” IC manage an internal or external disconnect unit of the meter (e.g. electricity breaker, gas valve) in order to connect or disconnect – partly or entirely – the premises of the consumer to / from the supply. The state diagram and the possible state transitions are shown in Figure 17.

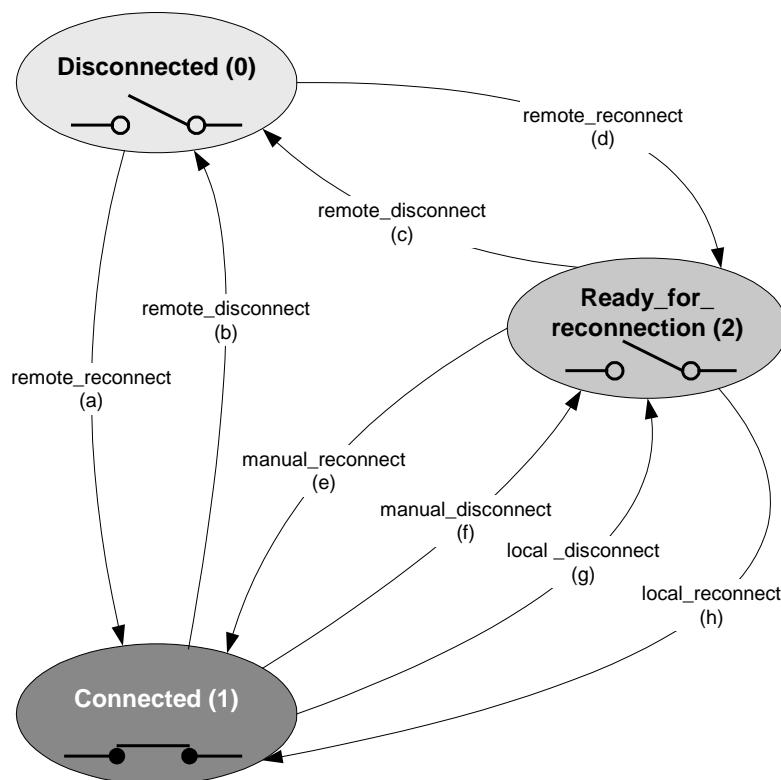


Figure 17 – State diagram of the Disconnect control IC

Disconnect and reconnect can be requested:

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

The states and state transitions of the Disconnect control IC are shown in Table 25. The possible state transitions depend on the control mode. The Disconnect control object doesn't feature a memory, i.e. any commands are executed immediately.

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

Table 25 – 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. |

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

Attribute description

| logical_name | Identifies the “Disconnect control” object instance. See 6.2.21 and 6.2.39. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|----------------------|--|--------|-------|--------------|--------|-------|-----|-----|--------------|---------------|--|--|--------------|--|--|--------|--------|-------|--------|--------|-------|-------|-----|-----|-----|-----|-----|-----|-----|-----|-----|---|---|---|---|---|---|---|---|-----|---|---|---|---|---|---|---|---|-----|---|---|---|---|---|---|---|---|-----|---|---|---|---|---|---|---|---|-----|---|---|---|---|---|---|---|---|-----|---|---|---|---|---|---|---|---|-----|---|---|---|---|---|---|---|---|
| output_state | Shows the actual physical state of the device connection the supply. boolean: TRUE = (BOOLEAN 1) Connected, FALSE = (BOOLEAN 0) Disconnected | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | NOTE 1 In electricity metering, the supply is connected when the disconnector device is closed. NOTE 2 In gas and water metering, the supply is connected when the valve is open. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| control_state | Shows the internal state of the disconnect control object. enum: (0) Disconnected, (1) Connected, (2) Ready_for_reconnection | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| control_mode | Configures the behaviour of the disconnect control object for all triggers, i.e. the possible state transitions. <table border="1" style="width: 100%; border-collapse: collapse;"><thead><tr><th rowspan="2">control_mode</th><th colspan="3">Disconnection</th><th colspan="3">Reconnection</th></tr><tr><th>Remote</th><th>Manual</th><th>Local</th><th>Remote</th><th>Manual</th><th>Local</th></tr></thead><tbody><tr><td>enum:</td><td>(b)</td><td>(c)</td><td>(f)</td><td>(g)</td><td>(a)</td><td>(d)</td><td>(e)</td><td>(h)</td></tr><tr><td>(0)</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td><td>—</td></tr><tr><td>(1)</td><td>x</td><td>x</td><td>x</td><td>x</td><td>—</td><td>x</td><td>x</td><td>—</td></tr><tr><td>(2)</td><td>x</td><td>x</td><td>x</td><td>x</td><td>x</td><td>—</td><td>x</td><td>—</td></tr><tr><td>(3)</td><td>x</td><td>x</td><td>—</td><td>x</td><td>—</td><td>x</td><td>x</td><td>—</td></tr><tr><td>(4)</td><td>x</td><td>x</td><td>—</td><td>x</td><td>x</td><td>—</td><td>x</td><td>—</td></tr><tr><td>(5)</td><td>x</td><td>x</td><td>x</td><td>x</td><td>—</td><td>x</td><td>x</td><td>x</td></tr><tr><td>(6)</td><td>x</td><td>x</td><td>—</td><td>x</td><td>—</td><td>x</td><td>x</td><td>x</td></tr></tbody></table> 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. | | | | | | | | control_mode | Disconnection | | | Reconnection | | | Remote | Manual | Local | Remote | Manual | Local | enum: | (b) | (c) | (f) | (g) | (a) | (d) | (e) | (h) | (0) | — | — | — | — | — | — | — | — | (1) | x | x | x | x | — | x | x | — | (2) | x | x | x | x | x | — | x | — | (3) | x | x | — | x | — | x | x | — | (4) | x | x | — | x | x | — | x | — | (5) | x | x | x | x | — | x | x | x | (6) | x | x | — | x | — | x | x | x |
| control_mode | Disconnection | | | Reconnection | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | Remote | Manual | Local | Remote | Manual | Local | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| enum: | (b) | (c) | (f) | (g) | (a) | (d) | (e) | (h) | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| (0) | — | — | — | — | — | — | — | — | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| (1) | x | x | x | x | — | x | x | — | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| (2) | x | x | x | x | x | — | x | — | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| (3) | x | x | — | x | — | x | x | — | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| (4) | x | x | — | x | x | — | x | — | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| (5) | x | x | x | x | — | x | x | x | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| (6) | x | x | — | x | — | x | x | x | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

Method description

| | |
|---------------------------------|--|
| remote_disconnect (data) | Forces the “Disconnect control” object into ‘disconnected’ state if remote disconnection is enabled (control mode > 0). data ::= integer (0) |
| remote_reconnect (data) | Forces the “Disconnect control” object into the ‘ready_for_reconnection’ state if a direct remote reconnection is disabled (control_mode = 1, 3, 5, 6). Forces the “Disconnect control” object into the ‘connected’ state if a direct remote reconnection is enabled (control_mode = 2, 4). data ::= integer (0) |

4.5.9 Limiter (class_id = 71, version = 0)

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

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

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

Attribute description

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

| | |
|--|---|
| emergency_profile | An <i>emergency_profile</i> is defined by three elements: <i>emergency_profile_id</i> , <i>emergency_activation_time</i> , <i>emergency_duration</i> . An emergency profile is activated if the <i>emergency_profile_id</i> element matches one of the elements on the <i>emergency_profile_group_id_list</i> , and time matches the <i>emergency_activation_time</i> and <i>emergency_duration</i> element: <i>emergency_profile ::= structure</i> { <i>emergency_profile_id:</i> long-unsigned, <i>emergency_activation_time:</i> octet-string, <i>emergency_duration:</i> double-long-unsigned } Where: - the <i>emergency_activation_time</i> element defines the date and time when the <i>emergency_profile</i> activated. The octet-string is encoded as specified in 4.1.6.1 for <i>date-time</i> , - the <i>emergency_duration</i> element defines the duration in seconds, for which the <i>emergency_profile</i> is activated. When an emergency profile is active, the <i>emergency_profile_active</i> attribute is set to TRUE. |
| emergency_profile_group_id_list | Defines a list of group id-s of the emergency profile. The emergency profile can be activated only if <i>emergency_profile_id</i> element of the <i>emergency_profile</i> attribute matches one of the elements on the <i>emergency_profile_group_id_list</i> : array <i>emergency_profile_group_id</i> <i>emergency_profile_group_id ::=</i> long-unsigned |
| emergency_profile_active | Indicates that the <i>emergency_profile</i> is active. |
| actions | Defines the scripts to be executed when the monitored value crosses the threshold for minimal duration time. <i>action ::= structure</i> { <i>action_over_threshold:</i> action_item, <i>action_under_threshold:</i> action_item } Where: - <i>action_over_threshold</i> defines the action when the value of the attribute monitored crosses the threshold in upwards direction and remains over threshold for minimal over threshold duration time; - <i>action_under_threshold</i> defines the action when the value of the attribute monitored crosses the threshold in the downwards direction and remains under threshold for minimal under threshold duration time. <i>action_item ::= structure</i> { <i>script_logical_name:</i> octet-string, <i>script_selector:</i> long-unsigned } |

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

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

The parameters can be changed as usual. If the value of an attribute changes and this attribute is present in the *parameter_list* attribute, the identifier and the value of that attribute is automatically captured to the *changed_parameter* attribute. The time when the change of the parameter occurred is captured in the *capture_time* attribute. These attributes may be captured then by a “Profile generic” object. In this way, a log of all parameter changes can be built. For the OBIS code of the Parameter monitor log objects, see 7.4.5.

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

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

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

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

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

Attribute description

| | |
|--------------------------|--|
| logical_name | Identifies the “Parameter monitor” object instance. See 6.2.14. |
| changed_parameter | Holds the identifier and the value of the most recently changed parameter. structure { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer, attribute_value: CHOICE -- CHOICE as specified in the “Data” interface class } |
| capture_time | Provides data and time information showing when the value of the <i>changed_parameter</i> attribute has been captured. <i>date-time</i> is formatted as specified in 4.1.6.1. |

parameter_list Holds the list of parameters managed by a given instance of the “Parameter monitor” IC.

```
parameter_list ::= array parameter_list_element
```

parameter_list_element ::= structure
{
 class_id: long-unsigned,
 logical_name: octet_string,
 attribute_index: integer
}

NOTE 4 The list of parameters monitored may be changed by using the *add_parameter* or *delete_parameter* methods or writing this attribute.

Method description

| | |
|------------------------------------|--|
| add_parameter (data) | Adds one parameter to the <i>parameter_list</i> . data ::= parameter_list_element NOTE 5 A parameter can be logged as soon as it is added to the list. Adding an element to the list does not affect the <i>buffer</i> of the “Profile generic” object capturing the <i>changed_parameter</i> attribute. |
| delete_parameter (data) | Deletes one parameter from the <i>parameter_list</i> . data ::= parameter_list_element NOTE 6 When a parameter is deleted from the parameter list, its changes will not be logged any more. Removing an element from the list does not affect the <i>buffer</i> of the “Profile generic” object capturing the <i>changed_parameter</i> attribute. |

4.5.11 Sensor manager interface class

4.5.11.1 General

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

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

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

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

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

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

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

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

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

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

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

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

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

Attribute description

logical_name Identifies the “Sensor manager” object instance. See 6.4.4.

serial_number Identifies the sensor (->site information).

NOTE 3 For simple sensors, the serial number may be held by “Data” objects with appropriate OBIS codes if this is the only information required. See Gas related general purpose objects, Configuration objects in Table 96.

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

| | |
|-----------------------------|---|
| output_type | describes the physical output of the sensor (->site information) |
| | enum: (0) not specified, (1) 4–20 mA, (2) 0–20 mA (3) 0–5 V, (4) 0–10 V, (5) Pt100, (6) Pt500, (7) Pt1000, (8) HART 128 manufacturer specific All other values are reserved. |
| adjustment_method | Describes the sensor adjustment method, e.g. by 3-Measuring-point-equation. |
| sealing_method | Type of seals applied to the sensor. |
| | enum: (0) none, (1) mechanical (e.g. wire or protective sticker); (2) electronic (e.g. contact); (3) software (e.g. password protection) |
| raw_value | Physical value from the sensor (e.g. voltage from a pressure to voltage converter). For the possible data type choices, see the “Register” IC in 4.3.2. |
| scaler_unit | The scaler and the unit of the <i>raw_value</i> . For the definition of <i>scaler_unit</i> , see the “Register” IC in 4.3.2. |
| status | Status of last <i>raw_value</i> captured (for the definition of status, see the “Extended register” IC. The status keeps information like: - sensor failure, - sensor activated / inactivated. The meaning of the elements of the <i>status</i> shall be provided for each “Sensor manager” object instance. |
| capture_time | Provides a “Sensor manager” object specific date and time information showing when the value of the attribute <i>raw_value</i> has been captured. <i>date_time</i> is formatted as specified in 4.1.6.1. For the possible data type choices, see the “Extended register” interface class in 4.3.3. |
| raw_value_thresholds | Provides the threshold values to which the <i>raw_value</i> attribute is compared. array threshold threshold: The threshold is of the same type as the raw-data. |
| raw_value_actions | Defines the scripts to be executed when the <i>raw_value</i> crosses the corresponding threshold. The attribute <i>raw_value_actions</i> has exactly the same number of elements as the attribute <i>raw_value_thresholds</i> . The ordering of the <i>action_items</i> corresponds to the ordering of the thresholds. For the specification of actions, see the <i>Register monitor</i> IC in 4.5.6. |

| | |
|-----------------------------------|---|
| processed_value | References the attribute holding the processed value of the raw data provided by the sensor. processed_value_definition ::= structure |
| | { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer } |
| | Note that more than one "Sensor manager" objects and processed value objects may belong to the same sensor. |
| processed_value_thresholds | Provides the threshold values to which the processed value is compared. array threshold threshold: The threshold is of the same type as the value processed. (referenced by the processed-value attribute). |

| | |
|--------------------------------|--|
| processed_value_actions | Defines the scripts to be executed when the processed value crosses the corresponding threshold. The attribute <i>processed_value_actions</i> has exactly the same number of elements as the attribute <i>processed_value_thresholds</i> . The ordering of the <i>action_items</i> corresponds to the ordering of the thresholds. For the specification of actions, see the <i>Register monitor</i> IC in 4.5.6. |
|--------------------------------|--|

Method description

| | |
|---------------------|---|
| reset (data) | This method resets the <i>raw_value</i> to the default value. The default value is an instance specific constant. The attribute <i>capture_time</i> is set to the time of the reset execution. data ::= integer (0) |
|---------------------|---|

4.5.11.3 Example for absolute pressure sensor

Figure 18 illustrates the definition of relevant upper and lower thresholds.

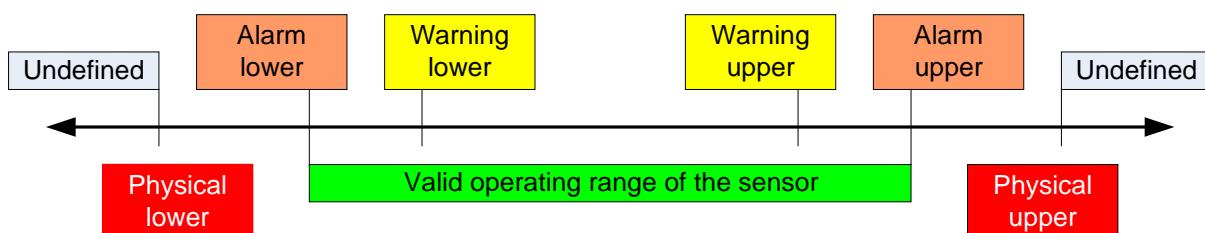


Figure 18 – Definition of upper and lower thresholds

Table 26 and Table 27 show examples of the various thresholds and the actions performed when the thresholds are crossed.

Table 26 – Explicit presentation of threshold value arrays

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

Table 27 – 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 |

4.5.12 Arbitrator**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;

NOTE 3 Names of the actors may be specified in project specific companion specifications.

- the list of actions requested, in the form of a bit-string. Each bit corresponds to one element of the *actions* array: for the actions requested the bit is set to 1, for the actions not requested (inactions) the bit is set to 0. An actor may request none, one, several or all actions in a single request in one invocation. The reason to allow requesting multiple actions in a single request is to allow the actor to request an action, and at the same time to prevent another actor reversing that action.

NOTE 4 For example, an actor may request disconnecting the supply and preventing another actor to reconnect it.

NOTE 5 An earlier action request by an actor can be cleared by not requesting the same action (i.e. by requesting an inaction) in another invocation of the *request_action* method by the same actor. With this, a request for inhibiting an action is lifted.

The *most_recent_requests_table* attribute holds the list of the most recent request of each actor, in the form of an array of bit-strings: each element in the array represents the last request of an actor, and each bit in the bit-string represents one action / inaction requested.

When the *request_action* method is invoked by an actor the AP carries out the following activities:

- it checks the *permissions_table* attribute entry for the given actor to see if the actions requested are permitted or not;
- it updates the *most_recent_requests_table* attribute by setting or clearing the bit in the bit-string for that actor for each action requested that is also permitted (bit is set); or not requested / not permitted (bit is cleared);
- it applies the *weightings_table* for the *most_recent_requests_table*: for each bit set in the *most_recent_requests_table* the corresponding weight of each actor is applied;
- for each action, the weights are summed; and then
- if there is a unique highest total weight for an action, this value is written to the *last_outcome* attribute and the corresponding script is executed. If there is no highest unique total weight, nothing happens.

4.5.12.2 Arbitrator (class_id = 68, version = 0)

| Arbitrator | 0...n | class_id = 68, version = 0 | | | |
|--------------------------------------|--------------|----------------------------|------|------------------|------------|
| Attributes | 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 () | m | | | | |
| 2. reset () | o | | | | |

Attribute description

logical_name Identifies the “Arbitrator” object instance. See 6.2.40.

actions Defines the actions that can be requested.

actions ::= array action_item

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

Where:

- script_logical_name: defines the *logical_name* of the “Script table” object;
- script_selector: defines the script_identifier of the script to be executed.

Entries that are intended to inhibit other actions are represented by

| | |
|-----------------------------------|---|
| | null-scripts, i.e. pointing to script_identifier (0) of a “Script table” object. |
| permissions_table | Contains the permissions for each actor to request actions. permissions_table ::= array actor_permissions actor_permissions ::= bit-string Each entry represents the permissions for one actor to request each action. Each bit in the bit-string corresponds to one action in the <i>actions</i> array. The length of the bit-string shall be the same as the number of elements in the <i>actions</i> array. The leading bit corresponds to the first element and the trailing bit corresponds to the last element in the <i>actions</i> array. The bits shall be set for actions allowed and cleared for actions not allowed. NOTE 1 These <i>actor_permissions</i> model business rules rather than acting as a form of access control. |
| weightings_table | Holds the weight allocated for each actor and to each possible action of that actor. weightings_table ::= array actor_weighting_list actor_weighting_list ::= array actor_action_weight actor_action_weight ::= long-unsigned The number and order of elements in the <i>weightings_table</i> array shall be the same as in the <i>permissions_table</i> array. The number of elements in the <i>actor_weighting_list</i> array shall be the same as in the <i>actions</i> array. The first element shows the weight for the first action and the last element shows the weight of the last action. NOTE 2 It is preferred using powers of 2 as weights. Using different weights for different actors and actions helps to avoid situations when there is no unique outcome after evaluating the action request (in which case no action is performed). |
| most_recent_requests_table | Holds the most recent requests of each actor. most_recent_requests_table ::= array most_recent_request most_recent_request ::= bit-string Each entry represents the most recent request of an actor. The number and order of elements in the <i>most_recent_requests_table</i> array shall be the same as in the <i>permissions_table</i> array. The length of the <i>most_recent_request</i> bit-string shall be the same as the number of elements in the <i>actions</i> array. The leading bit |

corresponds to the first element and the trailing bit corresponds to the last element in the *actions* array.

For each action that has been requested and which is also permitted the bit is set. For each action that is not requested (inaction) or not allowed the bit is cleared.

| | |
|---------------------|--|
| last_outcome | Contains the outcome of the most recent request that has resulted a unique highest total weight for an action requested and therefore performed. The number identifies a bit in the <i>request_action_list</i> bit-string: the number 1 corresponds to the leading bit and consequently to the first element in the <i>actions</i> array. |
|---------------------|--|

Method description

| | |
|----------------------------------|---|
| request_action (data) | Defines the actions that are requested by an actor. |
|----------------------------------|---|

```
data ::= structure
{
    request_actor:          unsigned,
    request_action_list:    bit-string
}
```

Where:

- *request_actor* is an index into the corresponding arrays. The number 1 identifies the first entry in the *permissions_table*, the first entry of the *weightings_table* and the first entry in the *most_recent_requests_table* arrays. The number *n* identifies the highest entry in these arrays;
- *request_action_list* identifies the action(s) to be performed. For each action requested the bit is set. For each action not requested (inaction) the bit is not set. The length of the bit-string shall be the same as the number of elements in the *actions* array. The leading bit corresponds to the first element and the trailing bit corresponds to the last element.

Although several actions can be requested, only one action (modelled by a script) will be actually performed, provided that it is permitted for that actor and there is a single highest total outcome. As specified above, an action may be a null-script.

| | |
|---------------------|---|
| reset (data) | Clears the configurable fields of the object instance, that is: <ul style="list-style-type: none"> – clears all bits in the <i>permissions_table</i> attribute; – sets all values in the <i>weightings_table</i> attribute to zero; – clears all bits in the <i>most_recent_requests_table</i> attribute; – sets the <i>last_outcome</i> attribute to zero. |
|---------------------|---|

NOTE 3 Following a reset it can be expected that every invocation of *request_action* will leave the *last_outcome* attribute equal to zero, and no action will be performed since the elements in the *permissions_table* attribute are all cleared.

```
data ::= integer (0)
```

4.5.13 Modelling examples: tariffication and billing

Figure 19 shows an example of modelling tariff parametrization and management using COSEM objects.

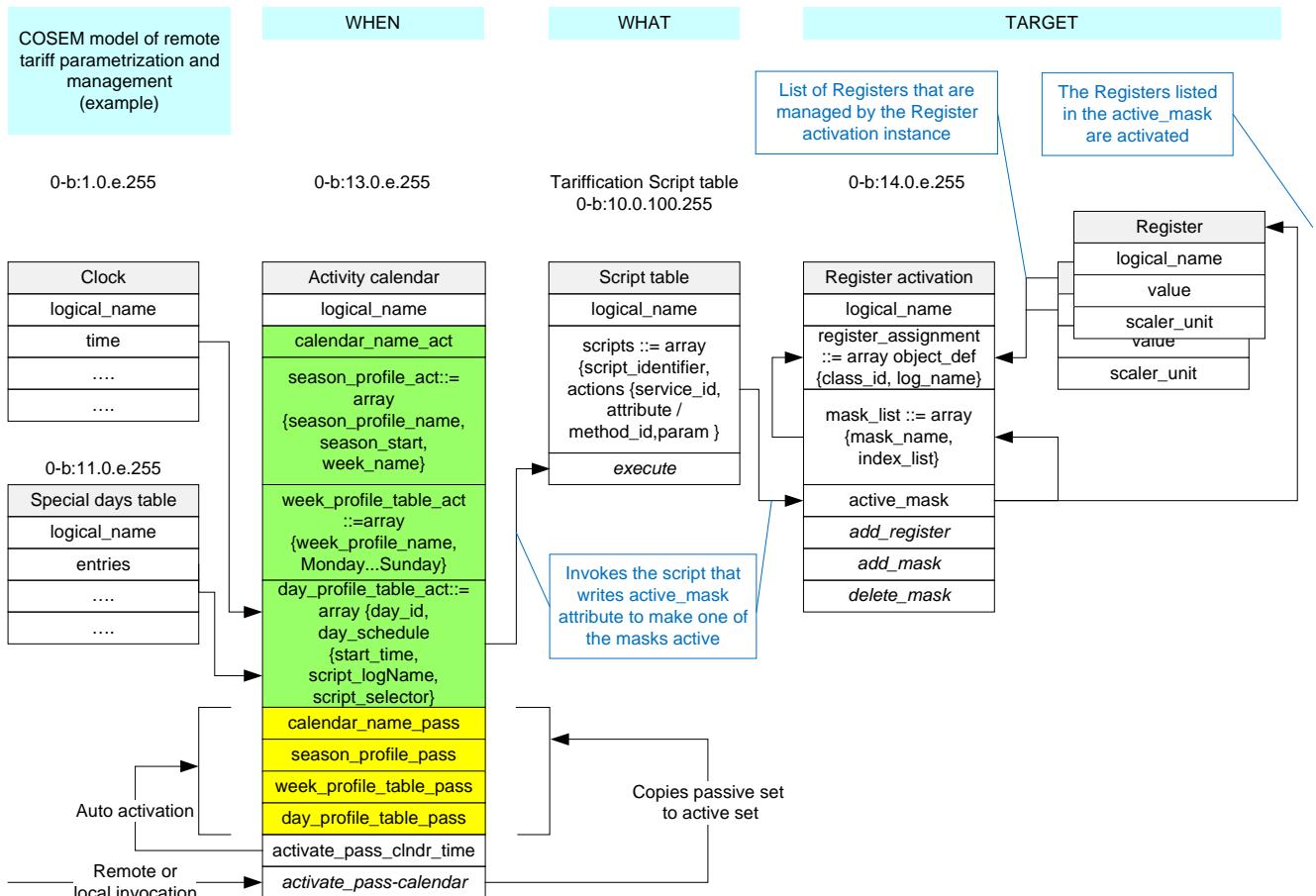


Figure 19 – COSEM tariffication model (example)

Figure 20 shows an example of modelling parametrization and management of billing using COSEM objects.

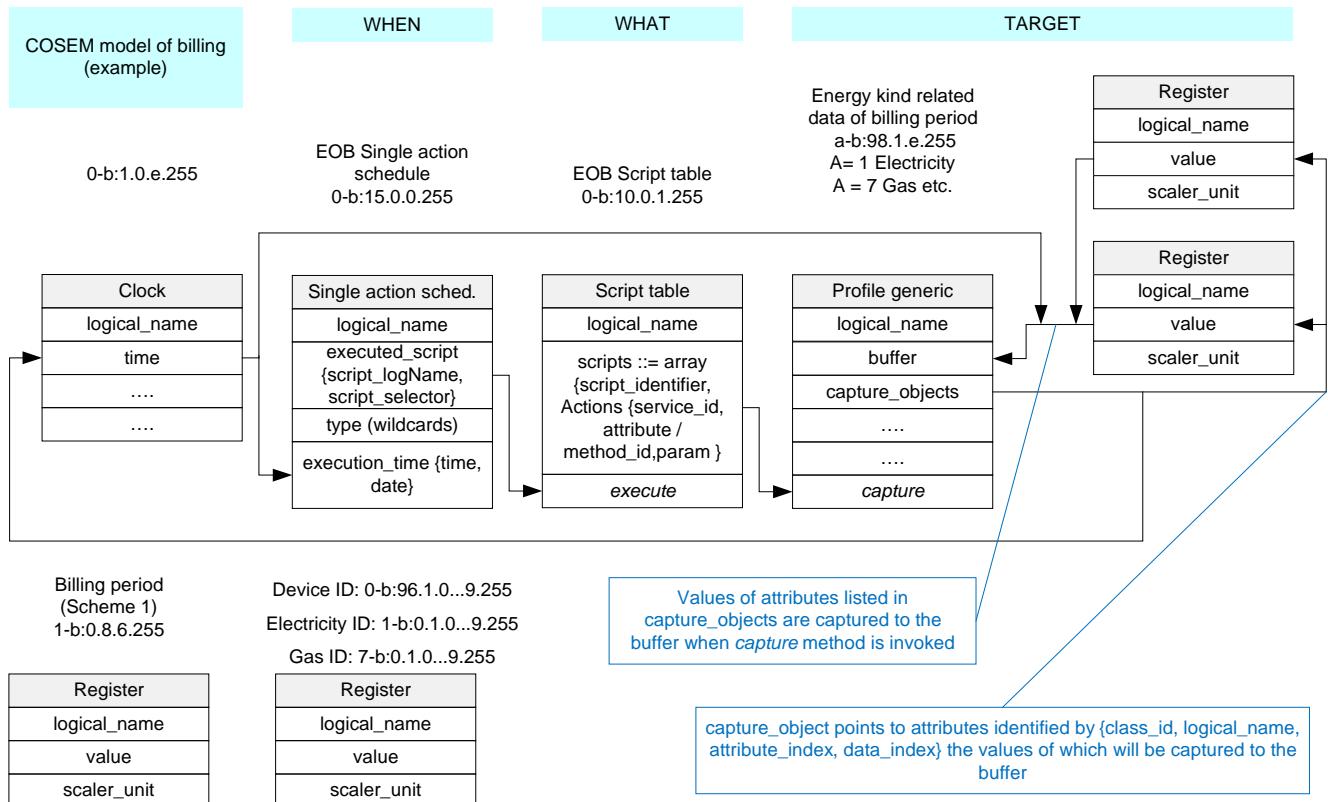


Figure 20 – COSEM billing model (example)

4.6 Payment metering related interface classes

4.6.1 Overview of the COSEM accounting model

The COSEM accounting model contains four interlinked interface classes: "Account", "Credit", "Charge" and the "Token gateway" IC. These classes are concerned with accounting for energy, not with delivery of that energy. The "Account" is linked to its associated "Credit", "Charge" and "Token gateway" objects by use of the value group D and B field such that an "Account" with D=0 should be linked to a "Token gateway" with D=40 and have a "Credit" objects with D=10 and "Charge" objects with D=20. Whereas an "Account" with D=1 should have "token gateway" with D=41, "Credit" objects with D=11 and "Charge" objects with D=21 etc. Multiple "Token gateway", "Credit" and "Charge" objects related to the same "Account" are identified using different values in the value group E field.

An "Account" object contains summary information and coordinates information pertaining to Credits and Charges. There is a single "Account" object per supply, for example, electricity import has one "Account" object, but a system that also has micro-generation could have a second "Account" to deal with the export of generated electricity; the second "Account" might or might not be accessible via the same Application Association (AA) as the first.

A "Credit" object contains detailed information about one source of funds. There is one or (usually) more "Credit" object(s) associated with an "Account": for example, one object for token credit and one object for emergency credit. Both of these objects can receive credit amounts from tokens, but emergency credit can only receive credit amounts when it has been consumed (entirely or partially) and when *credit_configuration* has bit 2 (Requires the credit amount to be paid back) set.

There are several types of credit listed in IEC/TR 62055-21:2005, and these are the types supported by the "Credit" IC. There can be zero or more instances of each type of Credit.

- **token_credit:** Credit that is transferred to a meter operating in prepayment mode, normally in the form of Credit Tokens;

NOTE 1 In a meter operating in credit mode or managed payment mode, a "Credit" object configured with type *token_credit* is used for recording the amount of credit used since last synchronised with a client.

NOTE 2 The content of the token is not defined by this document and may be an amount of money or another quantity that can be accounted in a way that is equivalent to the currency used by the meter.

- **reserved_credit:** Credit that is held in reserve, which is released under specific conditions;
- **emergency_credit:** Accounting functions that deal with the calculation and transacting of credit that is released only under emergency situations. Usually the amount of emergency credit used is recovered from subsequently purchased credit token

NOTE 3 Emergency above refers to a time when a consumer does not have any token credit, and not to any safety related situation.

- **time_based_credit:** Credit that is released on a scheduled time basis;
- **consumption_based_credit:** Credit that is released on the basis of a schedule of consumption levels. For example if a consumer keeps their consumption below a threshold, the system may release a predefined amount of credit.

A "Charge" object contains detailed information about one sink of expenditure, that is, one way in which credit is being used up. There can be one or (usually) more "Charge" objects associated with an "Account" for instance, one for energy usage, one for standing charge, and possibly one paying off a debt such as an installation charge.

There are several types of charge listed in IEC/TR 62055-21:2005 but the following types, distinguished by the trigger for collection, are the ones most useful in the COSEM accounting model. There can be zero or more instances of each type of "Charge" IC.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 161/492 |
|-----------------------|------------|-------------------------|---------|

- **consumption_based_collection:** describes charges that are collected according to the amount of consumption that has occurred in a tariff. A price per unit is assigned to each tariff register of the energy consumed
NOTE 4 Tariffs cannot be applied when currency is in time or energy units.
- **time_based_collection:** describes charges that are collected regularly according to the passage of time, independent of consumption in that period. This may be used to collect standing charges, or debt charge to be paid off over a period of time;
- **payment_event_based_collection:** describes charges that are collected from every top-up that is received, typically for debt repayment. These may be expressed as **amount-based**, where a fixed amount is taken from each top-up credit received (for example, the consumer pays £2 out of every vend regardless of the vend amount), or **percentage_based_collection** where a proportion of the amount of top-up credit received is taken (for example, with every vend the consumer pays 20% of the vend amount). Bit 0 (Percentage based collection) of the *charge_configuration* attribute of the "Charge" object specifies the method of *event_based_collection*. Figure 21 gives a general view of the account model.

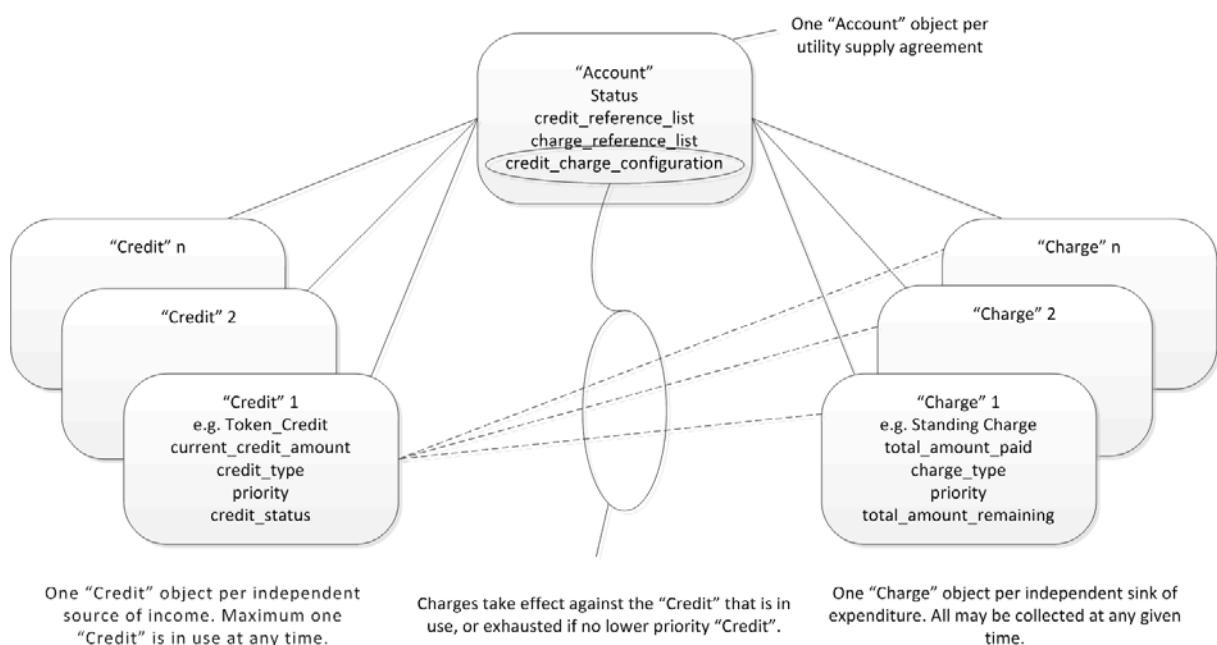


Figure 21 – Outline Account model

Figure 22 shows instances of "Account", "Credit" and "Charge" interface classes with some of their attributes and the relationships between those attributes. In this example:

- 1) There is one "Account", two "Credit" and two "Charge" objects configured;
- 2) "Credit" 1 is of type *token_credit* and the *low_credit_threshold* and *limit* attributes are configured to be 0;
- 3) Interaction between multiple classes is covered in this diagram. Detailed configuration of individual "Credit" and "Charge" objects is not shown.

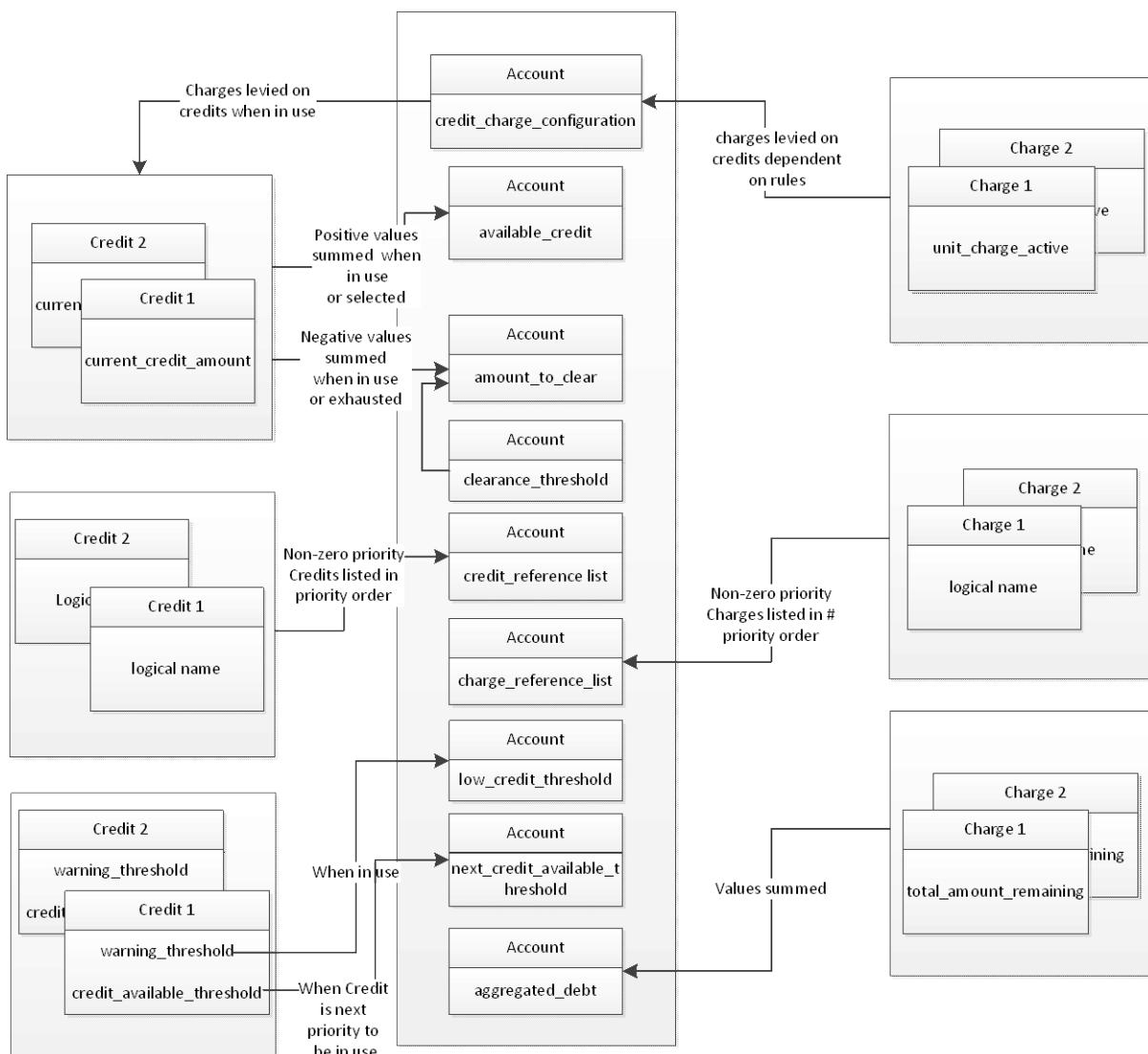


Figure 22 – Diagram of attribute relationships

4.6.2 Account (class_id = 111, version = 0)

Instances of the “Account” IC manage all the necessary elements related to the supervision of the “Credit” objects and the “Charge” objects referenced by a particular instance of the “Account” IC.

The operation of the payment metering function will be defined within the configuration of “Charge”, “Credit”, and “Account” objects and disconnection rules.

NOTE 1 Disconnection rules are either application specific or can be modelled using an instance of the “Arbitrator” IC, see 4.5.12.

If explicitly specified for a particular project, it is permissible for accounting to switch from credit to prepayment mode, or from prepayment to credit mode, once an accounting configuration has been correctly set up and the “Account” object has been activated. In the absence of any such specification the operating mode should remain fixed for any particular “Account” once it has been made active.

| 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 |

Attribute description

| | |
|--------------------------------|--|
| logical_name | Identifies the “Account” object instance. See 6.2.16. |
| account_mode_and_status | <p>Defines the payment_mode, enumerated below, and the status of the “Account”, also enumerated.</p> <pre>account_mode_and_status ::= structure { payment_mode: enum: (1) Credit mode, (2) Prepayment mode account_status: enum: (1) New (inactive) account, (2) Account active, (3) Account closed }</pre> |
| | <p>The payment_mode is an indication of a notional prepayment or credit/managed payment mode.</p> <p>NOTE 2 Payment_mode does not force some other actions in the meter, or cause a change of behaviour.</p> <p>The “Credit” and “Charge” objects associated with an “Account” object do not operate unless the “Account” object is in the active state. A New (inactive) “Account” object is passive and will become active at <i>account_activation_time</i> or invocation of the <i>activate_account</i> method.</p> |
| current_credit_in_use | <p>This attribute is an index into the <i>credit_reference_list</i> indicating which “Credit” object is <i>In use</i>.</p> |
| current_credit_status | <p>This attribute provides the status of the current “Credit” object <i>In use</i> and some information about the next priority “Credit” object.</p> <p>Bit 0 = <i>in_credit</i>, set when the <i>available_credit</i> is above zero,</p> <p>Bit 1 = <i>low_credit</i>, set when the “Account” object’s <i>available_credit</i> level has passed below the “Account” object <i>low_credit_threshold</i>,</p> <p>NOTE 3 The <i>low_credit_threshold</i> attribute of the associated “Credit” object <i>In use</i> is echoed in that attribute.</p> <p>Bit 2 = <i>next_credit_enabled</i>, set when the “Account” object <i>credit_reference_list</i> contains at least one other “Credit” object with non-zero priority, regardless of credit level,</p> <p>Bit 3 = <i>next_credit_selectable</i>, set when the next item in the “Account” object <i>credit_reference_list</i> contains a lower priority “Credit” object with non-zero priority with a <i>credit_configuration</i> bit 1 (Requires confirmation) set such that it requires confirmation (for example “Emergency Credit” that is selectable but has not yet been selected),</p> <p>NOTE 4 The next “Credit” object becomes selectable when the immediately higher priority “Credit” object has reached the “Account” object’s <i>next_credit_available_threshold</i>.</p> <p><i>next_credit_selected</i>, set when the “Account” object <i>credit_reference_list</i> contains a “Credit” object of lower</p> |

Bit 4 = 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

NOTE 6 When the next priority “Credit” object becomes the current “Credit” object then all bits have to update.

NOTE 7 A “Credit” object becomes *Selectable* when it is the next priority “Credit” and the *next_credit_available_threshold* attribute (reflecting the *credit_available_threshold* attribute in the “Credit”) of the “Account” object is greater than or equal to the *available_credit* in the “Account” object. However if the *available_credit* becomes greater than the *next_credit_available_threshold* due to the selection of the “Credit” the status of the “Credit” remains (2) *Selected*. If the *available_credit* becomes greater than the *next_credit_available_threshold* due to a top up or method invocation to increase the *current_credit_amount* of a “Credit” objects that is (2) *Selected* or (3) *In use* then this will change the status of the “Credit” object to (0) *Enabled*.

| | |
|-------------------------|--|
| available_credit | The <i>available_credit</i> attribute is the sum of the positive <i>current_credit_amount</i> values in the instances of the “Credit” class that: <ul style="list-style-type: none"> - are listed in the “Account” object <i>credit_reference_list</i>; - and have a “Credit” object <i>credit_status</i> (2) <i>Selected/Invoked</i> or (3) <i>In use</i>. This attribute holds the aggregate amount of credit available associated with the “Account” object. This value is positive or zero. NOTE 8 Negative values of “Credit” object <i>current_credit_amount</i> attributes are summed in <i>amount_to_clear</i> . See also the vertical axis in Figure 4. double-long; scaled according to the <i>currency</i> attribute. |
|-------------------------|--|

| | |
|------------------------|---|
| amount_to_clear | This value is the sum of: <ul style="list-style-type: none"> - all negative values of <i>current_credit_amount</i> in “Credit” objects that: <ul style="list-style-type: none"> - are listed in the “Account” object <i>credit_reference_list</i> and - have a “Credit” object <i>credit_status</i> (4) <i>Exhausted</i>, - have the <i>credit_configuration</i> bit 2 (Requires the credit amount to be paid back) is cleared, - the negative (Value * -1) of the amount of credit used from all “Credit” objects where the <i>credit_configuration</i> bit 2 (Requires the credit amount to be paid back) is set; and - the negative (Value * -1) of the value of the “Account” object <i>clearance_threshold</i> attribute; |
|------------------------|---|

See also [Figure 24](#).

NOTE 9 “Credit” objects where the *credit_configuration* bit 2 is set are referred to as “repayable” credits.

Credit used is the difference between the *preset_credit_amount* and the *current_credit_amount* (thus a positive value) of a “Credit” object when the “Credit” is (3) *In use* or (4) *Exhausted*. In the case of non-repayable credits the credit used is not relevant.

| | |
|---------------------------------------|--|
| amount_to_clear (continued) | <p>NOTE 10 The payment meter's application process might have specific functional behaviour to allow a consumer to live in emergency credit or not. This functionality is not within the Scope of this Companion Specification</p> <p>This attribute is significant when the meter has accumulated a temporary debt, for instance, while an emergency credit is <i>In use</i>, and/or during a friendly credit period.</p> <p>This attribute contains the minimum credit that the meter will need to receive (via token receipts and method invocations on "Credit" objects) in order to clear negative credits and also make "Credits" marked as repayable (for example an emergency credit), enabled again after they have been in the (3) <i>In use</i>, (2) <i>Selected/Invoked</i> or (4) <i>Exhausted</i> state.</p> <p>NOTE 11 For an explanation of the process of distributing top ups between "Credit" objects please refer to attribute 12 <i>token_gateway_configuration</i>.</p> <p>NOTE 12 Where a meter is being used in prepayment mode, <i>amount_to_clear</i> is usually the amount needed to reverse a disconnection. Amount to clear is shown on the vertical axis in Figure 4 as a negative value.</p> <p>double-long, scaled according to the <i>currency</i> attribute.</p> |
| clearance_threshold | <p>This attribute is used in conjunction with the <i>amount_to_clear</i>, and is included in the description of that attribute.</p> <p>This attribute becomes relevant when a meter has consumed all credit sources and has "Credit" objects with <i>current_credit_amount</i> attributes that have values less than zero.</p> <p>This represents the value of credit that the <i>available_credit</i> attribute must reach in order to make repayable "Credits" enabled again.</p> <p>NOTE 13 To achieve this, it is clear that enough credit must be added to the meter to ensure that the <i>current_credit_available</i> attribute on all listed "Credit" objects is zero or greater.</p> <p>double-long, scaled according to the <i>currency</i> attribute.</p> |
| aggregated_debt | <p>This attribute is a simple sum of <i>total_amount_remaining</i> of all the "Charge" objects which are listed in the "Account" object <i>charge_reference_list</i>, where bit 1 (Continuous collection) of the <i>charge_configuration</i> is cleared.</p> <p>NOTE 14 This value is not interchangeable with <i>amount_to_clear</i>, it is provided to assist external accounting.</p> <p>double-long, scaled according to the <i>currency</i> attribute.</p> |
| credit_reference_list | <p>This attribute is an array of logical names, identifying a collection of "Credit" objects that operate with this "Account" object. The elements in the array shall appear in priority order (priority 1 being first to priority n being last).</p> <p>Priority 0 credits shall NOT appear in this list, as by definition they are not enabled.</p> <p>credit_reference_list ::= array credit_reference credit_reference ::= octet-string</p> |
| charge_reference_list | <p>This attribute is an array of logical names, identifying a set of "Charge" objects that operate with this "Account" object. The elements in the array shall appear in priority order (priority 1 being first to priority n being last). Priority 0 charges shall NOT appear in this list, as by definition they are not applied.</p> <p>charge_reference_list ::= array charge_reference charge_reference ::= octet-string</p> |

credit_charge_configuration This attribute maps out which Charges are to be collected from which Credits.

If the array has zero elements then it is assumed that any Charge may be collected from any Credit in accordance with the “Credit” objects *credit_status* being (3) *In use* or (4) *Exhausted*.

If there are entries in this array then they represent each Charge that can be collected from each Credit.

If there is a Credit from which no Charges are collected then that Credit is not consumed and will remain unchanged until a Charge is configured.

NOTE 15 Collection of a Charge will cease when reaching zero, in cases where the appropriate “Charge” object has bit 1 (continuous collection) of its *charge_configuration* cleared. This will not alter the “Account” *credit_charge_configuration*.

```
credit_charge_configuration ::= array
    credit_charge_configuration_element
credit_charge_configuration_element ::= structure
{
    credit_reference:          octet-string,
    charge_reference:          octet-string,
    collection_configuration:  bit-string
}
```

Where *credit_reference* and *charge_reference* contain the *logical_name* of the relevant “Credit” and “Charge” object.

collection_configuration ::= bit-string

This element defines behaviour under specific conditions.

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.

NOTE 16 The meter application shall know internally whether it is in a supply disconnected, load limiting period, or friendly credit period, and shall take appropriate action based on the value of this attribute.

NOTE 17 It is implicit that charges are also collected at times when these specific conditions are not present.

token_gateway_configuration This attribute is designed to configure how a new top-up token from the “Token gateway” is to be apportioned, such that a configurable percentage of the token amount is distributed to each “Credit” object.

NOTE 18 The distribution of token top up amounts is also affected by the values of the *current_credit_amount*, *credit_type* and *credit_status* attributes of the “Credit” object.

If there are restrictions on how the credit token received through the “Token gateway” object is distributed, then this attribute determines the minimum proportion of the credit token that is attributed to each “Credit” object referenced in the array.

The proportions in this array shall not add up to more than 100%. If the sum of the proportions is less than 100% then the remainder will be added to the “Credit” objects using the rules below for an empty “*token_gateway_configuration*” attribute.

NOTE 19 It is possible that some “Credits” will get more than their allocated proportion as a result of the process highlighted above. The credit token is always 100% distributed across the “Credit” objects.

NOTE 20 The connection between a meter’s one or more “Token gateway” objects and a specific “Account” object is not explicitly shown in the model. The management of multiple “Account” objects and multiple token gateways shall be the subject of project specific companion specifications. However in the normal case an “Account” object has one “Token gateway” object and all tokens received shall follow the rules associated with the “Account” object with which it is associated (by application of the value group D field; see also 4.6.1).

```
token_gateway_configuration ::= array
                                token_gateway_configuration_element
```

```
token_gateway_configuration_element ::= structure
{
    credit_reference          octet-string,
    token_proportion         unsigned;
}
```

Where:

- *credit_reference* contains the *logical_name* of the relevant “Credit” and “Charge” object;
- *token_proportion* is scaled as one percent per integer step from 0 to 100.

If there are no entries in the array then it is assumed that there are no restrictions on processing the credit token within the meter’s payment application and credit should be applied first to the lowest priority “Credit” object with its *credit_status* set to (3) *In use* or (4) *Exhausted*, then apportioned to the other “Credit” objects in ascending order of the “Credit” object *priority* (starting with priority n and ending with priority 1).

If the “Credit” object requires a limited amount of top up (for example, an Emergency “Credit” object that is being repaid in full) then the credit used (*preset_credit_amount* less *current_credit_amount* before the top up) is taken from the Token amount and any surplus is applied to higher priority Credits following the same rules. At the point when the credit used is repaid the “Credit” will move from (3) *In use* or (4) *Exhausted* to (0) *Enabled*.

See also Note 11.

| | |
|--------------------------------|--|
| account_activation_time | Defines the time when the object itself invokes the specific method <i>activate_account</i> . A definition with "not specified" notation in all fields of the attribute will not be acted upon. Partial "not specified" notation in some fields of date and time are not allowed. When this time is in the past, this field indicates the time at which the "Account" object was activated. octet-string, formatted as specified in 4.1.6.1 for <i>date_time</i> . |
| account_closure_time | Defines the time when the object itself invokes the specific method <i>close_account</i> . A definition with "not specified" notation in all fields of the attribute will not be acted upon. Partial "not specified" notation in just some fields of date and time are not allowed. When this time is in the past, this field indicates the time at which the "Account" object was closed. octet-string, formatted as specified in 4.1.6.1 for <i>date_time</i> . |
| currency | Defines the "currency" unit used by all functions of an "Account" object. The <i>charge_per_unit</i> in the <i>unit_charge</i> attribute of "Charge" objects has an additional scaling factor that can be applied. The units could be currency or other units such as kWh, minutes; or other consumption units for different types of meters. When the units are monetary, then the name is expressed using the 3 character international symbol (GBP, USD, etc.) as defined in ISO 4217. currency ::= structure { currency_name: utf8-string, currency_scale: integer, currency_unit: enum } Where: currency_name: utf8-string as per ISO 4217 OR min = minutes, hrs = hours, sec = seconds, kWh = kilowatt hours, Whr = Watt hours, mt3 = m ³ , ft3 = ft ³ Jle = Joule OR Defined by project specific companion specification such that all characters used are lower case. The <i>currency_scale</i> indicates the exponent of a decimal multiple or submultiple of the base unit in which the relevant attribute values are expressed. Negative values indicate submultiples, NOTE 21 For example: if the currency is Euros and the values are expressed in tenths of one cent (one thousandth of a Euro) then the scalar will have value -3 (minus 3). |

| | |
|---|---|
| currency (continued) | currency_unit: enum: (0) time, (1) consumption, (2) monetary |
| <p>The <i>currency_unit</i> is an enumerated value that indicates the generic type of currency:</p> <ul style="list-style-type: none"> - time (in minutes, seconds, hours etc.); - consumption (in kWhrs, Whrs, m³, Whrs etc.); or monetary (GBP, USD, EUR etc.). | |
| low_credit_threshold | <p>This attribute reflects the <i>warning_threshold</i> attribute of the “Credit” object currently <i>In use</i>. This threshold can be used to generate a warning to the consumer, for example. It has no other function.</p> <p>double-long, scaled according to the <i>currency</i> attribute.</p> <p>NOTE 22 The value of this attribute will change depending on which “Credit” object is <i>In use</i> at the time of the query.</p> |
| <p>NOTE 22 The value of this attribute will change depending on which “Credit” object is <i>In use</i> at the time of the query.</p> | |
| next_credit_available_threshold | <p>This threshold reflects the <i>credit_available_threshold</i> attribute of the next-priority “Credit” object (except when there is no next priority “Credit”; in which case this attribute has the value -2 147 483 648 (largest possible negative value)).</p> <p>When the <i>available_credit</i> attribute of the “Account” object becomes less than or equal to this value, the <i>credit_status</i> attribute of the next priority “Credit” object changes to:</p> <ol style="list-style-type: none"> (1) <i>Selectable</i> in the case when the <i>credit_configuration</i> attribute of the “Credit” object concerned has the bit 1 (Requires confirmation) set; (2) <i>Selected/Invoked</i> in the case where the bit 1 (Requires confirmation) is cleared and the <i>next_credit_available_threshold</i> is greater than zero; (3) <i>In use</i> in the case where the bit 1 (Requires confirmation) is cleared and the <i>next_credit_available_threshold</i> is equal to zero. <p>NOTE 23 The “next-priority” refers to the next “Credit”, in priority order, that has not yet been selected. This attribute will change depending on which “Credit” is next in priority order.</p> <p>The <i>next_credit_available_threshold</i> will change depending on which “Credit” is <i>In use</i>.</p> <p>double-long, scaled according to the <i>currency</i> attribute.</p> |
| max_provision | <p>This attribute affects the operation of “Charge” objects that are configured with <i>charge_type</i> equal to (2) <i>payment_event_based_collection</i>.</p> <p>This attribute holds the limit amount scaled as defined in the <i>currency</i> attribute across all “Charge” objects with <i>charge_type</i> (2) <i>payment_event_based_collection</i>. The limit is related to the time period defined in <i>max_provision_period</i>.</p> <p>NOTE 24 The combination of this attribute and <i>max_provision_period</i> are intended to provide a limit to the total value of collections from top-ups within, for example, one week.</p> <p>NOTE 25 If a consumer vends many times in a short period, it is possible that he pays more than is required by the utility into one of his instantiated “Charge” objects (due to the unpredictability of payment event based collection).</p> <p>long-unsigned, scaled according to the <i>currency</i> attribute.</p> |

| | |
|-----------------------------|--|
| max_provision_period | This attribute holds the time period applicable for the <i>max_provision</i> attribute. This is specified in seconds. double-long |
|-----------------------------|--|

Method description

| | |
|-------------------------------|---|
| activate_account(data) | This method allows the activation of the “Account”. The <i>account_status</i> element of <i>account_mode_and_status</i> will be set to (2) <i>Account active</i> . NOTE 26 The <i>account_activation_time</i> will be set to the time of invoking this method. data ::= integer (0) |
|-------------------------------|---|

| | |
|----------------------------|---|
| close_account(data) | This method forces the closure of the Account. NOTE 27 This may be done pursuant to a change of supplier or change of tenancy or any other reason. The <i>account_status</i> element of <i>account_mode_and_status</i> will be set to (3) <i>Account closed</i> . The <i>account_closure_time</i> is set to the time of invoking this method. |
|----------------------------|---|

| | |
|----------------------------|---|
| reset_account(data) | NOTE 28 When this method is invoked the Account will no longer be active. As such its attributes will cease to change along with the attributes of all “Credit” and “Charge” objects listed within the <i>credit_reference_list</i> and <i>charge_reference_list</i> until such time that the “Account” object becomes active again, or the “Credit” and “Charge” objects are referenced from another “Account” object. data ::= integer (0) |
|----------------------------|---|

| | |
|-------------------------------------|--|
| 4.6.3 Credit interface class | |
| 4.6.3.1 General | If the <i>account_status</i> of the attribute <i>account_mode_and_status</i> is not equal to (2) <i>Active account</i> , then this method sets the attributes of the “Account” to default values except where an attribute does not have a default value, in which case the behaviour shall be project specific. If the <i>account_status</i> element of the attribute <i>account_mode_and_status</i> is equal to 1 (New, inactive account), then this method shall have no effect. data ::= integer (0) |

4.6.3 Credit interface class**4.6.3.1 General**

Instances of the “Credit” IC allow the management of a credit that can be consumed by charges. There are several different credit types; each “Credit” object characterizes itself by the values of its attributes.

All “Credits” associated with one supply are listed in the *credit_reference_list* attribute of the “Account” object. “Credits” move between states by:

- top-ups;
- the adjustment of *current_credit_amount* by method invocation; or
- the decrement of credit by charges.

This is explained in the state diagram in 4.6.3.2. Subclause 4.6.1 lists “Credit” types as defined in IEC/TR 62055-21:2005.

4.6.3.2 Credit states

The credit states only have meaning when *priority* is non-zero. They are shown in Table 28 and Figure 23. The state transitions are shown in Table 29.

Table 28 – 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. |

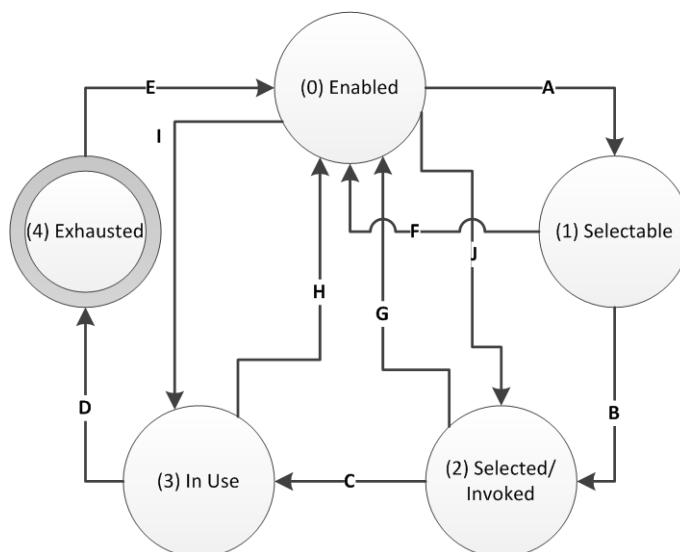


Figure 23 – Credit States when priority >0

Table 29 – 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> |
| 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 <i>Exhausted</i>).</p> |

| Credit state | From | To | Conditions |
|--------------|-------------|--------------------------|--|
| 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> |

4.6.3.3 Current credit status flags

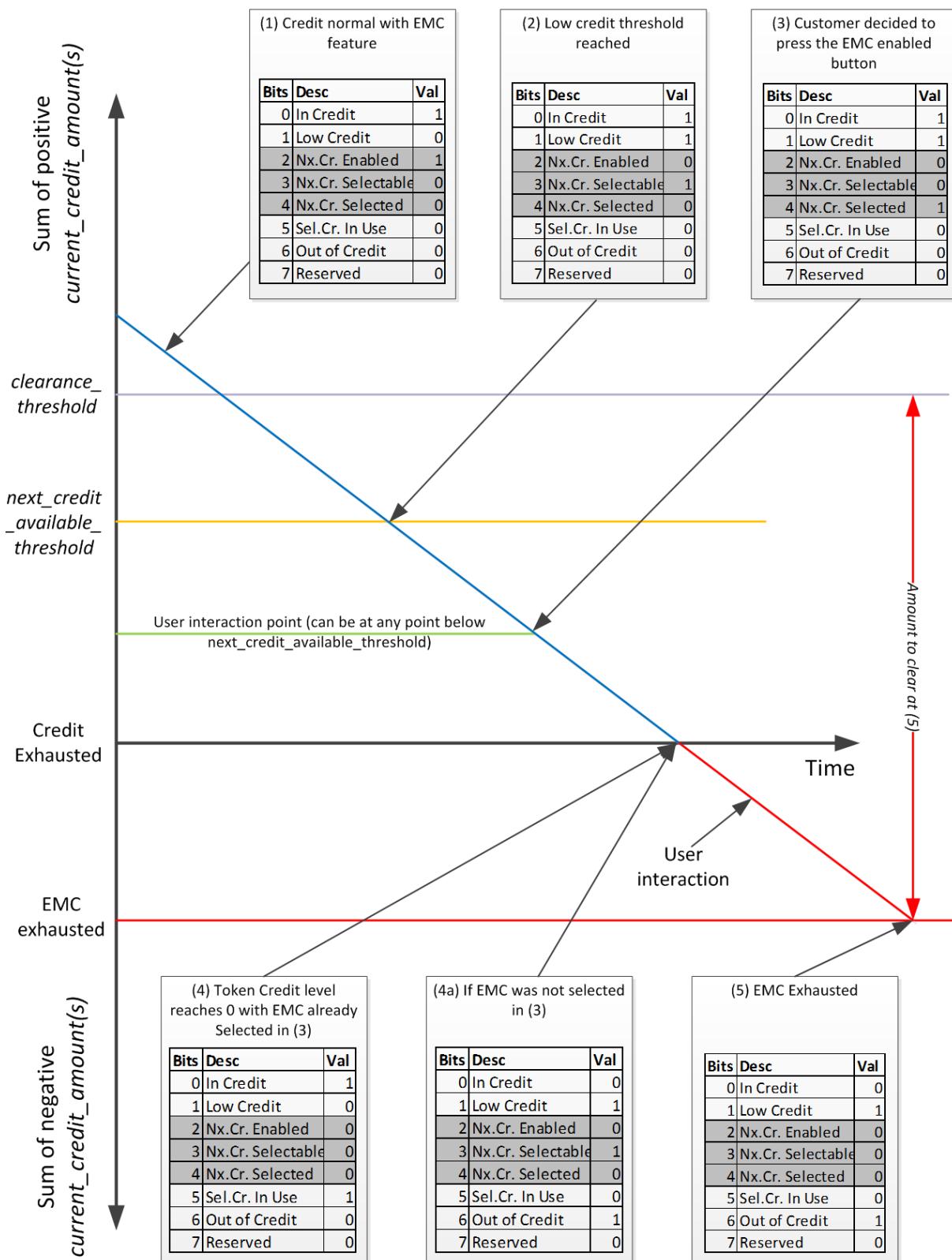
The significance of the *current_credit_status* flags (this is held by the *current_credit_status* attribute of the "Account" object) is explained in Figure 24 below.

In Figure 24 an example is given for possible cases involving a payment metering system with two "Credit" objects: Token Credit and Emergency Credit.

NOTE This is one possible implementation but others are possible and should be specified in project specific companion specifications.

There are two options for selection of selectable credits; either selection of a "Credit" whilst the current "Credit" is *In use* (case 4) or when the current "Credit" has been exhausted (case 4a).

In the case of credit being selected while current credit is *In use* (case 4), credit will start to be consumed from the next credit immediately after current credit has exhausted. In the case (4a) the current credit will become exhausted and potentially continue to be decremented by charges until the consumer takes action by selecting the selectable credit or adding a top up.

Figure 24 – Operation of *current_credit_status* flags

4.6.3.4 Credit (class_id = 112, version = 0)

| 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 |

Attribute description

| | |
|------------------------------|---|
| logical_name | Identifies the “Credit” object instance. See 6.2.16. |
| current_credit_amount | Provides the credit value of this particular “Credit” object. (See Figure 25). NOTE 1 This value is increased and decreased by invoking the methods of this object, the action of top-ups, and the collection of charges. This value contributes to the <i>available_credit</i> attribute in the “Account” object from which the “Credit” object is referenced. double-long, scaled according to the <i>currency</i> attribute of the “Account” object. |
| credit_type | This is an enumeration which identifies the type of Credit that this object represents. The type indicates which attributes are expected to be processed for this “Credit” object, but the actual processing is directed by values of other attributes including the configuration flags. The types available are: enum: (0) token_credit, (1) reserved_credit, (2) emergency_credit, (3) time_based_credit, (4) consumption_based_credit |

| | |
|-----------------------------|---|
| priority | <p>Describes the activation priority of this “Credit” object.</p> <p>Value 1 is the highest priority and value 255 the lowest.</p> <p>Every “Credit” object shall have a different priority, except in the case of priority 0.</p> <p>NOTE 2 Behaviour will be undefined if there are multiple “Credit” objects configured with the same non-zero priority.</p> <p>A value of 0 indicates that the “Credit” object is never activated, although it can still be configured and its <i>current_credit_amount</i> can be adjusted by invoking its methods, but it cannot receive top ups.</p> <p>When a “Credit” object of priority zero is configured it shall not appear in the “Account” <i>credit_reference_list</i>, it will be not considered as part of the <i>available_credit</i> calculation and it shall not be decremented by charges.</p> <p>See the <i>credit_reference_list</i> attribute of the “Account” object for more information.</p> |
| warning_threshold | <p>Holds a threshold value for <i>current_credit_amount</i>. When <i>current_credit_amount</i> is decremented to the value of <i>warning_threshold</i>, a warning is triggered for the consumer that credit is low.</p> <p>NOTE 3 The <i>low_credit_threshold</i> attribute of the associated “Account” object echoes this attribute of the currently <i>In use</i> “Credit” object.</p> <p>double-long, scaled according to the <i>currency</i> attribute of the “Account” object.</p> |
| limit | <p>This attribute holds a threshold value for <i>current_credit_amount</i>. When <i>current_credit_amount</i> is decremented to the value of <i>limit</i>, then <i>credit_status</i> becomes (4) <i>Exhausted</i>.</p> <p>NOTE 4 This is typically set to zero in a meter operating in prepayment mode. For a meter operating in credit mode the highest-priority “Credit” object would normally have a limit equal to the largest possible negative number.</p> <p>double-long, scaled according to the <i>currency</i> attribute of the “Account” object.</p> |
| credit_configuration | <p>Allows configuring the behaviour of the “Credit” object.</p> <p>If bit 0 is set then the credit item requires a visual indication on the meter display when it becomes selected.</p> <p>If bit 1 is set, confirmation is required from the consumer before moving into this type of “Credit” (for example this is the case of an emergency credit object, where the consumer is required to press a button before the “Credit” is (2) Selected/Invoked)</p> <p>If bit 2 is set then this indicates that this “Credit” amount requires repayment and consequently the credit used will contribute to the <i>amount_to_clear</i> attribute in the “Account” object that references the particular “Credit” object.</p> <p>If bit 3 is set then the <i>current_credit_amount</i> can be cleared by an end of billing period action (using a script acting upon the <i>set_amount_to_value</i> method. It is not recommended to configure “Credits” to permit this unless the meter is operating in credit mode).</p> <p>If bit 4 is set then this “Credit” object will be permitted to receive credit from tokens.</p> |

| | |
|--|--|
| credit_configuration (continued) | 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 |
| credit_status | Driven by the prepayment application to indicate the state that the "Credit" object is in. The states appear in Figure 23 above. Depending on the type of the "Credit" and its configuration, the value of this attribute is driven by the application based on the values of the <i>current_credit_amount</i> , <i>limit</i> , <i>preset_credit_amount</i> and <i>credit_available_threshold</i> attributes of the "Credit" objects and the <i>amount_to_clear</i> attribute of the "Account". When the <i>amount_to_clear</i> attribute of the 'Account" object referencing this "Credit" object transitions from a negative <i>value</i> to zero the EMC status shall be (0) <i>Enabled</i> again. NOTE 5 When a "Credit" object has a <i>credit_status</i> (4) <i>Exhausted</i> then this "Credit" cannot be invoked again until the "Credit" transitions to (0) <i>Enabled</i> . If this is the lowest priority "Credit" object, then charges may still be applied to this object. |

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*.

For additional explanation see Table 28.

| | |
|-----------------------------------|---|
| preset_credit_amount | <p>This attribute is a value that is set as an initial amount of credit that is available for the “Credit” object.</p> <p>The value is added to the <i>current_credit_amount</i> attribute:</p> <ol style="list-style-type: none"> 1) when the <i>credit_status</i> becomes (2), <i>Selected/Invoked</i> if <i>credit_configuration</i> bit 1 (Requires confirmation) is set, and the application confirmation occurs, or 2) when the <i>credit_status</i> becomes (3) <i>In use</i> if <i>credit_configuration</i> bit 1 (Requires confirmation) is cleared unless the <i>credit_status</i> is (4) <i>Exhausted</i>, or 3) when the method <i>invoke_credit</i> is invoked, if the <i>credit_configuration</i> bit 2 (Requires the credit amount to be paid back) is set, or 4) when the <i>date_time</i> in <i>period</i> occurs which is implicit. <p>When a “Credit” does not require a preset amount, the <i>preset_credit_amount</i> shall be 0 and the “Credit” can receive credit amounts from credit tokens or by invocation of the <i>update_amount</i> and <i>set_amount_to_value</i> methods. The value of <i>preset credit amount</i> does not change unless a new value is written by the client .</p> <p>In the case of “Credits” of type <i>emergency_credit</i>:</p> <ul style="list-style-type: none"> - the <i>preset_credit_amount</i> attribute shall be used, - the “Credit” shall only be able to receive credit amounts from tokens when: <ul style="list-style-type: none"> - the status is (3) <i>In use</i> or (4) <i>Exhausted</i>; - some (or all) credit has been used; and - it is required to be paid back (<i>credit_configuration</i> has bit 2 (Requires the credit amount to be paid back) set). <p>NOTE 6 When the <i>current_credit_amount</i> has reached the <i>limit</i> then the <i>credit_status</i> attribute will become (4) <i>Exhausted</i>. If the <i>credit_configuration</i> bit 2 (Requires the credit amount to be paid back) is set, then the credit used is the difference between <i>preset_credit_amount</i> and <i>current_credit_amount</i> (positive value), and it would be usual that it is repaid by the addition of a credit token or by means of invoking a method. After paying back the “Credit” the <i>current_credit_amount</i> will be zero but the <i>credit_status</i> will be (0) Enabled in the case of <i>amount_to_clear</i> = 0 or (4) Exhausted in the case where <i>amount_to_clear</i> is less than zero.</p> <p>If this attribute is set to zero there is no behaviour associated with it.</p> <p>double-long, scaled according to the <i>currency</i> attribute of the “Account” object.</p> |
| credit_available_threshold | <p>A threshold value related to the “Account” object <i>available_credit</i>.</p> <p>When the <i>available_credit</i> on the “Account” object decrements to the <i>credit_available_threshold</i> on the next-priority “Credit” object, the <i>credit_status</i> of that object changes to:</p> <ol style="list-style-type: none"> a) 1 (Selectable) if the <i>credit_configuration</i> bit 1 (Requires confirmation) is set, or b) 2 (Selected/Invoked) if the <i>credit_configuration</i> bit 1 (Requires confirmation) is cleared. <p>NOTE 7 This reflects whether the “Credit” requires confirmation before it is put into use.</p> <p>NOTE 8 A “Credit” will not be <i>In use</i> until exhaustion of a higher-priority “Credit”, but if selected before a higher priority “Credit” is exhausted the value of the <i>current_credit_amount</i> attribute will contribute to <i>available_credit</i> in the associated “Account”.</p> <p>double-long, scaled according to the <i>currency</i> attribute of the “Account” object.</p> |

| | |
|---------------------------------------|--|
| period | Where the <i>credit_type</i> = 3 (time_based_credit) or <i>credit_type</i> = 4 (consumption_based_credit), this attribute holds the time at which the <i>current_credit_amount</i> is set automatically to the <i>preset_credit_amount</i> . octet-string, formatted as specified in 4.1.6.1 for <i>date_time</i> . Wildcards are allowed. If all fields are wildcards, the <i>preset_credit_amount</i> will never be added. |
| Method description | |
| update_amount (data) | This method adjusts the value of the <i>current_credit_amount</i> attribute. Positive values adjust the <i>current_credit_amount</i> positively. Negative values are also permitted. data ::= double-long, scaled according to the <i>currency</i> attribute of the "Account" object. |
| set_amount_to_value (data) | This method sets the value of the <i>current_credit_amount</i> attribute. The amount previously in the updated attribute shall be given as the response parameter. data ::= double-long, scaled according to the <i>currency</i> attribute of the "Account" object. Returns double-long, scaled according to the <i>currency</i> attribute of the "Account" object. |
| invoke_credit (data) | This method is invoked to bring this "Credit" object to <i>credit_status</i> (2) Selected/Invoked if it has a <i>credit_configuration</i> bit 1 is set (Requires confirmation), and the <i>credit_status</i> is (1) Selectable. The mechanism for selecting the "Credit" is not in the scope of COSEM and shall be specified by the implementer (e.g. button push, meter process, script etc.) data ::= integer (0) |

4.6.3.5 Additional notes

NOTE 1 Although it is usual that tokens cause the incrementing of *current_credit_amount* and application of "Charge" objects cause decrement, these inputs are applied by means of signed addition and it could be possible to accept tokens or "Charge" objects with negative values.

NOTE 2 Behaviour of the emergency_credit type is determined by the *preset_credit_amount* attribute and the 'Requires the credit amount to be paid back' option in *credit_configuration* attribute.

When the "Credit" object is selected/invoked, the value of the *preset_credit_amount* attribute is added to the value of the *current_credit_amount* attribute (which is normally zero at this point in its lifecycle).

NOTE 3 To replenish emergency credit and make its status (0) Enabled after being (3) In use, sufficient payment should be received to take the *available_credit* of the "Account" object up to at least the *clearance_threshold*.

This will necessarily involve providing sufficient funds to repay the negative value of *current_credit_amount* of the "Credit" objects providing EMC function. To allow for the joint management of multiple "Credit" instances the credit used values for each "Credit" object are aggregated into the *amount_to_clear* together with the *clearance_threshold* attribute of the relevant "Account" object. When *amount_to_clear* reaches zero the status (4) Exhausted is by definition cleared on all associated "Credit" objects.

NOTE 4 The "Account" object *token_gateway_configuration* attribute determines the distribution of credit to "Credit" objects.

Figure 25 shows interaction of the *current_credit_amount* attributes of two "Credit" Objects in a payment metering system. It can be seen that the Emergency Credit, when selected contributes to the *available_credit* but does not become *In use* until the Token "Credit" has become *Exhausted*. At the point where Emergency "Credit" is *In use* it starts contributing to the *amount_to_clear*. When the Emergency "Credit" is *In use* and contributing to *amount_to_clear*, the *amount_to_clear* also takes into consideration the clearance threshold. The clearance threshold is only important when Token "Credit" *current_credit_amount* is zero or below.

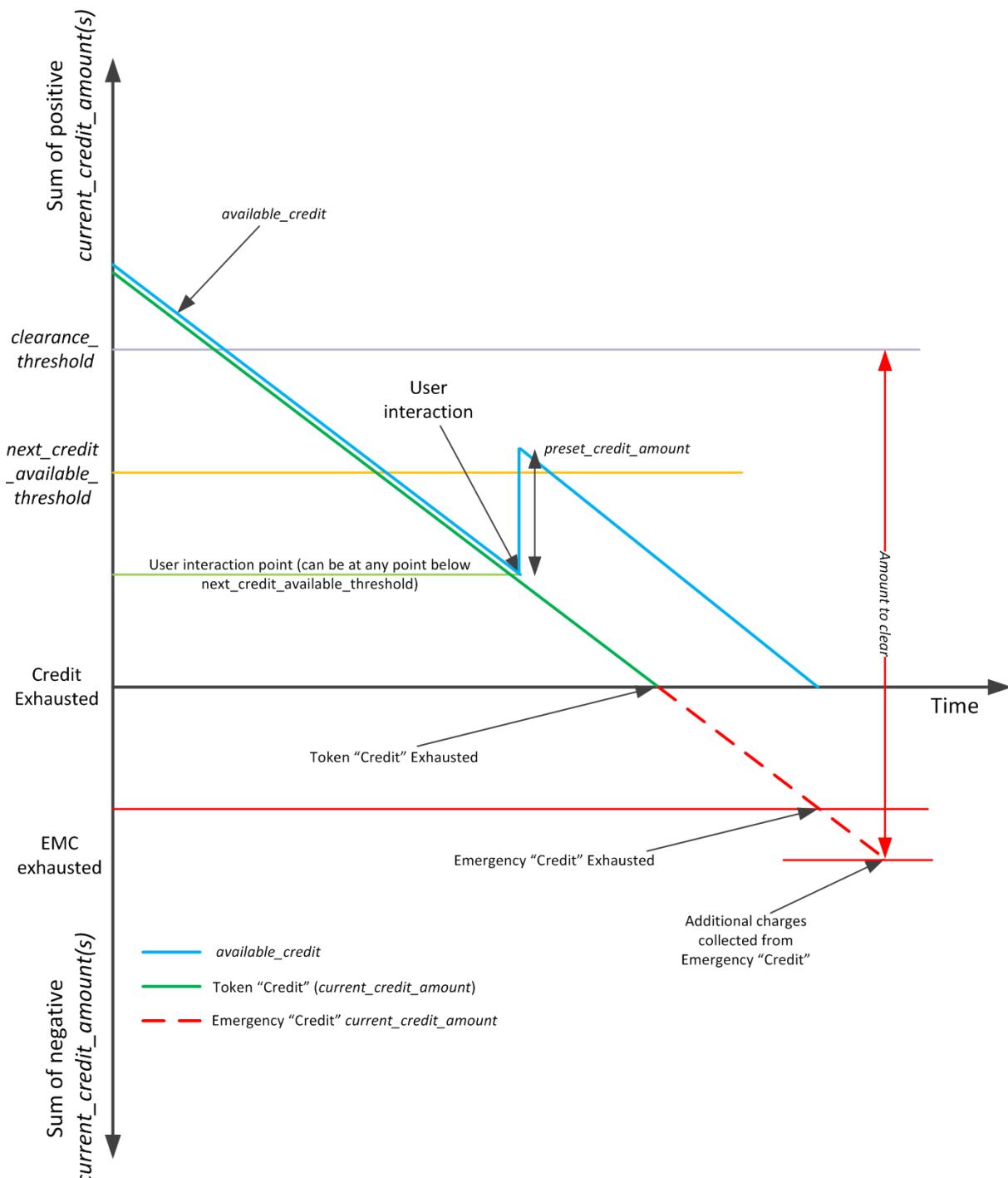


Figure 25 – Interaction of `current_credit_amount` and `available_credit` with Token "Credit" and Emergency "Credit"

As a result of invoking the Emergency "Credit" the `available_credit` will become bigger than `next_credit_available_threshold` but the Emergency "Credit" will not change `credit_status` from (1) `Selected` to (0) `Enabled`. However if the top up or method invocation to `set_amount_to_value`, forced the `available_credit` (by means of increasing the `current_credit_amount` of a "Credit" object) to be more than `next_credit_available_threshold` then the status shall be forced to (0) `Enabled`.

4.6.4 Charge (class_id = 113, version = 0)

Instances of the “Charge” IC allow the management of a single Charge. Depending on the attributes configured such as amount per price and the period, the Charge is taken at appropriate times from the “Credit” object *In use*.

NOTE 1 The details of the collection (charge-taking) cycle may be project dependent, but should be indicated in a project specific companion specification since they can affect third-party estimates of current values.

Each “Charge” object characterises itself by the values of its attributes.

All “Charges” associated with one supply are referenced in the *charge_reference_list* attribute of the related “Account” object.

| 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 |

Attribute description

| | |
|---------------------------|--|
| logical_name | Identifies the "Charge" object instance. See 6.2.16. |
| total_amount_paid | Holds the total amount collected for this "Charge" object. It is not normally reset while the "Account" remains active. NOTE 2 This value is incremented by the application at the same time and at the same rate as the charge that is collected. |
| charge_type | Designates the type of collection associated with this instance of the "Charge" class. enum: (0) consumption_based_collection, (1) time_based_collection, (2) payment_event_based_collection |
| | The <i>charge_type</i> indicates which attributes are expected to be processed for this "Charge" object. The processing is also influenced by the <i>charge_configuration</i> attribute. NOTE 3 In the case of <i>payment_event_based_collection</i> charges this attribute shall dictate the mechanism of charge collection. The application shall collect <i>payment_event_based_collection</i> charges from appropriate "Credits" as soon as credit is distributed from a new token. |
| priority | Describes the priority of this particular "Charge" when making collection from a "Credit" object. Every "Charge" object shall have a different priority, except in the case of priority 0. A value of 1 represents highest priority and a value of 255 the lowest. A value of 0 indicates that the "Charge" is not activated, though it can still be configured and its <i>total_amount_remaining</i> can be adjusted by invoking the appropriate methods of the "Charge" object. "Charge" objects with priority value 0 shall not appear in the <i>charge_reference_list</i> of the "Account" object, but "Charge" objects with <i>priority <> 0</i> do necessarily appear in the <i>charge_reference_list</i> of an "Account". |
| unit_charge_active | Defines the active price, that is the amount charged per unit consumed, per unit time, or per payment received, in terms of the currency attribute of the relevant "Account" instance and where relevant, the <i>scaler_unit</i> attribute of the object identified by the <i>commodity_reference</i> structure. unit_charge_active ::= structure { charge_per_unit_scaling: charge_per_unit_scaling_type, commodity_reference: commodity_reference_type, charge_table: charge_table_type } Where: charge_per_unit_scaling_type ::= structure { commodity_scale: integer, price_scale: integer } |

```

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

charge_table_type ::= array charge_table_element

charge_table_element ::= structure
{
    index:             octet-string,
    charge_per_unit:  long
}

```

Explanations related to charge_per_unit_scaling

Where the *charge_type* = (0) *consumption_based_collection* the field in *charge_per_unit_scaling* is expressed as the exponent (to the base of 10) of the multiplication factor of the base unit in which the relevant attribute values are expressed internally:

- *commodity_scale* is applied to the units associated with *commodity_reference*,
- *price_scale* is applied to the *currency_scale* of the *currency* attribute of the relevant "Account" object.

Where the *charge_type* IS NOT (0) *consumption_based_collection*, the *commodity_scale* shall be set to 0 and the *price_scale* is expressed as the exponent (to the base of 10) of the *currency_scale* to be applied to the *currency* attribute of the "Account" object.

NOTE 4 For example, suppose the primary *currency_name* is Euros (as determined in the "Account" object) with values expressed in units of one cent (one hundredth of a Euro) and the commodity being supplied is active import electrical energy with values expressed in units of 1 000 Wh (one kWh) from the *scaler_unit* attribute of the *commodity_reference*). Then for a price in tenths of a cent per kilowatt-hour the price scale will have value -3 (minus-three) since it is in thousandths of a base unit and the commodity scale will have value 0 (zero) since it is in base units (i.e. kWh).

For the case where the primary currency is kWh or similarly related to the *commodity_scale* then a TOU tariff is not possible and as such the *charge_table* element of the *unit_charge_active* and *unit_charge_passive* shall not be relevant but the *commodity_scale* and *price_scale* shall be the same value.

NOTE 5 The *commodity_scale* and *price_scale* do not imply any particular rate of collection.

Explanations related to charge_type

When the *charge_type* = (0) *consumption_based_collection* the *commodity_reference* identifies a *scaler_unit* attribute of a total register, measuring whatever is being supplied, for instance, electrical import energy.

When the *charge_type* = (1) *time_based_collection* or (2) *payment_event_based_collection*, then the *commodity_reference* shall be a structure of zero elements.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 185/492 |
|-----------------------|------------|-------------------------|---------|

Explanations to charge_table elements

Each element in the array *charge_table* is a structure identifying what is being charged for and collected.

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.

NOTE 6 For example, if the end consumer generates energy then the "Charge" should be negative, meaning that the utility has to pay the end consumer for the energy he produces.

NOTE 7 It is also possible to model exported energy using a separate "Account" object and other related objects.

Where the *charge_type* = (1) *time_based_collection* or (2) *payment_event_based_collection*, 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.

| | |
|------------------------------------|---|
| unit_charge_passive | Holds the details of prices to be applied at the occurrence of the activation date. Its data structure is the same as the <i>unit_charge_active</i> . On activation, the whole structure of <i>unit_charge_passive</i> is copied into <i>unit_charge_active</i> . |
| unit_charge_activation_time | Defines the time when the object itself invokes the specific method <i>activate_unit_charge_passive</i> . A definition with "not specified" notation in all fields of the attribute will deactivate this automatism. Partial "not specified" notation in just some fields of date and time is not allowed. octet-string, formatted as specified in 4.1.6.1 for <i>date_time</i> |
| period | Where <i>charge_type</i> = (0) <i>consumption_based_collection</i> or (1) <i>time_based_collection</i> it defines the period over which the Charge will be collected. If period is set to zero there is no automatic collection of this "Charge": collection is done by invoking the <i>collect</i> method. NOTE 8 Implementations may vary: it is possible to collect the whole charge in a single collection, or to divide the collection into parts. Where <i>charge_type</i> = (2) <i>payment_event_based_collection</i> this attribute is not used. double-long-unsigned, expressed in seconds. |
| charge_configuration | This attribute is a bit-string defined as follows: charge_configuration ::= bit-string Bit 0 = Percentage based collection, Bit 1 = Continuous collection If bit 0 is set and <i>charge_type</i> = (2) <i>payment_event_based_collection</i> then this "Charge" object will collect a proportion of each top-up in accordance with the proportion attribute. This applies only to token top ups and not to |

method invocations.

NOTE 9 The proportion attribute of the "Charge" object and the provision attribute of the "Account" object provide more information on this collection.

If bit 1 is set then collection will not terminate when the *total_amount_remaining* is equal to zero. If bit 1 is cleared then charge collection will terminate when the *total_amount_remaining* attribute is equal to zero.

| | |
|-------------------------------|--|
| last_collection_time | Holds the <i>date_time</i> when the last collection took place. This includes incremental collection made against consumption. octet-string, formatted as specified in 4.1.6.1 for <i>date_time</i> |
| last_collection_amount | Holds the last collection amount relating to the <i>last_collection_time</i> attribute. double-long, scaled according to the <i>price_scale</i> element of <i>unit_charge_active</i> . |
| total_amount_remaining | Where <i>charge_configuration</i> bit 1 (Continuous collection) is cleared, this attribute holds the total amount remaining for this "Charge" object. NOTE 10 The value of this attribute is initially configured by invoking the <i>set_total_amount_remaining</i> method. NOTE 11 This attribute is used to limit the collections for repayment of a debt. It is not a direct measure of the consumer's liability. The value is not allowed to go negative (it is set to zero instead) and is not incremented by collection of a "Charge" with a negative price. Where <i>charge_configuration</i> bit 1 (Continuous collection) is set, this attribute is zero. NOTE 12 See also the Additional Notes at the end of the "Charge" IC specification. double-long, scaled according to the <i>price_scale</i> element of <i>unit_charge_active</i> . |
| proportion | Where the <i>charge_configuration</i> bit 0 (Percentage based collection) is set and <i>charge_type</i> = (2) <i>payment_event_based_collection</i> , then this attribute is the proportion of each top-up amount processed within the "Token gateway" object linked via the "Account" object to this "Charge" object. The range 0x0000 to 0x2710 (0 to 10,000) represents 0 to 100%. NOTE 13 The amount of collection that occurs may be limited by the <i>max_provision</i> and <i>max_provision_period</i> attributes of the "Account" object. NOTE 14 If a fixed amount is required to be collected at every top-up (<i>charge_type</i> = 2), then the amount to be taken should be set in the first element of the <i>charge_table</i> array in the <i>unit_charge_active</i> / <i>unit_charge_passive</i> attribute. Where either <i>charge_type</i> <> (2) <i>payment_event_based_collection</i> or <i>charge_configuration</i> bit 0 (Percentage based collection) is cleared then this attribute has no effect. NOTE 15 In the case when <i>charge_type</i> = (2) <i>payment_event_based_collection</i> and the collection of a fixed amount is required see <i>unit_charge_active</i> and <i>unit_charge_passive</i> . |

Method description

update_unit_charge(data) Allows updating a subset of unit charge values inside the *unit_charge_passive* attribute. The *charge_per_unit_scaling* and *commodity_reference* values are not affected by this method.

NOTE 16 It remains necessary to activate the *unit_charge_passive* so that the values can take effect.

```
data ::= array charge_table_element
charge_table_element ::= structure
{
```

```
    index:          octet-string,
    charge_per_unit: long
```

```
}
```

See the *charge_table_element* of the *unit_charge_active* attribute for a description of the elements.

NOTE 17 In the case where the index exists then the value is overwritten, and if the index does not exist then the new value is appended to the array.

activate_passive_unit_charge(data) Copies the whole structure of the *unit_charge_passive* attribute into the *unit_charge_active* attribute.

NOTE 18 This method has no effect on the collection activity or the priority of the "Charge" object.

```
data ::= integer (0)
```

collect (data) Where *charge_type* <> (0) *consumption_based_collection*, this method makes collection of the amount defined in *unit_charge_active* when *charge_configuration* bit 0 (percentage based collection) is cleared.
Where *charge_type* = (0) *consumption_based_collection*, this method has no effect.
data ::= integer (0)

update_total_amount_remaining(data) Allows the update of the *total_amount_remaining* attribute. The value is to be scaled according to the *price_scale* of *unit_charge_active*. The value is added to *total_amount_remaining*. The amount previously in *total_amount_remaining* attribute shall be given as the response parameter.

NOTE 19 This does not affect *total_amount_paid*.

data ::= double-long, scaled according to the *price_scale* of *unit_charge_active*.

and

returns double-long, scaled according to the *price_scale* of *unit_charge_active*.

set_total_amount_remaining(data) Sets the *total_amount_remaining* attribute. The value shall be not less than 0. The amount previously in *total_amount_remaining* attribute shall be given as the response parameter.

NOTE 20 *total_amount_remaining* is set without affecting *total_amount_paid*.

data ::= double-long, scaled according to the *price_scale* of *unit_charge_active*,

and

returns double-long, scaled according to the *price_scale* of *unit_charge_active*.

4.6.5 Token gateway (class_id = 115, version = 0)

An instance of the “Token gateway” IC implements the Token Carrier Interface.

NOTE 1 A single instance of the “Token gateway” object is instantiated for each “Account” object and hence each 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.) | date-time | | | | 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 | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

Attribute description

| | |
|------------------------------|---|
| logical_name | Identifies the “Token gateway” object instance. See 6.2.16. |
| token | Contains the unprocessed octet string of the most recently received or token for the purpose of capture in a history profile. |
| token_time | Contains, for the most recently received and processed token, the time and date at which a received token arrived at the DLMS/COSEM server. octet-string, formatted as specified in 4.1.6.1 for <i>date_time</i> |
| token_description | Contains a description of the token for the most recently received and processed token which is supplied by the meter’s application program. NOTE 2 For example this could contain the type of token (credit or engineering), and/or the token origin. The use of this attribute is to be described in the token specification or project specific companion specifications. token_description ::= array token_description_element token_description_element ::= octet-string |
| token_delivery_method | Reflects the route by which the last token was received. enum: (0) via remote communications, (1) via local communications, (2) via manual entry |
| token_status | <i>token_status</i> is a structure containing a generic enumeration that shows the status of the last token operation and an optional bit-string that can be associated with the token status giving extra information about the token <i>status_code</i> . NOTE 3 It is expected that the optional bit string content will be documented in a project specific companion 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.

```
token_status ::= structure
{
    status_code:          enum,
    data_value:           bit-string
}

status_code: enum:
(0) Token format result OK,
(1) Authentication result OK,
(2) Validation result OK,
(3) Token execution result OK,
(4) Token format failure,
(5) Authentication failure,
(6) Validation result failure,
(7) Token execution result failure,
(8) Token received and not yet processed
```

NOTE 5 On receipt of a token the initial state shall be set to 8 while the token is being processed and until another state can be deduced.

The use of the *data_value* bit-string field is to be defined in project specific companion specifications.

Method description

| | |
|---------------------|---|
| enter (data) | This method is invoked to transfer a token to the DLMS/COSEM server in the form of octet-string. If successful it may return the <i>token_status</i> attribute. data ::= octet-string, and returns <i>token_status</i> |
|---------------------|---|

Additional notes

There are a number of configurable limits that may be held by “Data” objects and relate to the behaviour and processing of tokens and some aspects of the payment metering system. Some of these limits have been defined in this specification, have standard OBIS codes and are described below:

“Max credit limit”: On receipt of a token the meter’s application process shall check that the addition of the credit token’s value to the associated credit object(s) will not force the *available_credit* in the “Account” object above the limit programmed in the “Max credit limit” object. If the received token is found to force the *available_credit* to be more than this limit then the meter shall reject the token and return the appropriate status.

“Max vend limit”: On receipt of a token the meter’s application process shall check that the value of the credit token is not more than the limit programmed in the “Max vend limit” object. If the received token is found to be more than this limit then the meter must reject the token and return the appropriate status.

See also 6.2.16.

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

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

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

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

Attribute description

| | |
|---------------------|--|
| logical_name | Identifies the “IEC local port setup” object instance. See 6.2.17. |
| default_mode | Defines the protocol used by the meter on the port. enum: (0) protocol according to IEC 62056-21:2002 (modes A...E), (1) protocol according to Clause 8 of DLMS UA 1000-2 Ed. 8.1:201. Using this enumeration value all other attributes of this IC are not applicable, (2) protocol not specified. Using this enumeration value, attribute 4), <i>prop_baud</i> is used for setting the communication speed on the port. All other attributes are not applicable. |
| default_baud | Defines the baud rate for the opening sequence. enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud |

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

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

This IC allows modelling and configuring communication channels according to Clause 8 of DLMS UA 1000-2 Ed. 8.1:201. 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 | | | | |

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

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

NOTE 3 The maximum value of the *inter-octet-time-out* attribute has been increased from 1000 ms to 6000 ms in order to allow using communication media, where long delays may occur. The default value has been changed to 25 ms to align with 8.4.4.3.4 of DLMS UA 1000-2 Ed. 8.1:201.

Attribute description

| | | | | | | | | | | | | | | | | | | | | | |
|---------------------------------|---|------|---------------------|-------------|--------------------------|-------------|-----------------------|------|---------------------------|------|-------------------|--------|---------------------|-----------------|--------------------------|-----------------|-----------------------|--------|------------------------------------|--------|-------------------|
| logical_name | Identifies the "IEC HDLC setup" object instance. See 6.2.19. | | | | | | | | | | | | | | | | | | | | |
| comm_speed | <p>The communication speed supported by the corresponding port.</p> <p>enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud</p> <p>This communication speed can be overridden if the HDLC mode of a device is entered through a special mode of another protocol.</p> | | | | | | | | | | | | | | | | | | | | |
| window_size_transmit | The maximum number of frames that a device or system can transmit before it needs to receive an acknowledgement from a corresponding station. During logon, other values can be negotiated. | | | | | | | | | | | | | | | | | | | | |
| window_size_receive | The maximum number of frames that a device or system can receive before it needs to transmit an acknowledgement to the corresponding station. During logon, other values can be negotiated. | | | | | | | | | | | | | | | | | | | | |
| max_info_length_transmit | The maximum information field length that a device can transmit. During logon, a smaller value can be negotiated. | | | | | | | | | | | | | | | | | | | | |
| max_info_length_receive | The maximum information field length that a device can receive. During logon, a smaller value can be negotiated. | | | | | | | | | | | | | | | | | | | | |
| inter_octet_time_out | Defines the time, expressed in milliseconds, over which, when no character is received from the primary station, the device will treat the already received data as a complete frame. | | | | | | | | | | | | | | | | | | | | |
| inactivity_time_out | Defines the time, expressed in seconds over which, when no frame is received from the primary station, the device will process a disconnection. When this value is set to 0, this means that the inactivity_time_out is not operational. | | | | | | | | | | | | | | | | | | | | |
| device_address | <p>Contains the physical device address of a device.</p> <p>In the case of one byte addressing:</p> <table> <tr> <td>0x00</td> <td>NO_STATION Address,</td> </tr> <tr> <td>0x01...0x0F</td> <td>Reserved for future use,</td> </tr> <tr> <td>0x10...0x7D</td> <td>Usable address space,</td> </tr> <tr> <td>0x7E</td> <td>'CALLING' device address,</td> </tr> <tr> <td>0x7F</td> <td>Broadcast address</td> </tr> </table> <p>In the case of two byte addressing:</p> <table> <tr> <td>0x0000</td> <td>NO_STATION address,</td> </tr> <tr> <td>0x0001...0x000F</td> <td>Reserved for future use,</td> </tr> <tr> <td>0x0010...0x3FFD</td> <td>Usable address space,</td> </tr> <tr> <td>0x3FFE</td> <td>'CALLING' physical device address,</td> </tr> <tr> <td>0x3FFF</td> <td>Broadcast address</td> </tr> </table> | 0x00 | NO_STATION Address, | 0x01...0x0F | Reserved for future use, | 0x10...0x7D | Usable address space, | 0x7E | 'CALLING' device address, | 0x7F | Broadcast address | 0x0000 | NO_STATION address, | 0x0001...0x000F | Reserved for future use, | 0x0010...0x3FFD | Usable address space, | 0x3FFE | 'CALLING' physical device address, | 0x3FFF | Broadcast address |
| 0x00 | NO_STATION Address, | | | | | | | | | | | | | | | | | | | | |
| 0x01...0x0F | Reserved for future use, | | | | | | | | | | | | | | | | | | | | |
| 0x10...0x7D | Usable address space, | | | | | | | | | | | | | | | | | | | | |
| 0x7E | 'CALLING' device address, | | | | | | | | | | | | | | | | | | | | |
| 0x7F | Broadcast address | | | | | | | | | | | | | | | | | | | | |
| 0x0000 | NO_STATION address, | | | | | | | | | | | | | | | | | | | | |
| 0x0001...0x000F | Reserved for future use, | | | | | | | | | | | | | | | | | | | | |
| 0x0010...0x3FFD | Usable address space, | | | | | | | | | | | | | | | | | | | | |
| 0x3FFE | 'CALLING' physical device address, | | | | | | | | | | | | | | | | | | | | |
| 0x3FFF | Broadcast address | | | | | | | | | | | | | | | | | | | | |

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

4.7.3.1 Overview

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

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

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

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

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

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

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

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

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

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

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

For OBIS codes, see clause 6.2.20.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 194/492 |
|-----------------------|------------|-------------------------|---------|

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

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

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

Attribute description

| | |
|-----------------------------|---|
| logical_name | Identifies the “IEC twisted pair setup” object instance. See clause 6.2.20. |
| mode | <p>This attribute specifies the working mode of this interface</p> <p>enum:</p> <ul style="list-style-type: none"> (0) inactive. The interface ignores all frames received, (1) always active, (2) to (127) reserved, (128) to (250) manufacturer specific. |
| comm_speed | <p>Holds the communication speed supported by the port.</p> <p>enum:</p> <ul style="list-style-type: none"> (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud <p>NOTE IEC 62056-3-1:2013 supports baud rates from 1 200 to 9 600.</p> |
| primary_address_list | <p>Holds the list of Primary Station Addresses (ADP) for which each logical device of the real equipment (the secondary station) has been programmed. See IEC 62056-3-1:2013, 5.2.4.</p> <p>primary_address_list_type ::= array unsigned</p> |
| tabi_list | <p>Represents the list of the TAB(i) for which the real equipment (the secondary station) has been programmed in the case of forgotten station call (see IEC 62056-3-1:2013).</p> <p>When using IEC 62056-3-1 profile with DLMS/COSEM, the <i>tabi_list</i> attribute is made of only one element of value 0, the value used for the discovery process.</p> <p>tabi_list_type ::= array tabi_element</p> <p>tabi_element: integer</p> |

4.7.3.3 MAC address

Secondary stations – typically metering end devices – hold a secondary station address (ADS). The length of the ADS shall be 6 octets and consists of the elements shown in Table 30. This address shall be worldwide unique and it shall be assigned by the manufacturer to the secondary station. The ADS assigned is valid through the lifetime of the secondary station.

Table 30 – 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 Blue Book Edition 10 Table 29. | 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. | | | |

4.7.3.4 Fatal error register

Each device implementing the DLMS/COSEM communication profile specified in IEC 62056-3-1:2013 shall provide an error register holding the result of the last communication with the primary station. The structure of the fatal error register shall be as specified in Table 31.

Table 31 – 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. |

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

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

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

Attribute description

| | |
|------------------------------|--|
| logical_name | Identifies the “Modem configuration” object instance. See 6.2.6. |
| comm_speed | <p>The communication speed between the device and the modem, not necessarily the communication speed on the WAN.</p> <p>enum:</p> <ul style="list-style-type: none"> (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud |
| initialization_string | <p>Contains all the necessary initialization commands to be sent to the modem in order to configure it properly. This may include the configuration of special modem features.</p> <p>array initialization_string_element</p> <pre>initialization_string_element ::= structure { request: octet-string, response: octet-string, delay_after_response: long-unsigned }</pre> <p>If the array contains more than one initialization_string_element, the requests are sent in a sequence. The next request is sent after the expected response matching the previous request and waiting a delay_after_response time [ms], to allow the modem to execute the request.</p> <p>NOTE It is assumed that the modem is pre-configured so that it accepts the initialization_string. If no initialization is needed, the initialization string is empty.</p> |

| | |
|--|--|
| modem_profile | Defines the mapping from Hayes standard commands/responses to modem specific strings. array modem_profile_element modem_profile_element: octet-string |
| The modem_profile array shall contain the corresponding strings for the modem used in following order: | |
| Element 0: OK, Element 1: CONNECT, Element 2: RING, Element 3: NO CARRIER, Element 4: ERROR, Element 5: CONNECT 1 200, Element 6: NO DIAL TONE, Element 7: BUSY, Element 8: NO ANSWER, Element 9: CONNECT 600, Element 10: CONNECT 2 400, Element 11: CONNECT 4 800, Element 12: CONNECT 9 600, Element 13: CONNECT 14 400, Element 14: CONNECT 28 800, Element 15: CONNECT 33 600, Element 16: CONNECT 56 000 | |

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

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

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

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

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

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

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

Attribute description

logical_name Identifies the “Auto answer” object instance. See 6.2.6.

mode Defines the working mode of the line when the device is auto answering.

enum:

- (0) line dedicated to the device,
- (1) shared line management with a limited number of calls allowed. Once the number of calls is reached, the window status becomes inactive until the next start date, whatever the result of the call,
- (2) shared line management with a limited number of successful calls allowed. Once the number of successful communications is reached, the window status becomes inactive until the next start date,
- (3) currently no modem connected,
- (200...255) manufacturer specific modes

listening_window Defines the time points when the communication window(s) become active (start_time) and inactive (end_time). The start_time implicitly defines the period.

EXAMPLE When the day of month is not specified (equal to 0xFF) this means that we have a daily listening window management. Daily, monthly ...window management can be defined.

array window_element

```
window_element ::= structure
{
    start_time: octet-string,
    end_time: octet-string
}
```

start_time and end_time are formatted as specified in 4.1.6.1 for *date-time*.

status Here the status of the window is defined.

enum:

- (0) Inactive: the device will manage no new incoming call. This status is automatically reset to Active when the next listening window starts,
- (1) Active: the device can answer to the next incoming call,
- (2) Locked: This value can be set automatically by the device or by a specific client when this client has completed its reading session and wants to give the line back to the customer before the end of the window duration. This status is automatically reset to Active when the next listening window starts.

| | |
|--------------------------------|---|
| number_of_calls | This number is the reference used in modes 1 and 2. When set to 0, this means there is no limit. |
| number_of_rings | Defines the number of rings before the meter connects the modem. Two cases are distinguished: The number of rings within the window defined by the attribute <i>listening_window</i> and the number of rings outside the <i>listening_window</i> . nr_rings_type ::= structure { nr_rings_in_window: unsigned (0 = no connection in the window), nr_rings_out_of_window : unsigned (0 = no connection outside the window) } If the number of rings inside and outside the window is the same, the modem always connects regardless of the settings of the <i>listening_window</i> . |
| list_of_allowed_callers | Contains an – optional – list of calling numbers which further limits the connectivity of the modem based on the calling number. It also controls the acceptance of wake-up calls or wake-up messages (e.g. SMS) from a calling number. This requires the presence of a calling line identification (CLI) service in the communication network used. list_of_allowed_callers ::= array list_of_allowed_callers_element list_of_allowed_callers_element ::= structure { caller_id: octet-string, call_type: enum } - the caller_id element holds a calling number from which calls or messages (e.g. SMS) are accepted. The wild-card characters '?' and '*' are supported. With '?' any single character matches, with '*' any character string matches. '*' can only be used at the beginning or at the end of a number, but neither in between nor alone. Example 1: "+994193500" = only calls from "+994193500" are accepted. Example 2: "+9941935????" = calls from all numbers in the range of "+9941935000" to "+99419359999" are accepted. Example 3: "7777*" = calls from all numbers starting with "7777" are accepted. Example 4: "*9000" = calls from all numbers ending with "9000" are accepted. - the call_type element defines the purpose of the call, i.e. if it's a standard CSD call or a wake-up call / wake-up message. enum: (0) = normal CSD call; the modem only connects if the calling number matches an entry in the list. This is tested in addition to all other attributes, e.g. <i>number_of_rings</i> , <i>listening_windows</i> , etc. (1) = wake-up request; calls or messages from this calling number are handled as wake-up requests. The wake-up request is processed immediately regardless of all other attributes like <i>number_of_rings</i> and <i>listening_window</i> (except if the calling number is also present in the list of normal CSD calls, see NOTE 3 below). The received message shall be completely empty; otherwise it is not treated as a wake-up message. |

| | |
|--------------------------------|---|
| list_of_allowed_callers | If the message contains a valid xDLMS APDU from a client in a pre-established AA, the corresponding xDLMS service is executed instead of processing the wake-up request. |
| (continued) | If the message is not empty but does not contain any valid xDLMS APDU from a client in a pre-established AA then the server does not react. |
| | For a call of type (1) the modem <i>does not</i> connect - independently of the outcome of the wake-up process. |
| | For a call of both type (1) and type (0): see NOTE 3. |
| | NOTE 3 If the same calling number is defined as initiator of a normal CSD call (type (0)) and as initiator of a wake-up request (type (1)) the following rule applies for the incoming calls: the wake-up request is only processed if the caller disconnects the line before the <i>number_of_rings</i> criterion (depending on <i>listening_window</i>) is reached. If the <i>number_of_rings</i> criterion is met then a modem connection is established. |
| | The <i>number_of_rings</i> parameter shall be set large enough to allow the initiator of the call to control the behaviour of the receiver of the call without any knowledge of the time instances of the rings at the receiver's side. |
| | NOTE 4 If the <i>list_of_allowed_callers</i> is empty (= array [0]) the auto answer function operates in type (0) = normal CSD call and the modem connects independently of the calling number. |

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

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

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

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

Attribute description

| | |
|---------------------|--|
| logical_name | Identifies the “Auto connect” object instance. See 6.2.6. |
| mode | <p>Controls the auto connect functionality in terms of the timing, the message type and the infrastructure to be used.</p> <p>Modes (1) to (3) are dedicated to CSD services.</p> <p>Modes (4) to (6) are dedicated to sending specific messages using a specific infrastructure.</p> <p>Modes (101) to (104) apply to packet switched network connections only (e.g. GPRS).</p> |

| | | | | | | | |
|--|--|-------|----------------|--|--|--|--|
| mode | enum: | | | | | | |
| (continued) | <ul style="list-style-type: none"> (0) no auto connect; the device never connects, (1) auto dialling allowed anytime, the values defined in the <i>calling_window</i> are ignored, (2) auto dialling allowed within the validity time of the <i>calling_window</i>, (3) "regular" auto dialling allowed within the validity time of the <i>calling_window</i>; "alarm" initiated auto dialling allowed anytime, (4) SMS sending via Public Land Mobile Network (PLMN), (5) SMS sending via PSTN, (6) email sending, (7...99) reserved, (101) the device is permanently connected to the communication network, (102) the device is permanently connected to the communication network within the validity time of the calling window. The device is disconnected outside the calling window. No connection possible outside the calling window, (103) the device is permanently connected to the communication network within the validity time of the calling window. The device is disconnected outside the calling window but it connects to the communication network as soon as the connect method is invoked, (104) the device is usually disconnected. It connects to the communication network as soon as the connect method is invoked, 105...199) reserved, (200...255) manufacturer specific modes. | | | | | | |
| repetitions | The maximum number of retries in case of unsuccessful connection attempts. | | | | | | |
| repetition_delay | <p>The time delay, expressed in seconds until an unsuccessful connection attempt can be repeated.</p> <p>repetition_delay = 0 means delay is not specified</p> | | | | | | |
| calling_window | <p>Contains the time points when the window becomes active (<i>start_time</i>), and inactive (<i>end_time</i>). The <i>start_time</i> implicitly defines the period.</p> <p>EXAMPLE When the day of month is not specified (equal to 0xFF) this means that the calling window is managed on a daily basis. Daily, monthly ...window management can be defined.</p> <table> <tr> <td>array</td> <td>window_element</td> </tr> <tr> <td colspan="2"> <pre>window_element ::= structure { start_time: octet-string, end_time: octet-string }</pre> </td> </tr> <tr> <td colspan="2">start_time and end_time are formatted as specified in 4.1.6.1 for <i>date-time</i>.</td> </tr> </table> | array | window_element | <pre>window_element ::= structure { start_time: octet-string, end_time: octet-string }</pre> | | start_time and end_time are formatted as specified in 4.1.6.1 for <i>date-time</i> . | |
| array | window_element | | | | | | |
| <pre>window_element ::= structure { start_time: octet-string, end_time: octet-string }</pre> | | | | | | | |
| start_time and end_time are formatted as specified in 4.1.6.1 for <i>date-time</i> . | | | | | | | |

| | |
|------------------------------|--|
| destination_list | Contains the list of destinations (for example phone numbers, email addresses or their combinations) where the message(s) have to be sent under certain conditions. The conditions and their link to the elements of the array are not defined here. |
| array | destination |
| destination ::= octet-string | |

Method description

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

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

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

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

Attribute description

logical_name Identifies the “GPRS modem setup” object instance. See 6.2.22.

APN Defines the access point name of the network.

PIN_code Holds the personal identification number.

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

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

quality_of_service ::= structure

```

{
    default:      qos_element,
    requested:   qos_element
}
qos_element ::= structure
{
    precedence:  unsigned,
    delay:       unsigned,
    reliability: unsigned,
    peak throughput: unsigned,
    mean throughput: unsigned
}

```

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

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

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

A GSM diagnostic “Profile generic” object is also available to capture the attributes of the GSM diagnostic object, see 7.4.5.

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

Attribute description

| | |
|----------------------|---|
| logical_name | Identifies the “GSM Diagnostic” object instance. For logical names, see 6.2.22. |
| operator | Holds the name of the network operator e.g. “YourNetOp” |
| status | Indicates the registration status of the modem. enum: (0) not registered, (1) registered, home network, (2) not registered, but MT is currently searching a new operator to register to, (3) registration denied, (4) unknown, (5) registered, roaming (6) ... (255) reserved |
| cs_attachment | Indicates the current circuit switched status. enum: (0) inactive, (1) incoming call, (2) active, (3) ... (255) reserved |

ps_status The *ps_status* value field indicates the packet switched status of the modem.

enum:

- (0) inactive,
 - (1) GPRS,
 - (2) EDGE,
 - (3) UMTS,
 - (4) HSDPA,
 - (5) ... (255) reserved
-

cell_info Represents the cell information:

cell_info_type ::= structure

```
{
    cell_ID:          long-unsigned,
    location_ID:     long-unsigned,
    signal_quality:  unsigned,
    ber:              unsigned
}
```

Where:

- cell_ID: Two-byte cell ID in hexadecimal format;
 - location_ID: Two-byte location area code (LAC) in hexadecimal format (e.g. "00C3" equals 195 in decimal);
 - signal_quality: Represents the signal quality:
 - (0) -113 dBm or less,
 - (1) -111 dBm,
 - (2...30) -109...-53 dBm,
 - (31) -51 or greater,
 - (99) not known or not detectable;
 - ber: Bit error (BER) measurement in percent:
 - (0...7) as RXQUAL_n values specified in ETSI GSM 05.08 8.2.4.
 - (99) not known or not detectable.
-

adjacent_cells array adjacent_cell_info

adjacent_cell_info ::= structure

```
{
    cell_ID:          long-unsigned,
    signal_quality:  unsigned
}
```

Where:

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

capture_time Holds the date and time when the data have been last captured.

date-time is formatted as specified in 4.1.6.1.

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

4.8.1 Overview

The M-Bus related interface classes specified in this subclause 4.8 are used in two different scenarios:

- a DLMS/COSEM server hosted by a M-Bus master and exchanging dedicated M-Bus APDUs with M-Bus slaves;
- b) a DLMS/COSEM client hosted by a M-Bus master and exchanging DLMS/COSEM APDUs with DLMS/COSEM servers hosted by M-Bus slaves;

In case a) instances of the following M-Bus interface classes are used to set up and manage the M-Bus media in the DLMS/COSEM server:

- M-Bus client (class_id = 72), see 4.8.3;
- M-Bus master port setup (class_id = 74), see 4.8.5;
- M-Bus diagnostic (class_id = 77, version = 0), see 4.8.7.

In case b) instances of the following M-Bus interface classes are used in the DLMS/COSEM server:

- DLMS/COSEM server M-Bus port setup (class_id = 76), see 4.8.6;
- M-Bus slave port setup (class_id = 25), see 4.8.2; and/or
- Wireless Mode Q channel (class_id = 73), see 4.8.4;
- M-Bus diagnostic (class_id = 77, version = 0), see 4.8.7.

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

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

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

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

Attribute description

logical_name Identifies the “M-Bus slave port setup” object instance. See 6.2.21.

default_baud Defines the baud rate for the opening sequence.

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

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

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

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

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

The M-Bus client device may have one or more physical M-Bus interfaces, which can be configured using instances of the “M-Bus master port setup” IC, see 4.8.5.

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

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

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

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

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

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

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

Attribute description

| | | |
|----------------------------|--|--|
| logical_name | Identifies the “M-Bus client” object instance. See 6.2.21. | |
| mbus_port_reference | Provides reference to an “M-Bus master port setup” object, used to configure an M-Bus port, each interface allowing to exchange data with one or more M-Bus slave devices. | |
| capture_definition | Provides the <i>capture_definition</i> for M-Bus slave devices. NOTE 1 This attribute can be pre-configured or written as part of the installation procedure. array capture_definition_element | |
| | capture_definition_element ::= structure { data_information_block: octet-string, value_information_block: octet-string } | |
| | NOTE 2 The elements data_information_block and value_information_block correspond to Data Information Block (DIB) and Value Information Block (VIB) described in EN 13757-3:2013, 6.2 and clause 7 respectively. | |

| | |
|------------------------------|--|
| capture_period | >= 1: Automatic capturing assumed. Specifies the capture period in seconds. 0: No automatic capturing: capturing is triggered externally or capture events occur asynchronously. |
| primary_address | Carries the primary address of the M-Bus slave device. The range is 0...250. Each M-bus device is bound to a channel of the M-Bus master. However, there is no direct link between the primary address and the channel number. NOTE 3 The specification of the B field of the OBIS codes limits the range to 1...64 within one logical device. See 7.3.2. If the slave device is already configured and thus, its primary address is different from 0, then this value shall be written to the <i>primary_address</i> attribute. From this moment, the data exchange with the M-Bus slave device is possible. Otherwise, the <i>slave_install</i> method shall be used; see below. NOTE 4 The <i>primary_address</i> attribute cannot be used to store a desired primary address for an unconfigured slave device. If the <i>primary_address</i> attribute is set, this means that the M-Bus client can immediately operate with this primary address, which is not the case with an unconfigured slave device. |
| identification_number | Carries the Identification Number element of the data header as specified in EN 13757-3:2013, 5.5. This attribute, together with attributes 7, 8 and 9 are filled with the values found in the first message received after installation. If in subsequent messages these values are not the same, the message is discarded. |
| manufacturer_id | Carries the Manufacturer Identification element of the data header as specified in EN 13757-3:2013, 5.6. |
| version | Carries the Version element of the data header as specified in EN 13757-3:2013, 5.7. |
| device_type | Carries the Device type identification element of the data header as specified in EN 13757-3:2013, 5.8, Table 6. |
| access_number | Carries the Access Number element of the data header as specified in EN 13757-3:2013, 5.9. |
| status | Carries the Status byte element of the data header as specified in EN 13757-3:2013, 5.10, Table 7 and 8. It is updated with every readout of the M-Bus slave device. |
| alarm | Carries the Alarm state specified in EN 13757-3:2013 Annex D. It is updated with every readout of the M-Bus slave device. |
| configuration | Carries the Configuration field (previously: Signature field) as specified in EN 13757-3:2013, 5.12. It contains information about the encryption mode and the number of encrypted bytes. It is updated with every readout of the M-Bus slave device. |
| encryption_key_status | Provides information on the status of the encryption key. See also Annex B. enum: (0) no encryption key, (1) encryption_key set, (2) encryption_key transferred, (3) encryption_key set and transferred, (4) encryption_key in use. |

Method description

| | | | | | | | |
|-------------------------------------|--|-------------------------|---------------|--------------------------|---------------|-------|--------|
| slave_install (data) | <p>Installs a slave device, which is yet unconfigured (its primary address is 0). data ::= unsigned</p> <p>This method can be successfully invoked only if the current value of the <i>primary_address</i> attribute is 0. The following actions are performed:</p> <ul style="list-style-type: none"> - the M-Bus address 0 is checked for presence of a new device; - if no uninstalled M-Bus slave is found, the method invocation fails; - if the <i>slave_install</i> method is invoked with "0" as a parameter, then the <i>primary_address</i> attribute is assigned automatically. This is done by checking the <i>primary_address</i> attribute of all M-Bus client objects in the DLMS/COSEM device and then selecting the first unused number. The <i>primary_address</i> attribute is set to this address and it is then transferred to the M-Bus slave device; - if the <i>slave_install</i> method is invoked with a primary address (other than 0) as a parameter, then the <i>primary_address</i> attribute is set to this value and it is then transferred to the M-Bus slave device. | | | | | | |
| | <p>NOTE 5 Unconfigured slave devices are configured with primary address as specified in EN 13757-3:2013 Annex E.5.</p> | | | | | | |
| slave_deinstall (data) | <p>De-installs the slave device. The main purpose of this service is to de-install the M-Bus slave device and to prepare the master for the installation of a new device. The following actions are performed:</p> <ul style="list-style-type: none"> - the M-Bus address is set to 0 in the M-Bus slave device; - the encryption key transferred previously to the M-Bus slave device is destroyed; the default key is not affected; - the <i>encryption_key_status</i> is set to (0): no encryption_key; - the attribute <i>primary_address</i> is also set to 0. | | | | | | |
| | <p>NOTE 6 A new M-Bus slave can be installed only once the value of the <i>primary_address</i> attribute is 0.</p> | | | | | | |
| | <p data-bbox="420 1230 695 1262">data ::= unsigned (0)</p> | | | | | | |
| capture (data) | <p>Captures values – as specified by the <i>capture_definition</i> attribute – from the M-Bus slave device.</p> <p data-bbox="420 1343 668 1374">data ::= integer (0)</p> | | | | | | |
| reset_alarm (data) | <p>Resets alarm state of the M-Bus slave device.</p> <p data-bbox="420 1423 668 1455">data ::= integer (0)</p> | | | | | | |
| synchronize_clock (data) | <p>Synchronizes the clock of the M-Bus slave device with that of the M-Bus client device.</p> <p data-bbox="420 1545 668 1576">data ::= integer (0)</p> | | | | | | |
| data_send (data) | <p>Sends data to the M-Bus slave device.</p> <p data-bbox="420 1635 1044 1666">data ::= array data_definition_element</p> <p data-bbox="420 1675 901 1706">data_definition_element ::= structure</p> <p data-bbox="420 1715 436 1747">{</p> <table style="margin-left: 20px;"> <tr> <td data-bbox="500 1747 786 1778">data_information_block:</td> <td data-bbox="881 1747 1024 1778">octet-string,</td> </tr> <tr> <td data-bbox="500 1778 786 1810">value_information_block:</td> <td data-bbox="881 1778 1024 1810">octet-string,</td> </tr> <tr> <td data-bbox="500 1810 563 1841">data:</td> <td data-bbox="881 1810 1008 1841">CHOICE</td> </tr> </table> | data_information_block: | octet-string, | value_information_block: | octet-string, | data: | CHOICE |
| data_information_block: | octet-string, | | | | | | |
| value_information_block: | octet-string, | | | | | | |
| data: | CHOICE | | | | | | |

| | |
|----------------------------------|---|
| data_send (data) | { (continued) -- simple data types null-data [0], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], utf8-string [12], bcd [13], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], float32 [23], float64 [24] } |
| set_encryption_key (data) | <p>Sets the encryption key in the M-Bus client and enables encrypted communication with the M-Bus slave device.</p> <p>data ::= octet-string (encryption_key)</p> <p>After the installation of the M-Bus slave, the M-Bus client holds an empty encryption key. With this, encryption of M-Bus frames is disabled.</p> <p>Encryption can be disabled by invoking the <i>set_encryption_key</i> method with an octet_string of zero length.</p> <p>Changing the encryption key requires two steps:</p> <ul style="list-style-type: none"> - first, the key is sent to the M-Bus slave, encrypted with the default key, using the <i>transfer_key</i> method; - second, the key is set in the M-Bus master using the <i>set_encryption_key</i> method. |
| transfer_key (data) | <p>Transfers an encryption key to the M-Bus slave device.</p> <p>data ::= octet-string (encrypted_key)</p> <p>Before encrypted M-Bus frames can be used, an operational encryption key has to be sent to the M-Bus slave, by invoking the <i>transfer_key</i> method. The method invocation parameter is the operational encryption key encrypted with the default key of the M-Bus slave device. The M-Bus frame sent is not encrypted. After the execution, the encryption is enabled and all further telegrams are encrypted.</p> <p>Each M-Bus slave device shall be delivered with a default encryption key. A new encryption key can be set in the M-Bus client by invoking the <i>set_encryption_key</i> method with the new encryption key as a parameter.</p> <p>With further invocations of the <i>transfer_key</i> method, new encryption keys can be sent to the M-Bus slave. The method invocation parameter is the new encryption key encrypted with the default key. The M-Bus frame is encrypted.</p> <p>When an M-Bus slave is de-installed, the encryption key is destroyed but the default key is not affected. Encryption remains disabled until a new encryption key is transferred.</p> |

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

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

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

Attribute description

| | |
|-----------------------|--|
| logical_name | Identifies the “Wireless Mode Q channel” object instance. See 6.2.21. |
| addr_state | Defines whether or not the device has been assigned an address since last power up of the device. enum: (0) not assigned an address yet, (1) assigned an address either by manual setting or by automated method. |
| device_address | The currently assigned address of the device on the network. |
| address_mask | The group address the device will respond to when short form addressing is used. |

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

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

| M-Bus master port setup | 0...n | class_id = 74, version = 0 | | | |
|--------------------------|--------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. comm_speed (static) | enum | 0 | 7 | 3 | x + 0x08 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|---------------------|--|
| logical_name | Identifies the “M-Bus master port setup” object instance. See 6.2.21. |
| comm_speed | The communication speed supported by the port enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud |

4.8.6 DLMS/COSEM server M-Bus port setup (class_id = 76, version = 0)

Instances of the “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 (wM-Bus) communication 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 | | | | |

Attribute description

logical_name Identifies the “DLMS/COSEM server M-Bus port setup” object instance. See 6.2.21.

M-Bus_profile_selection References an M-Bus communication port setup object describing the physical capabilities for wired or wireless communication. The referenced object is either an “M-Bus slave port setup” object (class_id = 25) or a “Wireless Mode Q channel” object (class_id = 73).

M-Bus_port_communication_state Carries the communication status of the M-Bus node.
See EN 13757-4:2013, 11.6.3 Table 27.

enum:

- (0) No access: Meter provides no access windows (unidirectional meter),
- (1) Temporary no access: Meter supports bidirectional access in general, but there is no access window after this transmission (e.g. temporary no access in order to keep duty cycle limits or to limit energy consumption),
- (2) Limited access: Meter provides a short access windows only immediately after this transmission (e.g. battery operated meter),
- (3) Unlimited access: Meter provides unlimited access at least until next transmission (e.g. mains powered devices)

This attribute is relevant only in wM-Bus.

| | |
|-------------------------------|--|
| M-Bus_Data_Header_Type | Carries the type of the M-Bus Data Header, derived from the Cl_{TL} value from the current communication. |
| | In the case of a push operation the default value (2) shall be applied. enum: (0) M-Bus_Data_Header_Type == None_M-Bus_Data_Header, the value of the Cl_{TL} field shall be 0x10 when segmentation is not used, and shall be 0x00 .. 0x1F when segmentation is used (with the FIN bit set to 0 in all segments except in the last segment); (1) M-Bus_Data_Header_Type == Short_M-Bus_Data_Header, the value of the Cl_{TL} field shall be 0x61 in an M-Bus frame sent by a master and 0x7D in a an M-Bus frame sent by a slave; (2) M-Bus_Data_Header_Type == Long_M-Bus_Data_Header, the value of the Cl_{TL} field shall be 0x60 in an M-Bus frame sent by a master and 0x7C in a an M-Bus frame sent by a slave. |
| primary_address | Carries the primary address of the M-Bus slave device. See DLMS UA 1000-2 Ed. 8.1:201 Table 102. If the slave device is already configured and thus, its primary address is different from 0, then the data exchange with the M-Bus slave device is possible. |
| identification_number | Carries the Identification Number element of the Data Header as specified in EN 13757-3:2013, 5.5. In the case of unidirectional communication this value – together with attributes 6, 7, 8 and 9 – shall be parametrized. In the case of bi-directional communication this attribute – together with attributes 6, 7, 8 and 9 – are filled with the values found in the first message received after installation by different M-Bus network management interfaces. NOTE 1 If in subsequent messages these values are not the same, the message is discarded. |
| manufacturer_id | Carries the Manufacturer Identification element of the Data Header as specified in EN 13757-3:2013, 5.6. |
| version | Carries the Version element of the Data Header as specified in EN 13757-3:2013, 5.7. |
| device_type | Carries the Device type identification element of the Data Header as specified in EN 13757-3:2013, 5.8, Table 6. |
| max_pdu_size | Contains length capability available from M-Bus lower layers (expressed in bytes). Specifies the maximum length of the DLMS payload (an APDU or a part of it) that can be carried by one M-Bus frame. In the case of long messages, either block transfer provided by the DLMS/COSEM application layer or segmentation provided by the transport layer or both mechanisms can be used. |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 214/492 |
|-----------------------|------------|-------------------------|---------|

| | |
|-------------------------|---|
| listening_window | Defines the time points when the point-to-point communication window(s) become active (start_time) and inactive (end_time). The start_time implicitly defines the period. |
|-------------------------|---|

EXAMPLE When the day of month is not specified (equal to 0xFF) this means that we have a daily listening window management. Daily, monthly ...window management can be defined.

```
array window_element
window_element ::= structure
{
    start_time: octet-string,
    end_time: octet-string
}
```

start_time and end_time are formatted as specified in 4.1.6.1 for *date-time*.

This attribute is relevant only in wM-Bus.

4.8.7 M-Bus diagnostic (class_id = 77, version = 0)

Instances of the IC “M-Bus diagnostic” hold information related to the operation of the M-Bus network, like current signal strength, channel identifier, link status to the M-Bus network and counters related to 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 | | | | |

Attribute description

| | |
|---------------------------------|--|
| logical_name | Identifies the “M-Bus diagnostic” object instance. See 6.2.21. |
| received_signal_strength | When the wM-Bus profile is used, this attribute holds the value of signal strength of the last wM-Bus frame received, expressed in dBm. This attribute is relevant only for wireless bidirectional M-Bus communication. |
| channel_Id | When the wM-Bus profile is used, this attribute holds the identification of the channel currently used. The default value is 0. This attribute is relevant only for wM-Bus communication. |
| link_status | When the wM-Bus profile is used, this attribute holds the current status of link to the M-Bus network. The possible stati are as specified in EN 13757-5:2015, 9.7.4.1.7 Link State. enum: (0) Default (data never received), (1) Link in normal operation, (2) Link temporarily interrupted, (3) Link permanently interrupted |
| | <p>NOTE 1 If synchronisation with network is lost, trials start automatically to re-establish the link by performing radio scans. As long as re-synchronisation attempts are in progress, the link_status is temporarily interrupted.</p> <p>Link_status changes to normal operation when the link is re-established.</p> <p>Link_status changes to permanently interrupted , when the manufacturer specified number of attempts is exceeded.</p> <p>This attribute is relevant only for wM-Bus communication.</p> |
| broadcast_frames_counter | Holds the broadcast-frames-counter values with time stamp of last frame received and differentiated by client identifiers. array broadcast_frame_counter_definition broadcast_frame_counter_definition ::= structure { client_id: unsigned, counter: double-long-unsigned, time_stamp: date-time } The default value is 0. |
| transmissions_counter | Counts the number of frames transmitted by the related M-Bus port. The transmission counter is incremented at the beginning of each transmission phase. A client system can write this variable to update the counter. When the transmissions counter reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0. |

| | |
|-------------------------------|---|
| 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. The default value is 0. |
| 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. |
| capture_time | <p>Holds the time stamp of the most recent change of the value of the attributes 2, 4, 6, 7 or 8.</p> <pre>capture_time ::= structure { attribute_id: unsigned, time_stamp: date-time }</pre> |

Method description

| | |
|---------------------|--|
| reset (data) | Clears all counters, received_signal_strength, link_status and capture_time. data ::= integer(0) NOTE 2 Channel_id management is outside the scope of the specification of this IC. See EN 13757-4:2013. |
|---------------------|--|

4.9 Interface classes for setting up data exchange over the Internet

4.9.1 TCP-UDP setup (class_id = 41, version = 0)

This IC allows modelling the setup of the TCP or UDP sub-layer of the COSEM TCP or UDP based transport layer of a TCP-UDP/IP based communication profile.

In TCP-UDP/IP based communication profiles, all AAs between a physical device hosting one or more COSEM client application processes and a physical device hosting one or more COSEM server APs rely on a single TCP or UDP connection. The TCP or UDP entity is wrapped in the COSEM TCP-UDP based transport layer. Within a physical device, each AP – client AP or server logical device – is bound to a Wrapper Port (WPort). The binding is done with the help of the SAP Assignment object. See 4.4.5.

On the other hand, a COSEM TCP or UDP based transport layer may be capable to support more than one TCP or UDP connections, between a physical device and several peer physical devices hosting COSEM APs.

When a COSEM physical device supports various data link layers – for example Ethernet and PPP – an instance of the TCP-UDP setup object is necessary for each of them.

| TCP-UDP setup | | 0...n | class_id = 41, version = 0 | | | |
|-------------------------|------------|------------------|----------------------------|-------------|-------------|-------------------|
| Attributes | | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name | (static) | octet-string | | | | x |
| 2. TCP-UDP_port | (static) | long-unsigned | | | | x + 0x08 |
| 3. IP_reference | (static) | octet-string | | | | x + 0x10 |
| 4. MSS | (static) | long-unsigned | 40 | 65...535 | 576 | x + 0x18 |
| 5. nb_of_sim_conn | (static) | unsigned | 1 | | | x + 0x20 |
| 6. inactivity_time_out | (static) | long-unsigned | | | 180 | x + 0x28 |
| Specific methods | <i>m/o</i> | | | | | |

Attribute description

logical_name Identifies the “TCP-UDP setup” object instance. See 6.2.22.

TCP-UDP_port Holds the TCP-UDP port number on which the physical device is listening for the DLMS/COSEM application.

For DLMS/COSEM, the following port numbers have been registered by the IANA. See <http://www.iana.org/assignments/port-numbers>

dlms/cosem 4059/TCP DLMS/COSEM

dlms/cosem 4059/UDP DLMS/COSEM

IP_reference References an IP setup object by its logical name. The referenced object contains information about the IP Address settings of the IP layer supporting the TCP-UDP layer.

MSS With the help of the Maximum Segment Size (MSS) option, a TCP can indicate the maximum receive segment size to its partner. Note, that:

- this option shall only be sent in the initial connection request (i.e. in segments with the SYN control bit sent);
- if this option is not present, conventionally MSS is considered as its default value, 576;
- MSS is not negotiable; its value is indicated by this attribute.

| | |
|----------------------------|--|
| nb_of_sim_conn | The maximum number of simultaneous connections the COSEM TCP-UDP based transport layer is able to support. |
| inactivity_time_out | <p>Defines the time, expressed in seconds over which, if no frame is received from the COSEM client, the inactive TCP connection shall be aborted.</p> <p>When this value is set to 0, this means that the <i>inactivity_time_out</i> is not operational. In other words, a TCP connection, once established, in normal conditions – no power failure, etc. – will never be aborted by the COSEM server.</p> <p>Note, that all actions related to the management of the inactivity time-out function – measuring the inactivity time, aborting the TCP connection if the time-out is over, etc. – are managed inside the TCP-UDP layer implementation.</p> |

4.9.2 IPv4 setup (class_id = 42, version = 0)

NOTE 1 Compared to earlier editions of the Blue Book, this specification provides improvements in presenting the attributes. As this does not constitute technical changes, the version of the IC remains 0.

This IC allows modelling the setup of the IPv4 layer, handling all information related to the IP Address settings associated to a given device and to a lower layer connection on which these settings are used.

There shall be an instance of this IC in a device for each different network interface implemented. For example, if a device has two interfaces (using the TCP-UDP/IPv4 profile on both of them), there shall be two instances of the IPv4 setup IC in that device: one for each of these interfaces.

| IPv4 setup | 0...n | class_id = 42, version = 0 | | | |
|----------------------------------|----------------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. DL_reference (static) | octet-string | | | | x + 0x08 |
| 3. IP_address | double-long-unsigned | | | | x + 0x10 |
| 4. multicast_IP_address | array | | | | x + 0x18 |
| 5. IP_options | array | | | | x + 0x20 |
| 6. subnet_mask | double-long-unsigned | | | | x + 0x28 |
| 7. gateway_IP_address | double-long-unsigned | | | | x + 0x30 |
| 8. use_DHCP_flag (static) | boolean | | | | x + 0x38 |
| 9. primary_DNS_address | double-long-unsigned | | | | x + 0x40 |
| 10. secondary_DNS_address | double-long-unsigned | | | | x + 0x48 |
| Specific methods | m/o | | | | |
| 1. add_mc_IP_address (data) | o | | | | x + 0x60 |
| 2. delete_mc_IP_address (data) | o | | | | x + 0x68 |
| 3. get_nb_mc_IP_addresses (data) | o | | | | x + 0x70 |

Attribute description

logical_name Identifies the “IPv4 setup” object instance. See 6.2.22.

DL_reference References a Data link layer (e.g. Ethernet or PPP) setup object by its logical name.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 219/492 |
|-----------------------|------------|-------------------------|---------|

| | |
|-----------------------------|---|
| | The referenced object contains information about the specific settings of the data link layer supporting the IP layer. |
| IP_address | <p>Carries the value of the IP address (IPv4) of this physical device on the network to which the device is connected via the referenced lower layer interface. It can be either (static) or (dynamic). In the latter case, dynamic IP address assignment (for example DHCP) is used.</p> <p>If no IP address is assigned, the value is 0.</p> <p>EXAMPLE The IPv4 address 192.168.0.1 (in dotted decimal notation) corresponds to C0A80001 (hexa) which gives 3232235521 (double-long-unsigned).</p> |
| multicast_IP_address | <p>Contains an array of IP addresses. IP addresses in this array shall fall into the multicast group address range (“Class D” addresses, including IP addresses in the range of 224.0.0.0 to 239.255.255.255). When a device receives an IP datagram with one of these IP addresses in the destination IP address field, it shall consider that this datagram is addressed to it.</p> <p>multicast_IP_address ::= array double-long-unsigned</p> |
| IP_options | <p>Contains the necessary parameters to support the selected IP options – for example Datagram time-stamping or security services (IPSec).</p> <pre>IP_options ::= array IP_options_element IP_options_element ::= structure { IP_Option_Type: unsigned, IP_Option_Length: unsigned, IP_Option_Data: octet-string }</pre> <p>NOTE In all cases, as specified in RFC 791, the IP_Option_Length field includes the total length of all three fields: IP_Option_Type, IP_Option_Length and IP_Option_Data.</p> <p>Allowed IP_Option_Types:</p> <ul style="list-style-type: none"> - Security: IP_Option_Type = 0x82, IP_Option_Length = 11 If this option is present, the device shall be allowed to send security, compartmentation, handling restrictions and TCC (closed user group) parameters within its IP Datagrams. The IP_Option_Data shall contain the value of the Security, Compartments, Handling Restrictions and Transmission Control Code values, as specified in RFC 791. - Loose Source and Record Route: IP_Option_Type = 0x83 If this option is present, the device shall supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information. The IP_Option_Length and IP_Option_Data values are specified in RFC 791. - Strict Source and Record Route: IP_Option_Type = 0x89 If this option is present, the device shall supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information. The IP_Option_Length and IP_Option_Data values are specified in RFC 791. - Record Route: IP_Option_Type = 0x07 If this option is present, the device shall as well: <ul style="list-style-type: none"> - send originated IP Datagrams with that option, providing means to record the route of these Datagrams; - as a router, send routed IP Datagrams with the route option adjusted according to this option. |

The IP_Option_Length and IP_Option_Data values are specified in RFC 791.

- Internet Timestamp: IP_Option_Type = 0x44

If this option is present, the device shall as well:

- send originated IP Datagrams with that option, providing means to time-stamp the datagram in the route to its destination;
- as a router, send routed IP Datagrams with the time-stamp option adjusted according to this option.

The IP_Option_Length and IP_Option_Data values are specified in RFC 791.

subnet_mask Contains the subnet mask.

When sub-networking is used in a network segment, each device concerned shall behave conforming to the sub-networking rules. In order to do that, the device, besides of its IP address, needs also to know, how the IP address is structured within this sub-networked segment. The *subnet_mask* attribute carries this information.

With IPv4, the *subnet_mask* is a 32 bits word, expressed exactly in the same format as an IP Address (for example 255.255.255.0), but has another meaning: the '0' bits of the *subnet_mask* indicate the portion of the IP Address which is still used as Device_ID on a sub-networked IP Network ².

gateway_IP_address Contains the IP Address of the gateway device.

In most IP implementations, there is code in the module that handles outgoing datagrams to decide if a datagram can be sent directly to the destination on the local network or if it shall be sent to a gateway. In order to be able to send non-local datagrams to the gateway, the device shall know the IP address of the gateway device assigned to the given network segment.

If no IP address is assigned, the value is 0.

use_DHCP_flag When this flag is set to TRUE, the device uses DHCP (Dynamic Host Configuration Protocol) to dynamically determine the *IP_address*, *subnet_mask* and *gateway_IP_address* parameters.

On the other hand, when this flag is set to FALSE, the *IP_address*, *subnet_mask* and *gateway_IP_address* parameters shall be locally set.

primary_DNS_address The IP Address of the primary Domain Name Server (DNS).
If no IP address is assigned, the value is 0.

secondary_DNS_address The IP Address of the secondary Domain Name Server (DNS).
If no IP address is assigned, the value is 0.

Method description

add_mc_IP_address(IP_Address) Adds one multicast IP address to the *multicast_IP_address* array.
IP_Address ::= double-long-unsigned

delete_mc_IP_address(IP_Address) Deletes one IP Address from the *multicast_IP_address* array. The IP Address to be deleted is identified by its value.
IP_Address ::= double-long-unsigned

get_nb_of_mc_IP_addresses(data) Returns the number of IP Addresses contained in the *multicast_IP_address* array.
data ::= unsigned

² See more about sub-networking in RFC 940 and RFC 950.

4.9.3 IPv6 setup (class_id = 48, version = 0)

NOTE See also Annex C.

The IPv6 setup IC allows modelling the setup of the IPv6 layer, handling all information related to the IPv6 address settings associated to a given device and to a lower layer connection on which these settings are used.

There shall be an instance of this IC in a device for each different network interface implemented. For example, if a device has two interfaces (using the UDP/IP and/or TCP/IP profile on both of them), there shall be two instances of the IPv6 setup IC in that device: one for each of these interfaces.

| IPv6 setup | | 0...n | class_id = 48, version = 0 | | | |
|-------------------------------|----------|------------------|----------------------------|-------------|-------------|-------------------|
| Attributes | | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name | (static) | octet-string | | | | x |
| 2. DL_reference | (static) | octet-string | | | | x + 0x08 |
| 3. address_config_mode | (static) | enum | | | 0 | x + 0x10 |
| 4. unicast_IPv6_addresses | | array | | | | x + 0x18 |
| 5. multicast_IPv6_addresses | (static) | array | | | | x + 0x20 |
| 6. gateway_IPv6_addresses | (static) | array | | | 0 | x + 0x28 |
| 7. primary_DNS_address | (static) | octet-string | | | 0 | x + 0x30 |
| 8. secondary_DNS_address | (static) | octet-string | | | 0 | x + 0x38 |
| 9. traffic_class | (static) | unsigned | 0 | 63 | 0 | x + 0x40 |
| 10. neighbor_discovery_setup | (static) | array | | | | x + 0x48 |
| Specific methods | | m/o | | | | |
| 1. add_IPv6_address (data) | | o | | | | |
| 2. remove_IPv6_address (data) | | o | | | | |

Attribute description

| | |
|-------------------------------|--|
| logical_name | Identifies the “IPv6 setup” object instance. See 6.2.22. |
| DL_reference | References a Data link layer setup object by its logical name. The referenced object contains information about the specific settings of the data link layer supporting the IPv6 layer. |
| address_config_mode | Defines the IPv6 address configuration mode enum: (0) Auto-configuration (default), (1) DHCPv6, (2) Manual, (3) ND (Neighbour Discovery) |
| | NOTE The <i>address_config_mode</i> is common for all IPv6 addresses managed by an instance of the IPv6 setup class. |
| unicast_IPv6_addresses | Carries unicast IPv6 address(es) assigned to the related interface of the physical device on the network (unique local unicast, link local unicast and / or global unicast addresses). An IPv6 address can be either (static) or (dynamic) or both. <i>unicast_IPv6_addresses ::= array octet-string</i> The format of each unicast IPv6 address shall be as specified in RFC 3513. If no unicast IPv6 address is assigned to the interface, an array of zero |

elements is present (default).

To reset all unicast IPv6 address(es) configured, an array of zero elements shall be written.

| | |
|---------------------------------|---|
| multicast_IPv6_addresses | Contains an array of IPv6 addresses used for multicast. multicast_IPv6_addresses ::= array octet-string The format of each multicast IPv6 address shall be as specified in RFC 3513 . If no multicast IPv6 address is assigned to the interface, an array of zero elements is present (default). To reset all multicast IPv6 address(es) configured, an array of zero elements shall be written. |
| gateway_IPv6_addresses | Contains the IPv6 addresses of the IPv6 gateway device. gateway_IPv6_addresses ::= array octet-string The format of each gateway IPv6 address shall be as specified in RFC 3513. If no gateway IPv6 address is assigned to the interface, an array of zero elements is present (default). To reset all gateway IPv6 address(es) configured, an array of zero elements shall be written. |
| primary_DNS_address | Contains the IPv6 address of the primary Domain Name Server (DNS). If no IPv6 address is assigned, the length of the octet-string shall be 0. |
| secondary_DNS_address | Contains the IPv6 address of the secondary Domain Name Server (DNS). If no IPv6 address is assigned, the length of the octet-string shall be 0. |
| traffic_class | Contains the traffic class element of the IPv6 header. The 6 most-significant bits are used for DSCP (Differentiated Services Codepoint), which is used to classify packets as specified in RFC 2474, clause 3. |
| neighbor_discovery_setup | Contains the configuration to be used for both routers and hosts to support the Neighbor Discovery protocol for IPv6 (RFC 4861). array neighbor_discovery_setup neighbor_discovery_setup ::= structure { RS_max_retry: unsigned, RS_retry_wait_time: long-unsigned, RA_send_period: double-long-unsigned } Where: RS_max_retry Gives the maximum number of router solicitation retries to be performed by a node if the expected router advertisement has not been received. Range: 1-255 Default value: 3 RS_retry_wait_time Gives the waiting time in milliseconds between two successive router solicitation retries. Range: 0-65 535 Default value: 10 000 |

| | |
|----------------|--|
| RA_send_period | Gives the router advertisement transmission period in seconds. |
| Range: | 0-4 294 967 295 |

Method description

add_IPv6_address (data) Adds one IPv6 address for the physical interface to the IPv6 address array.

data ::= structure

```
{
    IPv6_address_type: enum,
    IPv6_address: octet_string
}
```

Where IPv6_address_type defines the type of IPv6 address to add:

enum: (0) unicast,
(1) multicast,
(2) gateway

IPv6_address specifies the IPv6 address to add. The new address will be automatically added at the end of the list. If an address has to be added to a specific position, this can be done by writing the whole array.

remove_IPv6_address (data) Removes one IPv6 address for the physical interface to the IPv6 address array.

data ::= structure

```
{
    IPv6_address_type: enum,
    IPv6_address: octet_string
}
```

Where IPv6_address_type defines the type of IPv6 address to remove:

enum: (0) unicast,
(1) multicast,
(2) gateway

IPv6_address specifies the IPv6 address to remove.

4.9.4 MAC address setup (class_id = 43, version = 0)

NOTE 1 The name and the use of this interface class has been changed in Edition 10 of the Blue Book from "Ethernet setup" to "MAC address setup" to allow a more general use, without changing the version.

Instances of this IC hold the MAC address of the physical device (or, more generally, a device or software.) There shall be an instance of this IC for each network interface of a physical device.

NOTE 2 In the case of the three-layer HDLC based communication profile, the MAC address (lower HDLC address) is carried by an IEC HDLC setup object.

NOTE 3 In the case of the S-FSK PLC communication profile, the MAC address is carried by a S-FSK Phy&MAC setup object.

| MAC address setup | 0...n | class_id = 43, version = 0 | | | | |
|--------------------------|--------------|----------------------------|------|------|------------|--|
| Attributes | Data type | Min. | Max. | Def. | Short name | |
| 1. logical_name (static) | octet-string | | | | x | |
| 2. MAC_address | octet-string | | | | x + 0x08 | |
| Specific methods | m/o | | | | | |

Attribute description

| | |
|---------------------|--|
| logical_name | Identifies the “MAC address setup” object instance. See 6.2.20, 6.2.22 and 6.2.25. |
| mac_address | Holds the MAC address. |

4.9.5 PPP setup (class_id = 44, version = 0)

NOTE 1 Compared to earlier editions of the Blue Book, this specification provides improvements in presenting the attributes. As this does not constitute technical changes, the version of the IC remains 0.

This IC allows modelling the setup of interfaces using the PPP protocol, by handling all information related to PPP settings associated to a given physical device and to a lower layer connection on which these settings are used. There shall be an instance of this IC for each network interface of a physical device.

| PPP setup | 0...n | class_id = 44, version = 0 | | | |
|--------------------------------|-------------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. PHY_reference (static) | octet-string | | | | x + 0x08 |
| 3. LCP_options (static) | LCP_options_type | | | | x + 0x10 |
| 4. IPCP_options (static) | IPCP_options_type | | | | x + 0x18 |
| 5. PPP_authentication (static) | PPP_auth_type | | | | x + 0x20 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|----------------------|--|
| logical_name | Identifies the “PPP setup” object instance. See 6.2.22. |
| PHY_reference | References another object by its logical_name. The object referenced contains information about the specific physical layer interface, supporting the PPP layer. |
| LCP_options | Contains the necessary parameters to support the selected LCP configuration options. LCP_options_type ::= array LCP_options_type_element LCP_options_type_element ::= structure { LCP_Option_Type: unsigned, LCP_Option_Length: unsigned, LCP_Option_Data: CHOICE { structure [2] -- for Callback-data boolean [3] -- for ProtF-Compr and AdCtr-Compr, double-long-unsigned [6] -- for ACCM and Mag-Num, unsigned [17] -- for FCS-Alternatives, long-unsigned [18] -- for MRU and Auth-Prot } } |
| NOTE 2 | In all cases, as specified in IETF STD 51 / RFC 1661, the LCP_Option_Length field includes the total length of all three fields: LCP_Option_Type, LCP_Option_Length and LCP_Option_Data. |

| | |
|-----------------------------------|---|
| LCP_options (continued) | NOTE 3 For assigned values see <i>Point-to-Point (PPP) Protocol Field Assignments</i> , available at: http://www.iana.org/assignments/ppp-numbers/ppp-numbers.xml |
|-----------------------------------|---|

The supported LCP_Option_Types are the following:

- Maximum-Receive-Unit (MRU), LCP_Option_Type = 1. See IETF STD 51 / RFC 1661.

This configuration option may be sent to inform the peer that the implementation can receive larger packets, or to request that peer send smaller packets. The default value is 1 500 octets;

- Async-Control-Character-Map (ACCM), LCP_Option_Type = 2. See IETF STD 51 / RFC 1662.

This configuration option provides a method to negotiate the use of control character transparency on asynchronous links;

- Authentication-Protocol, LCP_Option_Type = 3. See IETF STD 51 / RFC 1661.

This configuration option provides a method to negotiate the use of a specific protocol for authentication. By default, authentication is not required. The value indicates the authentication protocol used on the given PPP link. Possible values are:

0x0000 – No authentication protocol is used,
0xc023 – The PAP protocol is used,
0xc223 – The CHAP protocol is used,
0xc227 – The EAP protocol is used.

- Magic-Number, LCP_Option_Type = 5. See IETF STD 51 / RFC 1661.

This configuration option provides a method to detect looped-back links and other data link layer anomalies;

- Protocol-Field-Compression (PFC), LCP_Option_Type = 7. See IETF STD 51 / RFC 1661.

This configuration option provides a method to negotiate the compression of the PPP protocol fields;

- Address-and-Control-Field-Compression (ACFC), LCP_Option_Type = 8. See IETF STD 51 / RFC 1661.

This configuration option provides a method to negotiate the compression of the data link layer address and control fields;

- FCS-Alternatives, LCP_Option_Type = 9. See RFC 1570.

This configuration option provides a method for an implementation to specify another FCS format to be sent by the peer, or to negotiate away the FCS altogether. The value of the FCS-Alter (FCS Alternatives) options field identifies the FCS used. This field is one octet, and is comprised of the "logical or" of the following values:

bit 1 Null FCS,
bit 2 CCITT 16-bit FCS,
bit 4 CCITT 32-bit FCS.

- Callback, LCP_Option_Type = 13. See RFC 1570.

This configuration option provides a method for an implementation to request a dial-up peer to call back. This provides enhanced security by ensuring that the remote site can connect only from a single location as defined by the callback number.

callback_data ::= structure

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 226/492 |
|-----------------------|------------|-------------------------|---------|

| | |
|--------------------|---|
| LCP_options | { |
| (continued) | callback_active: boolean, // default: false, callback_data_length: unsigned, callback_operation: unsigned, callback_message: octet-string |
| | } |
| Where: | |
| | <ul style="list-style-type: none"> - the callback-active field indicates whether the callback option is active on this PPP link; - the callback_operation field indicates the contents of the Message field; callback_operation: unsigned (0) Location determined by user authentication, (1) Dialling string, (2) Location identifier, (3) E.164 number, (4) X.500 distinguished name, (5) Unassigned, (6) Location is determined during CBCP negotiation. |

the callback_message field is zero or more octets, and its general contents are determined by the callback_operation field. The actual format of the information is site or application specific.

| | |
|---------------------|--|
| IPCP_options | Contains the necessary parameters for the IP Control Protocol – the Network Control Protocol module of the PPP – that allow the negotiation of desirable Internet Protocol parameters. For details on IPCP, please refer to RFC 1332. IPCP_options_type ::= array IPCP_options_type_element |
|---------------------|--|

IPCP_options_type_element ::= structure

```

{
    IPCP_Option_Type:           unsigned,
    IPCP_Option_Length:         unsigned,
    IPCP_Option_Data:           CHOICE
    {
        array                  [1] -- for Pref-Peer-IP,
        -- each IP address is of type double-long-unsigned
        boolean                 [3] -- for GAO and USIP,
        double-long-unsigned     [6] -- for Pref-Local-IP,
        long-unsigned            [18] -- for IP-Comp-Prot
    }
}

```

NOTE 4 In all cases, as specified in RFC 1332, the IPCP_Option_Length field includes the total length of all three fields: IPCP_Option_Type, IPCP_Option_Length and IPCP_Option_Data.

The supported IPCP_Option_Types are the following:

- IP-Compression-Protocol (IP-Comp-Prot) IPCP_Option_Type = 2. See RFC 1332.

This configuration option provides a way to negotiate the use of a specific compression protocol. By default, compression is not enabled. Possible values are:

| | |
|------------------------------------|--|
| IPCP_options (continued) | <p>0x0000 – No IP Compression is used (default), 0x002d – Van Jacobson Compressed TCP/IP (RFC 1332), 0x0003 – Robust Header Compression (ROHC) (RFC 3241), 0x0061 – IP Header Compression (RFC 2507, RFC 3544)</p> <ul style="list-style-type: none"> - Preferred-Local-IP-Address (Pref-Local-I), IPCP_Option_Type = 3. See RFC 1332. This configuration option provides a way to negotiate the IP address to be used on the local end of the link. It allows the sender of the Configure-Request to state, which IP-address is desired, or to request that the peer provide the information. The peer can provide this information by NAK-ing the option, and returning a valid IP-Address; <p>NOTE 5 In RFC 1332 the name of option Type 3 is “IP-Address”.</p> <p>NOTE 6 Option types 20, 21 and 22 have been specified for the purposes of DLMS/COSEM; they are not specified in RFC 1332.</p> <ul style="list-style-type: none"> - Preferred-Peer-IP-Addresses (Pref-Peer-IP), IPCP_Option_Type = 20. This configuration option provides a way to negotiate the IP Address to be used on the remote end of the link. When the Grant-Access-Only-to-Pref-Peer-on-List (GAO) parameter is set to be TRUE, the device shall accept PPP connection only with a remote device having one of the IP Addresses on this list. When the Use-Static-IP-Pool (USIP) parameter is set to TRUE, the COSEM Server device shall try to assign one of these IP Addresses to the remote device; - Grant-Access-Only-to-Pref-Peer-on-List (GAO), IPCP_Option_Type = 21. This configuration option indicates whether the device can accept PPP connection only with peer devices with IP Address on the above list or not. Its default value is FALSE; - Use-Static-IP-Pool (USIP), IPCP_Option_Type = 22. This configuration option indicates whether the device should try to assign one of the IP Addresses of the Preferred-Peer-IP-Addresses to the remote end device during the IP Address negotiation phase or not. |
| PPP_authentication | <p>Contains the parameters required by the PPP authentication procedure used.</p> <p>PPP_auth_type ::= CHOICE</p> <pre>{ null-data [0] -- used when no authentication is required, structure [2] -- PAP_login or CHAP_algorithm or EAP_params }</pre> <p>PAP_login ::= structure</p> <pre>{ user-name: octet-string, PAP-password: octet-string }</pre> <p>CHAP_algorithm ::= structure</p> <pre>{ user-name: octet-string, algorithm_id: unsigned }</pre> <p>Possible values for CHAP-algorithm-id parameter today are as follows: 0x05: CHAP with MD5 (default), see RFC 1994.</p> |

| | |
|-----------------------------------|---|
| PPP_ | 0x06 – SHA-1, |
| Authentication | 0x80 – MS-CHAP, see RFC 2433 |
| (continued) | 0x81 – MS-CHAP-2, see RFC 2759. |
| | NOTE New values can be used as become assigned. |
| | When CHAP is used, a “secret” is also required to verify the “challenge” sent by the client. This “secret” is not accessible in the PPP setup object. |
| NOTE 6 | The PPP Challenge Handshake Authentication Protocol (CHAP) is specified in RFC 1994. |
| EAP_params ::= structure | |
| { | |
| md5_challenge (Type 4): | boolean, |
| one_time_password (OTP, Type 5): | boolean, |
| generic_token_card (GTC, Type 6): | boolean |
| } | |
| NOTE 7 | The Extensible Authentication Protocol (EAP) is specified in RFC 3748. |

4.9.6 SMTP setup (class_id = 46, version = 0)

This IC allows setting up data exchange using the SMTP protocol.

| SMTP modem setup | 0...n | class_id = 46, version = 0 | | | |
|-------------------------------|------------------|-----------------------------------|-------------|-------------|-------------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. server_port (static) | long-unsigned | | | 25 | x + 0x08 |
| 3. user_name (static) | octet-string | | | | x + 0x10 |
| 4. login_password (static) | octet-string | | | | x + 0x18 |
| 5. server_address (static) | octet-string | | | | x + 0x20 |
| 6. sender_address (static) | octet-string | | | | x + 0x28 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|-----------------------|--|
| logical_name | Identifies the “SMTP setup” object instance. See 6.2.22. |
| server_port | Defines the value of the TCP-UDP port related to this protocol. By default, this value is the SMTP port numberID assigned by IANA: - smtp 25/tcp, smtp 25/udp Simple Mail Transfer |
| user_name | Defines the user name to be used for the login to the SMTP server. |
| login_password | Password to be used for login. When the string is void, this means that there is no authentication. |
| server_address | Defines the server address as an octet string. This server address can be a name, which shall be resolvable by the primary DNS or the secondary DNS. In the case when it is directly the IP address of the server, which is specified here, it shall be a string in dotted format. EXAMPLE: 163.187.45.87. |
| sender_address | Defines the sender address as an octet string. This sender address can be a name. In the case when it is directly the IP address of the sender, which is specified here, it will be a string in dotted format. |

4.10 Interface classes for setting up data exchange using S-FSK PLC

4.10.1 General

This subclause specifies COSEM interface classes to set up and manage the protocol layers of DLMS/COSEM S-FSK PLC communication profile:

- the S-FSK Physical layer and the MAC sub-layer as defined in IEC 61334-5-1:2001 and IEC 61334-4-512:2001;
- the LLC sub-layer as specified in IEC 61334-4-32:1996.

The MIB variables / logical link parameters specified in IEC 61334-4-512:2001, IEC 61334-5-1:2001, IEC 61334-4-32:1996 and ISO/IEC 8802-2:1998 respectively have been mapped to attributes and/or methods of COSEM ICs. The specification of these elements has been taken from the above standards and the text has been adapted to the DLMS/COSEM environment.

NOTE IEC 61334-4-512:2001 also specifies some management variables to be used on the Client side. However, the Client side object model is not covered in this document.

For definitions related to S-FSK PLC profile see 3.2.

4.10.2 Overview

COSEM objects for setting up the S-FSK PLC channel and the LLC layer, if implemented, shall be located in the Management Logical Device of COSEM servers.

Figure 26 shows an example with a COSEM physical device comprising three logical devices. Each logical device shall contain a Logical Device Name (LDN) object. Each logical device contains one or more Association objects, one for each client supported.

NOTE As in this example there is more than one logical device, the mandatory Management Logical Device contains a SAP Assignment object instead of a Logical Device Name object.

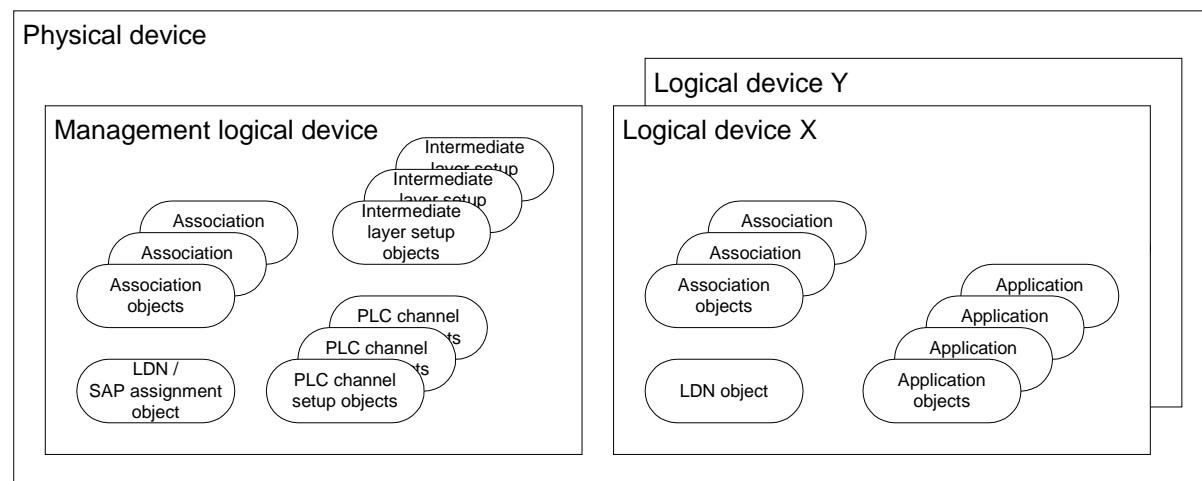


Figure 26 – Object model of DLMS/COSEM servers

The management logical device contains the setup objects of the physical and MAC layers of the PLC channel, as well as setup objects for the intermediate layer(s). It may contain further application objects.

The other logical devices, in addition to the Association and Logical Device Name objects mentioned above, contain further application objects, holding parameters and measurement values.

IEC 61334-4-512:2001 uses DLMS named variables to model the MIB objects and specifies their DLMS name in the range 8...184. For compatibility with existing implementations, the short names 8...400 [sic] are reserved for devices using the IEC 61334-5-1:2001 S-FSK PLC profiles without COSEM. Therefore, when mapping the attributes and methods of the COSEM objects specified in this document to DLMS named variables (SN mapping) this range shall not be used.

Table 32 shows the mapping of MIB variables to attributes and/or methods of COSEM ICs.

Note that on the one hand, not all MIB variables specified in IEC 61334-4-512:2001 have been mapped to attributes and methods of COSEM ICs. On the other hand, some new management variables are specified in this document.

Table 32 – Mapping IEC 61334-4-512:2001 MIB variables to COSEM IC attributes / methods

| Name | Reference (unless otherwise indicated) | Interface class | class_id / attribute / method |
|--|---|---|-------------------------------|
| S-FSK Physical layer management | | | |
| delta-electrical-phase | variable 1 | S-FSK Phy&MAC set-up (class_id = 50, version = 1) | 50 / Attr. 3 |
| max-receiving-gain | variable 2 | | 50 / Attr. 4 |
| max-transmitting-gain | – | | 50 / Attr. 5 |
| search-initiator-threshold | – | | 50 / Attr. 6 |
| frequencies | – | | 50 / Attr. 7 |
| transmission-speed | – | | 50 / Attr. 15 |
| MAC layer management | | | |
| mac-address | variable 3 | S-FSK Phy&MAC set-up (class_id = 50, version = 1) | 50 / Attr. 8 |
| mac-group-addresses | variable 4 | | 50 / Attr. 9 |
| repeater | variable 5 | | 50 / Attr. 10 |
| repeater-status | – | | 50 / Attr. 11 |
| search-initiator time-out | – | S-FSK MAC synchronization timeouts (class_id = 52, version = 0) | 52 / Attr. 2 |
| synchronization-confirmation-time-out | variable 6 | | 52 / Attr. 3 |
| time-out-not-addressed | variable 7 | | 52 / Attr. 4 |
| time-out-frame-not-OK | variable 8 | | 52 / Attr. 5 |
| min-delta-credit | variable 9 | 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 | | | |
|-------------------------------|-------------------------------|---|---------------|
| 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 |

4.10.3 S-FSK Phy&MAC set-up (class_id = 50, version = 1)

NOTE 1 The use of version 0 of this interface class is deprecated.

An instance of the “S-FSK Phy&MAC set-up” class stores the data necessary to set up and manage the physical and the MAC layer of the PLC S-FSK lower layer profile.

| S-FSK Phy&MAC setup | 0...n | class_id = 50, version = 1 | | | |
|--|------------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. initiator_electrical_phase (static) | enum | 0 | 3 | | x + 0x08 |
| 3. delta_electrical_phase (dyn.) | enum | 0 | 6 | | x + 0x10 |
| 4. max_receiving_gain (static) | unsigned | | | | x + 0x18 |
| 5. max_transmitting_gain (static) | unsigned | | | | x + 0x20 |
| 6. search_initiator_threshold (static) | unsigned | | | 98 | x + 0x28 |
| 7. frequencies (static) | frequencies_type | | | | x + 0x30 |
| 8. mac_address (dyn.) | long-unsigned | | | FFE | x + 0x38 |
| 9. mac_group_addresses (static) | array | | | | x + 0x40 |
| 10. repeater (static) | enum | | | | x + 0x48 |
| 11. repeater_status (dyn.) | boolean | | | | x + 0x50 |
| 12. min_delta_credit (dyn.) | unsigned | | | | x + 0x58 |
| 13. initiator_mac_address (dyn.) | long-unsigned | | | | x + 0x60 |
| 14. synchronization_locked (dyn.) | boolean | | | | x + 0x68 |
| 15. transmission_speed (static) | enum | 0 | 6 | 3 | x + 0x70 |
| Specific methods | m/o | | | | |

Attribute description

logical_name Identifies the “S-FSK Phy&MAC setup” object instance. See 6.2.23

initiator_electrical_phase Holds the MIB variable *initiator-electrical-phase* (variable 18) specified in IEC 61334-4-512:2001 5.8.

It is written by the client system to indicate the phase to which it is connected.

- enum:
 - (0) Not defined (default),
 - (1) Phase 1,
 - (2) Phase 2,
 - (3) Phase 3

NOTE 2 This enumeration is different from that of IEC 61334-4-512.

delta_electrical_phase Holds the MIB variable *delta-electrical-phase* (variable 1) specified in IEC 61334-4-512:2001 5.2 and IEC 61334-5-1:2001 3.5.5.3.

It indicates the phase difference between the client's connecting phase and the server's connecting phase. The following values are predefined:

enum:

- (0) Not defined: the server is temporarily not able to determine the phase difference,
- (1) The server system is connected to the same phase as the client system.

The phase difference between the server's connecting phase and the client's connecting phase is equal to:

- (2) 60 degrees,
- (3) 120 degrees,
- (4) 180 degrees,
- (5) -120 degrees,

(6) -60 degrees

| | |
|-----------------------------------|---|
| max_receiving_gain | Holds the MIB variable <i>max-receiving-gain</i> (variable 2) specified in IEC 61334-4-512:2001 5.2 and IEC 61334-5-1:2001 3.5.5.3. Corresponds to the maximum allowed gain bound to be used by the server system in the receiving mode. The default unit is dB. NOTE 3 In IEC 61334-4-512:2001, no units are specified. The possible values of the gain may depend on the hardware. Therefore, after writing a value to this attribute, the value should be read back to know the actual value. |
| max_transmitting_gain | Holds the value of the <i>max-transmitting-gain</i> . Corresponds to the maximum attenuation bound to be used by the server system in the transmitting mode. The default unit is dB. The possible values of the gain may depend on the hardware. Therefore, after writing a value to this attribute, the value should be read back to know the actual value. |
| search_initiator_threshold | This attribute is used in the intelligent search initiator process. If the value of the initiator signal is above the value of this attribute, a fast synchronization process is possible. The default value is 98 dB μ V. |
| frequencies | Contains frequencies required for S-FSK modulation. frequencies_type ::= structure { mark_frequency: double-long-unsigned, space_frequency: double-long-unsigned } The default unit is Hz. |
| mac_address | Holds the MIB variable <i>mac-address</i> (variable 3) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. NOTE 4 MAC addresses are expressed on 12 bits. Contains the value of the address of the physical attachment (MAC address) associated to the local system. In the unconfigured state, the MAC address is "NEW-address". This attribute is locally written by the CIASE when the system is registered (with a Register service). The value is used in each outgoing or incoming frame. The default value is "NEW-address". This attribute is set to NEW: <ul style="list-style-type: none">- by the MAC sub-layer, once the time-out-not-addressed delay is exceeded;- when a client system "resets" the server system. See the 4.10.4. When this attribute is set to NEW: <ul style="list-style-type: none">- the system loses its synchronization (function of the MAC-sublayer);- the <i>mac_group_address</i> attribute is reset (array of 0 elements);- the system automatically releases all AAs which can be released. NOTE 5 The second item is not present in IEC 61334-4-512:2001. The predefined MAC addresses are shown in Table 33. |

| | |
|----------------------------|--|
| mac_group_addresses | Holds the MIB variable <i>mac-group-address</i> (variable 4) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. Contains a set of MAC group addresses used for broadcast purposes. array mac-address mac-address ::= long-unsigned |
| | <p>The ALL-configured-address, ALL-physical-address and NO-BODY addresses are not included in this list. These ones are internal predefined values.</p> <p>This attribute shall be written by the initiator using DLMS services to declare specific MAC group addresses on a server system.</p> <p>This attribute is locally read by the MAC sublayer when checking the destination address field of a MAC frame not recognized as an individual address or as one of the three predefined values (ALL-configured-address, ALL-physical-address and NO-BODY).</p> |
| repeater | <p>Holds the MIB variable <i>repeater</i> (variable 5) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6.</p> <p>It specifies whether the server system effectively repeats all frames or not.</p> <p>enum: (0) never repeater, (1) always repeater, (2) dynamic repeater</p> <p>If the <i>repeater</i> variable is equal to 0, the server system should never repeat the frames.</p> <p>If it is set to 1, the server system is a repeater: it has to repeat all frames received without error and with a current credit greater than zero.</p> <p>If it is set to 2, then the repeater status can be dynamically changed by the server itself.</p> <p>NOTE 6 The value 2 value is not specified in IEC 61334-4-512.</p> <p>This attribute is internally read by the MAC sub-layer each time a frame is received.</p> <p>The default value shall be specified in project specific companion specifications.</p> |
| repeater_status | <p>Holds the current <i>repeater status</i> of the device.</p> <p>boolean: (0) FALSE = no repeater, (1) TRUE = repeater</p> |
| min_delta_credit | <p>Holds the MIB variable <i>min-delta-credit</i> (variable 9) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6.</p> <p>NOTE 7 Only the three least significant bits are used.</p> <p>The Delta Credit (DC) is the subtraction of the Initial Credit (IC) and Current Credit (CC) fields of a correct received MAC frame. The delta-credit minimum value of a correct received MAC frame, directed to a server system, is held by this variable.</p> <p>The default value is set to the maximal initial credit (see IEC 61334-5-1:2001 4.2.3.1 for further explanations on the credit and the value of MAX_INITIAL_CREDIT). A client system can reinitialise this variable by setting its value to the maximal initial credit.</p> |

| | |
|-------------------------------|---|
| initiator_mac_address | Holds the MIB variable <i>initiator-mac-address</i> specified in IEC 61334-5-1:2001, 4.3.7.6. Its value is either the MAC address of the active-initiator or the NO-BODY address, depending on the value of the <i>synchronization_locked</i> attribute (see below). See also IEC 61334-5-1:2001 3.5.3, 4.1.6.3 and 4.1.7.2. |
| synchronization_locked | Holds the MIB variable <i>synchronization-locked</i> (variable 10) specified in IEC 61334-4-512:2001, 5.3. Controls the synchronization locked / unlocked state. See IEC 61334-5-1:2001 for more details. If the value of this attribute is equal to TRUE, the system is in the synchronization-locked state. In this state, the <i>initiator-mac-address</i> is always equal to the MAC address field of the active-initiator MIB object. See attribute 2 of the S-FSK Active initiator IC in 4.10.4. If the value of this attribute is equal to FALSE, the system is in the synchronization-unlocked state. In this state, the <i>initiator_mac_address</i> attribute is always set to the NO-BODY value: a value change in the MAC address field of the active-initiator MIB object does not affect the content of the <i>initiator_mac_address</i> attribute which remains at the NO-BODY value. The default value of this variable shall be specified in the implementation specifications. |
| | NOTE 8 In the synchronization-unlocked state, the server synchronizes on any valid frame. In the synchronization locked state, the server only synchronizes on frames issued or directed to the client system the MAC address of which is equal to the value of the <i>initiator_mac_address</i> attribute. |

Table 33 – 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 | |

4.10.4 S-FSK Active initiator (class_id = 51, version = 0)

An instance of the “S-FSK Active initiator” IC stores the data of the active initiator. The active initiator is the client system, which has last registered the server system with a CIASE Register request. See IEC 61334-4-511:2000, 7.2.

| S-FSK Active initiator | 0...n | class_id = 51, version = 0 | | | |
|-------------------------------|----------------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. active_initiator (dyn.) | initiator_descriptor | | | | x + 0x08 |
| Specific methods | m/o | | | | |
| 1. reset_NEW_not_synchronized | | | | | x + 0x10 |

Attribute description

| | |
|-------------------------|--|
| logical_name | Identifies the “S-FSK Active initiator” object instance. See 6.2.23. |
| active_initiator | <p>Holds the MIB variable <i>active-initiator</i> (variable 15) specified in IEC 61334-4-512:2001, 5.6.</p> <p>Contains the identifiers of the active initiator, which has last registered the system with a Register request. See IEC 61334-4-511:2000 7.2.</p> <p>The Initiator system is identified with its System Title, MAC address and L-SAP selector:</p> <pre>initiator_descriptor ::= structure { system_title: octet-string, MAC_address: long-unsigned, L_SAP_selector: unsigned }</pre> <p>The size and the structure of the system title may be specified in system specifications. When the system title is used as part of the initialisation vector of cryptographic algorithms, then the size shall meet the requirements applicable for the initialisation vector.</p> <p>The MAC_address element is used to update the <i>initiator-mac-address</i> MAC management variable when the system is configured in the synchronization-locked state. See the specification of the <i>initiator_mac_address</i> and the <i>synchronization_locked</i> attributes of the S-FSK Phy&MAC setup IC in 4.10.3.</p> <p>As long as the server is not registered by an active initiator, the L_SAP_selector field is set to 0 and the system_title field is equal to an octet string of 0s.</p> <p>The default value of the initiator-descriptor is: system_title = octet-string of 0s, MAC_address = NO-BODY and L_SAP_selector = 0.</p> <p>The value of this attribute can be updated by the invocation of the <i>reset_NEW_not_synchronized</i> method or by the CIASE Register service.</p> |

Method description

| | |
|-----------------------------------|--|
| reset_NEW_not_synchronized | Holds the MIB variable <i>reset-NEW-not-synchronized</i> (variable 17) specified in IEC 61334-4-512:2001, 5.8. |
| (data) | <p>Allows a client system to “reset” the server system. The submitted value corresponds to a client MAC address. The writing is refused if:</p> <ul style="list-style-type: none"> - the value does not correspond to a valid client MAC address or the predefined NO-BODY address; |

- the submitted value is different from the NO-BODY address and the *synchronization_locked* attribute is not equal to TRUE.

For the description of the Intelligent Search Initiator process, see DLMS UA 1000-2 Ed. 8.1:201, 10.4.5.7.

When this method is invoked, the following actions are performed:

- the system returns to the unconfigured state (UNC: MAC-address equals NEW-address). This transition automatically causes the synchronization lost (function of the MAC sub layer);
- the system changes the value of the *active_initiator* attribute: the *MAC_address* is set to the submitted value, the *L-SAP_selector* is set to the value 0 and the *system_title* is set to an octet-string of 0s;
- all AAs that can be released are released.

4.10.5 S-FSK MAC synchronization timeouts (class_id = 52, version = 0)

An instance of the “S-FSK MAC synchronization timeouts” IC stores the timeouts related to the synchronization process.

| S-FSK MAC synchronization timeouts | 0...n | class_id = 52, version = 0 | | | |
|--|---------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. search_initiator_timeout (static) | long-unsigned | | | | x + 0x08 |
| 3. synchronization_confirmation_timeout (static) | long-unsigned | | | | x + 0x10 |
| 4. time_out_not_addressed (static) | long-unsigned | | | | x + 0x18 |
| 5. time_out_frame_not_OK (static) | long-unsigned | | | | x + 0x20 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|---------------------------------|--|
| logical_name | Identifies the “S-FSK synchronization timeouts” object instance. See 6.2.23. |
| search_initiator_timeout | <p>This timeout supports the intelligent search initiator function.</p> <p>It defines the value of the time, expressed in seconds, during which the server system is searching for the initiator with the strongest signal.</p> <p>During this timeout, all initiators, which may be heard by the servers, are expected to talk.</p> <p>After the expiry of this timeout, the server will accept a Register request from the initiator having provided the strongest signal and it will be locked to that initiator.</p> <p>If the value of the timeout is equal to 0, this means that the feature is not used.</p> <p>The timeout is started at the beginning of the Search Initiator Phase, when the server receives the first frame with a valid initiator MAC address. The timeout is restarted when the Search Initiator Phase is over and the server locks on the initiator. During the Check Initiator Phase, it is restarted on the reception of each valid frame.</p> |

| | |
|--|--|
| search_initiator_timeout (continued) | A Fast synchronization may be performed if the level of signal is good enough (Level of initiator signal >= Search-Initiator-Threshold) and one of the MAC addresses (Source or Destination) is an Initiator MAC address. This means that the module (the meter) is next to a DC or next to a module that is already locked on that DC. The module locks in this case on that initiator. |
| synchronization_confirmation_timeout | <p>Holds the MIB variable <i>synchronization-confirmation-timeout</i> (variable 6) specified in IEC 61334-4-512:2001, 5.3 and IEC 61334-5-1:2001, 4.3.7.6.</p> <p>Defines the value of the time, expressed in seconds, after which a server system which just gets frame synchronized (detection of a data path equal to AAAA54C7 hex) will automatically lose its frame synchronization if the MAC sublayer does not identify a valid MAC frame. The timeout starts after the reception of the first four bytes of a physical frame.</p> <p>The value of this variable can be modified by a client system. This time-out ensures a fast desynchronization of a system, which has synchronized on a wrong physical frame. See IEC 61334-5-1:2001, 3.5.3 for more details.</p> <p>The default value of this variable should be specified in the implementation specifications.</p> <p>A value equal to 0 is equivalent to cancel the use of the related <i>synchronization_confirmation_timeout</i> counter.</p> |
| time_out_not_addressed | <p>Holds the MIB variable <i>time-out-not-addressed</i> (variable 7) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6.</p> <p>Defines the time, in minutes, after which a server system that has not been individually addressed:</p> <ul style="list-style-type: none"> - returns to the non configured state (UNC: MAC-address equals NEW-address): this transition automatically involves the loss of the synchronization (function of the MAC sub layer) and releasing all AAs that can be released; - loses its active initiator: the MAC address of the active-initiator is set to NO-BODY, the LSAP selector is set to the value 00 and the System Title is set to an octet-string of 0s. <p>Because broadcast addresses are not individual system addresses, the timer associated with the <i>time-out-not-addressed</i> delay ensures that a forgotten system will sooner or later return to the unconfigured state. It will be then discovered again.</p> <p>A forgotten system is a system, which has not been individually addressed for more than the "<i>time-out-not-addressed</i>" amount of time.</p> <p>The default value of this variable should be specified in the implementation specifications.</p> <p>A value equal to 0 is equivalent to cancel the use of the related time-out-not-addressed counter.</p> |

| | |
|------------------------------|--|
| time_out_frame_not_OK | Holds the MIB variable <i>time-out-frame-not-OK</i> (variable 8), specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. Defines the time, in seconds, after which a server system that has not received a properly formed MAC frame (incorrect NS field, inconsistent number of received sub frames, false Cyclic Redundancy Code checking) loses its frame synchronization. The default value of this variable shall be specified in the implementation specifications. A value equal to 0 is equivalent to cancel the use of the related <i>time-out-frame-not-OK</i> counter. |
|------------------------------|--|

4.10.6 S-FSK MAC counters (class_id = 53, version = 0)

An instance of the “S-FSK MAC counters” IC stores counters related to the frame exchange, transmission and repetition phases.

| S-FSK MAC counters | 0...n | class_id = 53, version = 0 | | | |
|-------------------------------------|----------------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. synchronization_register (dyn.) | array | | | | x + 0x08 |
| 3. desynchronization_listing (dyn.) | structure | | | | x + 0x10 |
| 4. broadcast_frames_counter (dyn.) | array | | | | x + 0x18 |
| 5. repetitions_counter (dyn.) | double-long-unsigned | | | 0 | x + 0x20 |
| 6. transmissions_counter (dyn.) | double-long-unsigned | | | 0 | x + 0x28 |
| 7. CRC_OK_frames_counter (dyn.) | double-long-unsigned | | | 0 | x + 0x30 |
| 8. CRC_NOK_frames_counter (dyn.) | double-long-unsigned | | | | x + 0x38 |
| Specific methods | m/o | | | | |
| 1. reset (data) | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

Attribute description

| | |
|---------------------|--|
| logical_name | Identifies the “S-FSK MAC counters” object instance. See 6.2.23. |
|---------------------|--|

synchronization_register Holds the MIB variable *synchronization-register* (variable 23), specified in IEC 61334-4-512:2001, 5.8.
array synchronization_couples

```
synchronization_couples ::= structure
{
    mac_address:          long-unsigned,
    synchronizations_counter: double-long-unsigned
}
```

This variable counts the number of synchronization processes performed by the system. Processes that lead to a synchronization loss due to the detection of a wrong initiator are registered. The other processes that lead to a synchronization loss (time-out, management writing) **are not registered**.

This variable provides a balance sheet of the different systems on which the server system is "potentially" able to synchronize.

A synchronization process is initialized when the Management Application Entity (connection manager) receives a MA_Sync.indication (Synchronization State = SYNCHRO_FOUND) primitive from the MAC Sublayer Entity. This process is registered in the synchronization-register variable only if the MA_Sync.indication (Synchronization State = SYNCHRO_FOUND) primitive is followed by one of the three primitives:

- 1) MA_Data.indication (DA, SA, MSDU) primitive;
- 2) MA_Sync.indication (Synchronization State = SYNCHRO_CONF, SA, DA);
- 3) MA_Sync.indication (Synchronization State = SYNCHRO_LOSS, Synchro Loss Cause = wrong_initiator, SA, DA).

NOTE The third primitive is only generated if the server system is configured in a synchronization-locked state. See 4.10.3.

Processes which lead to the generation of MA_Sync.indication (Synchronization State = SYNCHRO_LOSS) primitives indicating synchronization loss due to:

- the physical layer;
- the time-out-not-addressed counter;
- setting the mac_address attribute of the S-FSK Phy&MAC setup object to NEW; see 4.10.3; or
- invoking the reset_NEW_not_synchronized method of the S-FSK Active initiator object; see 4.10.4 (this is known as Management Writing)

are not taken into account in this variable.

For details on the MA_Sync.indication service primitive, see IEC 61334-5-1:2001, 4.1.7.1.

| | |
|--|--|
| synchronization_register (continued) | <p>If the synchronization process ends with one of the three primitives listed above, the synchronization-register variable is updated by taking into account the SA and DA fields of the primitive.</p> <p>The updating of the <i>synchronization-register</i> variable is carried out as follows:</p> <p>First, the Management Entity checks the SA and DA fields.</p> <ul style="list-style-type: none"> - If one of these fields corresponds to a client MAC address (CMA) the Entity: <ul style="list-style-type: none"> - checks if the client MAC address (CMA) appears in one of the couples contained in the synchronization-register variable; - if it appears, the related synchronizations-counter subfield is incremented; - if it does not appear, a new (mac-address, synchronizations-counter) couple is added. This couple is initialized to the (CMA, 1) value. - If none of the SA and DA fields correspond to a client MAC address, it is supposed that the system found its synchronization reference on a DiscoverReport type frame. In that case, the mac-address which should be registered in the synchronization-register variable is the predefined NEW value (FFE). The updating of the synchronization-register variable is carried out in the same way as it is done for a normal client MAC address (CMA). <p>When a <i>synchronizations-counter</i> field reaches the maximum value, it automatically returns to 0 on the next increment.</p> <p>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.</p> <p>The default value of this variable is an empty array.</p> |
| desynchronization_listing | <p>Holds the MIB variable <i>desynchronization-listing</i> (variable 24), specified in IEC 61334-4-512:2001, 5.8.</p> <p>structure</p> <pre>{ nb_physical_layer_desynchronization: double-long-unsigned, nb_time_out_not_addressed_desynchronization: double-long-unsigned, nb_timeout_frame_not_OK_desynchronization: double-long-unsigned, nb_write_request_desynchronization: double-long-unsigned, nb_wrong_initiator_desynchronization: double-long-unsigned }</pre> <p>This variable counts the number of desynchronizations that occurred depending on their cause. On reception of synchronization loss notification, the Management Entity updates this attribute by incrementing the counter related to the cause of the desynchronization.</p> <p>When one of the counters reaches the maximum value, it automatically returns to 0 on the next increment.</p> <p>The default value of this variable is a structure with all elements equal to 0.</p> |

| | |
|---------------------------------|---|
| broadcast_frames_counter | Holds the MIB variable <i>broadcast-frames-counter</i> (variable 19) specified in IEC 61334-4-512:2001, 5.8. array broadcast-couples broadcast-couples ::= structure { source-mac-address: long-unsigned, frames-counter: double-long-unsigned } It counts the broadcast frames received by the server system and issued from a client system (source-mac-address = any valid client-mac-address, destination-mac address = ALL-physical). The number of frames is classified according to the origin of the transmitter. The counter is incremented even if the LLC-destination-address is not valid on the server system. When the frames-counter field reaches its maximum value, it automatically returns to 0 on the next increment. The maximum number of broadcast-couples {source-mac-address, frames-counter} contained in this variable should be specified in the implementation specifications. When this maximum is reached, the updating of the variable follows a First-In-First-Out (FIFO) mechanism: only the newest source MAC addresses are memorized. |
| repetitions_counter | Holds the MIB variable <i>repetitions-counter</i> (variable 20) specified in IEC 61334-4-512:2001, 5.8. Counts the number of repetition phases. The repetition phases following a transmission are not counted. If the MAC sub-layer is configured in the no-repeater mode, this variable is not updated. The repetitions counter measures the activity of the system as a repeater. A received frame repeated five times (from CC=4 to CC=0) is counted only once in the repetitions counter since it corresponds to one repetition phase. The counter is incremented at the beginning of each repetition phase. When the repetitions counter reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0. |
| transmissions_counter | Holds the MIB variable <i>transmissions-counter</i> (variable 21) specified in IEC 61334-4-512:2001, 5.8. Counts the number of transmission phases. A transmission phase is characterized by the transmission and the repetition of a frame. A repetition phase, which follows the reception of a frame, is not counted. The transmission counter is incremented at the beginning of each transmission phase. A client system can write this variable to update the counter. When the transmissions counter reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0. |
| CRC_OK_frames_counter | Holds the MIB variable <i>CRC-OK-frames-counter</i> (variable 22) specified in IEC 61334-4-512:2001, 5.8 Counts the number of frames received with a correct Frame Check Sequence Field. When the CRC OK frames counter field reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0. |
| CRC_NOK_frames_counter | Counts the number of frames received with an incorrect Frame Check Sequence Field. When the CRC NOK frames counter field reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0. |

Method description

reset (data) Clears all counters.
data ::= integer (0)

4.10.7 IEC 61334-4-32 LLC setup (class_id = 55, version = 1)

An instance of the “IEC 61334-4-32 LLC setup” IC holds parameters necessary to set up and manage the LLC layer as specified in IEC 61334-4-32.

| IEC 61334-4-32 LLC setup | 0...n | class_id = 55, version = 1 | | | |
|------------------------------|---------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. max_frame_length (static) | long-unsigned | | | | x + 0x08 |
| 3. reply_status_list (dyn.) | array | | | | x + 0x10 |
| Specific methods | m/o | | | | |

Attribute description

logical_name Identifies the “IEC 61334-4-32 LLC setup” object instance. See 6.2.23.

max_frame_length Holds the length of the LLC frame in bytes. See IEC 61334-4-32:1996, 5.1.4.

For the S-FSK PLC profile, the minimum / default / maximum values are 26 / 134 / 242 bytes respectively. See IEC 61334-5-1:2001, 4.2.2.

NOTE For other lower layer profiles, see the corresponding values in the relevant specification.

reply_status_list Holds the MIB variable *reply-status-list* (variable 11) specified in IEC 61334-4-512:2001, 5.4.

Lists the L-SAPs that have a not empty RDR (Reply Data on Request) buffer, which has not already been read. The length of a waiting L-SDU is specified in number of sub frames (different from zero). The variable is locally generated by the LLC sub layer.

reply_status_list array reply_status

(continued)

```
reply_status ::= structure
{
    L_SAP_selector:      unsigned,
    length_of_waiting_L_SDUs: unsigned
}
```

length_of_waiting_L_SDUs in the case of the S-FSK profile is in number of sub-frames; valid values are 1 to 7.

4.10.8 S-FSK Reporting system list (class_id = 56, version = 0)

An instance of the “S-FSK Reporting system list” IC holds the list of reporting systems.

| S-FSK Reporting system list | 0...n | class_id = 56, version = 0 | | | |
|------------------------------------|--------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. reporting_system_list (dyn.) | array | | | | x + 0x08 |
| Specific methods | m/o | | | | |

Attribute description

logical_name Identifies the “S-FSK Reporting system list” object instance. See 6.2.23.

reporting_system_list Holds the MIB variable *reporting-system-list* (variable 16) specified in IEC 61334-4-512:2001, 5.7.
array system-title

system-title ::= octet-string

Contains the system-titles of the server systems which have made a DiscoverReport request and which have not already been registered. The list has a finite size and it is sorted upon the arrival. The first element is the newest one. Once full, the oldest ones are replaced by the new ones.

The reporting system list is updated:

- when a DiscoverReport CI_PDU is received by the server system (whatever its state: non configured or configured): the CIASE adds the reporting system-title at the beginning of the list, and verifies that it does not exist anywhere else in the list, if so it destroys the old one. A system-title can only be present once in the list;
- when a Register CI_PDU is received by the server system (whatever its state: non configured or configured): the CIASE checks the reporting-system list. If a system-title is present in the reporting-system-list and in the Register CI-PDU, the CIASE deletes the system-title in the reporting-system-list: this system is no more considered as a reporting system.

4.11 Interface classes for setting up the LLC layer for ISO/IEC 8802-2

4.11.1 General

This clause specifies the ICs available for setting up the ISO/IEC 8802-2 LLC layer, used in some DLMS/COSEM communication profiles, in the various types of operation.

For definitions related to the ISO/IEEE 8802-2 LLC layer see ISO/IEC 8802-2:1998, 1.4.2.

4.11.2 ISO/IEC 8802-2 LLC Type 1 setup (class_id = 57, version = 0)

An instance of the “ISO/IEC 8802-2 LLC Type 1 setup” IC holds the parameters necessary to set up the ISO/IEC 8802-2 LLC layer in Type 1 operation.

| ISO/IEC 8802-2 LLC Type 1 setup | 0...n | class_id = 57, version = 0 | | | |
|---------------------------------|---------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. max_octets_ui_pdu (static) | long unsigned | | | 128 | x + 0x08 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|-------------------|---|
| logical_name | Identifies the “ISO/IEC 8802-2 LLC Type 1 setup” object instance. See 6.2.24. |
| max_octets_ui_pdu | Refer to the appropriate MAC protocol specification for any limitation on the maximum number of octets in a UI PDU. No restrictions are imposed by the LLC sublayer. However, in the interest of having a value that all users of Type 1 LLC may depend upon, all MACs shall at least be capable of accommodating UI PDUs with information fields up to and including 128 octets in length. See ISO/IEC 8802-2:1998, 6.8.1 <i>Maximum number of octets in a UI PDU</i> . |

4.11.3 ISO/IEC 8802-2 LLC Type 2 setup (class_id = 58, version = 0)

An instance of the “ISO/IEC 8802-2 LLC Type 2 setup” IC holds the parameters necessary to set up the ISO/IEC 8802-2 LLC layer in Type 2 operation.

| ISO/IEC 8802-2 LLC Type 2 setup | 0...n | class_id = 58, version = 0 | | | |
|---|---------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. transmit_window_size_k (static) | unsigned | 1 | 127 | 1 | x + 0x08 |
| 3. receive_window_size_rw (static) | unsigned | 1 | 127 | 1 | x + 0x10 |
| 4. max_octets_i_pdu_n1 (static) | long unsigned | | | 128 | x + 0x18 |
| 5. max_number_transmissions_n2 (static) | unsigned | | | | x + 0x20 |
| 6. acknowledgement_timer (static) | long-unsigned | | | | x + 0x28 |
| 7. p_bit_timer (static) | long-unsigned | | | | x + 0x30 |
| 8. reject_timer (static) | long-unsigned | | | | x + 0x38 |
| 9. busy_state_timer (static) | long-unsigned | | | | x + 0x40 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|------------------------------------|--|
| logical_name | Identifies the “ISO/IEC 8802-2 LLC Type 2 setup” object instance. See 6.2.24. |
| transmit_window_size_k | The transmit window size (k) shall be a data link connection parameter that can never exceed 127. It shall denote the maximum number of sequentially numbered I PDUs that the sending LLC may have outstanding (i.e., unacknowledged). The value of k is the maximum number by which the sending LLC send state variable V(S) can exceed the N(R) of the last received I PDU. See ISO/IEC 8802-2:1998, 7.8.4 <i>Transmit window size, k</i> . |
| receive_window_size_rw | The receive window size (RW) shall be a data link connection parameter that can never exceed 127. It shall denote the maximum number of unacknowledged sequentially numbered I PDUs that the local LLC allows the remote LLC to have outstanding. It is transmitted in the information field of XID (see ISO/IEC 8802-2:1998, 5.4.1.1.2) and applies to the XID sender. The XID receiver shall set its transmit window (k) to a value less than or equal to the receive window of the XID sender to avoid overrunning the XID sender. See ISO/IEC 8802-2:1998, 7.8.6 <i>Receive window size, RW</i> . |
| max_octets_i_pdu_n1 | N1 is a data link connection parameter that denotes the maximum number of octets in an I PDU. Refer to the various MAC descriptions to determine the precise value of N1 for a given medium access method. LLC itself places no restrictions on the value of N1. However, in the interest of having a value of N1 that all users of Type 2 LLC may depend upon, all MACs shall at least be capable of accommodating I PDUs with information fields up to an including 128 octets in length. See ISO/IEC 8802-2:1998, 7.8.3 <i>Maximum number of octets in an I PDU, N1</i> . |
| max_number_transmissions_n2 | N2 is a data link connection parameter that indicates the maximum number of times that a PDU is sent following the running out of the acknowledgment timer, the P-bit timer, the reject timer, or the busy-state timer. See ISO/IEC 8802-2:1998, 7.8.2 <i>Maximum number of transmissions, N2</i> . |
| acknowledgement_timer | The acknowledgment timer is a data link connection parameter that shall define the time interval during which the LLC shall expect to receive an acknowledgment to one or more outstanding I PDUs or an expected response PDU to a sent unnumbered command PDU. The unit is seconds. See ISO/IEC 8802-2:1998, 7.8.1.1 <i>Acknowledgement timer</i> . |
| p_bit_timer | The P-bit timer is a data link connection parameter that shall define the time interval during which the LLC shall expect to receive a PDU with the F bit set to “1” in response to a sent Type 2 command with the P bit set to “1”. The unit is seconds. See ISO/IEC 8802-2:1998, 7.8.1.2 <i>P-bit timer</i> . |
| reject_timer | The reject timer is a data link connection parameter that shall define the time interval during which the LLC shall expect to receive a reply to a sent REJ PDU. The unit is seconds. See ISO/IEC 8802-2:1998, 7.8.1.3 <i>Reject timer</i> . |
| busy_state_timer | The busy-state timer is a data link connection parameter that shall define the timer interval during which the LLC shall wait for an indication of the clearance of a busy condition at the other LLC. The unit is seconds. See ISO/IEC 8802-2:1998, 7.8.1.4 <i>Busy-state timer</i> . |

4.11.4 ISO/IEC 8802-2 LLC Type 3 setup (class_id = 59, version = 0)

An instance of the “ISO/IEC 8802-2 LLC Type 3 setup” IC holds the parameters necessary to set up the ISO/IEC 8802-2 LLC layer in Type 3 operation.

| ISO/IEC 8802-2 LLC Type 3 setup | 0...n | class_id = 59, version = 0 | | | |
|---|---------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | X |
| 2. max_octets_acn_pdu_n3 (static) | long unsigned | | | | x + 0x08 |
| 3. max_number_transmissions_n4 (static) | unsigned | | | | x + 0x10 |
| 4. acknowledgement_time_t1 (static) | long unsigned | | | | x + 0x18 |
| 5. receive_lifetime_var_t2 (static) | long unsigned | | | | x + 0x20 |
| 6. transmit_lifetime_var_t3 (static) | long unsigned | | | | x + 0x28 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|------------------------------------|---|
| logical_name | Identifies the “ISO/IEC 8802-2 LLC Type 3 setup” object instance. See 6.2.24. |
| max_octets_acn_pdu_n3 | N3 is a logical link parameter that denotes the maximum number of octets in an ACn command PDU. Refer to the various MAC descriptions to determine the precise value of N3 for a given medium access method. LLC places no restrictions on the value of N3. See ISO/IEC 8802-2:1998, 8.6.2 <i>Maximum number of octets in an ACn command PDU, N3</i> . |
| max_number_transmissions_n4 | N4 is a logical link parameter that indicates the maximum number of times that an ACn command PDU is sent by LLC trying to accomplish a successful information exchange. Normally, N4 is set large enough to overcome the loss of a PDU due to link error conditions. If the medium access control sublayer has its own retransmission capability, the value of N4 may be set to one so that LLC does not itself requeue a PDU to the medium access control sublayer. See ISO/IEC 8802-2:1998, 8.6.1 <i>Maximum number of transmissions, N4</i> . |
| acknowledgement_time_t1 | The acknowledgment time is a logical link parameter that determines the period of the acknowledgment timers, and as such shall define the time interval during which the LLC shall expect to receive an ACn response PDU from a specific LLC from which the LLC is awaiting a response PDU. The acknowledgment time shall take into account any delay introduced by the MAC sublayer and whether the timer is started at the beginning or at the end of the sending of the ACn command PDU by the LLC. The proper operation of the procedure shall require that the acknowledgment time be greater than the normal time between the sending of an ACn command PDU and the reception of the corresponding ACn response PDU. If the medium access control sublayer performs its own retransmissions and if the logical link parameter N4 is set to one to prevent LLC from re-queuing a PDU, then the acknowledgment time T1 may be set to infinity, making the acknowledgment timers unnecessary. The unit is seconds. Infinity is indicated by all bits set to 1. See ISO/IEC 8802-2:1998, 8.6.4, <i>Acknowledgement time, T1</i> . |

| | |
|---------------------------------|---|
| receive_lifetime_var_t2 | <p>This time value is a logical link parameter that determines the period of all of the receive variable lifetime timers. T2 shall be longer by a margin of safety than the longest possible period during which the first transmission and all retries of a single PDU may occur. The margin of safety shall take into account anything affecting LLCs perception of the arrival time of PDUs, such as LLC response time, timer resolution, and variations in the time required for the medium access control sublayer to pass received PDUs to LLC.</p> <p>If the destruction of the received state variables is not desired, the value of time T2 may be set to infinity. In this case the receive variable lifetime timer need not be implemented.</p> <p>The unit is seconds. Infinity is indicated by all bits set to 1.</p> <p>See ISO/IEC 8802-2:1998, 8.6.5 <i>Receive lifetime variable, T2</i>.</p> |
| transmit_lifetime_var_t3 | <p>This time value is a logical link parameter that determines the minimum lifetime of the transmit sequence state variables. T3 shall be longer by a margin of safety than</p> <ol style="list-style-type: none"> 1) the logical link variable T2 at stations to which ACn commands are sent; and 2) the longest possible lifetime of an ACn command-response pair. The lifetime of an ACn command-response pair shall take into account the sum of processing time, queuing delays, and transmission time for the command and response PDUs at the local and remote stations. <p>If the destruction of the transmit state variables is not desired, the value of time T3 may be set to infinity. Note, if the receive variable lifetime parameter, T2 is set to infinity at remote stations to which ACn commands are sent, then the T3 parameter shall be set to infinity at the local station.</p> <p>The unit is seconds. Infinity is indicated by all bits set to 1.</p> <p>See ISO/IEC 8802-2:1998, 8.6.6 <i>Transmit lifetime variable, T3</i>.</p> |

4.12 Interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for PRIME networks

4.12.1 Overview

See also Annex D.

COSEM objects for data exchange using narrowband OFDM PLC profile for PRIME networks, if implemented, shall be located in the Management Logical Device of COSEM servers.

Figure 27 shows an example with a COSEM physical device comprising three logical devices.

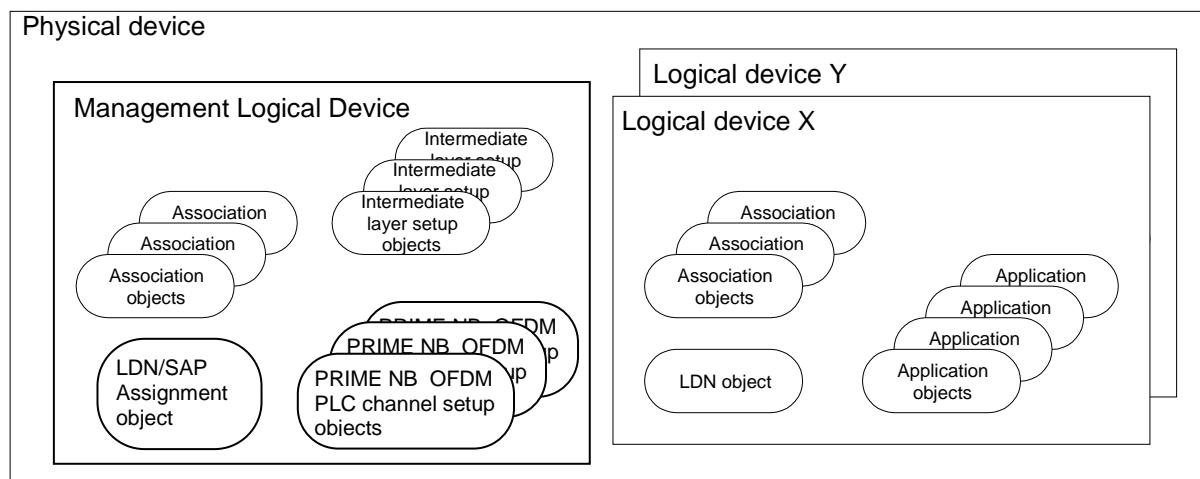


Figure 27 – Object model of DLMS/COSEM servers

Each logical device shall contain a Logical Device Name (LDN) object.

NOTE As in this example there is more than one logical device, the mandatory Management logical device contains a "SAP Assignment" object instead of a Logical Device object.

Each logical device contains one or more "Association" objects, one for each client supported.

The management logical device contains the setup objects of the physical and MAC layers of narrowband OFDM PLC profile for PRIME networks as well as setup objects for the intermediate layer(s). It may contain further application objects.

The other logical devices, in addition to the "Association" and Logical Device Name objects mentioned above, contain further application objects, holding parameters and measurement values.

To set up and manage the 61334-4-32 LLC SSCS, one IC is specified:

- “61334-4-32 LLC SSCS setup”, see 4.12.3.

To manage the PRIME NB OFDM PLC physical layer (PhL), one IC is specified:

- “PRIME NB OFDM PLC Physical layer counters”, see 4.12.5;

To set up and manage the PLC PRIME OFDM MAC layer, four ICs are specified:

- “PRIME NB OFDM PLC MAC setup”: see 4.12.6;
- “PRIME NB OFDM PLC MAC functional parameters”: see 4.12.7;
- “PRIME NB OFDM PLC MAC counters”: see 4.12.8;
- “PRIME NB OFDM PLC MAC network administration data”: see 4.12.9.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 250/492 |
|-----------------------|------------|-------------------------|---------|

For application identification, one IC is specified:

- “PRIME NB OFDM PLC Application identification”, see 4.12.11.

4.12.2 Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes

ITU-T G.9904:2012 defines variables in Table 10-1 and Table 10-2 for PHY PIB attributes, Table 10-3 to Table 10-8 for MAC PIB attributes and Table 10-9 for Applications PIB attributes.

Table 34 shows the mapping of PRIME NB OFDM PLC PIB attributes to attributes of COSEM ICs. Only variables related to the switch and Terminal nodes are mapped. Variables relevant for the base node are not mapped, because the base node acts as a client regarding the distribution network.

Table 34 – Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes

| Name | Identifier | Interface class | class_id / attribute |
|---|------------|--|----------------------|
| PHY PIB attributes – PHY read-only variable that provide statistical 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 | PRIME NB OFDM PLC MAC setup (class_id = 82, version = 0) | 82 / Attr. 2 |
| macMaxPromotionPdu | 0x0011 | | 82 / Attr. 3 |
| macMaxPromotionPduTxPeriod | 0x0012 | | 82 / Attr. 4 |
| macBeaconsPerFrame | 0x0013 | | 82 / Attr. 5 |
| macSCPMaxTxAttempts | 0x0014 | | 82 / Attr. 6 |
| macCtlReTxTimer | 0x0015 | | 82 / Attr. 7 |
| macMaxCtlReTx | 0x0018 | | 82 / Attr. 8 |
| macEMASMOOTHING | 0x0019 | | Not modelled |
| macSCPRBO | 0x0016 | | Not modelled |
| macSCPChSenseCount | 0x0017 | | Not modelled |

| MAC read-only variables that provide functional information⁴ | | | |
|--|--------|---|---------------|
| macLNID | 0x0020 | PRIME NB OFDM PLC MAC functional parameters (class_id = 83 version = 0) | 83 / Attr. 2 |
| macLSID | 0x0021 | | 83 / Attr. 3 |
| macSID | 0x0022 | | 83 / Attr. 4 |
| macSNA | 0x0023 | | 83 / Attr. 5 |
| macState | 0x0024 | | 83 / Attr. 6 |
| macSCPLength | 0x0025 | | 83 / Attr. 7 |
| macNodeHierarchyLevel | 0x0026 | | 83 / Attr. 8 |
| macBeaconSlotCount | 0x0027 | | 83 / Attr. 9 |
| macBeaconRxSlot | 0x0028 | | 83 / Attr. 10 |
| macBeaconTxSlot | 0x0029 | | 83 / Attr. 11 |
| macBeaconRxFrequency | 0x002A | | 83 / Attr. 12 |
| macBeaconTxFrequency | 0x002B | | 83 / Attr. 13 |
| macCapabilities | 0x002C | | 83 / Attr. 14 |
| MAC read-only variable that provide statistical 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 |
| 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. | | | |

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.

4.12.3 61334-4-32 LLC SSCS setup (class_id = 80, version = 0)

An instance of the “61334-4-32 LLC SSCS” (Service Specific Convergence Sublayer, 432 CL) setup IC holds addresses that are provided by the base node during the opening of the convergence layer, as a response to the establish request of the service node. They allow the service node to be part of the network managed by the base node.

| 61334-4-32 LLC 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 | | | | |

Attribute description

logical_name Identifies the “61334-4-32 LLC SSCS setup object instance”. See 6.2.25.

service_node_address Holds the value of the address assigned to the service node during its registration by the base node.
After deregistration, the value of this address is NEW, meaning 0xFFE.

base_node_address Holds the value of the base node address to which the service node is registered.
After deregistration this address is 0.

Method description

reset (data) This method is used for deallocating the service node address.
The value of the *service_node_address* becomes NEW and the value of the *base_node_address* becomes 0.

4.12.4 PRIME NB OFDM PLC Physical layer parameters

The physical layer parameters are not modelled.

4.12.5 PRIME NB OFDM PLC Physical layer counters (class_id = 81, version = 0)

An instance of the “PRIME NB OFDM PLC Physical layer counters” IC stores counters related to the physical layers exchanges. The objective of these counters is to provide statistical information for management purposes.

The attributes of instances of this IC shall be read only. They can be reset using the reset method.

| PRIME NB OFDM PLC Physical layer counters | 0...n | class_id = 81, version = 0 | | | |
|---|---------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. phy_stats_crc_incorrect_count (dyn.) | long-unsigned | | | | x + 0x08 |
| 3. phy_stats_crc_fail_count (dyn.) | long-unsigned | | | | x + 0x10 |
| 4. phy_stats_tx_drop_count (dyn.) | long-unsigned | | | | x + 0x18 |
| 5. phy_stats_rx_drop_count (dyn.) | long-unsigned | | | | x + 0x20 |
| Specific methods | m/o | | | | |
| 1. reset (data) | o | | | | |

Attribute description

| | |
|--------------------------------------|---|
| logical_name | Identifies the “PRIME NB OFDM PLC Physical layer counters” object instance. See 6.2.25. |
| phy_stats_crc_incorrect_count | PIB attribute 0x00A0: Number of bursts received on the physical layer for which the CRC was incorrect. |
| phy_stats_crc_failed_count | PIB attribute 0x00A1: Number of bursts received on the physical layer for which the CRC was correct, but the <i>Protocol</i> field of PHY header had invalid value. This count would reflect number of times corrupt data was received and the CRC calculation failed to detect it. |
| phy_stats_tx_drop_count | PIB attribute 0x00A2: Number of times when PHY layer received new data to transmit (PHY_DATA.request) and had to either overwrite on existing data in its transmit queue or drop the data in new request due to full queue. |
| phy_stats_rx_drop_count | PIB attribute 0x00A3: Number of times when the PHY layer received new data on the channel and had to either overwrite on existing data in its receive queue or drop the newly received data due to full queue. |

NOTE When a counter reaches the maximum value (0xFFFF), it is automatically rolled-over.

Method description

| | |
|---------------------|---|
| reset (data) | This method is used for resetting all the counters held by an instance of this interface. |
|---------------------|---|

4.12.6 PRIME NB OFDM PLC MAC setup (class_id = 82, version = 0)

An instance of the “PRIME NB OFDM PLC MAC setup” IC holds the necessary parameters to set up and manage the PRIME NB OFDM PLC MAC layer.

These attributes influence the functional behaviour of an implementation. These attributes may be defined external to the MAC, typically by the management entity and implementations may allow changes to their values during normal running, i.e. even after the device start-up sequence has been executed.

| PRIME NB OFDM PLC MAC setup | 0...n | class_id = 82, version = 0 | | | |
|---|--------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. mac_min_switch_search_time (static) | unsigned | 16 | 32 | 24 | x + 0x08 |
| 3. mac_max_promotion_pdu (static) | unsigned | 1 | 4 | 2 | x + 0x10 |
| 4. mac_promotion_pdu_tx_period (static) | unsigned | 2 | 8 | 5 | x + 0x18 |
| 5. mac_beacons_per_frame (static) | unsigned | 1 | 5 | 5 | x + 0x20 |
| 6. mac_scp_max_tx_attempts (static) | unsigned | 2 | 5 | 5 | x + 0x28 |
| 7. mac_ctl_re_tx_timer (static) | unsigned | 2 | 20 | 15 | x + 0x30 |
| 8. mac_max_ctl_re_tx (static) | unsigned | 3 | 5 | 3 | x + 0x38 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|------------------------------------|---|
| logical_name | Identifies the “PRIME NB OFDM PLC MAC setup” object instance. See 6.2.25. |
| mac_min_switch_search_time | PIB attribute 0x0010: Minimum time for which a service node in <i>Disconnected</i> status should scan the channel for beacons before it can broadcast PNPDU. This attribute is not maintained in base nodes. The unit of this attribute is seconds. |
| mac_max_promotion_pdu | PIB attribute 0x0011: Maximum number of PNPDUs that may be transmitted by a service node in a period of <i>mac_promotion_pdu_tx_period</i> seconds. This attribute is not maintained in base nodes. |
| mac_promotion_pdu_tx_period | PIB attribute 0x0012: Time quantum for limiting the number of PNPDUs transmitted from a service node. No more than <i>mac_max_promotion_pdu</i> may be transmitted in a period of <i>mac_promotion_pdu_tx_period</i> . The unit of this attribute is seconds. |
| mac_beacons_per_frame | PIB attribute 0x0013: Maximum number of beacon slots that may be provisioned in a frame. This attribute is maintained in base nodes. |
| mac_scp_max_tx_attempts | PIB attribute 0x0014: Number of times the CSMA algorithm would attempt to transmit requested data when a previous attempt was withheld due to PHY indicating channel busy. |
| mac_ctl_re_tx_timer | PIB attribute 0x0015: Number of seconds for which a MAC entity waits for acknowledgement of receipt of MAC control packet from its peer entity. On expiry of this time, the MAC entity may retransmit the MAC control packet. The unit of this attribute is seconds. |
| mac_max_ctl_re_tx | PIB attribute 0x0018: Maximum number of times a MAC entity will try to retransmit an unacknowledged MAC control packet. If the retransmit count reaches this maximum, the MAC entity shall abort further attempts to transmit the MAC control packet. |

NOTE When a counter reaches the maximum value (0xFFFF), it is automatically rolled-over.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 255/492 |
|-----------------------|------------|-------------------------|---------|

4.12.7 PRIME NB OFDM PLC MAC functional parameters (class_id = 83 version = 0)

The attributes of an instance of the “PRIME NB OFDM PLC MAC functional parameters” IC belong to the functional behaviour of MAC. They provide information on specific aspects.

The attributes of instances of this IC shall be read only.

| PRIME NB OFDM PLC MAC functional parameters | 0...n | class_id = 83, version = 0 | | | |
|---|---------------|----------------------------|--------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. mac_LNID (static) | long | 0 | 16 383 | | x + 0x08 |
| 3. mac_LSID (static) | unsigned | 0 | 255 | | x + 0x10 |
| 4. mac_SID (static) | unsigned | 0 | 255 | | x + 0x18 |
| 5. mac_SNA (static) | octet-string | | | | x + 0x20 |
| 6. mac_state (static) | enum | 0 | 3 | | x + 0x28 |
| 7. mac_scp_length (static) | long | | | | x + 0x30 |
| 8. mac_node_hierarchy_level (static) | unsigned | 0 | 63 | | x + 0x38 |
| 9. mac_beacon_slot_count (static) | unsigned | 0 | 7 | | x + 0x40 |
| 10. mac_beacon_rx_slot (static) | unsigned | 0 | 7 | | x + 0x48 |
| 11. mac_beacon_tx_slot (static) | unsigned | 0 | 7 | | x + 0x50 |
| 12. mac_beacon_rx_frequency (static) | unsigned | 0 | 31 | | x + 0x58 |
| 13. mac_beacon_tx_frequency (static) | unsigned | 0 | 31 | | x + 0x60 |
| 14. mac_capabilities (static) | long-unsigned | | | | x + 0x68 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|---------------------|--|
| logical_name | Identifies the “PRIME NB OFDM PLC MAC functional parameters” object instance. See 6.2.25. |
| mac_LNID | PIB attribute 0x0020: LNID allocated to this node at time of its registration. |
| mac_LSID | PIB attribute 0x0021: LSID allocated to this node at the time of its promotion. This attribute is not maintained if the node is in a <i>Terminal</i> functional state. |
| mac_SID | PIB attribute 0x0022: SID of the switch node through which this node is connected to the subnetwork. This attribute is not maintained in a base node. |
| mac_SNA | PIB attribute 0x0023: Subnetwork address to which this node is registered. The base node returns the SNA it is using. |

| | |
|---------------------------------|--|
| mac_state | PIB attribute 0x0024: Present functional state of the node. enum: (0) Disconnected, (1) Terminal, (2) Switch, (3) Base |
| mac_scp_length | PIB attribute 0x0025: The SCP length, in symbols, in present frame. |
| mac_node_hierarchy_level | PIB attribute 0x0026: Level of this node in subnetwork hierarchy. |
| mac_beacon_slot_count | PIB attribute 0x0027: Number of beacon slots provisioned in present frame structure. |
| mac_beacon_rx_slot | PIB attribute 0x0028: Beacon slot in which this device's switch node transmits its beacon. This attribute is not maintained in a base node. |
| mac_beacon_tx_slot | PIB attribute 0x0029: Beacon slot in which this device transmits its beacon. This attribute is not maintained in service nodes that are in a <i>Terminal</i> functional state. |
| mac_beacon_rx_frequency | PIB attribute 0x002A: Number of frames between receptions of two successive beacons. A value of 0x0 indicates beacons are received in every frame. This attribute is not maintained in a base node. |
| mac_beacon_tx_frequency | PIB attribute 0x002B: Number of frames between transmissions of two successive beacons. A value of 0x0 indicates beacons are transmitted in every frame. This attribute is not maintained in service nodes that are in a <i>Terminal</i> functional state. |
| mac_capabilities | PIB attribute 0x002C: This attribute defines the capabilities of the node. It is a bitmap each bit defining a capability. Bit 0: Switch Capable Bit 1: Packet Aggregation Bit 2: Contention Free Period Bit 3: Direct connection Bit 4: Multicast Bit 5: PHY Robustness Management Bit 6: ARQ Bit 7; Reserved for future use Bit 8: Direct Connection Switching Bit 9: Multicast Switching Capability Bit 10: PHY Robustness Management Switching Capability Bit 11: ARQ Buffering Switching Capability Bit 12 to 15: Reserved for future use |

4.12.8 PRIME NB OFDM PLC MAC counters (class_id = 84, version = 0)

An instance of the “PRIME NB OFDM PLC MAC counters” IC stores statistical information on the operation of the MAC layer for management purposes. The attributes of instances of this IC shall be read only. They can be reset using the reset method.

| PRIME NB OFDM PLC MAC counters | | 0...n | class_id = 84, version = 0 | | | |
|--------------------------------|----------|----------------------|----------------------------|---------------|------|------------|
| Attributes | | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name | (static) | octet-string | | | | x |
| 2. mac_tx_data_pkt_count | (dyn.) | double-long-unsigned | | 4 294 967 295 | | x + 0x08 |
| 3. mac_rx_data_pkt_count | (dyn.) | double-long-unsigned | | 4 294 967 295 | | x + 0x10 |
| 4. mac_tx_ctrl_pkt_count | (dyn.) | double-long-unsigned | | 4 294 967 295 | | x + 0x18 |
| 5. mac_rx_ctrl_pkt_count | (dyn.) | double-long-unsigned | | 4 294 967 295 | | x + 0x20 |
| 6. mac_csma_fail_count | (dyn.) | double-long-unsigned | | 4 294 967 295 | | x + 0x28 |
| 7. mac_csma_ch_busy_count | (dyn.) | double-long-unsigned | | 4 294 967 295 | | x + 0x30 |
| Specific methods | | m/o | | | | |
| 1. reset (data) | | o | | | | |

Attribute description

| | |
|-------------------------------|--|
| logical_name | Identifies the “PRIME NB OFDM PLC MAC counters” object instance. See 6.2.25. |
| mac_tx_data_pkt_count | PIB attribute 0x0040: Count of successfully transmitted MSDUs. |
| mac_rx_data_pkt_count | PIB attribute 0x0041: Count of successfully received MSDUs whose destination address was this node. |
| mac_tx_ctrl_pkt_count | PIB attribute 0x0042: Count of successfully transmitted MAC control packets. |
| mac_rx_ctrl_pkt_count | PIB attribute 0x0043: Count of successfully received MAC control packets whose destination was this node. |
| mac_csma_fail_count | PIB attribute 0x0044: Count of failed CSMA transmit attempts. |
| mac_csma_ch_busy_count | PIB attribute 0x0045: Count of number of times this node has to back off SCP transmission due to channel busy state. |

NOTE When a counter reaches the maximum value (0xFFFFFFFF), it is automatically rolled-over.

Method description

| | |
|---------------------|---|
| reset (data) | This method is used for resetting all the counters held by an instance of this interface class. data ::= integer (0) |
|---------------------|---|

4.12.9 PRIME NB OFDM PLC MAC network administration data (class_id = 85, version = 0)

This IC holds the parameters related to the management of the devices connected to the network.

| PRIME NB OFDM PLC MAC network administration data | 0...n | class_id = 85, version = 0 | | | |
|---|--------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. mac_list_multicast_entries (dyn.) | array | | | | x + 0x08 |
| 3. mac_list_switch_table (dyn.) | array | | | | x + 0x10 |
| 4. mac_list_direct_table (dyn.) | array | | | | x + 0x18 |
| 5. mac_list_available_switches (dyn.) | array | | | | x + 0x20 |
| 6. mac_list_phy_comm (dyn) | array | | | | x + 0x28 |
| Specific methods | m/o | | | | |
| 1. reset (data) | o | | | | |
| | | | | | |
| | | | | | |

Attribute description

| | |
|-----------------------------------|---|
| logical_name | Identifies the “PRIME NB OFDM PLC MAC network administration data” object instance. See 6.2.25. |
| mac_list_multicast_entries | <p>PIB attribute 0x0052: List of entries in multicast switching table. This list is not maintained in service nodes in a <i>Terminal</i> functional state.</p> <p>mac_list_multicast_entries_type ::= array mac_list_multicast_entries_element</p> <pre>mac_list_multicast_entries_element ::= structure { mcast_entry_LCID : integer, -- LCID of multicast group mcast_entry_members: long -- number of child nodes }</pre> <p>The number of child nodes is the number of the members of this group, including the Node itself.</p> |
| mac_list_switch_table | <p>PIB attribute 0x0053: Switch table. This table is not maintained by service nodes in a <i>Terminal</i> state.</p> <p>mac switch_table ::= array stbl_entry_LSID</p> <p>stbl_entry_LSID ::= long -- SID of attached Switch node</p> |
| mac_list_direct_table | <p>PIB attribute 0x0055: Direct table.</p> <p>mac_direct_table ::= array mac_direct_table_element</p> <pre>mac_direct_table_element ::= structure { dconn_entry_src_SID: long, dconn_entry_src_LNID: long, dconn_entry_src_LCID: long, dconn_entry_dst_SID: long, dconn_entry_dst_LNID: long, dconn_entry_dst_LCID: long, dconn_entry_DID: octet-string (size 6 bytes) }</pre> |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 259/492 |
|-----------------------|------------|-------------------------|---------|

| | |
|---|--|
| mac_list_direct_table (continued) | Where: <ul style="list-style-type: none"> - dconn_entry_src_SID is the SID of switch through which the source service node is connected; - dconn_entry_src_LNID is the NID allocated to the source service node; - dconn_entry_src_LCID is the LCID allocated to this connection at the source; - dconn_entry_dst_SID is the SID of the switch through which the destination service node is connected; - dconn_entry_dst_LNID is the NID allocated to the destination service node; - dconn_entry_dst_LCID is the LCID allocated to this connection at the destination; - dconn_entry_DID is the EUI-48 of the direct switch. |
| mac_list_available_switches | PIB attribute 0x0056: List of switch nodes whose beacons are received. <code>mac_list_available_switches ::= array mac_list_available_switches_element</code> <pre>mac_list_available_switches_element ::= structure { slist_entry_SNA: octet-string (size 6 bytes), slist_entry_LSID: long, slist_entry_level: integer, slist_entry_rx_level: integer, slist_entry_rx_snr: integer }</pre> <p>Where:</p> <ul style="list-style-type: none"> - slist_entry_SNA is EUI-48 of the subnetwork; - slist_entry_LSID is SID of this switch; - slist_entry_level is level of this switch in subnetwork hierarchy; - slist_entry_rx_level is the received signal level for this Switch; - slist_entry_rx_snr is the signal to noise ratio for this switch. |
| mac_list_phy_comm | PIB attribute 0x0057: List of PHY communication parameters. It is maintained in every node. For terminal nodes it contains only one entry for the switch the node is connected through. For other nodes is contains also entries for every directly connected child node. <code>mac_list_phy_comm ::= array phy_comm_element</code> |

```

mac_list_      phy_comm_element ::= structure
phy_comm        {
(continued)      phy_Comm_EUI:          octet-string,
                  phy_Comm_Tx_Pwr:    integer,
                  phy_Comm_Tx_Cod:    integer,
                  phy_Comm_Rx_Cod:    integer,
                  phy_Comm_Rx_Lvl:    integer,
                  phy_Comm_SNR:       integer,
                  phy_Comm_Tx_Pwr_Mod: integer,
                  phy_Comm_Tx_Cod_Mod: integer,
                  phy_Comm_Rx_Cod_Mod: integer
}

```

Where:

- `phy_Comm_EUI` is the EUI-48 of the other device;
- `phy_Comm_Tx_Pwr` is the Tx power of GPDU packets sent to the device;
- `phy_Comm_Tx_Cod` is the Tx coding of GPDU packets sent to the device;
- `phy_Comm_Rx_Cod` is the Rx coding of GPDU packets received from the device;
- `phy_Comm_Rx_Lvl` is the Rx power level of GPDU packets received from the device;
- `phy_Comm_SNR` is the SNR of GPDU packets received from the device;
- `phy_Comm_Tx_Pwr_Mod` is the number of times the Tx power was modified;
- `phy_Comm_Tx_Cod_Mod` is the number of times the Tx coding was modified;
- `phy_Comm_Rx_Cod_Mod` is the number of times the Rx coding was modified.

Method description

| | |
|---------------------|---|
| reset (data) | This method is used for resetting all the entries (to an array of 0 elements) of the attributes 2 to 6 of the instance of this interface class. data ::= integer (0) |
|---------------------|---|

4.12.10 PRIME NB OFDM PLC MAC address setup (class_id = 43, version = 0)

An instance of the MAC address setup IC holds the EUI-48 MAC address of the device. The size of this octet string is 6 due to the fact that this address is a EUI-48 and is unique. See also 4.9.4 and 6.2.25.

4.12.11 PRIME NB OFDM PLC Application identification (class_id = 86, version = 0)

An instance of the “PRIME NB OFDM PLC Application identification IC” holds identification information related to administration and maintenance of PRIME NB OFDM PLC devices. They are not communication parameters but allow the device management.

| PRIME NB OFDM PLC Application identification | 0...n | class_id = 86, version = 0 | | | |
|--|---------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. firmware_version (static) | octet-string | | 128 | | x + 0x08 |
| 3. vendor_Id (static) | long-unsigned | | | | x + 0x10 |
| 4. product_Id (static) | long-unsigned | | | | x + 0x18 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|-------------------------|--|
| logical_name | Identifies the device setup object instance. See 6.2.25. |
| firmware_version | PIB attribute 0x0075: Textual description of the firmware version running on the device. |
| vendor_Id | PIB attribute 0x0076: Unique vendor identifier assigned by PRIME Alliance. |
| product_Id | PIB attribute 0x0077: Vendor assigned unique identifier for specific product. |

4.13 Interface classes for setting up and managing the DLMS/COSEM narrowband OFDM PLC profile for G3-PLC networks

4.13.1 Overview

This subclause 4.13 specifies interface classes for setting up and managing the MAC and 6LoWPAN Adaptation layers of the DLMS/COSEM G3-PLC profile, based on ITU-T G.9903:2014.

NOTE 1 The use of version 0 of these interface classes based on ITU-T G.9903 Amd. 1:2013 – see 5.4.21 to 5.4.23 – is deprecated.

For this purpose, the elements of the PAN Information Base (PIB) have been mapped to attributes of COSEM ICs.

COSEM objects for data exchange using G3-PLC, if implemented, shall be located in the Management Logical Device of COSEM servers.

To set up and manage the DLMS/COSEM G3-PLC profile layers (including PHY, IEEE 802.15.4 MAC and 6LoWPAN), three ICs are specified:

- “G3-PLC MAC layer counters”, see 4.13.3;
- “G3-PLC MAC setup”, see 4.13.4;
- “G3-PLC 6LoWPAN adaptation layer setup”, see 4.13.5.

An instance of the existing COSEM interface class “MAC address” (class_id = 43, version = 0) is needed to indicate the EUI-48 MAC address of the G3-PLC modem (corresponding to aExtendedAddress constant in IEEE 802.15.4).

IPv6 configuration is provided by an instance of “IPv6 setup” class.

NOTE 2 The PHY layer of ITU-T G.9903:2014 is out of scope of the G3-PLC setup ICs.

4.13.2 Mapping of G3-PLC PIB attributes to COSEM IC attributes

In terms of IEEE 802.15.4, a meter is a Reduced Function Device (RFD) while a concentrator / Neighbourhood Network Access Point (NNAP) is a Full Function Device (FFD) / PAN coordinator. In terms of DLMS/COSEM the meter is the server and the concentrator / NNAP is the client (or an agent for a client).

As COSEM models only the server and not the client, the G3-PLC setup classes concern only the RFD (Reduced Function Device) and not the PAN coordinator.

Table 35 shows the mapping of G3-PLC PIB attributes to attributes of COSEM interface classes.

Table 35 – 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 & read-write & write only variables ¹² | | | |
| 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 & read-write variables ^{3,4} | | | |
| adp_max_hops | 0x0F | G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 1) | 92 / Att. 2 |
| adp_weak_LQI_value | 0x1A | | 92 / Att. 3 |
| adp_security_level | 0x00 | | 92 / Att. 4 |
| adp_prefix_table | 0x01 | | 92 / Att. 5 |
| adp_routing_configuration | 0x09, 0x0A, 0x0D, 0x11-0x19, 0x1B, 0x1F | | 92 / Att. 6 |
| adp_broadcast_log_table_entry_TTL | 0x02 | | 92 / Att. 7 |
| adp_routing_table | 0x0C | | 92 / Att. 8 |
| adp_context_information_table | 0x07 | | 92 / Att. 9 |
| adp_blacklist_table | 0x1E | | 92 / Att. 10 |
| adp_broadcast_log_table | 0x0B | | 92 / Att. 11 |
| adp_group_table | 0x0E | | 92 / Att. 12 |
| adp_max_join_wait_time | 0x20 | | 92 / Att. 13 |
| adp_path_discovery_time | 0x21 | | 92 / Att. 14 |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 264/492 |
|-----------------------|------------|-------------------------|---------|

| Name | Identifier | Interface class | class_id / attribute | |
|--|------------|-----------------|----------------------|--|
| adp_active_key_index | 0x22 | | 92 / Att. 15 | |
| adp_metric_type | 0x03 | | 92 / Att. 16 | |
| adp_coord_short_address | 0x08 | | 92 / Att. 17 | |
| adp_disable_default_routing | 0xF0 | | 92 / Att. 18 | |
| adp_device_type | 0x10 | | 92 / Att. 19 | |
| ¹⁾ See ITU-T G.9903:2014 9.3.6.2.2 and 9.3.6.2.3. | | | | |
| ²⁾ The following attributes of the G3-PLC MAC sublayer IB attributes have been excluded as there is no need to expose them: <i>macBSN</i> , <i>macDSN</i> , <i>macAckWaitDuration</i> , <i>macFreqNotching</i> , <i>macTimeStampSupported</i> , <i>macPromiscuousMode</i> , <i>macSecurityEnabled</i> . | | | | |
| ³⁾ See ITU-T G.9903:2014 9.4.1.1. | | | | |
| ⁴⁾ The following attributes of the G3-PLC Adaptation sublayer IB attributes have been excluded as there is no need to expose them ; <i>adpSoftVersion</i> , <i>adpSnifferMode</i> . | | | | |
| NOTE Whereas in ITU-T G.9903:2014 the camel-case notation is used, in COSEM interface class specifications – and in this table – the underscore notation is used. | | | | |

4.13.3 G3-PLC MAC layer counters (class_id = 90, version = 1)

An instance of the “G3-PLC MAC layer counters” IC stores counters related to the MAC layer exchanges. The objective of these counters is to provide statistical information for management purposes.

The attributes of instances of this IC shall be read only. They can be reset using the reset method.

| G3-PLC MAC layer counters | 0...n | class_id = 90, version = 1 | | | |
|--|----------------------|----------------------------|---------------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. mac_Tx_data_packet_count (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x08 |
| 3. mac_Rx_data_packet_count (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x10 |
| 4. mac_Tx_cmd_packet_count (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x18 |
| 5. mac_Rx_cmd_packet_count (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x20 |
| 6. mac_CSMA_fail_count (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x28 |
| 7. mac_CSMA_no_ACK_count (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x30 |
| 8. mac_bad_CRC_count (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x38 |
| 9. mac_Tx_data_broadcast_count (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x40 |
| 10. mac_Rx_data_broadcast_count (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x48 |
| Specific methods | m/o | | | | |
| 1. reset (data) | O | | | | x + 0x50 |

Attribute description

NOTE When a counter reaches the maximum value (0xFFFFFFFF), it's automatically rolled-over.

| | |
|---------------------------------|---|
| logical_name | Identifies the “G3-PLC MAC layer counters” object instance. See 6.2.26. |
| mac_Tx_data_packet_count | PIB attribute 0x0101: Statistic counter of successfully transmitted data packets (MSDUs). |
| mac_Rx_data_packet_count | PIB attribute 0x0102: Statistic counter of successfully received data packets (MSDUs). |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 265/492 |
|-----------------------|------------|-------------------------|---------|

| | |
|------------------------------------|---|
| mac_Tx_cmd_packet_count | PIB attribute 0x0103: Statistic counter of successfully transmitted command packets. |
| mac_Rx_cmd_packet_count | PIB attribute 0x0104: Statistic counter of successfully received command packets. |
| mac_CSMA_fail_count | PIB attribute 0x0105: Counts the number of times when CSMA backoffs reach macMaxCSMABackoffs. |
| mac_CSMA_no_ACK_count | PIB attribute 0x0106: Counts the number of times when an ACK is not received while transmitting a unicast data frame (The loss of ACK is attributed to collisions). |
| mac_bad_CRC_count | PIB attribute 0x0109: Statistic counter of the number of frames received with bad CRC. |
| mac_Tx_data_broadcast_count | PIB attribute 0x0108: Statistic counter of the number of broadcast frames sent. |
| mac_Rx_data_broadcast_count | PIB attribute 0x0107: Statistic counter of successfully received broadcast packets. |

Method description

| | |
|---------------------|--|
| reset (data) | This method forces a reset of the object. By invoking this method, the value of all counters is set to 0. data::= integer (0) |
|---------------------|--|

4.13.4 G3-PLC MAC setup (class_id = 91, version = 1)

An instance of the “G3-PLC MAC setup” IC holds the necessary parameters to set up and manage the G3-PLC IEEE 802.15.4 MAC sub-layer.

These attributes influence the functional behaviour of an implementation. Implementations may allow changes to the attributes during normal running, i.e. even after the device start-up sequence has been executed.

| G3-PLC MAC setup | | 0...n | class_id = 91, version = 1 | | | |
|--|----------|----------------------|-----------------------------------|---------------|--------------------------|-------------------|
| Attributes | | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name | (static) | octet-string | | | | x |
| 2. mac_short_address | (dyn.) | long-unsigned | 0x0000 | 0xFFFF | 0xFFFF | x + 0x08 |
| 3. mac_RC_coord | (dyn.) | long-unsigned | 0x0000 | 0xFFFF | 0xFFFF | x + 0x10 |
| 4. mac_PAN_id | (dyn.) | long-unsigned | 0x0000 | 0xFFFF | 0xFFFF | x + 0x18 |
| 5. mac_key_table | (dyn.) | array | | | | x + 0x20 |
| 6. mac_frame_counter | (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x28 |
| 7. mac_tone_mask | (static) | bit-string | | | 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 |

Attribute description

| | |
|--------------------------|--|
| logical_name | Identifies the “G3-PLC MAC setup” object instance. See 6.2.26. |
| mac_short_address | PIB attribute 0x0053: The 16-bit address the device is using to communicate through the PAN. Its value shall be equal to 0xFFFF when the device does not have a short address. An associated device necessarily has a short address, so that a device cannot be in the state where it is associated but does not have a short address. |
| mac_RC_coord | PIB attribute 0x010F: Route cost to coordinator, to be used in the beacon payload as RC_COORD |
| mac_PAN_id | PIB attribute 0x0050: The 16-bit identifier of the PAN through which the device is operating. A value equal to 0xFFFF indicates that the device is not associated. |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 267/492 |
|-----------------------|------------|-------------------------|---------|

| | |
|--------------------------------------|---|
| mac_key_table | PIB attribute 0x0071: This attribute holds GMK keys required for MAC layer ciphering. The attribute can hold up to two 16-bytes keys. The Key Identifier value must be different for each key. For security reason, the key entries cannot be read, only written. array mac_GMK mac_GMK ::= structure { key_id: unsigned, key: octet-string } |
| | key_id The Key Identifier used to refer to this key, can take the value 0 or 1. |
| | key The AES-128 key used for ciphering the frames exchanged at MAC layer. |
| mac_frame_counter | PIB attribute 0x0077: The outgoing frame counter for this device, used when ciphering frames at MAC layer. |
| mac_tone_mask | PIB attribute 0x0110: Defines the tone mask to use during symbol formation. |
| mac_TMR_TTL | PIB attribute 0x010D: Maximum time to live of tone map parameters entry in the neighbour table in minutes. |
| mac_max_frame_retries | PIB attribute 0x0059: Maximum number of retransmissions. |
| mac_neighbour_table_entry_TTL | PIB attribute 0x010E: Maximum time to live for an entry in the neighbour table in minutes. |
| mac_neighbour_table | PIB attribute 0x010A: See ITU-T G.9903:2014 9.3.7.2 for CENELEC and FCC bands. The neighbour table contains information about all the devices within the POS of the device. One element of the table represents one PLC direct neighbour of the device. array neighbour_table neighbour_table ::= structure { short_address: long-unsigned, payload_modulation_scheme: boolean, tone_map: bit-string, modulation: enum, tx_gain: integer, tx_res: enum, tx_coeff: bit-string, lqi: unsigned, phase_differential: integer, TMR_valid_time: unsigned, neighbour_valid_time: unsigned } NOTE 1 This table is actualized each time any frame is received from a neighbour device, and each time a Tone Map Response is received. |
| short_address | The MAC Short Address of the node which this entry refers to. |

| | |
|--|---|
| <code>payload_modulation_scheme</code> | Payload Modulation scheme to be used when transmitting to this neighbour. FALSE: Differential, TRUE: Coherent |
| <code>tone_map</code> | 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. |
| <code>modulation</code> | 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. |
| <code>tx_gain</code> | Defines the Tx Gain to use to transmit frames to that device. |
| <code>tx_res</code> | Defines the Tx Gain resolution corresponding to one gain step. 0 : 6 dB, 1 : 3 dB |
| <code>tx_coeff</code> | 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. |
| mac_high_priority_window_size | PIB attribute 0x0100: The high priority contention window size in number of slots. |
| mac_CSMA_fairness_limit | PIB attribute 0x010C: Channel access fairness limit. Specifies how many failed back-off attempts, back-off exponent is set to minBE. This attribute can take a value between $2 \times (\text{macMaxBE} - \text{macMinBE})$ and 255. |
| mac_beacon_randomization_window_length | PIB attribute 0x0111: Duration time in seconds for the beacon randomization. |
| mac_A | PIB attribute 0x0112: This parameter controls the adaptive CW linear decrease. |
| mac_K | PIB attribute 0x0113: Rate adaptation factor for channel access fairness limit. This attribute can take a value between 1 and macCSMFAccuracyLimit. |
| mac_min_CW_attempts | PIB attribute 0x0114: Number of consecutive attempts while using minimum CW. |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 270/492 |
|-----------------------|------------|-------------------------|---------|

| | |
|--------------------------------|---|
| 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): <ul style="list-style-type: none">- 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): <ul style="list-style-type: none">- Full Block interleaving;- Interleaver parameters n_i and n_j are swapped when $I(i,j) = 0$. |
| mac_FCC_legacy_mode | PIB attribute 0x0116: This read only attribute indicates the capability of the node. 0: The following configuration is used (legacy mode): <ul style="list-style-type: none">- 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): <ul style="list-style-type: none">- Coherent FCH modulation;- Full Block interleaving;- Interleaver parameters n_i and n_j are swapped when $I(i,j) = 0$;- Two RS blocks. |
| mac_max_BE | PIB attribute 0x0047: Maximum value of backoff exponent. It should always be greater than macMinBE. |
| mac_max_CSMA_backoffs | PIB attribute 0x004E: Maximum number of backoff attempts. |
| mac_min_BE | PIB attribute 0x004F: Minimum value of backoff exponent. |

Method description

| | |
|---|---|
| mac_get_neighbour_table_entry (data) | This method is used to retrieve the mac neighbour table for one MAC short address. It may be used to perform topology monitoring by the client. The method invocation parameter contains a mac_short_address. data ::= long-unsigned The response parameter includes the neighbour table for this mac_short_address. data ::= array neighbour_table where mac_neighbour_table is as defined in the <i>mac_neighbour_table</i> attribute of the present IC. |
|---|---|

4.13.5 G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 1)

An instance of the “G3-PLC 6LoWPAN adaptation layer setup” IC holds the necessary parameters to set up and manage the G3-PLC 6LoWPAN Adaptation layer.

These attributes influence the functional behaviour of an implementation. Implementations may allow changes to their values during normal running, i.e. even after the device start-up sequence has been executed.

| G3-PLC 6LoWPAN adaptation layer setup | | 0...n | class_id = 92, version = 1 | | | |
|---------------------------------------|----------|---------------|----------------------------|--------|--------|------------|
| Attributes | | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name | (static) | octet-string | | | | x |
| 2. adp_max_hops | (static) | unsigned | 1 | 14 | 8 | x + 0x08 |
| 3. adp_weak_LQI_value | (static) | unsigned | 0 | 255 | 52 | x + 0x10 |
| 4. adp_security_level | (static) | unsigned | 0 | 7 | 5 | x + 0x18 |
| 5. adp_prefix_table | (dyn.) | array | | | | x + 0x20 |
| 6. adp_routing_configuration | (static) | array | | | | x + 0x28 |
| 7. adp_broadcast_log_table_entry_TTL | (static) | long-unsigned | 0 | 65535 | 2 | x + 0x30 |
| 8. adp_routing_table | (dyn.) | array | | | | x + 0x38 |
| 9. adp_context_information_table | (dyn.) | array | | | | x + 0x40 |
| 10. adp_blacklist_table | (dyn.) | array | | | | x + 0x48 |
| 11. adp_broadcast_log_table | (dyn.) | array | | | | x + 0x50 |
| 12. adp_group_table | (dyn.) | array | | | | x + 0x58 |
| 13. adp_max_join_wait_time | (static) | long-unsigned | 0 | 1 023 | 20 | x + 0x60 |
| 14. adp_path_discovery_time | (static) | unsigned | 0 | 255 | 40 | x + 0x68 |
| 15. adp_active_key_index | (static) | unsigned | 0 | 1 | 0 | x + 0x70 |
| 16. adp_metric_type | (static) | unsigned | 0x00 | 0x0F | 0x0F | x + 0x78 |
| 17. adp_coord_short_address | (static) | long-unsigned | 0x0000 | 0xFFFF | 0x0000 | x + 0x80 |
| 18. adp_disable_default_routing | (static) | boolean | | | FALSE | x + 0x88 |
| 19. adp_device_type | (static) | enum | 0 | 2 | 2 | x + 0x90 |
| Specific methods | | m/o | | | | |

Attribute description

| | |
|---------------------------|--|
| logical_name | Identifies the “G3-PLC OFDM 6LoWPAN adaptation layer setup” object instance. See 6.2.26. |
| adp_max_hops | PIB attribute 0x0F: Defines the maximum number of hops to be used by the routing algorithm. |
| adp_weak_LQI_value | PIB attribute 0x1A: The weak link value defines the LQI value below which a link to a neighbour is considered as a weak link. A value of 52 represents an SNR of 3 dB. |
| adp_security_level | PIB attribute 0x00: The minimum security level to be used for incoming and outgoing adaptation frames. Only values 0 (no ciphering) and 5 (ciphering with 32 bits integrity code) are supported. |
| adp_prefix_table | PIB attribute 0x01: Contains the list of prefixes defined on this PAN. |

adp_routing_configuration The routing configuration element specifies all parameters linked to the routing mechanism described in ITU-T G.9903:2014. The elements are specified in 9.4.1.2 of that Recommendation.

NOTE 1 The Link cost calculation is provided in ITU-T G.9903:2014 Annex B.

| | |
|--|---|
| <pre>array routing_configuration routing_configuration::= structure { adp_net_traversal_time: unsigned, adp_routing_table_entry_TTL: long-unsigned, adp_Kr: unsigned, adp_Km: unsigned, adp_Kc: unsigned, adp_Kq: unsigned, adp_Kh: unsigned, adp_Krt: unsigned, adp_RREQ_retries: unsigned, adp_RREQ_RERR_wait: unsigned, adp_blacklist_table_entry_TTL: long-unsigned, adp_unicast_RREQ_gen_enable: boolean, adp_RLC_Enabled: boolean, adp_add_rev_link_cost: unsigned }</pre> | |
| adp_net_traversal_time | PIB attribute 0x11: Maximum time that a packet is expected to take to reach any node from any node in seconds. Range : 0-255 Default value : 20 |
| adp_routing_table_entry_TTL | PIB attribute 0x12: Maximum time-to-live of a routing table entry (in minutes). Range : 0-65 535 Default value : 60 |
| adp_Kr | PIB attribute 0x13: A weight factor for the Robust Mode to calculate link cost. Range : 0-31 Default value : 0 |
| adp_Km | PIB attribute 0x14: A weight factor for modulation to calculate link cost. Range : 0-31 Default value : 0 |
| adp_Kc | PIB attribute 0x15: A weight factor for number of active tones to calculate link cost. Range : 0-31 Default value : 0 |
| adp_Kq | PIB attribute 0x16: A weight factor for LQI to calculate route cost. Range : 0-50 |

| | | |
|-----------------------------------|---|--|
| | | Default value : 10 for CENELEC A band / 40 for FCC band. |
| adp_Kh | PIB attribute 0x17: A weight factor for hop to calculate link cost. Range : 0-31 Default value : 4 for CENELEC A band / 2 for FCC band. | |
| adp_Krt | PIB attribute 0x1B: A weight factor for the number of active routes in the routing table to calculate link cost. Range : 0-31 Default value : 0 | |
| adp_RREQ_retries | PIB attribute 0x18: The number of RREQ retransmission in case of RREP reception time out. Range : 0-255 Default value : 0 | |
| adp_RREQ_RERR_wait | PIB attribute 0x19: The number of seconds to wait between two consecutive RREQ – RERR generations. Range : 0-255 Default value : 30 | |
| adp_blacklist_table_entry_TTL | PIB attribute 0x1F: Maximum time-to-live of a blacklisted neighbour entry (in minutes). Range : 0-65 535 Default value : 10 | |
| adp_unicast_RREQ_gen_enable | PIB attribute 0x0D: If TRUE, the RREQ shall be generated with its "unicast RREQ" flag set to '1'. If FALSE, the RREQ shall be generated with its "unicast RREQ" flag set to '0'. Default value : TRUE | |
| adp_RLC_Enabled | PIB attribute 0x09: Enable the sending of RLCREQ frame by the device. Default value : FALSE | |
| adp_add_rev_link_cost | PIB attribute 0x0A: It represents an additional cost to take into account a possible asymmetry in the link. Range : 0-255 Default value : 0 | |
| adp_broadcast_log_table_entry_TTL | PIB attribute 0x02: Maximum time to live of an adpBroadcastLogTable entry (in minutes). | |

| | |
|--------------------------------------|---|
| adp_routing_table | PIB attribute 0x0C: Contains the routing table. array routing_table routing_table ::= structure { destination_address: long-unsigned, next_hop_address: long-unsigned, route_cost: long-unsigned, hop_count: unsigned, weak_link_count: unsigned, valid_time: long-unsigned } NOTE 2 This table is actualized each time a route is built or updated (triggered by data traffic) and each time the TTL timer expires. |
| destination_address | Address of the destination. |
| next_hop_address | Address of the next hop on the route towards the destination. |
| route_cost | Cumulative link cost along the route towards the destination. |
| hop_count | Number of hops of the selected route to the destination. Range: 0-14 NOTE 3 Practically the maximum allowed value is limited by adp_max_hops. |
| weak_link_count | Number of weak links to destination. Range : 0-14 NOTE 4 Practically the maximum allowed value is limited by adp_max_hops. |
| valid_time | Remaining time in minutes until when this entry in the routing table is considered valid. |
| adp_context_information_table | PIB attribute 0x07: Contains the context information associated to each CID extension field. array context_information_table context_information_table ::= structure { CID: bit-string, context_length: unsigned, context: octet-string, C: boolean, valid_lifetime: long-unsigned } |
| 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 – 0xFFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF |
| C | | Indicates if the context is valid for use in compression. FALSE : Only decompression is allowed, TRUE : Compression and decompression are allowed A context may be used for decompression purposes only. Moreover, recommendations made in RFC 6775 should be followed to take into account the propagation of the context to all nodes of the PAN. |
| | valid_lifetime | Remaining time in minutes during which the context information table is considered valid. It is updated upon reception of the advertised context. Range: 0-65 535 |
| adp_blacklist_table | PIB attribute 0x1E: Contains the list of the blacklisted neighbours. array blacklisted_neighbour_set blacklisted_neighbour_set ::= structure { blacklisted_neighbour_address: long-unsigned, valid_time: long-unsigned } | |
| | blacklisted_neighbour_address | The 16-bit address of the blacklisted neighbour. |
| | valid_time | Remaining time in minutes until which this entry in the blacklisted neighbour table is considered valid. |
| adp_broadcast_log_table | PIB attribute 0x0B: Contains the broadcast log table. NOTE 5 This table provides a list of the broadcast packets recently received by this device. array broadcast_log_table broadcast_log_table ::= structure { source_address: long-unsigned, sequence_number: unsigned, valid_time: long-unsigned } | |
| | source_address | The 16-bit source address of a broadcast packet. This is the address of the broadcast |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 276/492 |
|-----------------------|------------|-------------------------|---------|

| | |
|------------------------------------|---|
| | initiator. |
| sequence_number | The sequence number contained in the BC0 header. |
| valid_time | Remaining time in minutes until when this entry in the broadcast log table is considered valid. |
| adp_group_table | <p>PIB attribute 0x0E: Contains the group addresses to which the device belongs.</p> <p>array group_address</p> <p>group_address ::= long-unsigned</p> |
| | group_address Group address to which this node has been subscribed. |
| adp_max_join_wait_time | PIB attribute 0x20: Network join timeout in seconds for LBD. |
| adp_path_discovery_time | PIB attribute 0x21: Timeout for path discovery in seconds. |
| adp_active_key_index | PIB attribute 0x22: Index of the active GMK to be used for data transmission |
| adp_metric_type | PIB attribute 0x03: Metric Type to be used for routing purposes |
| adp_coord_short_address | PIB attribute 0x08: Defines the short address of the coordinator. |
| adp_disable_default_routing | PIB attribute 0xF0: If TRUE, the default routing (LOADng) is disabled. If FALSE, the default routing (LOADng) is enabled. |
| adp_device_type | <p>PIB attribute 0x10: Defines the type of the device connected to the modem:</p> <p>enum:</p> <ul style="list-style-type: none"> (0) PAN device, (1) PAN coordinator, (2) Not Defined |

4.14 ZigBee® setup classes

4.14.1 Overview

This clause 4.14 specifies COSEM interface classes required for the external configuration and management of a ZigBee® network to allow interfacing with a multi-part installation that internally uses ZigBee® communications. ZigBee® is a low-power radio communications technology and open standard that is operated by the ZigBee® Alliance, see www.zigbee.org.

ZigBee® is a registered trademark of the ZigBee® Alliance.

NOTE 1 A multi-part installation is one where the meter provides information and/or services to the householder on behalf of the utility. For example, the meter interacts with an in home display, and/or an external load control switch, and/or a smart appliance, to inform the customer of their usage in real time, to control heating devices, and possibly to disconnect peak loads when supply is constrained. While it is possible that the consumer will control the ZigBee® network, in normal operations the utility will control the radio system. This is to ensure that security is maintained for PAN, so that ZigBee® devices such as load switches controlled by the utility operate in a secure manner.

ZigBee® defines a local network of devices linked by radio, with routing and forwarding of messages and with encryption for privacy at network-level.

NOTE 2 Such a local network is known as a PAN – a Personal Area Network. This name is used in the ZigBee® community as ZigBee® is underpinned by the IEEE 802.15.4 standard, which uses the term PAN. This is broadly equivalent to a HAN (Home Area Network – name used in the context of smart metering in the UK) or PAN.

Each PAN has one device designated as ZigBee® coordinator, which has responsibility for creating and managing the network, and which normally acts also as a ZigBee® Trust Center for the management of ZigBee® network, PCLK's and APS Link keys.

There is a process of PAN creation (and corresponding destruction) which is performed by the coordinator; this declares the existence of the network without any devices apart from the coordinator forming part of it. Other ZigBee® devices can join a network created with cooperation of the coordinator, and equally can choose to leave, or can be invited to leave by the coordinator (this is not currently enforceable). Normally devices are members of the network indefinitely; they do not repeatedly join and leave. To create a PAN the coordinator has to receive an external trigger and needs to have setup information including:

- extended PAN ID;
- link keys or install code (for initial communication with new devices);
- radio channel information.

During the process of creating the PAN, the coordinator scans for nearby radio devices, “exchanges” keys, chooses the short addresses, and confirms use of radio channels. Details of the information available from the ZigBee® servers on each device are also exchanged.

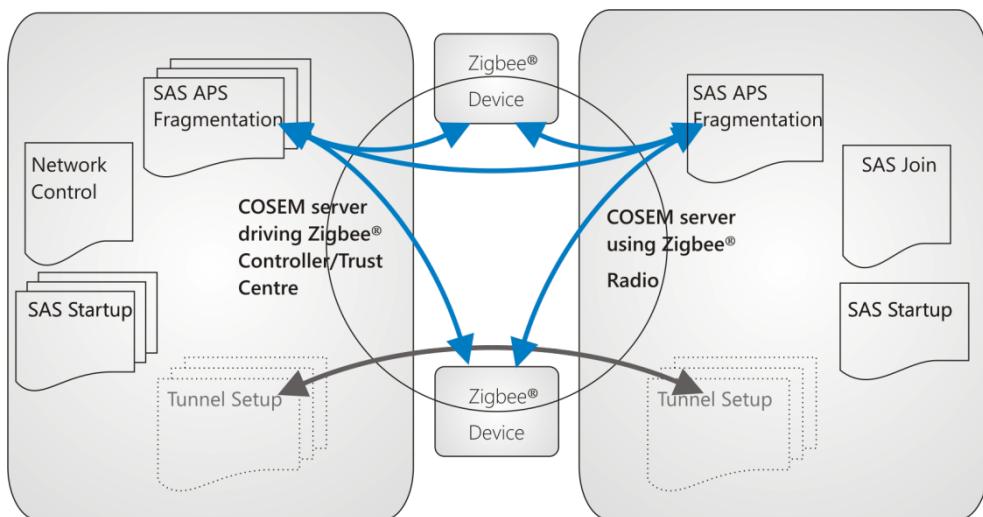
NOTE 3 Full details of the joining process are documented in ZigBee® 053474, the ZigBee® specification. More information on ZigBee® technology can be sought at <http://www.zigbee.org>.

Figure 28 shows an example architecture with a Comms hub on the left, that comprises a DLMS/COSEM server as well as the ZigBee® coordinator. The DLMS/COSEM server has interface objects needed to set up and control the ZigBee® network and may have other COSEM objects to support a metering application. Further, there are two native ZigBee® devices and another DLMS/COSEM server – an electricity meter or another meter type – on the right which is also a “normal” ZigBee® network device.

Any ZigBee® device can be “joined” to the network by remote control.

It is assumed that the Comms hub will also have a further network connection to the WAN, and it is assumed that this is managed by existing DLMS structures – e.g. PSTN, GSM/3G, PLC etc. – but this is out of scope of this Technical Specification.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 278/492 |
|-----------------------|------------|-------------------------|---------|

**Figure 28 – Example of a ZigBee® network**

The role of COSEM interface objects in the process of creating / destructing the PAN is to set up ZigBee® parameters and allow a DLMS/COSEM client to trigger actions, typically when commissioning the installation, in a system where WAN communications between a central system and a smart meter installation is by means of DLMS.

Operation of the ZigBee® network is not the responsibility of DLMS/COSEM. The DLMS/COSEM server is merely the vehicle for controlling the ZigBee® network by an external manager (DLMS/COSEM Client).

The use of ZigBee® setup classes in the ZigBee® coordinator and other DLMS/COSEM ZigBee® devices is shown in Table 36.

Table 36 – Use of ZigBee® setup COSEM interface classes

| ZigBee® coordinator | Other DLMS/COSEM ZigBee® devices | Reference |
|----------------------------------|----------------------------------|-----------|
| ZigBee® SAS startup | ZigBee® SAS startup | 4.14.2 |
| – | ZigBee® SAS join | 4.14.3 |
| ZigBee® SAS APS fragmentation | ZigBee® SAS APS fragmentation | 4.14.4 |
| ZigBee® network control | – | 4.14.5 |
| Optionally: ZigBee® tunnel setup | Optionally: ZigBee® tunnel setup | 4.14.6 |

This set of COSEM ICs supports the ZigBee® 2007 and ZigBee® PRO protocol stacks. The ZigBee® IP protocol stack is not supported at this time.

4.14.2 ZigBee® SAS startup (class_id = 101, version = 0)

NOTE 1 In the specification of the ZigBee® COSEM ICs, the length of the octet-strings is indicated for information.

Instances of this IC are used to configure a ZigBee® PRO device with information necessary to create or join the network. The functionality that is driven by this object and the effect on the network depends on whether the object is located in a ZigBee® coordinator or in another ZigBee® device.

| ZigBee® SAS startup | 0...n | class_id = 101, version = 0 | | | |
|--------------------------------|----------------------|-----------------------------|------|--------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. short_address (dyn.) | long-unsigned | | | 0xFFFF | x + 0x08 |
| 3. extended_pan_id (dyn.) | octet-string | | | 0 | x + 0x10 |
| 4. pan_id (dyn.) | long-unsigned | | | 0xFFFF | x + 0x18 |
| 5. channel_mask (dyn.) | double-long-unsigned | | | 0 | x + 0x20 |
| 6. protocol_version (static) | unsigned | 0x02 | | 0x02 | x + 0x28 |
| 7. stack_profile (static) | enum | | | 0x02 | x + 0x30 |
| 8. start_up_control (dyn.) | unsigned | | | 2 | x + 0x38 |
| 9. trust_center_address (dyn.) | octet-string | | | 0 | x + 0x40 |
| 10. link_key (dyn.) | octet-string | | | 0 | x + 0x48 |
| 11. network_key (dyn.) | octet-string | | | 0 | x + 0x50 |
| 12. use_insecure_join (static) | boolean | | | FALSE | x + 0x58 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|-------------------------|--|
| logical_name | Identifies the “ZigBee® SAS startup” object instance. See 6.2.27. |
| short_address | Defines the 16-bit address by which this device is known locally on the ZigBee® network. Value 0x0000 is given to a coordinator / Trust Center device, and only to these devices. NOTE 2 This short address will be issued to the device by the coordinator when the device joins the network. |
| extended_pan_ID | Defines the unique address by which the network is known externally. The length of the octet-string is 8 octets. |
| pan_ID | Defines the 16-bit address by which network is known locally. |
| channel_mask | Defines (as a bit mask) the set of radio channels which this PAN is permitted to use. Actual usage of channels within this set is determined by the coordinator on the basis of local conditions. The mask is as defined in the ZigBee® specification. |
| protocol_version | Defines the version of the ZigBee® protocol to be used on the network. |
| stack_profile | Identifies the capabilities of the ZigBee ® stack. enum: (1) ZigBee®, (2) ZigBee® PRO |
| start_up_control | Indicates the commissioning state of the ZigBee® device: 2: un-commissioned. Indicates that the device will seek to join the network if/when it is established; 0: commissioned. Indicates that the device should consider itself a part of the network indicated by the <i>extended_pan_id</i> attribute. In this case it will not perform any explicit join or rejoin operation. See NOTE 3 below. |

| | |
|-----------------------------|--|
| trust_center_address | Defines the unique extended address of the Trust Center used for this network. This is often, but not necessarily, the address of the ZigBee® coordinator. The length of the octet-string is 8 octets. |
| link_key | Defines the key value used to secure point-to-point communications between particular devices within the Smart Energy Profile. The way in which the raw key value is protected is not defined in this specification. This is the APS link key or the PCLK. It's down to the implementation if this attribute is Hashed or in the clear or even if this is used. If this is the Trust Center then this is not usually implemented. This is usually read only, but may need to be writable for testing. The length of the octet-string is 16 octets. |
| network_key | Defines the key value used to secure general communications between devices. The way in which the raw key value is protected is not defined in this specification. The network key value may be set internally by the coordinator. It's down to the implementation if this attribute is Hashed or in the clear or even if this is used. If this is the TC then this is not usually implemented. This is read only. The length of the octet-string is 16 octets. |
| use_insecure_join | Indicates whether the coordinator is permitted to allow insecure joining as defined by ZigBee® specification. |

NOTE 3 The “ZigBee® SAS startup” object is reflecting the state of the ZigBee® HAN to the WAN (or a diagnostic tool connected to a ZigBee® connected DLMS/COSEM Server). For example, if the object is in a Comms hub, then the attribute can be read out to show that the ZigBee® HAN has not been commissioned. The main function of this object is to provide information about a ZigBee® network to a client on the WAN (or via the optical port). The reason that there are multiple instances of the “ZigBee® SAS startup” object in Figure 28 is to express the idea that there may be multiple ZigBee® networks operated from a single Comms Hub.

4.14.3 ZigBee® SAS join (class_id = 102, version = 0)

Instances of this IC configure the behaviour of a ZigBee® PRO device on joining or loss of connection to the network. “ZigBee® SAS join” objects are present in all devices where a DLMS/COSEM server controls the ZigBee® Radio behaviour, but it is not used when a device is acting as coordinator (as the coordinator device creates rather than joins a network). “ZigBee® SAS join” objects can be factory configured, or configured using another communications technique – e.g. an optical port.

| ZigBee® SAS join | 0...n | class_id = 102, version = 0 | | | |
|-----------------------------------|---------------|-----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. scan_attempts (static) | unsigned | | | 3 | x + 0x08 |
| 3. time_between_scans (static) | long-unsigned | | | 1 | x + 0x10 |
| 4. rejoin_interval | long-unsigned | | | 60 | x + 0x18 |
| 5. rejoin_retry_interval (static) | long-unsigned | | | 900 | x + 0x20 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|------------------------------|---|
| logical_name | Identifies the “ZigBee® SAS join” object instance. See 6.2.27. |
| scan_attempts | Defines the number of consecutive scans that a ZigBee® device will perform on each attempt to rejoin a network, in the event of losing contact. |
| time_between_scans | Defines the period in seconds between consecutive scans in a single attempt to rejoin a network. |
| rejoin_interval | Defines the period in seconds that the device should wait after apparently becoming disconnected from a network, before attempting to rejoin. |
| rejoin_retry_interval | Defines the period in seconds for which the device should wait after a failed attempt to rejoin a network before trying again. |

Notes on usage

At boot time or when instructed to join a network, the device should complete up to three (3) (Note: as per the default) scan attempts to find a ZigBee® coordinator or router with which to associate.

If a device has not been commissioned, this means that when the user presses a button or uses another methodology to instruct the device to join a network, it will scan all of the channels up to three times (Note: as per the default) to find a network that allows joining. If it has already been commissioned, it should scan up to three times (Note: as per the default) to find its original PAN to join. (ZigBee® Pro devices should try to find a network using their original extended PAN ID and ZigBee® devices can only try to find a network using their original PAN ID).

Note on the *rejoin_retry_interval*

Imposes an upper bound on the *rejoin_interval* parameter – this is restarted if device is touched by human user, i.e. by a button press. This parameter is intended to restrict how often a device will scan to find its network in case the network is no longer present and therefore a scan attempt by the device would always fail (i.e., if a device finds it has lost network connectivity, it will try to re-join the network, scanning all channels if necessary). If the scan fails to successfully re-join, the device will wait for 15 minutes before attempting to re-join again. To be network friendly, it would be recommended to adaptively extend this time period if successive re-joins fail. It would also be recommended the device should try a re-join when triggered (via a control, button, etc.) and fall back to the *rejoin_retry_interval* if attempts to re-join fail again.

4.14.4 ZigBee® SAS APS fragmentation (class_id = 103, version = 0)

Instances of this IC configure the fragmentation feature of ZigBee® PRO transport layer. This fragmentation is not of concern to COSEM; the object merely allows configuration of the fragmentation function by an external manager (DLMS/COSEM client).

Instances of this IC are present in all devices where a DLMS/COSEM server controls the ZigBee® Radio behaviour.

| ZigBee® SAS APS fragmentation | 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 | | | | |

Attribute description

| | |
|---|--|
| logical_name | Identifies the “ZigBee® SAS APS fragmentation” object instance. See 6.2.27. |
| aps_interframe_delay | Defines the delay in milliseconds between sending two blocks of a fragmented transmission. |
| aps_max_window_size | Defines the maximum number of unacknowledged frames that can be transmitted consecutively. NOTE These initial values will allow for installation, and setting these values will depend on the specification of the ZigBee® Interface. |
| 4.14.5 ZigBee® network control (class_id = 104, version = 0) | |

There will be a single instance of the “ZigBee® network control” IC in any device that can act as a ZigBee® coordinator controlled by the DLMS/COSEM client. This class allows interaction between a DLMS/COSEM client (head-end system) and a ZigBee® coordinator at times such as when the installation is commissioned.

| ZigBee® network control | 0..1 | class_id = 104, version = 0 | | | |
|----------------------------------|---------------|-----------------------------|------|-------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. enable_disable_joining | boolean | | | FALSE | x + 0x08 |
| 3. join_timeout (static) | long-unsigned | | | 60 | x + 0x10 |
| 4. active_devices (dyn.) | array | | | | x + 0x18 |
| Specific methods | m/o | | | | |
| 1. register_device (data) | m | | | | x + 0x20 |
| 2. unregister_device (data) | m | | | | x + 0x28 |
| 3. unregister_all_devices (data) | o | | | | x + 0x30 |
| 4. backup_PAN (data) | o | | | | x + 0x38 |
| 5. restore_PAN (data) | o | | | | x + 0x40 |
| 6. identify_device (data) | o | | | | x + 0x48 |
| 7. remove_mirror (data) | o | | | | x + 0x50 |
| 8. update_network_key (data) | o | | | | x + 0x58 |
| 9. update_link_key (data) | o | | | | x + 0x60 |
| 10. create_PAN (data) | m | | | | x + 0x68 |
| 11. remove_PAN (data) | m | | | | x + 0x70 |

Attribute description

| | |
|-------------------------------|---|
| logical_name | Identifies the “ZigBee® network control” object instance. See 6.2.27. |
| enable_disable_joining | A flag controlling whether devices are allowed to join the ZigBee® network. The flag is normally FALSE (joining disabled). At certain times – when a new device is expected to join – the flag is set externally to TRUE and then remains set for the duration of the <i>join_timeout</i> , or until externally reset if that happens sooner. |
| join_timeout | Defines the time period in seconds during which the coordinator device will permit joining of new devices, following setting of the <i>enable_disable_joining</i> flag. |

active_devices This attribute shows all of the currently authorised devices within the ZigBee® PAN.

| | |
|---------------------------------|---------------|
| array | active_device |
| active_device ::= structure | |
| { | |
| mac_address: | octet-string, |
| status: | bit-string, |
| maxRSSI: | integer, |
| averageRSSI: | integer, |
| minRSSI : | integer, |
| maxLQI: | unsigned, |
| averageLQI: | unsigned, |
| minLQI : | unsigned, |
| last_communication_date-time: | octet-string, |
| number_of_hops: | unsigned, |
| transmission_failures: | unsigned, |
| transmission_successes: | unsigned, |
| application_version: | unsigned, |
| stack_version: | unsigned |
| } | |

The length of the mac_address is 8 octets.

The length of the status is 8 bits.

The Max/Average/Min values are taken over the last 24 h period for each device. Period is from 00:00:01 to 00:00:00. They are always historical i.e. they refer to the last day.

status ::= bit-string[8]

| | |
|---------|--------------------------------------|
| Bit 0 = | Authorised on PAN |
| Bit 1 = | Actively reporting on PAN |
| Bit 2 = | Unauthorised on PAN but has reported |
| Bit 3 = | Authorised after swap-out |
| Bit 4 = | SEP Transmitting |
| Bit 4 = | Reserved |
| Bit 6 = | Reserved |
| Bit 7 = | Reserved |

Other elements are detailed in the ZigBee® specification.

Method description

register_device (data) This method is called externally to instruct the coordinator that a device should be added to the list of authorised devices. This method does not actually join the devices to the network, they are just authorised to join at some time in the future.

```
data ::= structure
{
    ieee_address:          octet-string,
    key_type:              enum,
    key:                   octet-string,
    device_type:           enum,
}
```

Where:

- ieee_address holds the IEEE address of the device. The length of the octet-string is 8 octets;
- key_type determines the type of content of key.
enum:

| | |
|-------------|--------------------------|
| (0) | Pre-configured Link Key, |
| (1) | Install code, |
| (2)...(255) | Reserved, |

NOTE 1 Install Codes will be subject to agreement across the members who are implementing a ZigBee® network. Therefore the length of install codes, the use of a CRC, and how they are padded when transported in this method needs are subject to agreement as part of a project specific companion specification.

- key is a pre-configured link key or install code. This will depend on the device being authorised to join the network. Its maximum length is 16 octets,
- device_type is linked to the enumeration shown below so that the coordinator has a method of understanding the possible services that the joining device may require.

NOTE 2 For example, it is likely that an implementation would require Mirror function for Gas, Water and Heat Meters connected by ZigBee®. However, determination of which ZigBee® support is needed for which device will be dependent on the organisation designing the smart metering solution, perhaps a government or a large utility.

| | |
|--|---------------------------------------|
| register_device (data) (continued) | device-type ::= enum |
| | Value |
| | Description |
| | (0) Electricity Meter |
| | (1) Gas Meter |
| | (2) Water Meter |
| | (3) Thermal Meter |
| | (4) Pressure Meter |
| | (5) Heat Meter |
| | (6) Cooling Meter |
| | (7) Electric Vehicle charging Meter |
| | (8) PV Generation Meter |
| | (9) Wind Turbine Generation Meter |
| | (10) Water Turbine Generation Meter |
| | (11) Micro Generation Meter |
| | (12) Solar Hot Water Generation Meter |
| | (13) ZigBee® Controlled Load Switch |
| | (14) ZigBee® Based Boost Button |
| | (128) IHD |
| | (129) Range extender |
| | (130) CAD (Consumer Access Device) |
| | (131) Thermostat |
| | (132) Prepayment Terminal |
| | (133) ZigBee® Controlled Load Switch |
| | (134) ZigBee® Based Boost Button |

| | |
|--|---|
| unregister_device (data) | This method is called externally to instruct the coordinator that a device should leave the network. data ::= octet-string It holds the ieee_address. The length of the octet-string is 8 octets. |
| unregister_all_devices (data) | This method is called externally to instruct the coordinator that all devices should leave the network. data ::= integer (0) NOTE 4 The likely use of this function is to ensure a network is empty of devices prior to destroying it. |

**backup_PAN
(data)** This method instructs the coordinator to create a back-up of information that would be necessary to re-create the PAN.

NOTE 5 The storage location of the back-up is not currently defined and is an internal function of the DLMS/COSEM server.

data ::= integer (0)

Method invocation return parameters are detailed below:

```
data ::= structure
{
    date_time:          octet-string,
    extended_PAN_ID:   octet-string,
    devices_to_backup: array device_to_backup
}
```

Where:

- date-time refers to the time at which the backup process is due to begin. It is formatted as specified in 4.1.6.1;
- extended_PAN_ID identifies the PAN. The length of the octet-string is 8.

```
device_to_backup ::= structure
{
    MAC_address:      octet-string,
    hashed_TC_link_key: octet-string
}
```

Where:

- MAC_address hold the MAC address. The length of the octet-string is 8;
- the length of octet-string holding the hashed_TC_link_key is 16 octets.

NOTE 6 The method of hashing the link key is not part of this specification; it is defined in the ZigBee® Smart Energy Specification. MMO is currently used.

**restore_PAN
(data)** This method instructs the coordinator to restore a PAN using backup information. The storage location of the back-up is not currently defined and is an internal function of the DLMS/COSEM Server.

```
data ::= structure
{
    extended_PAN_ID:   octet-string,
    devices_to_restore: array device_to_restore
}
```

Where:

- extended_PAN_ID identifies the PAN. The length of the octet-string is 8;

**restore_PAN
(data)** (continued) device_to_restore ::= structure
{
 MAC_address: octet-string,
 hashed_TC_link_key: octet-string
}

Where:

- MAC_address holds the MAC address. The length of the octet-string is 8;
- the length of octet-string holding the hashed_TC_link_key is 16 octets.

NOTE 7 The method of hashing the link key is not part of this specification; it is defined in the ZigBee® Smart Energy Specification. MMO is currently used.

**identify_device
(data)** This method is called externally to instruct a device to identify itself to an engineer present on site, for example by sounding a buzzer.
data ::= ieee_address

ieee_address: octet-string

ieee_address holds the IEEE address of the device. The length of the octet-string is 8 octets.

**remove_mirror
(data)** This method causes the removal of a ZigBee® mirror that reflects the real device identified by the mac_address parameter.

data ::= structure
{
 mac_address: octet-string,
 mirror_control: bit-string
}

Where:

- MAC_address holds the MAC address. The length of the octet-string is 8;
- the mirror_control parameter is provided to support the execution of implementation-specific actions, which should be defined in a project specific companion specification.

| | | |
|-------------------------------------|----------------|---|
| remove_mirror (continued) | EXAMPLE | The following example is one of the possible options for the bit-string functionality. |
| | 0 = | Force Gas Meter Removal. NOTE Where a device removal is forced, the keys values for this device are removed; an APS ack from the device is not required. |
| | 1 = | Clear all Mirror Data |
| | 2 = | Clear Consumption registers / indexes |
| | 3 = | Clear Demand & Max Demand registers |
| | 4 = | Clear ZigBee® attributes |
| | 5 = | Clear MPAN |
| | 6 = | Clear Billing Information |
| | 7 = | Clear Logs |
| | 8 = | Clear OTA Firmware waiting |
| | 9...14 = | Reserved |
| | 15 = | Action all |

Should the bit be set then the action is carried out.
The full meaning and actions that the DLMS/COSEM server should undertake are implementation-specific actions, which should be defined in a project specific companion specification.

| | |
|----------------------------------|---|
| update_network_key (data) | This method requests that the ZigBee® coordinator updates the network key and propagate it to all devices on the PAN. For details of ZigBee® key management, see the ZigBee® specification. data ::= integer (0) |
| update_link_key (data) | This method requests that the ZigBee® coordinator updates the link key and propagate it to the identified device on the PAN. For details of ZigBee® key management, see the ZigBee® specification. data ::= ieee_address ieee_address: octet-string ieee_address holds the IEEE address of the device. The length of the octet-string is 8 octets. |
| create_PAN (data) | This method is called externally to instruct a coordinator to create a network using the configuration held in the ZigBee® SAS startup object. data ::= integer (0) |
| remove_PAN (data) | This method is called externally to instruct a coordinator to destroy a network by turning off the ZigBee® radio and removing all of the settings associated with the current PAN. data ::= integer (0) |

4.14.6 ZigBee® tunnel setup (**class_id = 105, version = 0**)

A ZigBee® tunnel is established between two ZigBee® PRO devices to allow DLMS APDUs to be transferred between them. The tunnel in effect extends WAN connectivity to ZigBee® devices not connected to the WAN through a ZigBee® device connected to the same ZigBee® network and connected to the WAN.

The ZigBee® tunnel setup objects would be present on the coordinator and on all other DLMS/COSEM devices that are not connected to the WAN.

Creation of the tunnel is managed on demand and invisibly from the point of view of the DLMS/COSEM client. The target device is implicitly identified by the COSEM addressing information.

NOTE 1 See also the Gateway specification in DLMS UA 1000-2 Ed. 8.1:201.

| 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 | <i>m/o</i> | | | | |

Attribute description

| | |
|---------------------------------------|--|
| logical_name | Identifies the “ZigBee® tunnel setup” object instance. See 6.2.27. |
| maximum_incoming_transfer_size | Defines the maximum size, in octets, of the data packet that can be transferred to the tunnel client in the payload of a single TransferData (TransferData is a ZigBee® parameter) command. Behaviour on receipt of a larger packet is not defined in this specification. |
| maximum_outgoing_transfer_size | Defines the maximum size, in octets, of the data packet that can be sent from the tunnel client in the payload of a single TransferData command. Behaviour on transmission of a larger packet is not defined in this specification. See NOTE 2 below. |
| protocol_address | The <i>protocol_address</i> is implementation-specific, which should be defined in a project specific companion specification. The length of the octet-string is 6 octets. |
| close_tunnel_timeout | Defines the time, in seconds that the ZigBee® server waits before closing an inactive tunnel on its own (without waiting for the <i>CloseTunnel</i> Command from the client within the ZigBee® specification) and freeing its resources The timer is re-started with each reception of a command. |

NOTE 2 The word *outcoming* is unusual, but is adopted for commonality with ZigBee® specifications.

5 Maintenance of the interface classes

5.1 New versions of interface classes

Any modification of an existing IC affecting the transmission of service requests or responses results in a new version (version ::= version+1) and shall be documented accordingly. The following rules shall be followed:

- a) new attributes and methods may be added;
- b) existing attributes and methods may be invalidated BUT the indices of the invalidated attributes and methods shall not be re-used by other attributes and methods;
- c) if these rules cannot be met, then a new IC shall be created;

Any modification of ICs will be recorded by moving the old version of an IC into Clause 5.4.

5.2 New interface classes

The DLMS UA reserves the right to be the exclusive administrator of interface classes.

5.3 Removal of interface classes

Besides one association object and the logical device name object no instantiation of an IC is mandatory within a meter. Therefore, even unused ICs will not be removed from the standard. They will be kept to ensure compatibility with possibly existing implementations.

5.4 Previous versions of interface classes

5.4.1 General

This chapter lists those IC specifications which were included in previous editions of this document. The previous IC versions differ from the current versions by at least one attribute and/or method and by the version number.

For new implementations in metering devices, only the current versions should be used.

Communication drivers at the client side should also support previous versions.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 291/492 |
|-----------------------|------------|-------------------------|---------|

5.4.2 Profile generic (class_id = 7, version = 0)

The version listed here was valid in edition 1 and it is replaced as of edition 2 and 3.

| Profile generic | 0...n | class_id = 7, version = 0 | | | |
|-----------------------------|----------------------|---------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. buffer (dyn.) | array | | | x | x + 0x08 |
| 3. capture_objects (static) | array | | | | x + 0x10 |
| 4. capture_period (static) | double-long-unsigned | | | x | x + 0x18 |
| 5. sort_method (static) | enum | | | x | x + 0x20 |
| 6. sort_object (static) | ObjectDefinition | | | x | x + 0x28 |
| 7. entries_in_use (dyn.) | double-long-unsigned | 0 | | 0 | x + 0x30 |
| 8. profile_entries (static) | double-long-unsigned | 1 | | 1 | x + 0x38 |
| Specific methods | m/o | | | | |
| 1. reset () | | | | | x + 0x58 |
| 2. capture () | | | | | x + 0x60 |
| 3. get_buffer_by_range () | | | | | x + 0x68 |
| 4. get_buffer_by_index () | | | | | x + 0x70 |

Attribute description

buffer The buffer attribute contains a sequence of entries. Each entry contains values of the captured objects (as returned by calling *read (current_value)*). The sequence is ordered according to the specified sort method. The buffer gets filled by subsequent calls to *capture ()*.

array entry

entry ::= structure

Instance Specific

Def. The buffer is empty at installation.

Remark: Reading the entire buffer delivers only those entries which are “in use”

capture_objects Specifies the list of capture objects (registers, clocks and profiles) that are assigned to this profile object. Upon a call of the *capture ()* service, the specified attributes of these objects are copied into the buffer of the profile.

array ObjectDefinition

ObjectDefinition ::= structure

{

logical_name: octet-string,
class_id: long-unsigned,
attribute_index: unsigned

}

where attribute_index is a pointer to the attribute within the object. attribute_index = 1 refers to the first attribute (i.e. *logical_name*), attribute_index = 2 to the 2nd, etc.)

| | |
|---------------------------|--|
| capture_period | >= 1: Automatic capturing assumed. Specifies the capturing period in seconds 0: no automatic capturing, capturing is trigger externally or capture events occur asynchronously |
| sort_method | If the profile is unsorted, it works as a „first in first out“ buffer (it is hence sorted by capturing, and not necessarily by the time maintained in the clock object). If the buffer is full, the next call to <i>capture ()</i> will push out the first (oldest) entry of the buffer to make space for the new entry. If the profile is sorted, a call to <i>capture ()</i> will store the new entry at the appropriate position in the buffer, moving all following entries and probably losing the least interesting entry. If the new entry would enter the buffer after the last entry and if the buffer is already full, the new entry will not be retained at all. enum: 1: fifo, 2: lifo (last in first out), 3: largest, 4: smallest, 5: nearest_to_zero, 6: farthest_from_zero |
| Def. | fifo |
| sort_object | If the profile is sorted, this attribute specifies the register or clock that the ordering is based upon. ObjectDefinition see above Def. no object to sort by (only possible with <i>sort_method</i> = fifo or lifo) |
| entries_in_use | Counts the number of entries stored in the buffer. After a call of <i>reset ()</i> the buffer does not contain any entries, and this value is zero. Upon each subsequent call of <i>capture ()</i> , this value will be incremented up to the maximum number of entries that will get stored (see <i>profile_entries</i>). double-long-unsigned 0...profile_entries Def. 0 |
| profile_entries | Specifies, how many entries should be retained in the buffer. double-long-unsigned 1... (limited by physical size) Def. 1 |
| Method description | |
| reset (data) | Clears the buffer. The buffer has no valid entries afterwards; <i>entries_in_use</i> is zero after this call. This call does not trigger any additional operations of the capture objects, specifically, it does not reset any captured buffers or registers. data = integer (0) |
| capture (data) | Copies the values of the objects to capture into the buffer by calling <i>read (<object_attribute>)</i> of each capture object. Depending on the <i>sort_method</i> and the actual state of the buffer this produces a new entry or a replacement for the less significant entry. As long as not all entries are already used, the <i>entries_in_use</i> attribute will be incremented. This call does not trigger any additional operations within the capture objects such as <i>capture ()</i> or <i>reset ()</i> . Note that only some attributes of the captured objects might be stored, not the complete object. data = integer (0) |

| | | |
|--|---|--|
| write (...) | Any write access to one of the attributes describing the static structure of the buffer will automatically call a <i>reset ()</i> and this call will propagate to all other profiles capturing this profile. If writing to <i>profile_entries</i> is attempted with a value too large for the buffer, it will be set to the maximum possible value (restricted by physical size, typically laid down in the firmware). | |
| get_buffer_by_range (data) | Reads all entries between the given limits. | |
| restricting_object | in | ObjectDefinition Defines the register or clock restricting the range of entries to be retrieved. |
| from_value | in | instance_specific Oldest or smallest entry to retrieve. |
| to_value | in | instance_specific Newest or largest entry to retrieve. |
| selected_values | in | List of columns to retrieve: array ObjectDefinition If the array is empty (has no entries), all captured data is returned. Otherwise, only the columns specified in the array are returned. The type <i>ObjectDefinition</i> is specified above (<i>Capture_Objects</i>) |
| entries | out | See <i>entries_in_use</i> above. |
| array (of <i>instance_specific_value</i>) | out | Sorted sequence of entries containing the requested values of the capture objects. |
| data ::= structure {restricting_object; from_value; to_value; selected_values} In the response "data" is an array of <i>instance_specific_value</i> . | | |
| get_buffer_by_index (data) | Read all entries between the given limits. | |
| from_index | in | double-long-unsigned First entry to retrieve. |
| to_index | in | double-long-unsigned Last entry to retrieve. |
| from_selected_value | in | long-unsigned index of first value to retrieve |
| to_selected_value | in | long-unsigned index of last value to retrieve. |
| entries | out | See <i>entries_in_use</i> above. |
| array of <i>instance_specific_value</i> | out | Sorted sequence of entries containing the requested values of the capture objects. |
| data ::= structure {from_index; to_index; selected_values} In the response "data" is an array of <i>instance_specific_value</i> . | | |

5.4.3 Association SN (class_id = 12, version = 0)

The version listed here was valid in Edition 1 and it is replaced as of Edition 2 and 3.

| Device Association View | 0..1 ³ | class_id = 12, version = 0 | | | |
|-------------------------------|-------------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. object_list (static) | objlist_type | | | | x + 0x08 |
| Specific methods | m/o | | | | |
| 1. getlist_by_classid | o | | | | |
| 2. getobj_by_logicalname | o | | | | |
| 3. read_by_logicalname () | o | | | | |
| 4. get_attributes&services () | o | | | | |
| 5. change_LLS_secret () | o | | | | |
| 6. change_HLS_secret () | o | | | | |
| 7. get_HLS_challenge () | o | | | | |
| 8. reply_to_HLS_challenge () | o | | | | |

Attribute description

logical_name Identifies the type of the client-server association.

object_list Contains the list of all objects with their short_name (DLMS ObjectName of the first attribute), class_id, version⁴ and logical_name.
 Objlist_type ::= array objlist_element
 objlist_element ::= structure
 {
 short_name: long,
 class_id: long-unsigned,
 version = unsigned,
 logical_name: octet-string
 }

Method description

getlist_by_classid (data) Delivers the subset of the object_list for a specific class_id.
 data ::= class_id: long-unsigned

For the response: data ::= objlist_type

getobj_by_logicalname (data) Delivers the entry of the object_list for a specific logical_name and class_id.
 data ::= structure
 {
 class_id: long-unsigned,
 logical_name: octet-string
 }

³ Per client server association

⁴ NOTE Within an client-server association, there are never two objects with the same class_id and logical_name differing only in the versions.

| | |
|---|--|
| | For the response: data ::= objlist_element |
| read_by_logicalname (data) | <p>Reads attributes for specific objects. The objects are specified by their class_id and their logical_name.</p> <p>data ::= array AttributIdentification</p> <p>AttributIdentification ::= structure</p> <pre>{ class_id: long-unsigned, logical_name: octet-string, attribute_index: unsigned }</pre> <p>where attribute_index is a pointer (i.e. offset) to the attribute within the object. attribute_index = 0 delivers all attributes⁵, attribute_index = 1 delivers the first attribute (i.e. <i>logical_name</i>), etc.</p> <p>For the response: data is according to the type of the attribute.</p> |
| get_attributes&services (data) | <p>Delivers information about the access rights to the attributes and services within the actual association. The objects are specified by their class_id and their logical_name.</p> <p>array ObjectIdentification</p> <p>ObjectIdentification ::= structure</p> <pre>{ class_id: long-unsigned, logical_name: octet-string }</pre> <p>For the response</p> <p>data ::= array AccessDescription</p> <p>AccessDescription ::= structure</p> <pre>{ read_attributes: bit-string, write_attributes: bit-string, services: bit-string }</pre> <p>The position in the bit-string identifies the attribute/service (first position ↔ first attribute, first position ↔ first service) and the value of the bit specifies whether the attribute/service is available (bit set) or not available (bit clear).</p> |
| change_LLS_secret (data) | <p>Changes the LLS secret (for example password).</p> <p>data ::= octet-string new LLS secret</p> |
| change_HLS_secret (data) | <p>Changes the HLS secret (for example encryption key).</p> <p>data ::= octet-string⁶ new HLS secret</p> |

⁵ If at least one attribute has no read access right under the current association, then a *read_by_logicalname ()* to attribute index = 0 reveals the error message “scope-of-access-violated” (comp. IEC 61334-4-41:1996, p. 221).

⁶ The structure of the “new secret” depends on the security mechanism implemented. The “new secret” may contain additional checkbits and it may be encrypted.

| | |
|--------------------------------------|---|
| get_HLS_challenge (data) | Asks the server for the client “challenge” (for example random number). data ::= octet-string client challenge |
| reply_to_HLS_challenge (data) | Delivers the “secretly” processed “challenge” back to the server. data ::= octet-string client’s response to the challenge If the authentication is accepted, then the response is successful [0]. If the authentication is not accepted, then the response is set to data-access-error [1]. |

5.4.4 Association SN (class_id = 12, version = 1)

This IC allows modelling of AAs between a server and a client, using the application context *short name referencing*. A COSEM logical device may have one instance of this IC for each association the device is able to support.

The **short_name** of the current “Association SN” object itself is fixed within the COSEM context. It is given in Clause 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 | | | | x + 0x30 |
| 4. get_attributes&methods (data) | o | | | | x + 0x38 |
| 5. change_LLS_secret (data) | o | | | | x + 0x40 |
| 6. change_HLS_secret (data) | o | | | | x + 0x48 |
| 7. reserved from previous versions | | | | | |
| 8. reply_to_HLS_authentication (data) | o | | | | x + 0x58 |

Attribute description

logical_name Identifies the “Association SN” object instance. See 6.2.28.

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

objlist_type ::= array objlist_element

objlist_element ::= structure

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

selective access (see 4.1.4) to the attribute *object_list* may be available (optional). The selective access parameters are as defined below.

Parameters for selective access to the *object_list* attribute

| Access selector value | Parameter | Comment |
|-----------------------|---|--|
| 1 | class_id: long-unsigned | Delivers the subset of the object_list for a specific class_id. For the response: data ::= objlist_type |
| 2 | structure { class_id: long-unsigned, logical_name: octet-string } | Delivers the entry of the object_list for a specific class_id and logical_name. For the response: data ::= objlist_element |

Method description

| | |
|--|---|
| read_by_logicalname | Reads attributes for selected objects. The objects are specified by their class_id and their <i>logical_name</i> . With this method, the parameterized access feature can also be used. data ::= array attribute_identification attribute_identification ::= structure { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer } where attribute_index is a pointer (i.e. offset) to the attribute within the object. attribute_index 0 delivers all attributes ^a , attribute_index 1 delivers the first attribute (i.e. <i>logical_name</i>), etc.). For the response: data is according to the type of the attribute. |
| get_attributes&methods (data) | Delivers information about the access rights to the attributes and methods within the actual association. The objects are specified by their class_id and their <i>logical_name</i> . With this method, the parameterized access feature can also be used. data ::= array object_identification object_identification ::= structure { class_id: long-unsigned, logical_name: octet-string } For the response data ::= array access_description access_description ::= structure { read_attributes: bit-string, write_attributes: bit-string, methods: bit-string } The position in the bit-string identifies the attribute/method (first position ↔ first attribute, first position ↔ first method) and the value of the bit specifies whether the attribute/method is available (bit set) or not available (bit clear). |

Depending on the implementation, some attributes or methods of some objects may not be needed. In this case, such attributes or methods may not be accessible (neither read access, nor write access to attributes, no access to methods).

| | |
|---|--|
| change_LLS_secret (data) | Changes the LLS secret (for example password). data ::= octet-string new LLS secret |
| change_HLS_secret (data) | Changes the HLS secret (for example encryption key). data ::= octet-string ^b new HLS secret |
| reply_to_HLS_authentication (data) | The remote invocation of this method delivers the client's "secretly" processed "challenge StoC" (f(StoC)) back to the server as the <i>data</i> service parameter of the invoked Write.request service. data ::= octet-string client's response to the challenge data ::= octet-string server's response to the challenge If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back. |

^a If at least one attribute has no read access right under the current association, then a *read_by_logicalname ()* to attribute index 0 reveals the error message "scope of access violation" (see IEC 61334-4-41:1996, p. 221).

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

5.4.5 Association SN (class_id = 12, version = 2)

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

The **short_name** of the "Association SN" object itself is fixed within the COSEM context. See 4.1.3.

| 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. reserved from previous versions | o | | | | |
| 2. reserved from previous versions | o | | | | |
| 3. read_by_logicalname (data) | o | | | | x + 0x30 |
| 4. reserved from previous versions | o | | | | |
| 5. change_secret (data) | o | | | | x + 0x40 |
| 6. reserved from previous versions | o | | | | |
| 7. reserved from previous versions | | | | | |
| 8. reply_to_HLS_authentication (data) | o | | | | x + 0x58 |

Attribute description

| | |
|---------------------------|---|
| logical_name | Identifies the “Association SN” object instance. See 6.2.28. |
| object_list | <p>Contains the list of all objects with their <i>base_name</i> (<i>short_name</i>), <i>class_id</i>, <i>version</i> and <i>logical_name</i>. The <i>base_name</i> is the DLMS <i>objectName</i> of the first attribute (<i>logical_name</i>).</p> <pre>objlist_type ::= array objlist_element objlist_element ::= structure { base_name: long, class_id: long-unsigned, version: unsigned, logical_name: octet-string }</pre> <p><i>selective access</i> (see 4.1.4) to the attribute <i>object_list</i> may be available. The access selector values and their parameters are as defined below.</p> |
| access_rights_list | <p>Contains the access rights to attributes and methods.</p> <p>The link between the <i>object_list</i> and the <i>access_rights_list</i> is the <i>base_name</i>, present in both the <i>objlist_element</i> structure and the <i>access_right_element</i> structure. Therefore, the <i>base_names</i> on the two lists shall be the same. The number – and preferably, the order – of the elements in the array of <i>objlist_element</i> and the array of <i>access_right_element</i> shall also be the same.</p> <pre>access_rights_type ::= array access_rights_element access_rights_element ::= structure { base_name: long, attribute_access: attribute_access_descriptor, method_access: method_access_descriptor } attribute_access_descriptor ::= array attribute_access_item attribute_access_item ::= structure { attribute_id: integer, access_mode: enum: (0) no_access, (1) read_only, (2) write_only, (3) read_and_write, (4) authenticated_read_only, (5) authenticated_write_only, (6) authenticated_read_and_write access_selectors: CHOICE { null-data [0], array integer [1] } }</pre> |

| | |
|------------------------------------|--------------------|
| method_access_descriptor ::= array | method_access_item |
|------------------------------------|--------------------|

| | |
|----------------------------------|--|
| method_access_item ::= structure | |
|----------------------------------|--|

| | |
|---|--|
| <pre>{ method_id: integer, access_mode: enum: (0) no_access, (1) access, (2) authenticated_access }</pre> | |
|---|--|

selective access (see 4.1.4) to the attribute *access_rights_list* may be available (optional). The access selector values and their parameters are as defined below.

security_setup_reference

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

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

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

Method description
read_by_logicalname(data)

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

data ::= array attribute_identification

attribute_identification ::= structure

| | |
|--|--|
| <pre>{ class_id: long-unsigned, logical_name: octet-string, attribute_index: integer }</pre> | |
|--|--|

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

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

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

NOTE 1 If at least one attribute has no read access right under the current association, then a *read_by_logicalname()* to attribute index 0 reveals the error message "scope-of-access-violated", see DLMS UA 1000-2 Ed. 8.1:201, 9.5, page 211/310.

| | |
|---|--|
| change_secret (data) | Changes the LLS or HLS secret (for example password). data ::= octet-string new secret NOTE 2 The structure of the "new secret" depends on the security mechanism implemented. The "new secret" may contain additional check bits and it may be encrypted. NOTE 3 In the case of HLS with GMAC, the (HLS_) secret is held by the "Security setup" object referenced in attribute |
| reply_to_HLS_authentication (data) | The remote invocation of this method delivers to the server the result of the secret processing by the client of the server's challenge to the client, f(StoC), as the data service parameter of the Read.request primitive invoked with parameterised access. data ::= octet-string client's response to the challenge If the authentication is accepted, then the response (Read.confirm primitive) contains Result == OK and the result of the secret processing by the server of the client's challenge to the server, f(CtoS) in the data service parameter of the Read.response service. data ::= octet-string server's response to the challenge If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back. |

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

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

The **short_name** of the "Association SN" object itself is fixed within the COSEM context. 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 | | | | |
| 4. reserved from previous versions | o | | | | |
| 5. change_secret (data) | o | | | | |
| 6. reserved from previous versions | o | | | | |
| 7. reserved from previous versions | | | | | |
| 8. reply_to_HLS_authentication (data) | o | | | | |
| 9. add_user (data) | o | | | | |
| 10. remove_user (data) | o | | | | |

Attribute description

| | |
|---------------------------|--|
| logical_name | Identifies the “Association SN” object instance. See 6.2.28. |
| object_list | <p>Contains the list of all objects with their base_name (short_name), class_id, version and <i>logical_name</i>. The base_name is the DLMS objectName of the first attribute (<i>logical_name</i>).</p> <pre>objlist_type ::= array objlist_element objlist_element ::= structure { base_name: long, class_id: long-unsigned, version: unsigned, logical_name: octet-string }</pre> <p><i>selective access</i> (see 4.1.4) to the attribute <i>object_list</i> may be available. The access selector values and their parameters are as defined below.</p> |
| access_rights_list | <p>Contains the access rights to attributes and methods.</p> <p>The link between the <i>object_list</i> and the <i>access_rights_list</i> is the base_name, present in both the objlist_element structure and the access_right_element structure. Therefore, the base_names on the two lists shall be the same. The number – and preferably, the order – of the elements in the array of objlist_element and the array of access_right_element shall also be the same.</p> <pre>access_rights_type ::= array access_rights_element access_rights_element ::= structure {</pre> |

```

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

attribute_access_descriptor ::= array      attribute_access_item

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

    access_selectors: CHOICE
    {
        null-data      [0],
        array integer   [1]
    }
}

method_access_descriptor ::= array      method_access_item
method_access_item ::= structure
{
    method_id:         integer,
    access_mode:       enum:
                        (0)  no_access,
                        (1)  access,
                        (2)  authenticated_access
}
selective access (see 4.1.4) to the attribute access_rights_list may be available
(optional). The access selector values and their parameters are as defined
below.

```

security_setup_reference

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

| | |
|---------------------|--|
| user_list | <p>Contains the list of users allowed to use the AA managed by the given instance of the “Association SN” IC.</p> <p>array user_list_entry</p> <pre>user_list_entry ::= structure { user_id: unsigned, user_name: visible-string }</pre> <p>Where:</p> <ul style="list-style-type: none"> - user_id is the identifier of the user (this value is carried by the calling-AE-invocation-id field of the AARQ); - user_name is the name of the user. <p>If the user_list attribute is empty – i.e. it is an array of 0 elements – any user can use the AA, i.e. the calling-AE-invocation-id field of the AARQ is ignored.</p> <p>If the user_list attribute is not empty then only the users in the list can establish the AA, i.e. the calling-AE-invocation-id field of the AARQ shall be present and its value shall match one of user_ids in the user_list or else, the AA is not established.</p> |
| current_user | <p>Holds the identifier of the current user.</p> <p>current_user ::= user_list_entry (see above)</p> <p>If the user_list is empty, then current_user shall be a structure {user_id: unsigned 0, user_name: visible string of 0 elements}</p> |

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

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

Method description

| | |
|---|---|
| read_by_logicalname (data) | <p>Reads attributes for selected objects. The objects are specified by their <i>class_id</i> and their <i>logical_name</i>. With this method, the parameterized access feature can also be used.</p> <p>data ::= array attribute_identification</p> <pre>attribute_identification ::= structure { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer }</pre> <p>where <i>attribute_index</i> is a pointer (i.e. offset) to the attribute within the object. <i>attribute_index</i> 0 delivers all attributes; <i>attribute_index</i> 1 delivers the first attribute (i.e. <i>logical_name</i>), etc.).</p> <p>For the response: data is according to the type of the attribute.</p> <p>NOTE 2 If at least one attribute has no read access right under the current association, then a <i>read_by_logicalname ()</i> to attribute index 0 reveals the error message "scope-of-access-violated", see DLMS UA 1000-2 Ed. 8.1:201, 9.5, page 211/310.</p> |
| change_secret (data) | <p>Changes the LLS or HLS secret (for example password).</p> <p>data ::= octet-string new secret</p> <p>NOTE 3 The structure of the "new secret" depends on the security mechanism implemented. The "new secret" may contain additional check bits and it may be encrypted.</p> <p>NOTE 4 In the case of HLS with GMAC, the (HLS_) secret is held by the "Security setup" object referenced in attribute</p> |
| reply_to_HLS_authentication (data) | <p>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.</p> <p>data ::= octet-string client's response to the challenge</p> <p>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.</p> <p>data ::= octet-string server's response to the challenge</p> <p>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.</p> |
| add_user (data) | <p>Adds a user to the user_list.</p> <p>data ::= user_list_entry (see above)</p> |
| remove_user (data) | <p>Removes a user from the user_list.</p> <p>data ::= user_list_entry (see above)</p> |

5.4.7 Association LN (class_id = 15, version = 0)

This IC allows modelling of AAs between a server and a client, using the application context *logical name referencing*. Each AA is modelled by one Association LN object.

| Association LN | 0...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 |

Attribute description

| | |
|---------------------|--|
| logical_name | Identifies the “Association LN” object instance. See 6.2.28. |
| object_list | Contains the list of visible COSEM objects with their class_id, version, logical name and the access rights to their attributes and methods within the given application association. object_list_type ::= array object_list_element |
| | object_list_element ::= structure { class_id: long-unsigned, version: unsigned, logical_name: octet-string, access_rights: access_right } access_right ::= structure { attribute_access: attribute_access_descriptor, method_access: method_access_descriptor } |
| | attribute_access_descriptor ::= array attribute_access_item |

```

attribute_access_item ::= structure
{
    attribute_id:           integer,
    access_mode:            enum:
                            (0) no_access,
                            (1) read_only,
                            (2) write_only,
                            (3) read_and_write

    access_selectors: CHOICE
    {
        null-data          [0],
        array integer      [1]
    }
}

method_access_descriptor ::= array      method_access_item
method_access_item ::=      structure
{
    method_id:           integer,
    access_mode:         boolean
}

```

Where:

- the attribute_access_descriptor and the method_access_descriptor always contain all implemented attributes or methods;
- access_selectors contain a list of the supported selector values;
- selective access (see 4.1.4) to the attribute object_list may be available (optional). The selective access parameters are as defined below.

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

```

associated_partners_type ::= structure
{
    client_SAP:           integer,
    server_SAP:           long-unsigned
}

```

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

The range for the server_SAP is 0x000...0x3FFF.

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

```

context_name_type ::=      CHOICE
{
    context_name_structure [2],
    octet-string           [9]
}

```

The application context name is specified as OBJECT IDENTIFIER in DLMS UA 1000-2 Ed. 8.1:201 9.4.2.2.2.

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

```
context_name_structure ::= structure
{
    joint_iso_ctt_element:           unsigned,
    country_element:                unsigned,
    country_name_element:           long-unsigned,
    identified_organization_element: unsigned,
    DLMS_UA_element:                unsigned,
    application_context_element:    unsigned,
    context_id_element:              unsigned
}
```

Example 1: In the case of context_id(1) the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 01 11 01 (all values are hexadecimal).

When the context_name_type is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See DLMS UA 1000-2 Ed. 8.1:201 11.4.

Example 2: In the case of context_id(1) the A-XDR encoding is: 09 07 60 85 74 05 08 01 01 (all values are hexadecimal).

xDLMS_context_info Contains all the necessary information on the xDLMS context for the given AA.
xDLMS-context-type ::= structure

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

Where:

- the conformance element contains the xDLMS conformance block supported by the server;
- the 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 DLMS UA 1000-2 Ed. 8.1:201 Clause 9.5);
- 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 DLMS UA 1000-2 Ed. 8.1:201 Clause 9.5);
- the dlms_version_number element contains the DLMS version number supported by the server;
- the quality_of_service element is not used;
- the cyphering_info, in an active association, contains the dedicated key parameter of the xDLMS initiateRequest APDU (see DLMS UA 1000-2 Ed. 8.1:201 Clause 9.5).

| | |
|--------------------------------------|---|
| authentication_mechanism_name | Contains the name of the authentication mechanism for the AA. |
|--------------------------------------|---|

```
mechanism_name_type ::= CHOICE
{
    mechanism_name_structure [2],
    octet-string [9]
}
```

The authentication mechanism name is specified as an OBJECT IDENTIFIER in DLMS UA 1000-2 Ed. 8.1:201 9.4.2.2.3.

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

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

Example 3: In the case of mechanism_id(1) the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 02 11 01 (all values are hexadecimal):

When the mechanism_name_type is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See DLMS UA 1000-2 Ed. 8.1:201 11.4.

EXAMPLE 4: In the case of mechanism_id(1) the A-XDR encoding is: 09 07 60 85 74 05 08 02 01 (all values are hexadecimal).

No mechanism_name is required when no authentication is used.

| | |
|---------------------------|--|
| LLS_secret | Contains the authentication value for the LLS authentication process. |
| association_status | Indicates the current status of the association, which is modelled by the object. enum: (0) non-associated, (1) association-pending, (2) associated |

Parameters for selective access to the *object_list* attribute

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

| Access selector | Access parameter | Comment |
|-----------------|------------------|--|
| 1 | NULL | All information excluding the 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. |

Method description

| | |
|---|--|
| reply_to_HLS_authentication (data) | The remote invocation of this method delivers the client's "secretly" processed "challenge StoC" (f(StoC)) back to the server as the <i>data</i> service parameter of the ACTION.request primitive invoked. data ::= octet-string client's response to the challenge If the authentication is accepted, then the response (ACTION.confirm primitive) contains Result == OK and the server's "secretly" processed "challenge CtoS" (f(CtoS)) back to the client in the <i>data</i> service parameter of the response service. data ::= octet-string server's response to the challenge If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back. |
| change_HLS_secret (data) | Changes the HLS secret (for example encryption key). data ::= octet-string ^a new HLS secret |
| add_object (data) | Adds the referenced object to the object_list. data ::= object_list_element (see above) |
| remove_object (data) | Removes the referenced object from the object_list. data ::= object_list_element (see above) |

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

5.4.8 Association LN (class_id = 15, version = 1)

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

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

Attribute description

logical_name Identifies the “Association LN” object instance. See 6.2.28

object_list Contains the list of visible COSEM objects with their class_id, version, logical name and the access rights to their attributes and methods within the given application association.
 object_list_type ::= array object_list_element
 object_list_element ::= structure
 {
 class_id: long-unsigned,
 version: unsigned,
 logical_name: octet-string,
 access_rights: access_right
 }
 access_right ::= structure
 {
 attribute_access: attribute_access_descriptor,
 method_access: method_access_descriptor
 }
 attribute_access_descriptor ::= array attribute_access_item

```

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

    access_selectors: CHOICE
    {
        null-data          [0],
        array integer      [1]
    }
}
method_access_descriptor ::= array      method_access_item

method_access_item ::= structure
{
    method_id:      integer,
    access_mode:   enum:
                    (0)  no_access,
                    (1)  access,
                    (2)  authenticated_access
}

```

Where:

- the attribute_access_descriptor and the method_access_descriptor elements always contain all implemented attributes or methods;
 - access_selectors contain a list of the supported selector values.
- selective access* (see 4.1.4) to the attribute *object_list* may be available (optional). The selective access parameters are as defined below.

| | |
|-------------------------------|--|
| associated_partners_id | <p>Contains the identifiers of the COSEM client and server (logical device) APs within the physical devices hosting these APs, which belong to the AA modelled by the “Association LN” object.</p> <p>associated_partners_type ::= structure</p> <pre> { client_SAP: integer, server_SAP: long-unsigned } </pre> <p>The range for the client_SAP is 0...0x7F.</p> <p>The range for the server_SAP is 0x0000...0x3FFF.</p> <p>The SAPs shall be in the range allowed by the data type and the media.</p> |
|-------------------------------|--|

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

context_name_type ::= CHOICE

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

The application context name is specified as OBJECT IDENTIFIER in DLMS UA 1000-2 Ed. 8.1:201 9.4.2.2.2.

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

context_name_structure ::= structure

```
{
    joint_iso_ctt_element: unsigned,
    country_element: unsigned,
    country_name_element: long-unsigned,
    identified_organization_element: unsigned,
    DLMS_UA_element: unsigned,
    application_context_element: unsigned,
    context_id_element: unsigned
}
```

Example 1: In the case of context_id(1) the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 01 11 01 (all values are hexadecimal).

When the context_name_type is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See DLMS UA 1000-2 Ed. 8.1:201 11.4.

Example 2: In the case of context_id(1) the A-XDR encoding is: 09 07 60 85 74 05 08 01 01 (all values are hexadecimal).

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

xDLMS_context_type ::= structure

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

Where:

- the conformance element contains the xDLMS conformance block supported by the server;
- the max_receive_pdu_size element contains the maximum length for an xDLMS APDU, expressed in bytes that the client may send. This is the same as the server-max-receive-pdu-size parameter of the xDLMS initiateResponse APDU;
- the max_send_pdu_size element, in an active AA contains the maximum length for an xDLMS APDU, expressed in bytes that the server may send. This is the same as the client-max-receive-pdu-size parameter of the xDLMS initiateRequest APDU;
- the dlms_version_number element contains the DLMS version number supported by the server;

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 314/492 |
|-----------------------|------------|-------------------------|---------|

-
- the quality_of_service element is not used;
 - the cyphering_info element, in an active association, contains the dedicated key parameter of the xDLMS initiateRequest APDU. See DLMS UA 1000-2 Ed. 8.1:201 9.5.
-

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

```
mechanism_name_type ::= CHOICE
{
    mechanism_name_structure [2],
    octet-string [9]
}
```

The authentication mechanism name is specified as an OBJECT IDENTIFIER in DLMS UA 1000-2 Ed. 8.1:201 9.4.2.2.3.

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

```
mechanism_name_structure ::= structure
```

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

Example 3: In the case of mechanism_id(1) the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 02 11 01 (all values are hexadecimal):

When the mechanism_name_type is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See DLMS UA 1000-2 Ed. 8.1:201 11.4.

EXAMPLE 4: In the case of mechanism_id(1) the A-XDR encoding is: 09 07 60 85 74 05 08 02 01 (all values are hexadecimal).

No mechanism_name is required when no authentication is used.

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

NOTE 2 In the case of HLS with GMAC, the (HLS)_secret is held by the Security setup object referenced in attribute 9.

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

enum: (0) non-associated,
(1) association-pending,
(2) associated

security_setup_reference References the Security setup object by its logical name. The referenced object manages security for a given Association LN object instance.

A SET operation on an attribute of an association LN object becomes effective when this association object is used to establish a new association.

Parameters for selective access to the object_list attribute

- If no selective access is requested, (no Access_Selection_Parameters parameter is present in the GET.request (.indication) service primitive for the object_list attribute) the corresponding .response (.confirmation) service shall contain all object_list_elements of the object_list attribute.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 315/492 |
|-----------------------|------------|-------------------------|---------|

- When selective access is requested to the `object_list` attribute (the `Access_Selection_Parameter` parameter is present), the response shall contain a ‘filtered’ list of `object_list_elements`, as follows:

| Access selector | Access parameter | Comment |
|-----------------|-----------------------------|---|
| 1 | NULL | All information excluding the <code>access_rights</code> shall be included in the response. |
| 2 | <code>class_list</code> | <p>Access by <code>class_id</code>. In this case, only those <code>object_list_elements</code> of the <code>object_list</code> shall be included in the response, which have a <code>class_id</code> equal to one of the <code>class_id</code>-s of the <code>class_list</code>.</p> <p>No <code>access_right</code> information is included.</p> <p><code>class_list ::= array class_id</code></p> <p><code>class_id: long-unsigned</code></p> |
| 3 | <code>object_id_list</code> | <p>Access by object. The full information record of object instances on the <code>object_id_list</code> shall be returned.</p> <p><code>object_id_list ::= array object_id</code></p> <p><code>object_id ::= structure</code></p> <p>{</p> <p style="padding-left: 20px;"><code>class_id: long-unsigned,</code></p> <p style="padding-left: 20px;"><code>logical_name: octet-string</code></p> <p>}</p> |
| 4 | <code>object_id</code> | <p>The full information record of the required COSEM object instance shall be returned.</p> <p><code>object_id ::= structure</code></p> <p>See above.</p> |

Method description

| | |
|---|---|
| reply_to_HLS_authentication (data) | <p>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, $f(\text{StoC})$, as the <code>data</code> service parameter of the ACTION.request primitive invoked.</p> <p><code>data ::= octet-string</code> client’s response to the challenge</p> <p>If the authentication is accepted, then the response (ACTION.confirm primitive) contains <code>Result == OK</code> and the result of the secret processing by the server of the client’s challenge to the server, $f(\text{CtoS})$ in the <code>data</code> service parameter of the response service.</p> <p><code>data ::= octet-string</code> server’s response to the challenge</p> <p>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.</p> |
| change_HLS_secret (data) | <p>Changes the HLS secret (for example encryption key).</p> <p><code>data ::= octet-string</code> new HLS secret</p> <p>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.</p> |
| add_object (data) | <p>Adds the referenced object to the <code>object_list</code>.</p> <p><code>data ::= object_list_element</code> (see above)</p> |
| remove_object (data) | <p>Removes the referenced object from the <code>object_list</code>.</p> <p><code>data ::= object_list_element</code> (see above)</p> |

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

NOTE 1 This new version 2 of the "Association LN" IC supports the client user identification process, see 4.4.2.

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

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

Attribute description

| | |
|---------------------|--|
| logical_name | Identifies the “Association LN” object instance. See 6.2.25. |
| object_list | Contains the list of visible COSEM objects with their class_id, version, <i>logical_name</i> and the access rights to their attributes and methods within the given AA. object_list_type ::= array object_list_element |
| | object_list_element ::= structure |
| | { |
| | class_id: long-unsigned, |
| | version: unsigned, |
| | logical_name: octet-string, |
| | access_rights: access_right |
| | } |

| | |
|-----------------------------------|---|
| object_list (continued) | <pre> access_right ::= structure { attribute_access: attribute_access_descriptor, method_access: method_access_descriptor } attribute_access_descriptor ::= array attribute_access_item attribute_access_item ::= structure { attribute_id: integer, access_mode: enum: (0) no_access, (1) read_only, (2) write_only, (3) read_and_write, (4) authenticated_read_only, (5) authenticated_write_only, (6) authenticated_read_and_write access_selectors: CHOICE { null-data [0], array integer [1] } } </pre> |
| | <p>method_access_descriptor ::= array method_access_item</p> <p>method_access_item ::= structure</p> <pre> { method_id: integer, access_mode: enum: (0) no_access, (1) access, (2) authenticated_access } </pre> <p>Where:</p> <ul style="list-style-type: none"> - the attribute_access_descriptor and the method_access_descriptor elements always contain all implemented attributes or methods; - access_selectors contain a list of the supported selector values. <p><i>selective access</i> (see 4.1.4) to the attribute <i>object_list</i> may be available (optional). The selective access parameters are as defined below.</p> |

| | |
|-------------------------------|--|
| associated_partners_id | <p>Contains the identifiers of the COSEM client and server (logical device) APs within the physical devices hosting these APs, which belong to the AA modelled by the “Association LN” object.</p> <p>associated_partners_type ::= structure</p> <pre> { client_SAP: integer, server_SAP: long-unsigned } </pre> |
|-------------------------------|--|

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

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

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

| | |
|---------------------------------|--|
| application_context_name | In the COSEM environment, it is intended that an application context pre-exists and is referenced by its name during the establishment of an AA. This attribute contains the name of the application context for that AA. context_name_type ::= CHOICE { context_name_structure [2], octet-string [9] } |
|---------------------------------|--|

The application context name is specified as OBJECT IDENTIFIER in DLMS UA 1000-2 Ed. 8.1:201 9.4.2.2.2.

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

context_name_structure ::= structure

```
{
    joint_iso_ctt_element:           unsigned,
    country_element:                unsigned,
    country_name_element:           long-unsigned,
    identified_organization_element: unsigned,
    DLMS_UA_element:                unsigned,
    application_context_element:     unsigned,
    context_id_element:              unsigned
}
```

Example 1: In the case of context_id(1) the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 01 11 01 (all values are hexadecimal).

When the context_name_type is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See DLMS UA 1000-2 Ed. 8.1:201 11.4.

Example 2: In the case of context_id(1) the A-XDR encoding is: 09 07 60 85 74 05 08 01 01 (all values are hexadecimal).

| | |
|--|--|
| xDLMS_context_info | Contains all the necessary information on the xDLMS context for the given AA. xDLMS_context_type ::= structure { conformance: bit-string, max_receive_pdu_size: long-unsigned, max_send_pdu_size: long-unsigned, dlms_version_number: unsigned, quality_of_service: integer, cyphering_info: octet-string } |
| Where: <ul style="list-style-type: none">- the conformance element contains the xDLMS conformance block supported by the server. The length of the bit-string is 24 bits.- the max_receive_pdu_size element contains the maximum length for an xDLMS APDU, expressed in bytes that the client may send. This is the same as the server-max-receive-pdu-size parameter of the xDLMS initiateResponse APDU;- the max_send_pdu_size element, in an active AA contains the maximum length for an xDLMS APDU, expressed in bytes that the server may send. | |

- This is the same as the client-max-receive-pdu-size parameter of the xDLMS initiateRequest APDU;
- the dlms_version_number element contains the DLMS version number supported by the server;
 - the quality_of_service element is not used;
 - the cyphering_info element – in an active AA – contains the dedicated key parameter of the xDLMS initiateRequest APDU. See DLMS UA 1000-2 Ed. 8.1:201 clause 9.5.

| | |
|--------------------------------------|---|
| authentication_mechanism_name | Contains the name of the authentication mechanism for the AA. <code>mechanism_name_type ::= CHOICE</code> <code>{</code> <code> mechanism_name_structure [2],</code> <code> octet-string [9]</code> <code>}</code> |
|--------------------------------------|---|

The authentication mechanism name is specified as an OBJECT IDENTIFIER in DLMS UA 1000-2 Ed. 8.1:201 9.4.2.2.3.

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

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

Example 3: In the case of mechanism_id(1) the A-XDR encoding is: 02 07 11 02 11 10 12 02 F4 11 05 11 08 11 02 11 01 (all values are hexadecimal):

When the mechanism_name_type is encoded as an octet-string, it holds the value of the OBJECT IDENTIFIER. See DLMS UA 1000-2 Ed. 8.1:201 11.4.

EXAMPLE 4: In the case of mechanism_id(1) the A-XDR encoding is: 09 07 60 85 74 05 08 02 01 (all values are hexadecimal).

No mechanism_name is required when no authentication is used.

| | |
|---------------------------------|--|
| secret | Contains the secret for the LLS or HLS authentication process. |
| | NOTE 2 In the case of HLS with GMAC, the (HLS_)secret is held by the “Security setup” object referenced in attribute 9, security_setup_reference. |
| association_status | Indicates the current status of the association, which is modelled by the object. enum: (0) non-associated, (1) association-pending, (2) associated |
| security_setup_reference | References a “Security setup” object by its logical name. The referenced object manages security for a given “Association LN” object instance. |

| | |
|---------------------|--|
| user_list | <p>Contains the list of users allowed to use the AA managed by the given instance of the "Association LN" IC.</p> <pre>array user_list_entry user_list_entry ::= structure { user_id: unsigned, user_name: visible-string }</pre> <p>Where:</p> <ul style="list-style-type: none"> - user_id is the identifier of the user (this value is carried by the calling-AE-invocation-id field of the AARQ); - user_name is the name of the user. <p>If the <i>user_list</i> attribute is empty – i.e. it is an array of 0 elements – any user can use the AA, i.e. the calling-AE-invocation-id field of the AARQ is ignored.</p> <p>If the <i>user_list</i> attribute is not empty then only the users in the list can establish the AA, i.e. the calling-AE-invocation-id field of the AARQ shall be present and its value shall match one of user_ids in the <i>user_list</i> or else the AA is not established.</p> |
| current_user | <p>Holds the identifier of the current user.</p> <pre>current_user ::= user_list_entry (see above)</pre> <p>If the <i>user_list</i> is empty, then <i>current_user</i> shall be a structure {user_id: unsigned 0, user_name: visible string of 0 elements}</p> |

Parameters for selective access to the *object_list* attribute

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

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

Method description

| | |
|---|---|
| reply_to_HLS_authentication (data) | The remote invocation of this method delivers to the server the result of the secret processing by the client of the server's challenge to the client, f(StoC), as the <i>data</i> service parameter of the ACTION.request primitive invoked. data ::= octet-string client's response to the challenge If the authentication is accepted, then the response (ACTION.confirm primitive) contains Result == OK and the result of the secret processing by the server of the client's challenge to the server, f(CtoS) in the <i>data</i> service parameter of the response service. data ::= octet-string server's response to the challenge If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back. |
| change_HLS_secret (data) | Changes the HLS secret (for example encryption key). data ::= octet-string new HLS secret The structure of the "new secret" depends on the security mechanism implemented. The "new secret" may contain additional check bits and it may be encrypted. |
| add_object (data) | Adds the referenced object to the <i>object_list</i> . data ::= object_list_element (see above) |
| remove_object (data) | Removes the referenced object from the <i>object_list</i> . data ::= object_list_element (see above) |
| add_user (data) | Adds a user to the <i>user_list</i> . data ::= user_list_entry (see above) |
| remove_user (data) | Removes a user from the <i>user_list</i> . data ::= user_list_entry (see above) |

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

Instances of this IC contain the necessary information on the security policy applicable and the security suite in use within a particular AA, between two systems identified by their client system title and server system title respectively. They also contain methods to increase the level of security and to transfer the global keys. See DLMS UA 1000-2 Ed. 8.1:201 Clause 9.2.

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

Attribute description

| | |
|---|---|
| logical_name | Identifies the “Security setup” object instance. See 6.2.31. |
| security_policy | Enforces authentication and/or encryption algorithm provided with security_suite. enum: (0) nothing, (1) all messages to be authenticated, (2) all messages to be encrypted, (3) all messages to be authenticated and encrypted (4)...(15) reserved |
| security_suite | Specifies authentication, encryption and key transport algorithm. enum: (0) AES-GCM-128 for authenticated encryption and AES-128 for key wrapping (1) ...(15) reserved |
| client_system_title | Carries the (current) client system title: <ul style="list-style-type: none"> - in the S-FSK PLC environment, the active initiator sends its system title using the CIASE protocol; NOTE 1 It is also held by the <i>active_initiator</i> attribute of the S-FSK Active initiator object; see 4.10.4; - during confirmed or unconfirmed AA establishment, it is carried by the calling-AP-title field of the AARQ APDU; - If a client system title has already been sent during a registration process, like in the case of the S-FSK PLC profile, the client system title carried by the AARQ APDU should be the same. If not, the AA shall be rejected and appropriate diagnostic information shall be sent. - in a pre-established AA, it can be written by the client using an unsecured SET / Write service. |
| server_system_title | Carries the server system title. <ul style="list-style-type: none"> - in the S-FSK PLC environment, the server sends its system title during the discover process, using the CIASE protocol; - during confirmed AA establishment, it is carried by the responding-AP-title field of the AARE APDU. <p>This attribute shall be read only.</p> |
| Method description | |
| security_activate (data) | Activates and strengthens the security policy: enum: (0) nothing, (1) all messages to be authenticated, (2) all messages to be encrypted, (3) all messages to be authenticated and encrypted |
| The new security policy applies as soon as the method invocation has been confirmed with success. | |
| NOTE 2 The security policy can only be strengthened. | |

5.4.11 IEC local port setup (class_id = 19, version = 0)

Instances of this IC define the operational parameters for communication using IEC 62056-21:2002. Several ports can be configured.

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

Attribute description

| | |
|---------------------|--|
| logical_name | Identifies the “IEC local port setup” object instance. See 6.2.17. |
| default_mode | Defines the protocol used by the meter on the port. enum: (0) protocol according to IEC 62056-21:2002 (modes A...E) (1) protocol according to DLMS UA 1000-2 Ed. 8.1:201 Clause 8. Using this enumeration value all other attributes of this IC are not applicable. |

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

5.4.12 IEC HDLC setup, (class_id = 23, version = 0)

An instance of the “IEC HDLC setup” contains all data necessary to set up a communication channel according to Clause 8 of DLMS UA 1000-2 Ed. 8.1:201. Several communication channels can be configured.

| IEC HDLC setup | 0...n | class_id = 23, version = 0 | | | |
|-----------------------------------|------------------------|----------------------------|--------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name | (static) octet-string | | | | x |
| 2. comm_speed | (static) enum | 0 | 9 | 5 | x + 0x08 |
| 3. window_size_transmit | (static) unsigned | 1 | 7 | 1 | x + 0x10 |
| 4. window_size_receive | (static) unsigned | 1 | 7 | 1 | x + 0x18 |
| 5. max_info_field_length_transmit | (static) unsigned | 32 | 128 | 128 | x + 0x20 |
| 6. max_info_field_length_receive | (static) unsigned | 32 | 128 | 128 | x + 0x28 |
| 7. inter_octet_time_out | (static) long-unsigned | 20 | 1000 | 25 | x + 0x30 |
| 8. inactivity_time_out | (static) long-unsigned | 0 | | 120 | x + 0x38 |
| 9. device_address | (static) long-unsigned | 0x0010 | 0x3FFD | | x + 0x40 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|---------------------------------|---|
| logical_name | Identifies the “IEC HDLC setup” object instance. See 6.2.19. |
| comm_speed | The communication speed supported by the corresponding port: enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud This communication speed can be overridden if the HDLC mode of a device is entered through a special mode of another protocol. |
| window_size_transmit | The maximum number of frames that a device or system can transmit before it needs to receive an acknowledgement from a corresponding station. During logon, other values can be negotiated. |
| window_size_receive | The maximum number of frames that a device or system can receive before it needs to transmit an acknowledgement to the corresponding station. During logon, other values can be negotiated. |
| max_info_length_transmit | The maximum information field length that a device can transmit. During logon, a smaller value can be negotiated. |
| max_info_length_receive | The maximum information field length that a device can receive. During logon, a smaller value can be negotiated. |
| inter_octet_time_out | Defines the time, expressed in milliseconds, over which, when no character is received from the primary station, the device will treat the already received data as a complete frame. |

| | | | | | | | | | | | | | | | | | | | | | |
|----------------------------|---|------|---------------------|-------------|--------------------------|-------------|-----------------------|------|---------------------------|------|-------------------|--------|---------------------|-----------------|--------------------------|-----------------|-----------------------|--------|------------------------------------|--------|-------------------|
| inactivity_time_out | Defines the time, expressed in seconds over which, when no frame is received from the primary station, the device will process a disconnection. When this value is set to 0, this means that the <i>inactivity_time_out</i> is not operational. | | | | | | | | | | | | | | | | | | | | |
| device_address | <p>Contains the physical device address of a device:</p> <p>In the case of single byte addressing:</p> <table> <tr><td>0x00</td><td>NO_STATION Address,</td></tr> <tr><td>0x01...0x0F</td><td>Reserved for future use,</td></tr> <tr><td>0x10...0x7D</td><td>Usable address space,</td></tr> <tr><td>0x7E</td><td>'CALLING' device address,</td></tr> <tr><td>0x7F</td><td>Broadcast address</td></tr> </table> <p>In the case of double byte addressing:</p> <table> <tr><td>0x0000</td><td>NO_STATION address,</td></tr> <tr><td>0x0001...0x000F</td><td>Reserved for future use,</td></tr> <tr><td>0x0010...0x3FFD</td><td>Usable address space,</td></tr> <tr><td>0x3FFE</td><td>'CALLING' physical device address,</td></tr> <tr><td>0x3FFF</td><td>Broadcast address</td></tr> </table> | 0x00 | NO_STATION Address, | 0x01...0x0F | Reserved for future use, | 0x10...0x7D | Usable address space, | 0x7E | 'CALLING' device address, | 0x7F | Broadcast address | 0x0000 | NO_STATION address, | 0x0001...0x000F | Reserved for future use, | 0x0010...0x3FFD | Usable address space, | 0x3FFE | 'CALLING' physical device address, | 0x3FFF | Broadcast address |
| 0x00 | NO_STATION Address, | | | | | | | | | | | | | | | | | | | | |
| 0x01...0x0F | Reserved for future use, | | | | | | | | | | | | | | | | | | | | |
| 0x10...0x7D | Usable address space, | | | | | | | | | | | | | | | | | | | | |
| 0x7E | 'CALLING' device address, | | | | | | | | | | | | | | | | | | | | |
| 0x7F | Broadcast address | | | | | | | | | | | | | | | | | | | | |
| 0x0000 | NO_STATION address, | | | | | | | | | | | | | | | | | | | | |
| 0x0001...0x000F | Reserved for future use, | | | | | | | | | | | | | | | | | | | | |
| 0x0010...0x3FFD | Usable address space, | | | | | | | | | | | | | | | | | | | | |
| 0x3FFE | 'CALLING' physical device address, | | | | | | | | | | | | | | | | | | | | |
| 0x3FFF | Broadcast address | | | | | | | | | | | | | | | | | | | | |

5.4.13 IEC twisted pair (1) setup (class_id = 24, version = 0)

NOTE The use of version 0 of the IEC twisted pair (1) setup IC is deprecated.

This IC allows modelling and configuring communication channels according to IEC 62056-31:1999. Several communication channels can be configured.

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

Attribute description

| | |
|-----------------------------|--|
| logical_name | Identifies the "IEC twisted pair setup" object instance. See 6.2.20. |
| secondary_address | <p>Secondary_address memorizes the ADS of the secondary station that corresponds to the real equipment.</p> <p>octet-string (SIZE(6))</p> |
| primary_address_list | <p>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.</p> <p>primary_address_list_type ::= array primary_address_element</p> <p>primary_address_element ::= octet-string</p> <p>The length of the octet-string is one octet.</p> |

| | |
|------------------|--|
| tabi_list | <i>tabi_list</i> represents the list of the TAB(i) for which the real equipment (the secondary station) has been programmed in case of forgotten station call. |
| | <pre>tabi_list_type ::= array tabi_element tabi_element ::= integer</pre> |

| | |
|--------------------|---|
| fatal_error | <i>fatal_error</i> represents the last occurrence of one of the fatal errors of the protocols described in IEC 62056-31:1999. The initial default value of this variable is 0x00. Then, each fatal error is spotted. enum: (0) No-error, (1) t-EP-1F, (2) t-EP-2F, (3) t-EL-4F, (4) t-EL-5F, (5) eT-1F, (6) eT-2F, (7) e-EP-3F, (8) e-EP-4F, (9) e-EP-5F, (10) e-EL-2F |
|--------------------|---|

5.4.14 PSTN modem configuration (class_id = 27, version = 0)

NOTE The name of this IC was changed to "Modem configuration" in Edition 6.0.

An instance of the "PSTN modem configuration" IC stores data related to the initialization of modems, which are used for data transfer from/to a device. Several modems can be configured.

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

Attribute description

| | |
|---------------------|---|
| logical_name | Identifies the "PSTN modem configuration" object instance. See 6.2.6. |
|---------------------|---|

| | |
|------------------------------|---|
| comm_speed | The communication speed between the device and the modem, not necessarily the communication speed on the WAN. enum: (0) 300 baud, (1) 600 baud, (2) 1 200 baud, (3) 2 400 baud, (4) 4 800 baud, (5) 9 600 baud, (6) 19 200 baud, (7) 38 400 baud, (8) 57 600 baud, (9) 115 200 baud |
| initialization_string | This data contains all the necessary initialization commands to be sent to the modem in order to configure it properly. This may include the configuration of special modem features. array initialization_string_element initialization_string_element ::= structure { request: octet-string, response: octet-string } If the array contains more than one initialization string element, they are subsequently sent to the modem after receiving an answer matching the defined response. REMARK It is assumed that the modem is pre-configured so that it accepts the initialization_string. If no initialization is needed, the initialization string is empty. |
| modem_profile | This data defines the mapping from Hayes standard commands/responses to modem specific strings. array modem_profile_element modem_profile_element: octet-string The <i>modem_profile</i> array shall contain the corresponding strings for the modem used in following order : Element 0: OK, Element 1: CONNECT, Element 2: RING, Element 3: NO CARRIER, Element 4: ERROR, Element 5: CONNECT 1 200, Element 6: NO DIAL TONE, Element 7: BUSY, Element 8: NO ANSWER, Element 9: CONNECT 600, Element 10: CONNECT 2 400, Element 11: CONNECT 4 800, Element 12: CONNECT 9 600, Element 13: CONNECT 14 400, Element 14: CONNECT 28 800, Element 15: CONNECT 36 600, Element 16: CONNECT 56 000 |

5.4.15 Auto answer (class_id = 28, version = 0)

This IC allows modelling how the device manages the “Auto answer” function of the modem, i.e. answering of incoming calls. Several modems can be configured.

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

Attribute description

logical_name Identifies the “Auto answer” object instance. See 6.2.6.

mode Defines the working mode of the line when the device is auto answer.

enum:

- (0) line dedicated to the device,
- (1) shared line management with a limited number of calls allowed. Once the number of calls is reached, the window status becomes inactive until the next start date, whatever the result of the call,
- (2) shared line management with a limited number of successful calls allowed. Once the number of successful communications is reached, the window status becomes inactive until the next start date,
- (3) currently no modem connected,
- (200...255) manufacturer specific modes

listening_window Contains the start and end instant when the window becomes active (for the start instant), and inactive (for the end instant). The start_date defines implicitly the period.

EXAMPLE When the day of month is not specified (equal to 0xFF) this means that we have a daily share line management. Daily, monthly ...window management can be defined.

array window_element

```
window_element ::= structure
{
    start_time: octet-string,
    end_time: octet-string
}
```

start_time and end_time are formatted as specified in 4.1.6.1 for *date-time*.

| | |
|------------------------|---|
| status | Here is defined the status of the window. enum: (0) Inactive: the device will manage no new incoming call. This status is automatically reset to Active when the next listening window starts, (1) Active: the device can answer to the next incoming call, (2) Locked: This value can be set automatically by the device or by a specific client when this client has completed its reading session and wants to give the line back to the customer before the end of the window duration. This status is automatically reset to Active when the next listening window starts. |
| number_of_calls | This number is the reference used in modes 1 and 2. When set to 0, this means there is no limit. |
| number_of_rings | Defines the number of rings before the meter connects the modem. Two cases are distinguished: number of rings within the window defined by attribute <i>listening_window</i> and number of rings outside the <i>listening_window</i> . nr_rings_type ::= structure { nr_rings_in_window: unsigned, (0: no connect in window) nr_rings_out_of_window: unsigned (0: no connect out of window) } |

5.4.16 PSTN auto dial (class id = 29, version = 0)

NOTE The name of this IC was changed to "Auto connect" in Edition 6.0.

An instance of the “PSTN auto dial” IC stores data related to the management data transfer between the device and the modem to perform auto dialling. Several modems can be configured. The logical names shall be as defined in 6.2.6.

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

Attribute description

| | |
|---------------------|---|
| logical_name | Identifies the “PSTN auto dial” object instance. See 6.2.6. |
| mode | <p>Defines if the device can perform auto-dialling.</p> <p>enum:</p> <ul style="list-style-type: none">(0) no auto dialling,(1) auto dialling allowed anytime,(2) auto dialling allowed within the validity time of the calling window,(3) “regular” auto dialling allowed within the validity time of the calling |

window; “alarm” initiated auto dialling allowed anytime,
 (200...255) manufacturer specific modes

| | |
|-------------------------|---|
| repetitions | The maximum number of trials in the case of unsuccessful dialling attempts. |
| repetition_delay | The time delay, expressed in seconds until an unsuccessful dial attempt can be repeated. repetition_delay 0 means delay is not specified |
| calling_window | Contains the start and end date/time stamp when the window becomes active (for the start instant), or inactive (for the end instant). The start_date defines implicitly the period. EXAMPLE When day of month is not specified (equal to 0xFF) this means that we have a daily share line management. Daily, monthly ...window management can be defined. array window_element window_element ::= structure { start_time: octet-string, end_time: octet-string } start_time and end_time are formatted as specified in 4.1.6.1 for <i>date-time</i> . |
| phone_list | Contains phone numbers, the device modem has to call under certain conditions. The link between entries in the array and the conditions are not contained in here. array phone_number phone_number ::= octet-string |

5.4.17 Auto connect (class_id = 29, version = 1)

This IC allows modelling the management of data transfer from the device to one or several destinations.

The messages to be sent, the conditions on which they shall be sent and the relation between the various modes, the calling windows and destinations are not defined here.

Depending on the mode, one or more instances of this IC may be necessary to perform the function of sending out messages.

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

Attribute description

| | |
|-------------------------|--|
| logical_name | Identifies the “Auto connect” object instance. See 6.2.6. |
| mode | Defines the mode controlling the auto dial functionality concerning the timing, the message type to be sent and the infrastructure to be used. enum: (0) no auto dialling, (1) auto dialling allowed anytime, (2) auto dialling allowed within the validity time of the calling window, (3) “regular” auto dialling allowed within the validity time of the calling window; “alarm” initiated auto dialling allowed anytime, (4) SMS sending via Public Land Mobile Network (PLMN), (5) SMS sending via PSTN, (6) email sending, (200..255) manufacturer specific modes |
| repetitions | The maximum number of trials in the case of unsuccessful dialling attempts. |
| repetition_delay | The time delay, expressed in seconds until an unsuccessful dial attempt can be repeated. repetition_delay 0 means delay is not specified |
| calling_window | Contains the start and end date/time stamp when the window becomes active (for the start instant), or inactive (for the end instant). The start_date defines implicitly the period. EXAMPLE When day of month is not specified (equal to 0xFF) this means that we have a daily share line management. Daily, monthly ...window management can be defined. array window_element window_element ::= structure { start_time: octet-string, end_time: octet-string } start_time and end_time are formatted as specified in 4.1.6.1 for <i>date-time</i> . |
| destination_list | Contains the list of destinations (for example phone numbers, email addresses or their combinations) where the message(s) have to be sent under certain conditions. The conditions and their link to the elements of the array are not defined here. array destination destination ::= octet-string |

5.4.18 S-FSK Phy&MAC setup (class_id = 50, version = 0)

NOTE 1 The use of this version is deprecated.

An instance of the “S-FSK Phy&MAC setup” IC stores the data necessary to set up and manage the physical and the MAC layer of the PLC S-FSK lower layer profile.

| S-FSK Phy&MAC setup | 0...n | class_id = 50, version = 0 | | | |
|--|------------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. initiator_electrical_phase (static) | enum | 0 | 3 | | x + 0x08 |
| 3. delta_electrical_phase (dyn.) | enum | 0 | 6 | | x + 0x10 |
| 4. max_receiving_gain (static) | unsigned | | | | x + 0x18 |
| 5. max_transmitting_gain (static) | unsigned | | | | x + 0x20 |
| 6. search_initiator_gain (static) | unsigned | | | | x + 0x28 |
| 7. frequencies (static) | frequencies_type | | | | x + 0x30 |
| 8. mac_address (dyn.) | long-unsigned | | | FFE | x + 0x38 |
| 9. mac_group_addresses (static) | array | | | | x + 0x40 |
| 10. repeater (static) | enum | | | 1 | x + 0x48 |
| 11. repeater_status (dyn.) | boolean | | | | x + 0x50 |
| 12. min_delta_credit (dyn.) | unsigned | | | | x + 0x58 |
| 13. initiator_mac_address (dyn.) | long-unsigned | | | | x + 0x60 |
| 14. synchronization_locked (dyn.) | boolean | | | | x + 0x68 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|-----------------------------------|---|
| logical_name | Identifies the “S-FSK Phy&MAC setup” object instance. See 6.2.23. |
| initiator_electrical_phase | Holds the MIB variable initiator-electrical-phase (variable 18) specified in IEC 61334-4-512:2001, 5.8. It is written by the client system to indicate the phase to which it is connected. enum: (0) Not defined (default), (1) Phase 1, (2) Phase 2, (3) Phase 3. NOTE 2 This enumeration is different from that of IEC 61334-4-512:2001. |
| delta_electrical_phase | Holds the MIB variable delta-electrical-phase (variable 1) specified in IEC 61334-4-512:2001, 5.2 and IEC 61334-5-1:2001, 3.5.5.3. It indicates the phase difference between the client's connecting phase and the server's connecting phase. The following values are predefined: enum: (0) Not defined: the server is temporarily not able to determine the phase difference, (1) The server system is connected to the same phase as the client system. The phase difference between the server's connecting phase and the client's connecting phase is equal to: (2) 60 degrees, (3) 120 degrees, (4) 180 degrees, (5) -120 degrees, (6) -60 degrees. |

| | |
|------------------------------|---|
| max_receiving_gain | Holds the MIB variable max-receiving-gain (variable 2) specified in IEC 61334-4-512:2001, 5.2 and in IEC 61334-5-1:2001, 3.5.5.3. Corresponds to the maximum allowed gain bound to be used by the server system in the receiving mode. The default unit is dB. NOTE 3 In IEC 61334-4-512:2001, no units is specified. The possible values of the gain may depend on the hardware. Therefore, after writing a value to this attribute, the value should be read back to know the actual value. |
| max_transmitting_gain | Holds the value of the max-transmitting-gain. Corresponds to the maximum attenuation bound to be used by the server system in the transmitting mode. The default unit is dB. The possible values of the gain may depend on the hardware. Therefore, after writing a value to this attribute, the value should be read back to know the actual value. |
| search_initiator_gain | This attribute is used in the intelligent search initiator process. If the value of the <i>max_receiving_gain</i> is below the value of this attribute, a fast synchronization process is possible. |
| frequencies | Contains frequencies required for S-FSK modulation. frequencies_type ::= structure { mark_frequency: double-long-unsigned, space_frequency: double-long-unsigned } The default unit is Hz. |
| mac_address | Holds the MIB variable mac-address (variable 3) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. NOTE 4 MAC addresses are expressed on 12 bits. Contains the value of the address of the physical attachment (MAC address) associated to the local system. In the unconfigured state, the MAC address is "NEW-address". This attribute is locally written by the CIASE when the system is registered (with a Register service). The value is used in each outgoing or incoming frame. The default value is "NEW-address". This attribute is set to NEW: <ul style="list-style-type: none">- by the MAC sub-layer, once the time-out-not-addressed delay is exceeded;- when a client system "resets" the server system. See the "S-FSK Active initiator" IC in Clause 4.10.4. When this attribute is set to NEW: <ul style="list-style-type: none">- the system loses its synchronization (function of the MAC-sublayer);- the <i>mac_group_address</i> attribute is reset (array of 0 elements);- the system automatically releases all AAs which can be released. NOTE 5 The second item is not present in IEC 61334-4-512:2001. The predefined MAC addresses are shown in Table 33. |

| | |
|----------------------------|--|
| mac_group_addresses | Holds the MIB variable mac-group-address (variable 4) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. Contains a set of MAC group addresses used for broadcast purposes. array mac-address mac-address ::= long-unsigned The ALL-configured-address, ALL-physical-address and NO-BODY addresses are not included in this list. These ones are internal predefined values. This attribute shall be written by the initiator using DLMS services to declare specific MAC group addresses on a server system. This attribute is locally read by the MAC sublayer when checking the destination address field of a MAC frame not recognized as an individual address or as one of the three predefined values (ALL-configured-address, ALL-physical-address and NO-BODY). |
| repeater | Holds the MIB variable repeater (variable 5) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. It specifies whether the server system effectively repeats all frames or not. enum: (0) never repeater, (1) always repeater, (2) dynamic repeater If the repeater variable is equal to 0, the server system should never repeat the frames. If it is set to 1, the server system is a repeater: it has to repeat all frames received without error and with a current credit greater than zero. If it is set to 2, then the repeater status can be dynamically changed by the server itself. NOTE 6 The value 2 value is not specified in IEC 61334-4-512:2001. This attribute is internally read by the MAC sub-layer each time a frame is received. The default value is 2, and the server system is a repeater (<i>repeater_status</i> = TRUE). |
| repeater_status | Holds the current repeater status of the device. boolean: FALSE = no repeater, TRUE = repeater |
| min_delta_credit | Holds the MIB variable min-delta-credit (variable 9) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6. NOTE 7 Only the three least significant bits are used. The Delta Credit (DC) is the subtraction of the Initial Credit (IC) and Current Credit (CC) fields of a correct received MAC frame. The delta-credit minimum value of a correct received MAC frame, directed to a server system, is held by this variable. The default value is set to the maximal initial credit (see IEC 61334-5-1:2001 4.2.3.1 for further explanations on the credit and the value of MAX_INITIAL_CREDIT). A client system can reinitialise this variable by setting its value to the maximal initial credit. |

| | |
|-------------------------------|---|
| initiator_mac_address | Holds the MIB variable initiator-mac-address specified in IEC 61334-5-1:2001, 4.3.7.6. Its value is either the MAC address of the active-initiator or the NO-BODY address, depending on the value of the synchronization_locked attribute (see below). See also IEC 61334-5-1:2001, 3.5.3, 4.1.6.3 and 4.1.7.2. If the value NO-BODY is written then the server mac_address (see the <i>mac_address</i> attribute) has to be set to NEW. |
| synchronization_locked | Holds the MIB variable synchronization-locked (variable 10) specified in IEC 61334-4-512:2001, 5.3. Controls the synchronization locked / unlocked state. See IEC 61334-5-1:2001 for more details. If the value of this attribute is equal to TRUE, the system is in the synchronization-locked state. In this state, the initiator-mac-address is always equal to the MAC address field of the active-initiator MIB object. See attribute 2 of the S-FSK Active initiator IC, in 4.10.4. If the value of this attribute is equal to FALSE, the system is in the synchronization-unlocked state. In this state, the <i>initiator_mac_address</i> attribute is always set to the NO-BODY value: a value change in the MAC address field of the active-initiator MIB object does not affect the content of the <i>initiator_mac_address</i> attribute which remains at the NO-BODY value. The default value of this variable shall be specified in the implementation specifications. NOTE 8 In the synchronization-unlocked state, the server synchronizes on any valid frame. In the synchronization locked state, the server only synchronizes on frames issued or directed to the client system the MAC address of which is equal to the value of the <i>initiator_mac_address</i> attribute. |

5.4.19 S-FSK IEC 61334-4-32 LLC setup (class_id = 55, version = 0)

An instance of the “S-FSK IEC 61334-4-32 LLC setup” IC holds parameters necessary to set up and manage the LLC layer as specified in IEC 61334-4-32:1996.

| S-FSK IEC 61334-4-32 LLC setup | 0...n | class_id = 55, version = 0 | | | |
|--------------------------------|--------------|----------------------------|------|------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | x |
| 2. max_frame_length (static) | unsigned | 26 | 242 | 134 | x + 0x08 |
| 3. reply_status_list (dyn.) | array | | | | x + 0x10 |
| Specific methods | m/o | | | | |

Attribute description

| | |
|-------------------------|---|
| logical_name | Identifies the “S-FSK IEC 61334-4-32 LLC setup” object instance. See 6.2.23. |
| max_frame_length | Holds the length of the LLC frame in bytes. See IEC 61334-4-32:1996, 5.1.4. In the case of the S-FSK profile, as specified in IEC 61334-5-1:2001, 4.2.2, the maximum value is 242, but lower values may be chosen due to performance considerations. |

| | |
|--------------------------|---|
| reply_status_list | Holds the MIB variable <i>reply-status-list</i> (variable 11) specified in IEC 61334-4-512:2001, 5.4. Lists the L-SAPs that have a not empty RDR (Reply Data on Request) buffer, which has not already been read. The length of a waiting L-SDU is specified in number of sub frames (different from zero). The variable is locally generated by the LLC sub layer. array reply_status |
| | <pre>reply_status ::= structure { L-SAP-selector: unsigned, length-of-waiting-L-SDU: unsigned }</pre> <p>length-of-waiting-LSDU in the case of the S-FSK profile is in number of sub-frames; valid values are 1 to 7.</p> |

5.4.20 M-Bus client (class_id = 72, version = 0)

Instances of this IC allow setting up M-Bus slave devices and to exchange data with them. Each M-Bus client object controls one M-Bus slave device. For details on the M-Bus dedicated application layer, see EN 13757-3:2004.

The M-Bus client device may have one or more physical M-Bus interfaces, which can be configured using instances of the M-Bus master port setup IC, see 4.8.5.

An M-Bus slave device is identified with its Primary Address, Identification Number, Manufacturer ID etc. as defined in EN 13757-3:2004 Clause 5, *Variable Data respond*. These parameters are carried by the respective attributes of the M-Bus client IC, see 5.4.17.

Values to be captured from an M-Bus slave device are identified by the *capture_definition* attribute, containing a list of data identifiers (DIB, VIB) for the M-Bus slave device. The data are captured periodically or on an appropriate trigger. Each data element is stored in an M-Bus value object, of IC “Extended register”. M-Bus value objects may be captured in M-Bus Profile generic objects, eventually along with other, not M-Bus specific objects.

Using the methods of M-Bus client objects, M-Bus slave devices can be installed and de-installed. It is also possible to send data to M-Bus slave devices and to perform operations like resetting alarms, setting the clock, transferring an encryption key etc.

NOTE 1 In Edition 10, the mapping of attributes to short name has been corrected, without a change in the version.

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

Attribute description

| | |
|----------------------------|--|
| logical_name | Identifies the “M-Bus client” object instance. See 6.2.21. |
| mbus_port_reference | Provides reference to an “M-Bus master port setup” object, used to configure an M-Bus port, each interface allowing to exchange data with one or more M-Bus slave devices. |
| capture_definition | <p>Provides the capture_definition for M-Bus slave devices.</p> <p>array capture_definition_element</p> <p>capture_definition_element ::= structure</p> <pre>{ data_information_block: octet-string, value_information_block: octet-string }</pre> <p>NOTE 2 The elements data_information_block and value_information_block correspond to Data Information Block (DIB) and Value Information Block (VIB) described in EN 13757-3:2013, 6.2 and clause 7 respectively.</p> |
| capture_period | <p>>= 1: Automatic capturing assumed. Specifies the capture period in seconds.</p> <p>0: No automatic capturing: capturing is triggered externally or capture events occur asynchronously.</p> |

| | |
|------------------------------|--|
| primary_address | Carries the primary address of the M-Bus slave device, in the range 0...250. If the slave device is already configured and thus, its primary address is different from 0, then this value shall be written to the <i>primary_address</i> attribute. From this moment, the data exchange with the M-Bus slave device is possible. Otherwise, the <i>slave_install</i> method shall be used; see below. NOTE 3 The <i>primary_address</i> attribute cannot be used to store a desired primary address for an unconfigured slave device. If the primary address attribute is set, this means that the M-Bus client can immediately operate with this primary address, which is not the case with an unconfigured slave device. |
| identification_number | Carries the Identification Number element of the data header as specified in EN 13757-3:2004, 5.4. It is either a fixed fabrication number or a number changeable by the customer, coded with 8 BCD packed digits (4 Byte), and which thus runs from 00 000 000 to 99 999 999. It can be preset at fabrication time with a unique number, but could be changeable afterwards, especially if in addition a unique and not changeable fabrication number (DIF = 0x0C, VIF = 0x78, see 7.2 is provided. |
| manufacturer_id | Carries the Manufacturer Identification element of the data header as specified in EN 13757-3:2004, 5.5. It is coded unsigned binary with 2 bytes. This <i>manufacturer_id</i> is calculated from the ASCII code of the IEC 62056-21 manufacturer ID (three uppercase letters), using the formula specified in EN 13757-3:2004, 5.5. |
| version | Carries the Version element of the data header as specified in EN 13757-3:2004, 5.6. It specifies the generation or version of the meter and depends on the manufacturer. It can be used to make sure, that within each version number the identification # is unique. |
| device_type | Carries the Device type identification element of the data header as specified in EN 13757-3:2004, 5.7, Table 3. |
| access_number | Carries the Access Number element of the data header as specified in EN 13757-3:2004, 5.8. It has unsigned binary coding, and it is incremented (modulo 256) by one before or after each RSP-UD from the slave. Since it can also be used to enable private end users to detect an unwanted over-frequently readout of their consumption meters, it should not be resettable by any bus communication. |
| status | Carries the Status byte element of the data header as specified in EN 13757-3:2004, 5.9, Table 4 and 5. |
| alarm | Carries the Alarm state specified in EN 13757-3:2004 Annex D. It is coded with data type D (Boolean, in this case 8 bit). Set bits signal alarm bits or alarm codes. The meaning of these bits is manufacturer specific. |

Method description

| | |
|-----------------------------|---|
| slave_install (data) | Installs a slave device, which is yet unconfigured (its primary address is 0). This method can be successfully invoked only if the value of the <i>primary_address</i> attribute is 0. The following actions are performed: <ul style="list-style-type: none">- the M-Bus address 0 is checked for presence of a new device.- if no uninstalled M-Bus slave is found, the method invocation fails;- if the <i>slave_install</i> method is invoked with no parameter, then the primary address is assigned automatically. This is done by checking the <i>primary_address</i> attribute of all M-Bus client objects in the DLMS/COSEM device and then selecting the first unused number. The <i>primary_address</i> attribute is set to this address and it is then transferred to the M-Bus slave device; |
|-----------------------------|---|

- if the *slave_install* method is invoked with a primary address as a parameter, then the *primary_address* attribute is set to this value and it is then transferred to the M-Bus slave device.

data ::= unsigned (no data, or a valid primary address)

NOTE 4 Unconfigured slave devices are configured with primary address as specified in EN 13757-3:2004, Annex E.5.

| | |
|-------------------------------|---|
| slave_deinstall (data) | De-installs the slave device. The main purpose of this service is to uninstall the M-Bus slave device and to prepare the master for the installation of a new device. |
|-------------------------------|---|

The following actions are performed:

- the M-Bus address is set to 0 in the M-Bus slave device;
- the encryption key transferred previously to the M-Bus slave device is destroyed; the default key is not affected.
- the attribute *primary_address* is also set to 0.

NOTE 5 A new M-Bus slave can be installed only, once the value of the *primary_address* attribute is 0.

data ::= integer (0)

| | |
|----------------|---|
| capture | Capture values – as specified by the <i>capture_definition</i> attribute – from the M-Bus slave device. |
|----------------|---|

data ::= integer (0)

| | |
|--------------------|--|
| reset_alarm | Reset alarm state of the M-Bus slave device. |
|--------------------|--|

data ::= integer (0)

| | |
|--------------------------|---|
| synchronize_clock | Synchronize the clock of the M-Bus slave device with that of the M-Bus client device. |
|--------------------------|---|

data ::= integer (0)

| | |
|------------------|---|
| data_send | Send data to the M-Bus slave device. data ::= array data_definition_element data_definition_element ::= structure { data_information_block: octet-string, value_information_block: octet-string data: CHOICE { -- simple data types null-data [0], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], utf8-string [12], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], float32 [23], float64 [24] } } |
|------------------|---|

| | |
|---------------------------|---|
| set_encryption_key | <p>Sets encryption key to be used with the M-Bus slave device.</p> <p>Changing the encryption key requires two steps: First, the key is sent to the M-Bus slave, encrypted with the master key, using the <i>transfer_key</i> method. Second, the key is set in the M-Bus master using the <i>set_encryption_key</i> method.</p> <p>data ::= octet-string (encryption_key)</p> <p>After the installation of the M-Bus slave, the M-Bus client holds an empty encryption key. With this, encryption of M-Bus telegrams is disabled. Encryption can be disabled by invoking the <i>set_encryption_key</i> method with null- data as a parameter.</p> |
| transfer_key | <p>Transfers an encryption key to the M-Bus slave.</p> <p>data ::= octet-string (encrypted_key)</p> <p>Each M-Bus slave device shall be delivered with a default encryption key.</p> <p>Before encrypted M-Bus telegrams can be used, an operational encryption key has to be sent to the M-Bus slave, by invoking the <i>transfer_key</i> method. The method invocation parameter is the operational encryption key encrypted with the default key of the M-Bus slave device. The M-Bus telegram sent is not encrypted. After the execution, the encryption is enabled and all further telegrams are encrypted.</p> <p>A new encryption key can be set in the M-Bus client by invoking the <i>set_encryption_key</i> method with the new encryption key as a parameter.</p> <p>With further invocations of the <i>transfer_key</i> method, new encryption keys can be sent to the M-Bus slave. The method invocation parameter is the new encryption key encrypted with the default key. The M-Bus telegram is encrypted.</p> <p>When an M-Bus slave is de-installed, the encryption key is destroyed, but the default key is not affected. Encryption remains disabled until a new encryption is transferred.</p> |

5.4.21 G3 NB OFDM PLC MAC layer counters (class_id = 90, version = 0)

An instance of the “G3 NB OFDM PLC MAC layer counters” IC stores counters related to the MAC layer exchanges. The objective of these counters is to provide statistical information for management purposes.

The attributes of instances of this IC shall be read only. They can be reset using the reset method.

| G3 NB OFDM PLC MAC layer counters | | 0...n | class_id = 90, version = 0 | | | |
|--|----------|----------------------|-----------------------------------|---------------|-------------|-------------------|
| Attributes | | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name | (static) | octet-string | | | | x |
| 2. mac_Tx_data_packet_count | (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x08 |
| 3. mac_Rx_data_packet_count | (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x10 |
| 4. mac_Tx_cmd_packet_count | (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x18 |
| 5. mac_Rx_cmd_packet_count | (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x20 |
| 6. mac_CSMA_fail_count | (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x28 |
| 7. mac_CSMA_no_ACK_count | (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x30 |
| 8. mac_bad_CRC_count | (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x38 |
| 9. mac_broadcast_count | (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x40 |
| 10. mac_multicast_count | (dyn.) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x48 |
| Specific methods | | m/o | | | | |
| 1. reset (data) | | o | | | | |

Attribute description

| | |
|---------------------------------|--|
| logical_name | Identifies the "G3 NB OFDM PLC MAC layer counters" object instance. See 6.2.26. |
| mac_Tx_data_packet_count | PIB attribute 0x02000101: Statistic counter of successfully transmitted data packets (MSDUs). |
| mac_Rx_data_packet_count | PIB attribute 0x02000102: Statistic counter of successfully received data packets (MSDUs). |
| mac_Tx_cmd_packet_count | PIB attribute 0x02000201: Statistic counter of successfully transmitted command packets. |
| mac_Rx_cmd_packet_count | PIB attribute 0x02000202: Statistic counter of successfully received command packets. |
| mac_CSMA_fail_count | PIB attribute 0x02000103: Counts the number of times when CSMA backoffs exceed macMaxCSMABackoffs. |
| mac_CSMA_no_ACK_count | PIB attribute 0x02000104: Counts the number of times when an ACK is not received while transmitting an unicast data frame (The loss of ACK is attributed to collisions). |
| mac_bad_CRC_count | PIB attribute 0x02000108: Statistic counter of the number of frames received with bad CRC. |
| mac_broadcast_count | PIB attribute 0x02000106: Statistic counter of the number of broadcast frames sent. |
| mac_multicast_count | PIB attribute 0x02000107: Statistic counter of the number of multicast frames sent. |

NOTE When a counter reaches the maximum value (0xFFFFFFFF), it's automatically rolled-over.

Method description

| | |
|---------------------|---|
| reset (data) | This method forces a reset of the object. By invoking this method, the value of all counters is set to 0. data ::= integer (0) |
|---------------------|---|

5.4.22 G3 NB OFDM PLC MAC setup (class_id = 91, version = 0)

An instance of the “G3 NB OFDM PLC MAC setup” IC holds the necessary parameters to set up and manage the G3 NB OFDM IEEE 802.15.4 MAC sub-layer.

These attributes influence the functional behaviour of an implementation. Implementations may allow changes to the attributes during normal running, i.e. even after the device start-up sequence has been executed.

| G3 NB OFDM PLC MAC setup | 0...n | class_id = 91, version = 0 | | | |
|--|----------------------|----------------------------|---------------|--------|------------|
| Attributes | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name (static) | octet-string | | | | X |
| 2. mac_short_address (dyn.) | long-unsigned | 0x0000 | 0xFFFF | 0xFFFF | x + 0x08 |
| 3. mac_coord_short_address (dyn.) | long-unsigned | 0x0000 | 0xFFFF | 0x0000 | x + 0x10 |
| 4. mac_PAN_id (dyn.) | long-unsigned | 0x0000 | 0xFFFF | 0xFFFF | x + 0x18 |
| 5. mac_max_orphan_timer (static) | double-long-unsigned | 0 | 4 294 967 295 | 0 | x + 0x20 |
| 6. mac_security_enabled (static) | boolean | | | TRUE | x + 0x28 |
| 7. mac_freq_notching (static) | boolean | | | FALSE | x + 0x30 |
| 8. mac_TMR_TTL (static) | double-long-unsigned | 0 | 262 143 | 120 | x + 0x38 |
| 9. mac_max_frame_retries (static) | unsigned | 0 | 10 | 5 | x + 0x40 |
| 10. mac_neighbour_table_entry_TTL (static) | double-long-unsigned | 0 | 262 143 | 15 300 | x + 0x48 |
| 11. mac_neighbour_table (dyn.) | array | | | | |
| Specific methods | m/o | | | | |
| 1. mac_get_neighbour_table_entry (data) | o | | | | |

Attribute description

| | |
|--------------------------------|--|
| logical_name | Identifies the “G3 NB OFDM PLC MAC setup” object instance. See 6.2.26. |
| mac_short_address | PIB attribute 0x01000112: Device short address. The 16-bit address the device is using to communicate on the PAN. Its value shall be equal to 0xFFFF when the device does not have a short address. An associated device necessarily has a short address, so that a device cannot be in the state where it is associated but does not have a short address. |
| mac_coord_short_address | PIB attribute 0x01000107: Coordinator short address. NOTE 1 The short address assigned to the coordinator through which the device is associated. |
| mac_PAN_id | PIB attribute 0x0100010F: PAN ID. NOTE 2 The 16-bit identifier of the PAN on which the device is operating. A value equal to 0xFFFF indicates that the device is not associated. |
| mac_max_orphan_timer | PIB attribute 0x02000109: The maximum number of seconds without communication with a particular device after which it is declared as an orphan. |
| mac_security_enabled | PIB attribute 0x01000111: Security enabled. boolean: TRUE : security enabled, FALSE: security disabled |

| | |
|--------------------------------------|---|
| mac_freq_notching | PIB attribute 0x0000006D: S-FSK 63 and 74 kHz frequency notching. boolean: TRUE : notching enabled, FALSE: notching disabled Default value is FALSE (disabled). |
| mac_TMR_TTL | PIB attribute 0x02000113: Maximum valid time of tone map parameters in the neighbour table in seconds. |
| mac_max_frame_retries | PIB attribute 0x0100010D: Maximum number of retransmissions. |
| mac_neighbour_table_entry_TTL | PIB attribute 0x02000114: Maximum valid time for an entry in the neighbour table in seconds. |
| mac_neighbour_table | PIB attribute 0x0000006B: See ITU-T G.9903 Amd. 1:2013 clause 9.3.7.2 for CENELEC and FCC bands. The neighbour table contains information about all the devices within the POS of the device. One element of the table represents one PLC direct neighbour of the device. array mac_neighbour_table_element mac_neighbour_table_element ::= structure { mac_short_address: long-unsigned, payload_modulation_scheme: boolean, tone_map: bit-string, modulation: enum, tx_gain: integer, tx_res: boolean, tx_coeff: array, lqi: unsigned, TMR_valid_time: double-long-unsigned, neighbour_valid_time: double-long-unsigned } NOTE 3 This table is actualized each time any frame is received from a neighbour device, and each time a Tone Map Response is received. |
| short_address | The MAC Short Address of the node which this entry refers to. |
| payload_modulation_scheme | 0: Differential, 1: Coherent |
| tone_map | The Tone Map parameter defines which frequency sub-band can be used for communication with the device. A bit set to 1 means that the frequency sub-band can be used, and a bit set to 0 means that frequency sub-band shall not be used. |
| modulation | The modulation type to use for communicating with the device. enum: (0) Robust Mode (1) DBPSK (2) DQPSK (3) D8PSK (4) 16-QAM |

| | |
|----------------------|---|
| | NOTE 4 The 16-QAM modulation is optional and only applicable for FCC band. |
| tx_gain | Defines the Tx Gain to use to transmit frames to that device. |
| tx_res | Defines the Tx Gain resolution corresponding to one gain step. 0 : 6 dB 1 : 3 dB |
| tx_coeff | A parameter that specifies transmitter gain for each group of tones represented by one valid bit of the tone map. The receiver measures the frequency-dependent attenuation of the channel and may request the transmitter to compensate for this attenuation by increasing the transmit power on sections of the spectrum that are experiencing attenuation in order to equalize the received signal. Each group of tones is mapped to a 4-bit value for CENELEC-A or a 2-bit value for FCC where a "0" in the most significant bit indicates a positive gain value, hence an increase in the transmitter gain scaled by TXRES is requested for that section and a "1" indicates a negative gain value, hence a decrease in the transmitter gain scaled by TXRES is requested for that section. Implementing this feature is optional and it is intended for frequency selective channels. If this feature is not implemented, the value zero shall be used. NOTE 5 One group of tones gathers 6 consecutive tones (or carriers) for CENELEC bands, and 3 consecutive tones for FCC band. |
| lqi | Link quality indicator NOTE 6 LQI value measured during reception of the PPDU from the neighbour. The LQI measurement is a characterization of the strength and/or quality of a received packet. |
| TMR_valid_time | Remaining time in seconds until which the tone map response parameters in the neighbour table are considered valid. - When the entry is created, this value shall be set to the default value 0. - When it reaches 0, a tone map request may be issued if data is sent to this device. Upon successful reception of a tone map response, this value is set to 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. |

Method description

| | |
|--|--|
| mac_get_neighbour_table_entry(data) | This method is used to retrieve the mac neighbour table for one MAC short address. It may be used to perform topology monitoring by the client. The method invocation parameter contains a mac_short_address. data ::= long-unsigned |
| The response parameter includes the neighbour table for this mac_short_address. | |
| data ::= array mac_neighbour_table_element where mac_neighbour_table-element is as defined in the <i>mac_neighbour_table</i> attribute of the present IC. | |

5.4.23 G3 NB OFDM PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 0)

An instance of the “G3 NB OFDM PLC 6LoWPAN adaptation layer setup” IC holds the necessary parameters to set up and manage the G3 NB OFDM PLC 6LoWPAN Adaptation layer.

These attributes influence the functional behaviour of an implementation. Implementations may allow changes to their values during normal running, i.e. even after the device start-up sequence has been executed.

| G3 NB OFDM PLC 6LoWPAN adaptation layer setup | | 0...n | class_id = 92, version = 0 | | | |
|---|------------|---------------|----------------------------|-------|------------------------------|------------|
| Attributes | | Data type | Min. | Max. | Def. | Short name |
| 1. logical_name | (static) | octet-string | | | | x |
| 2. adp_max_hops | (static) | unsigned | 1 | 14 | 8 | x + 0x10 |
| 3. adp_weak_LQI_value | (static) | unsigned | 0 | 255 | 3 for CENELEC A 5 for FCC | x + 0x18 |
| 4. adp_tone_mask | (static) | bit-string | | | All bits set to 1 | x + 0x20 |
| 5. adp_discovery_attempts_wait_time | (static) | long-unsigned | 1 | 3 600 | 60 | x + 0x28 |
| 6. adp_routing_configuration | (static) | array | | | | x + 0x30 |
| 7. adp_broadcast_log_table_entry_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 | <i>m/o</i> | | | | | |

Attribute description

| | |
|---|---|
| logical_name | Identifies the “G3 NB OFDM 6LoWPAN adaptation layer setup” object instance. See 6.2.26 |
| adp_max_hops | PIB attribute 0x10: Defines the maximum number of hops to be used by the routing algorithm. |
| adp_weak_LQI_value | PIB attribute 0x1B: The weak link value defines the threshold below which a direct neighbour is not taken into account during the commissioning procedure (compared to the LQI measured). |
| adp_tone_mask | PIB attribute 0x0F: Defines the Tone Mask to use during symbol formation. The length of the bit-string is 70 bits. |
| adp_discovery_attempts_wait_time | PIB attribute 0x06: Allows programming the maximum wait time between invocations of two consecutive network discovery primitives (in seconds). |
| adp_routing_configuration | <p>The routing configuration element specifies all parameters linked to the routing mechanism described in ITU-T G.9903 Amd. 1:2013. The elements are specified in clause 9.4.1.2 of that Recommendation.</p> <p>NOTE 1 The Link cost calculation is provided in ITU-T G.9903 Amd. 1:2013 Annex B.</p> <p>array routing_configuration</p> <pre>routing_configuration ::= structure { adp_net_traversal_time: long-unsigned, adp_routing_table_entry_TTL: double-long-unsigned, adp_Kr: unsigned, adp_Km: unsigned, adp_Kc: unsigned, adp_Kq: unsigned, adp_Kh: unsigned, adp_Krt: unsigned, adp_RREQ_retries: unsigned, adp_RREQ_RERR_wait: long-unsigned, adp_blacklist_table_entry_TTL: long-unsigned }</pre> <p>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</p> <p>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</p> <p>adp_Kr PIB attribute 0x14: A weight factor for ROBO to calculate link cost. Range: 0-31 Default value: 0</p> <p>adp_Km PIB attribute 0x15: A weight factor for modulation to calculate link cost. Range: 0-31 Default value: 0</p> |

| | |
|--|--|
| 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 |
| adp_broadcast_log_table_entry_TTL | PIB attribute 0x02: Defines the time while an entry in the adpBroadcastLogTable remains active in the table (in seconds). |
| adp_routing_table | PIB attribute 0x0C: The routing table contains information about the different routes in which the device is implicated. array routing_table routing_table ::= structure { destination_address: long-unsigned, next_hop_address: long-unsigned, route_cost: long-unsigned, hop_count: unsigned, weak_link_count: unsigned, status: enum, valid_time: unsigned } NOTE 2 This table is actualized each time a route is built or updated (triggered by data traffic) and each time the TTL timer expires. |

| | |
|--------------------------------------|---|
| destination_address | Address of the destination. |
| next_hop_address | Address of the next hop on the route towards the destination. |
| route_cost | Cumulative link cost along the route towards the destination. |
| hop_count | Number of hops of the selected route to the destination. Range: 0-15 NOTE 3 Practically the maximum allowed value is limited by adp_max_hops. |
| weak_link_count | Number of weak links to destination. Range : 0-15 |
| NOTE 4 | practically the maximum allowed value is limited by adp_max_hops. |
| status | Status of the routing table entry. enum : (0) Invalid route, (1) Valid route, (2) Route under construction |
| valid_time | Remaining time in seconds until which this entry in the routing table is considered valid. |
| adp_context_information_table | <p>PIB attribute 0x07: Contains the context information associated to each CID extension field.</p> <p>array context_information_table</p> <p>context_information_table ::= structure</p> <pre>{ CID: bit-string, context_length: unsigned, context: octet-string, C: boolean, valid_lifetime: long-unsigned }</pre> <p>CID Corresponds to the 4-bit context information used for source and destination addresses (SCI, DCI). Range: 0x00-0x0F</p> <p>context_length Indicates the length of the carried context (up to 128-bit contexts may be carried). Range: 0-128</p> <p>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</p> <p>C Indicates if the context is valid for use in compression 0 : Only decompression is allowed, 1 : Compression and decompression are allowed</p> <p>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.</p> |

| | | |
|--------------------------------|--|--|
| | valid_lifetime | Remaining time in minutes during which the context information table is considered valid. It is updated upon reception of the advertised context. Range: 0-65 535 |
| adp_blacklist_table | PIB attribute 0x25: Contains the list of the blacklisted neighbours. | |
| | array blacklisted_neighbour_set | |
| | blacklisted_neighbour_set ::= structure | |
| | { | |
| | blacklisted_neighbour_address: long-unsigned, | |
| | valid_time: long-unsigned | |
| | } | |
| | blacklisted_neighbour_address | The 16-bit address of the blacklisted neighbour. |
| | valid_time | Remaining time in minutes until which this entry in the blacklisted neighbour table is considered valid. |
| adp_broadcast_log_table | PIB attribute 0x0B: Contains the broadcast log table. | |
| | NOTE 5 This table provides a list of the broadcast packets recently received by this device. | |
| | array broadcast_log_table | |
| | broadcast_log_table ::= structure | |
| | { | |
| | source_address: long-unsigned, | |
| | sequence_number: unsigned, | |
| | time_to_live: long-unsigned | |
| | } | |
| | source_address | The 16-bit source address of a broadcast packet. This is the address of the broadcast initiator. |
| | sequence_number | The sequence number contained in the BC0 header. |
| | time_to_live | The remaining time to live of this entry in the broadcast log table, in seconds. |
| adp_group_table | PIB attribute 0x0E: Contains the group addresses to which the device belongs. | |
| | array group_table | |
| | group_table ::= structure | |
| | { | |
| | group_address: long-unsigned | |
| | } | |
| | group_address | Group address to which this node has been subscribed. |
| adp_max_join_wait_time | PIB attribute 0x21: Network join timeout in seconds for LBD. | |
| adp_path_discovery_time | PIB attribute 0x22: Timeout for path discovery in msec. | |
| adp_use_new_GMK_time | PIB attribute 0x23: The wait time in seconds for a device to use new GMK after rekeying. | |
| adp_exp_prec_GMK_time | PIB attribute 0x24: The time in seconds to keep PrecGMK after switching to a new GMK. | |

6 Relation to OBIS

6.1 General

This clause 6 specifies the use of COSEM interface objects to model various data items.

It also specifies the logical names of the objects. The naming system is based on OBIS, the Object Identification System: each logical name is an OBIS code.

OBIS codes are specified in the following clauses:

- 6.2 specifies the use and the logical names of abstract COSEM objects, i.e. objects not related to an energy type;
 - 6.3 specifies the use and logical names for electricity related COSEM objects;
 - 6.4 specifies the use and logical names for gas related COSEM objects;
- NOTE The use and the logical names of COSEM objects related to other media / energy types are under consideration.
- the detailed OBIS code allocations – for all media / energy types – are specified in clause 7.

Unless otherwise specified the use of value group B shall be:

- if just one object is instantiated, the value in value group B shall be 0;
- if more than one object is instantiated in the same physical device, the value group B shall number the measurement or communication channels as appropriate, from 1...64. This is indicated by the letter "b".

Unless otherwise specified the use of value group E shall be:

- if just one object is instantiated, value in value group E shall be 0;
- if more than one object is instantiated in the same physical device, the value group E shall number the instantiations from zero to the maximum value needed. This is indicated by the letter "e". For the values allocated, see clause 7.

All codes, which are not explicitly listed, but which are outside the manufacturer, utility or consortia specific ranges are reserved for future use.

6.2 Abstract COSEM objects

6.2.1 Use of value group C

Table 37 shows the use of value group C for abstract objects in the COSEM context. See also Table 53.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 352/492 |
|-----------------------|------------|-------------------------|---------|

Table 37 – 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" |
| 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" |
| | |
| 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" |
| 26 | COSEM objects for data exchange using S-FSK PLC networks |
| 27 | COSEM objects for ISO/IEC 8802-2 LLC layer setup |
| 28 | COSEM objects for data exchange using narrow-band OFDM PLC for PRIME networks |
| 29 | COSEM objects for data exchange using narrow-band OFDM PLC for G3-PLC networks |
| 30 | COSEM objects for data exchange using ZigBee® |
| 31 | Instances of IC "Wireless Mode Q" (M-Bus) |
| | |
| 40 | Instances of IC "Association SN/LN" |
| 41 | Instances of IC "SAP assignment" |
| 42 | COSEM logical device name |
| 43 | COSEM objects related to security: Instances of IC "Security setup" and "Data protection" |
| 44 | Instances of IC "Image transfer" |
| | |
| 65 | Instances of IC "Utility tables" |
| 66 | Instances of "Compact data" |
| 128...199 | Manufacturer specific COSEM related abstract objects |
| All other | Reserved |

6.2.2 Data of historical billing periods

COSEM provides three mechanisms to represent values of historical billing periods. This is shown in Figure 29 and is described below:

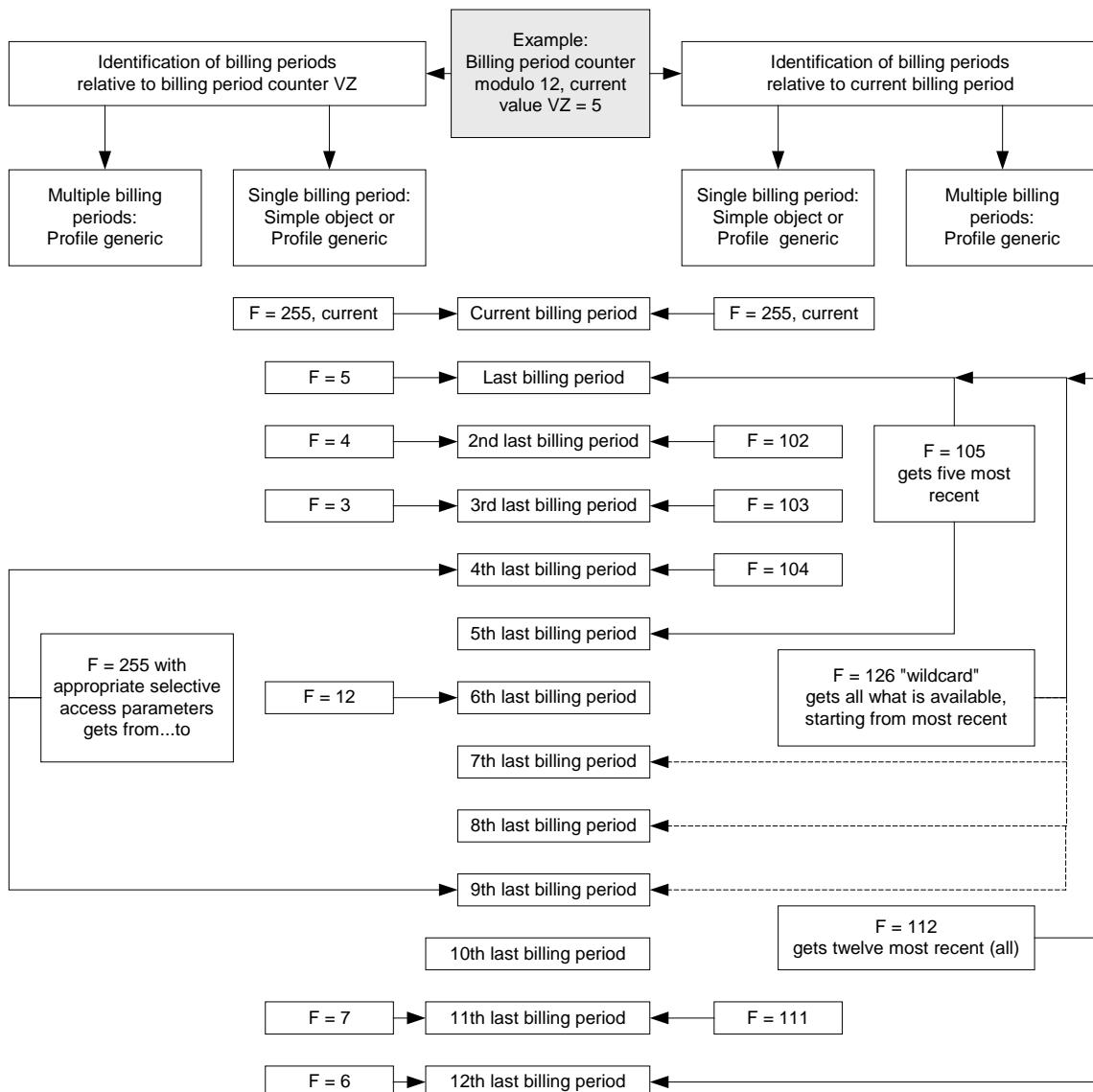


Figure 29 – Data of historical billing periods – example with module 12, VZ = 5

- 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;
- when values of historical billing periods are represented by “Profile generic” objects, more than one billing periods schemes may be used. The billing period scheme is identified by the billing period counter object captured in the profile.

6.2.3 Billing period values / reset counter entries

These values are represented by instances of the IC "Data".

For billing period / reset counters and for number of available billing periods the data type shall be *unsigned*, *long-unsigned* or *double-long-unsigned*. For time stamps of billing periods, the data type shall be *double-long-unsigned* (in the case of UNIX time), *octet-string* or *date-time* formatted as specified in 4.1.6.1.

These objects may be related to energy type – see also 6.3.3 and 6.4.3 – and channels.

When the values of historical periods are represented by “Profile generic” objects, the time stamp of the billing period objects shall be part of the captured objects.

| Billing period values / reset counter entries | IC | OBIS code | | | | | |
|---|----------------------|-----------|---|---|---|---|-----|
| | | A | B | C | D | E | F |
| For item names and OBIS codes see Table 56. | 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.

6.2.4 Other abstract general purpose OBIS codes

Program entries shall be represented by instances of the IC "Data" with data type *unsigned*, *long-unsigned* or *octet-string*.

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;
NOTE *Firmware version* can be used to distinguish between different releases of a firmware identified by the same *Firmware identifier*.
- Active firmware signature objects hold the digital signature of the currently active firmware. The digital signature algorithm is not specified here.

These three elements are inextricably linked to the currently active firmware.

Firmware identifiers may be also energy type and channel related.

Time entry values shall be represented by instances of IC “Data”, “Register” or “Extended register” with the data type of the value attribute *octet-string*, formatted as *date-time* in 4.1.6.1.

For detailed OBIS codes, see Table 56.

| 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.

6.2.5 Clock objects (class_id = 8)

Instances of the IC “Clock” – see 4.5.1 – control the system clock of the physical device.

“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.

“High resolution clock” objects are instances of the Interface class “Data”, with data type *long64-unsigned*. They hold the number of microseconds since 1970-01-01 00:00:00.

| 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 |
| High resolution clock | 1, Data ^a | 0 | b | 1 | 2 | e | 255 |

^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

6.2.6 Modem configuration and related objects

In this group, the following objects are available:

- Instances of the IC “Modem configuration” – see 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 |

6.2.7 Script table objects (class_id = 9)

Instances of the IC “Script table” – see 4.5.2 – control the behaviour of the device.

Several instances are predefined and normally available as hidden scripts only with access to the *execute ()* method. The following table contains only the identifiers for the “standard” instances of the listed scripts. Implementation specific instances of these scripts should use values different from zero in value group D.

- *MDI reset / End of billing period* “Script table” objects define the actions to be performed at the end of the billing period, for example the reset of maximum demand indicator registers and archiving data. If there are several billing period schemes available, then there shall be one script present in the array of scripts for each billing period scheme.
- *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;

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 356/492 |
|-----------------------|------------|-------------------------|---------|

- *Push* “Script table” objects hold scripts to activate the push operation. Normally every entry in the array of scripts calls the push method of one “Push setup” object instance;
- *Broadcast* “Script table” objects allow standardising utility wide the entry point into regularly needed functionality.

| Script table objects | IC | OBIS code | | | | | |
|---|-----------------|-----------|---|----|---|-----|-----|
| | | A | B | C | D | E | F |
| Global meter reset ^a Script table | 9, Script table | 0 | b | 10 | 0 | 0 | 255 |
| MDI reset / End of billing period ^a Script table | | 0 | b | 10 | 0 | 1 | 255 |
| Tariffication 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 |
| 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 Clause 7.5.1, Table 61. 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> | | | | | | | |

6.2.8 Special days table objects (class_id = 11)

Instances of the IC “Special days table” – see 4.5.4 – define and control the behaviour of the device regarding calendar functions on special days for clock control.

| Special days table objects | IC | OBIS code | | | | | |
|----------------------------|------------------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| Special days table | 11, Special days table | 0 | b | 11 | 0 | e | 255 |

6.2.9 Schedule objects (class_id = 10)

Instances of the IC “Schedule” – see 4.5.3 – define and control the behaviour of the device in a sequenced way.

| Schedule objects | IC | OBIS code | | | | | |
|------------------|--------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| Schedule | 10, Schedule | 0 | b | 12 | 0 | e | 255 |

6.2.10 Activity calendar objects (class_id = 20)

Instances of the IC “Activity calendar” – see 4.5.5 – define and control the behaviour of the device in a calendar-based way.

| Activity calendar objects | IC | OBIS code | | | | | |
|---------------------------|-----------------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| Activity calendar | 20, Activity calendar | 0 | b | 13 | 0 | e | 255 |

6.2.11 Register activation objects (class_id = 6)

Instances of the IC “Register activation” – see 4.3.5 – are used to handle different tariffication structures.

| Register activation objects | IC | OBIS code | | | | | |
|-----------------------------|------------------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| Register activation | 6, Register activation | 0 | b | 14 | 0 | e | 255 |

6.2.12 Single action schedule objects (class_id = 22)

Instances of the IC “Single action schedule” – see 4.5.7 – control the behaviour of the device. Implementation specific instances should use values different from zero in value group D.

Instances of Push “Single action schedule” objects activate scripts in Push “Script table” objects, which invoke the push method of the appropriate “Push setup” objects.

| Single action schedule objects | IC | OBIS code | | | | | |
|--|----------------------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| End of billing period Single action schedule | 22, Single action schedule | 0 | 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 |

6.2.13 Register monitor objects (class_id = 21)

Instances of the IC “Register monitor” – see 4.5.6 – control the register monitoring function of the device. They define the value to be monitored, the set of thresholds to which the value is compared, and the actions to be performed when a threshold is crossed.

In general, the logical name(s) shown in the table below shall be used. See also 6.3.9 and 6.3.10.

Alarm monitor objects monitor *Alarm register* or *Alarm descriptor* objects.

| Register monitor objects | IC | OBIS code | | | | | |
|--------------------------|----------------------|-----------|---|----|---|-------|-----|
| | | A | B | C | D | E | F |
| Register monitor | 21, Register monitor | 0 | b | 16 | 0 | e | 255 |
| Alarm monitor | | 0 | b | 16 | 1 | 0...9 | 255 |

6.2.14 Parameter monitor objects (class_id = 65)

Instances of the IC “Parameter monitor” – see 4.5.10 – control the Parameter monitoring function of the device. They define the list of parameters to be monitored and hold the identifier and the value of the last parameter changed, as well as the *capture_time*.

| Parameter monitor objects | IC | OBIS code | | | | | |
|---------------------------|-----------------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| Parameter monitor | 65, Parameter monitor | 0 | b | 16 | 2 | e | 255 |

6.2.15 Limiter objects (class_id = 71)

Instances of the IC “Limiter” handle the monitoring of values in normal and emergency conditions. See also 4.5.9.

| Limiter objects | IC | OBIS code | | | | | |
|-----------------|-------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| Limiter | 71, Limiter | 0 | b | 17 | 0 | e | 255 |

6.2.16 Payment metering related objects

Payment accounting can be applied to any commodity.

An instance of the “Account” – see 4.6.2 – IC holds the summary information for a given contract and lists the “Credit” and “Charge” objects used by that “Account”. If more than one “Account” is for any reason required in a given context then field D should be other than 0.

One or several instances of the “Credit” IC – see 4.6.3 – represent the different credit sources.

One or several instances of the “Charge” IC – see 4.6.4 – represent the different charges applicable.

One or more instances of the “Token gateway” IC – see 4.6.5. – are available to enter tokens. If only a single gateway is defined in a single “Account” then field E of the OBIS code shall be zero. If more than one “Token gateway” object is for any reason required in a single “Account” then field E should be other than 0.

The “Account” is linked to its associated “Credit”, “Charge” and “Token gateway” objects by use of the value group D and B field such that an “Account” with D=0 should be linked to a “Token gateway” with D=40 and have a “Credit” objects with D=10 and “Charge” objects with D=20. Whereas an “Account” with D=1 should have “token gateway” with D=41, “Credit” objects with D=11 and “Charge” objects with D=21 etc. Multiple “Credit” and “Charge” objects are identified using different values in the value group E field. See also Additional Notes there describing the “Max credit_limit” and „Max vend limit” objects there.

Instances of “Profile Generic” IC hold the history of the token credit and of the charge collections with a “Parameter Monitor” interface class monitoring a value used to trigger capture.

| Payment metering relegated 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 |
| Configurable limit objects | | | | | | | |
| Max credit limit | 01, Data | 0 | b | 19 | 50...59 | 1 | 255 |
| Max vend limit | 01, Data | 0 | b | 19 | 50...59 | 2 | 255 |

6.2.17 IEC local port setup objects (class_id = 19)

These objects define and control the behaviour of local ports using the protocol specified in IEC 62056-21:2002. See also 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 |

6.2.18 Standard readout profile objects (class_id = 7)

A set of objects is defined to carry the standard readout as it would appear with IEC 62056-21:2002 (modes A to D). See also 4.3.6.

| 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 |

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 |

6.2.19 IEC HDLC setup objects (class_id = 23)

Instances of the IC "IEC HDLC setup" – see 4.7.2 – hold the parameters of the HDLC based data link layer.

| IEC HDLC setup objects | IC | OBIS code | | | | | |
|------------------------|--------------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| IEC HDLC setup | 23, IEC HDLC setup | 0 | b | 22 | 0 | 0 | 255 |

6.2.20 IEC twisted pair (1) setup objects (class_id = 24)

An instance of the IC "IEC twisted pair (1) set up" IC – see 4.7.3 – stores the parameters necessary to manage a communication profile specified in IEC 62056-3-1:2013.

An instance of the IC "MAC address set up" IC stores the Secondary Station Address ADS.

An instance of the "Data" stores the Fatal Error register.

Instances of the IC "Profile generic" IC instances allow the configuration of IEC 62056-3-1 readout lists.

| IEC twisted pair (1) setup and related objects | IC | OBIS code | | | | | |
|--|--------------------------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| IEC twisted pair (1) setup | 24, IEC twisted pair (1) setup | 0 | b | 23 | 0 | 0 | 255 |
| IEC twisted pair (1) MAC address setup | 43, MAC address setup | 0 | b | 23 | 1 | 0 | 255 |
| IEC twisted pair (1) Fatal Error register | 1, Data | 0 | b | 23 | 2 | 0 | 255 |
| IEC 62056-3-1 Short readout | 7, Profile generic | 0 | b | 23 | 3 | 0 | 255 |
| IEC 62056-3-1 Long readout | | 0 | b | 23 | 3 | 1 | 255 |
| IEC 62056-3-1 Alternate readout profile 0 | | 0 | b | 23 | 3 | 2 | 255 |
| IEC 62056-3-1 Additional readout profile 1 | | 0 | b | 23 | 3 | 3 | 255 |
| IEC 62056-3-1 Additional readout profile 2 | | 0 | b | 23 | 3 | 4 | 255 |
| | | | | | | | |
| IEC 62056-3-1 Additional readout profile 7 | | 0 | b | 23 | 3 | 9 | 255 |

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 |

6.2.21 Objects related to data exchange over M-Bus

The following objects are available to model and control data exchange using the M-Bus protocol specified in the EN 13757 series:

- instances of the IC "M-Bus slave port setup" define and control the behaviour of M-Bus slave ports of a DLMS/COSEM device. See 4.8.2;
- instances of the IC "M-Bus client" are used to configure DLMS/COSEM devices as M-Bus clients. There is one "M-Bus client" object for each M-Bus slave. Value group B identifies the M-Bus channels. See 4.8.3;
- M-Bus value objects, instances of the IC "Extended register", hold the values captured from M-Bus slave devices on the relevant channel. The link between the M-Bus client setup objects and the M-Bus value objects is provided by the channel number.
- M-Bus "Profile generic" objects capture M-Bus value objects possibly along with other, not M-Bus specific objects;
- M-Bus "Disconnect control" objects control disconnect devices of M-Bus devices (e.g. gas valves);
- instances of the IC "Wireless mode Q" define and control the behaviour of the device regarding the communication parameters according to mode Q of EN 13757-5:2015. A node having more than one network address, i.e. a multi-homed node, will have multiple objects of these types. See 4.8.4;
- M-Bus control log objects are instances of the IC "Profile generic". They log the changes of the state of the disconnect devices;
- instances of the IC "M-Bus master port setup" define and control the behaviour of M-Bus master ports of DLMS/COSEM devices, allowing to exchange data with M-Bus slaves. See 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.

| 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.

6.2.22 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;
- Instances of the IC “GPRS modem setup” – see 4.7.7 – handle all information related to the setup of the GPRS modem;
- Instances of the IC “SMTP setup” – see 4.9.6 – handle all information related to the setup of the SMTP service.

NOTE The following objects have internet related OBIS codes, although they are not strictly related to use over the internet only.

- Instances of the IC “GSM diagnostic” – see 4.7.8 – handle all diagnostic information related to the GSM/GPRS network.
- Instances of the IC “Push setup” – see 4.4.8 – handle all information about the data to be pushed, the push destination and the method the data should be pushed.

| Objects to set up data exchange over the Internet | IC | OBIS code | | | | | |
|---|-----------------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| TCP-UDP setup | 41, TCP-UDP setup | 0 | b | 25 | 0 | 0 | 255 |
| IPv4 setup | 42, IPv4 setup | 0 | b | 25 | 1 | 0 | 255 |
| MAC address setup | 43, MAC address setup | 0 | b | 25 | 2 | 0 | 255 |
| PPP setup | 44, PPP setup | 0 | b | 25 | 3 | 0 | 255 |
| GPRS modem setup | 45, GPRS modem setup | 0 | b | 25 | 4 | 0 | 255 |
| SMTP setup | 46, SMTP setup | 0 | b | 25 | 5 | 0 | 255 |
| GSM diagnostic | 47, GSM diagnostic | 0 | b | 25 | 6 | 0 | 255 |
| IPv6 setup | 48, IPv6 setup | 0 | b | 25 | 7 | 0 | 255 |
| <i>Reserved for FTP setup</i> | | | | | | | |
| Push setup | 40, Push setup | 0 | b | 25 | 9 | 0 | 255 |

6.2.23 Objects for setting up data exchange using S-FSK PLC

In this group, the following objects are available:

- Instances of the IC “S-FSK Phy&MAC setup” – see 4.10.3 – handle all information related to setting up the PLC S-FSK lower layer profile specified in IEC 61334-5-1:2001.
- Instances of the IC “S-FSK Active initiator” – see 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.
- 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.
- 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.
- 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.
- Instances of the IC “S-FSK Reporting system list” – see 4.10.8 – hold information on reporting systems in the PLC S-FSK lower layer profile specified in IEC 61334-5-1:2001.

| Objects to set up data exchange using S-FSK PLC | IC | OBIS code | | | | | |
|---|---------------------------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| S-FSK Phy&MAC setup | 50, S-FSK Phy&MAC setup | 0 | b | 26 | 0 | 0 | 255 |
| S-FSK Active initiator | 51, S-FSK Active initiator | 0 | b | 26 | 1 | 0 | 255 |
| S-FSK MAC synchronization timeouts | 52, S-FSK MAC synchronization | 0 | b | 26 | 2 | 0 | 255 |
| S-FSK MAC counters | 53, S-FSK MAC counters | 0 | b | 26 | 3 | 0 | 255 |
| NOTE This is a placeholder for a Monitoring IC to be specified. | | | | | | | |
| IEC 61334-4-32 LLC setup | 55, IEC 61334-4-32 LLC setup | 0 | b | 26 | 5 | 0 | 255 |
| S-FSK Reporting system list | 56, S-FSK Reporting system list | 0 | b | 26 | 6 | 0 | 255 |

6.2.24 Objects for setting up the ISO/IEC 8802-2 LLC layer

In this group, the following objects are available:

- Instances of the IC “ISO/IEC 8802-2 LLC Type 1 setup” – see 4.11.2 – handle all information related to the LLC layer specified in ISO/IEC 8802-2:1998 in Type 1 operation.

- Instances of the IC “ISO/IEC 8802-2 LLC Type 2 setup” – see 4.11.3 – handle all information related to the LLC layer specified in ISO/IEC 8802-2:1998 in Type 2 operation.
- Instances of the IC “ISO/IEC 8802-2 LLC Type 3 setup” – see 4.11.4 – handle all information related to the LLC layer specified in ISO/IEC 8802-2:1998 in Type 3 operation.

| Objects to set up the ISO/IEC 8802-2 LLC layer | IC | OBIS code | | | | | |
|--|-------------------------------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| ISO/IEC 8802-2 LLC Type 1 setup | 57, ISO/IEC 8802-2 LLC Type 1 setup | 0 | 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 |

6.2.25 Objects for data exchange using narrowband OFDM PLC for PRIME networks

For setting up and managing data exchange using narrowband OFDM PLC for PRIME networks one instance of each following classes shall be implemented for each interface:

- an instance of the 61334-4-32 LLC SSCS setup – see 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.

| Objects for data exchange using Type 1 OFDM PLC | IC | OBIS code | | | | | |
|---|---|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| 61334-4-32 LLC SSCS setup | 80, 61334-4-32 LLC SSCS setup | 0 | 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 |
|--|--|---|---|----|---|---|-----|

6.2.26 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 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 |

6.2.27 ZigBee® setup objects

The following objects are available for setting up and managing a ZigBee® network; see also 4.14.

| Objects set up and manage ZigBee® networks | IC | OBIS code | | | | | |
|--|------------------------------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| ZigBee® SAS startup | 101, ZigBee® SAS Startup | 0 | b | 30 | 0 | e | 255 |
| ZigBee® SAS join | 102, ZigBee® SAS join | 0 | b | 30 | 1 | e | 255 |
| ZigBee® SAS APS fragmentation | 103, ZigBee® SAS APS fragmentation | 0 | b | 30 | 2 | e | 255 |
| ZigBee® network control | 104, ZigBee® network control | 0 | b | 30 | 3 | e | 255 |
| ZigBee® tunnel setup | 105 ,ZigBee® tunnel setup | 0 | b | 30 | 4 | e | 255 |

6.2.28 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 |

6.2.29 SAP assignment object (class_id = 17)

An instance of the IC “SAP assignment” – see 4.4.5 – holds information about the addresses (Service Access Points, SAPs) of logical devices within a physical device.

| SAP Assignment object | IC | OBIS code | | | | | |
|---|--------------------|------------------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| SAP assignment of current physical device | 17, SAP assignment | 0 | 0 | 41 | 0 | 0 | 255 |

6.2.30 COSEM logical device name object

Each COSEM logical device shall be identified by its Logical Device Name, unique worldwide. See 4.1.8.2. It is held by the *value* attribute of a “Data” object, with data type *octet-string* or *visible-string*. For short name referencing, the *base_name* of the object is fixed. See 4.1.3.

| 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. | | | | | | | |

6.2.31 Information security related objects

Instances of the IC “Security setup” – see 4.4.7 – are used to set up the message security features. For each Association object, there is one Security setup object managing security within that AA. See 5.4.4 and 5.4.5. Value group E numbers the instances.

| Security setup objects | IC | OBIS code | | | | | |
|-------------------------------|--------------------|------------------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| Security setup | 64, Security setup | 0 | 0 | 43 | 0 | e | 255 |

Invocation counter objects hold the invocation counter element of the initialization vector. They are instances of the IC “Data”. The value in value group B identifies the communication channel.

NOTE The same client may use different communication channels e.g. a remote port and a local port. The invocation counter on the different channels may be different.

The value in value group E shall be the same as in the logical name of the corresponding “Security setup” object.

| 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 version of the Blue Book, these objects were called Frame counter objects. | | | | | | | |
| ^a If the IC “Data” is not available, “Register” (with scaler = 0, unit = 255) may be used. | | | | | | | |

Instances of the IC “Data protection” – see 4.4.9 – are used to apply / remove protection on COSEM data, i.e. sets of attributes values, method invocation and return parameters. Value 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 |

6.2.32 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 |

6.2.33 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.

| 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 |

6.2.34 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 |

6.2.35 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 Table 56.

| 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.

6.2.36 Metering point ID objects

One object is available to store a media type independent metering point ID. It is held by the *value* attribute of a "Data" object, with data type ***double-long-unsigned*, *octet-string*, *visible-string*, *utf8-string*, *unsigned*, *long-unsigned***.

| Metering point ID objects | IC | OBIS code | | | | | |
|---------------------------|----------------------|-----------|---|----|---|----|-----|
| | | A | B | C | D | E | F |
| Metering point ID | 1, Data ^a | 0 | 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.

6.2.37 Parameter changes and calibration objects

A set of simple COSEM objects describes the history of the configuration of the device. All values are modelled by instances of the IC "Data".

| Parameter changes objects | IC | OBIS code | | | | | |
|--|----------------------|-----------|---|----|---|---|-----|
| | | A | B | C | D | E | F |
| For names and OBIS codes see Table 56. | 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.

6.2.38 I/O control signal objects

A series of objects are available to define and control the status of I/O lines of the physical metering equipment.

The status is held by the *value* attribute of a "Data" object, with data type *octet-string* or *bit-string*. Alternatively, the status is held by a "Status mapping" object, see 4.3.10, which holds both the status word and the mapping of its bits to the reference table. If there are several I/O control status objects used, it is allowed to combine them into an instance of the IC "Profile generic" or "Register table", using the OBIS code of the global state of I/O control signals object. See also Table 56.

| 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.

6.2.39 Disconnect control objects (class_id = 70)

Instances of the IC “Disconnect control” – see 4.5.8 – manage internal or external disconnect units (e.g. electricity breaker, gas valve) in order to connect or disconnect – partly or entirely – the premises of the consumer to / from the supply. See also 6.2.21.

| Disconnect control objects | IC | OBIS code | | | | | |
|----------------------------|------------------------|-----------|---|----|---|----|-----|
| | | A | B | C | D | E | F |
| Disconnect control | 70, Disconnect control | 0 | b | 96 | 3 | 10 | 255 |

6.2.40 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 |

6.2.41 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 4.3.10, 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 Table 56.

| 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.

6.2.42 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 4.3.10, 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 Table 56.

| 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.

Internal operating status objects can also be related to an energy type. See 6.3.7.

6.2.43 Battery entries objects

A series of objects are available for holding information relative to the battery of the device. These objects are instances of IC “Data”, “Register” or “Extended register” as appropriate.

| Battery entries objects | IC | OBIS code | | | | | |
|--|--|-----------|---|----|---|-------|-----|
| | | A | B | C | D | E | F |
| For names and OBIS codes see Table 56. | 1, Data, 3, Register or 4, Extended register | 0 | b | 96 | 6 | 0...6 | 255 |

6.2.44 Power failure monitoring objects

A series of objects are available for power failure monitoring:

- For simple power failure monitoring, it is possible to count the number of power failure events affecting all three phases, one of the three phases, any of the phases, and the auxiliary supply;
- For advanced power failure monitoring, it is possible to define a time threshold to make a distinction between short and long power failure events. It is possible to count the number of such long power failure events separately from the short ones, as well as to store their time of occurrence and duration (time from power down to power up) in all three phases, in one of the three phases and in any of the phases;
- The number of power failure events objects are represented by instances of the IC “Data”, “Register” or “Extended register” with data types *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*;
- The power failure duration, time and time threshold data are represented by instances of the IC “Data”, “Register” or “Extended register” with appropriate data types;
- If power failure duration objects are represented by instances of the IC “Data”, then the default scaler shall be 0, and the default unit shall be the second.

| Power failure monitoring objects | IC | OBIS code | | | | | |
|--|--|-----------|---|----|---|--------|-----|
| | | A | B | C | D | E | F |
| For names and OBIS codes see Table 56. | 1, Data, 3, Register or 4, Extended register | 0 | b | 96 | 7 | 0...21 | 255 |

These objects may be collected in a “Power failure event log” object. See Table 71.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 370/492 |
|-----------------------|------------|-------------------------|---------|

6.2.45 Operating time objects

A series of objects are available for holding the cumulated operating time and the various tariff registers of the device. These objects are instances of the IC “Data”, “Register” or “Extended register”. The data type shall be *unsigned*, *long-unsigned* or *double-long-unsigned* with appropriate scaler and unit. If the IC “Data” is used, the unit shall be the second by default.

| Operating time objects | IC | OBIS code | | | | | |
|--|--|-----------|---|----|---|--------|-----|
| | | A | B | C | D | E | F |
| For names and OBIS codes see Table 56. | 1, Data, 3, Register or 4, Extended register | 0 | b | 96 | 8 | 0...63 | 255 |

6.2.46 Environment related parameters objects

A series of objects are available to store environmental related parameters. They are held by the *value* attribute of instances of the IC “Register” or “Extended register”, with appropriate data types.

| Environment related parameters objects | IC | OBIS code | | | | | |
|--|-------------------------------------|-----------|---|----|---|-------|-----|
| | | A | B | C | D | E | F |
| For names and OBIS codes see Table 56. | 3, Register or 4, Extended register | 0 | b | 96 | 9 | 0...2 | 255 |

6.2.47 Status register objects

A series of objects are available to hold statuses that can be captured in load profiles. See also Table 56.

| 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. | | | | | | | |

The status register is held by the *value* attribute of a “Data” object, with data type *bit-string*, *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-unsigned* or *octet-string*. It carries binary information from a bitmap. Its contents is not specified.

Alternatively, the status register may be held by the *status_word* attribute of a “Status mapping” object, see 4.3.9. The *mapping_table* attribute holds mapping information between the bits of the status word and entries of a reference table.

6.2.48 Event code objects

In the meter or in its environment, various events may be generated. A series of objects are available to hold an identifier of a most recent event (event code). Different instances of event code objects may be captured in different instances of event logs; see 6.2.58.

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 Table 56. | 1, Data, 3, Register, or 4, Extended register | 0 | b | 96 | 11 | 0...99 | 255 |

Events may also set flags in error registers and alarm registers. See also 6.2.56.

6.2.49 Communication port log parameter objects

A series of objects are available to hold various communication log parameters. They are represented by instances of IC “Data”, “Register” or “Extended register”.

| Communication port log parameter objects | IC | OBIS code | | | | | |
|--|---|-----------|---|----|----|-------|-----|
| | | A | B | C | D | E | F |
| For names and OBIS codes see Table 56. | 1, Data, 3, Register, or 4, Extended register | 0 | b | 96 | 12 | 0...6 | 255 |

6.2.50 Consumer message objects

A series of objects are available to store information sent to the energy end-user. The information may appear on the display of the meter and / or on a consumer information port.

| Consumer message objects | IC | OBIS code | | | | | |
|--|---|-----------|---|----|----|------|-----|
| | | A | B | C | D | E | F |
| For names and OBIS codes see Table 56. | 1, Data, 3, Register, or 4, Extended register | 0 | b | 96 | 13 | 0, 1 | 255 |

6.2.51 Currently active tariff objects

A series of objects are available to hold the identifier of the currently active tariff. They carry the same information as the *active_mask* attribute of the corresponding “Register activation” object.

| Currently active tariff objects | IC | OBIS code | | | | | |
|--|---|-----------|---|----|----|--------|-----|
| | | A | B | C | D | E | F |
| For names and OBIS codes see Table 56. | 1, Data, 3, Register, or 4, Extended register | 0 | b | 96 | 14 | 0...15 | 255 |

6.2.52 Event counter objects

A series of objects are available to count events. The number of the events is held by the value attribute.

| Event counter objects | IC | OBIS code | | | | | |
|--|---|-----------|---|----|----|--------|-----|
| | | A | B | C | D | E | F |
| For names and OBIS codes see Table 56. | 1, Data, 3, Register, or 4, Extended register | 0 | b | 96 | 15 | 0...99 | 255 |

6.2.53 Profile entry digital signature objects

Instances of “Data”, “Register” or “Extended register” objects hold digital signatures of “Profile generic” object buffer entries. If the *capture_object* attribute of a “Profile generic” object contains a reference to a “Profile entry digital signature” object, then the digital signature is calculated and captured together with the other attribute values.

The security context is determined by the “Security setup” object which is which is visible in the same AA (object_list).

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 Table 56. | 1, Data, 3, Register, or 4, Extended register | 0 | b | 96 | 16 | 0...9 | 255 |

6.2.54 Meter tamper event related objects

A series of objects are available to register characteristics of various meter tamper events. These objects are instances of the IC “Data”, Register” or “Extended register”. The data type shall be *unsigned*, *long-unsigned* or *double-long-unsigned* with appropriate scaler and unit. For time stamps, the data type shall be *double-long-unsigned* (in the case of UNIX time), *octet-string* or *date-time* formatted as specified in 4.1.6.1.

- 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.

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 Table 56. | 1, Data, 3, Register, or 4, Extended register | 0 | b | 96 | 20 | e | 255 |

6.2.55 Error register objects

A series of objects are used to communicate error indications of the device. The different error registers are held by the *value* attribute of “Data” objects, with data type or *bit-string*, *octet-string*, *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*.

The individual bits of the error register may be set and cleared by a pre-defined selection of events – see 6.2.48. Depending on the type of the error, some errors may clear themselves when the reason setting the error flag disappears.

If more than one of those objects is used, it is allowed to combine them into one instance of the IC “Profile generic”. In this case, the captured objects are the *value* attributes “Data” objects, the capture period is 1 to have just actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively, an instance of the IC “Register table” can be used.

Error register objects can also be related to an energy type and to a channel. See 7.4.2, 7.5.5.2, 7.6.4.2, 7.7.4.2, 7.9.4.2.

| Error register objects | IC | OBIS code | | | | | |
|------------------------------|----------------------|-----------|---|----|----|-------|-----|
| | | A | B | C | D | E | F |
| Error register 1...10 object | 1, Data ^a | 0 | b | 97 | 97 | 0...9 | 255 |
| Error profile object | 7, Profile generic | 0 | b | 97 | 97 | 255 | 255 |
| Error table object | 61, Register table | 0 | b | 97 | 97 | 255 | 255 |

^a If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

6.2.56 Alarm register, Alarm filter and Alarm descriptor objects

A number of objects are available to hold alarm registers. The different alarm registers are held by the value attribute of "Data" objects, with data type *bit-string*, *octet-string*, *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*. When selected events occur, they set the corresponding flag and the device may raise an alarm. Depending on the type of alarm, some alarms may clear themselves when the reason setting the alarm flag disappears.

If more than one of those objects is used, it is also allowed to combine them into one instance of the IC "Profile generic". In this case, the captured objects are the *value* attributes of "Data" objects, the capture period is 1 to have just actual values, the sort method is FIFO, and the profile entries are limited to 1. Alternatively, an instance of the IC "Register table" can be used.

Alarm filter objects are available to define if an event is to be handled as an alarm when it appears. The different alarm filters are held by the value attribute of "Data" objects, with data type *bit-string*, *octet-string*, *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*. The bit mask has the same structure as the corresponding alarm register object. If a bit in the alarm filter is set, then the corresponding alarm is enabled, otherwise it is disabled. *Alarm filter* objects act on *Alarm register* and *Alarm descriptor* objects the same way.

Alarm descriptor objects are available to persistently hold the occurrence of alarms. The different alarm descriptors are of the same type as the corresponding *Alarm register*. When a selected event occurs, the corresponding flag is set in the *Alarm register* as well as in the *Alarm descriptor* objects. An alarm descriptor flag remains set even if the corresponding alarm condition has disappeared. Alarm descriptor flags do not reset themselves; they can be reset by writing the value attribute only.

NOTE The alarm conditions, the structure of the *Alarm register* / *Alarm filter* / *Alarm descriptor* objects are subject to a project specific companion specification.

| Alarm register, Alarm filter and Alarm descriptor objects | IC | OBIS code | | | | | |
|---|----------------------|-----------|---|----|----|---------|-----|
| | | A | B | C | D | E | F |
| Alarm register objects 1...10 | 1, Data ^a | 0 | b | 97 | 98 | 0...9 | 255 |
| Alarm register profile object | 7, Profile generic | 0 | b | 97 | 98 | 255 | 255 |
| Alarm register table object | 61, Register table | 0 | b | 97 | 98 | 255 | 255 |
| Alarm filter objects 1...10 | 1, Data ^a | 0 | b | 97 | 98 | 10...19 | 255 |
| Alarm descriptor objects 1...10 | 1, Data ^a | 0 | b | 97 | 98 | 20...29 | 255 |

^a If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

6.2.57 General list objects

Instances of the IC "Profile generic" are used to model lists of any kind of data, for example measurement values, constants, statuses, events. They are modelled by "Profile generic" objects. One standard object per billing period scheme is defined.

List objects may be also related to an energy type and to a channel.

| General list objects | IC | OBIS code | | | | | |
|---|--------------------|-----------|---|----|---|---|------------------|
| | | A | B | C | D | E | F |
| For names and OBIS codes see 7.4.3. | 7, Profile generic | 0 | b | 98 | d | e | 255 ^a |
| ^a F = 255 means a wildcard here. See 7.11.3. | | | | | | | |

6.2.58 Event log objects

Instances of the IC “Profile generic” are used to store Event logs. Event logs may be also media related. In this case, the value of value group A shall be the relevant media identifier. See also 7.4.5, 7.5.5.4, 7.8.6.4.

| Event log objects | IC | OBIS code | | | | | |
|---|--------------------|-----------|---|----|----|---|------------------|
| | | A | B | C | D | E | F |
| Event log | 7, Profile generic | a | b | 99 | 98 | e | 255 ^a |
| ^a F = 255 means a wildcard here. See 7.11.3. | | | | | | | |
| NOTE 1 Event logs may capture for example the time of occurrence of the event, the event code and other relevant data. | | | | | | | |
| NOTE 2 Project specific companion specifications may specify a more precise meaning of the instances of the different event logs, i.e. the data captured and the number of events captured. | | | | | | | |

6.2.59 Inactive objects

Inactive objects are objects, which are present in the meter, but which do not have an assigned functionality. Inactive instances of any IC may be present. See also 7.3.3.2.

| Inactive objects | IC | OBIS code | | | | | |
|------------------|-----|-----------|---|-----|---|---|-----|
| | | A | B | C | D | E | F |
| Inactive objects | Any | 0 | b | 127 | 0 | e | 255 |

6.3 Electricity related COSEM objects

6.3.1 Value group D definitions

The different ways of processing measurement values as defined by value group D – see 7.5.2.1 – are modelled as shown in Table 38.

Table 38 – Representation of various values by appropriate ICs

| Type of value | Represented by |
|---------------------------------|---|
| cumulative values | Instances of IC "Register" or "Extended register". |
| maximum and minimum values | Instances of IC "Profile generic" with sorting method <i>maximum</i> or <i>minimum</i> , depth according to implementation and captured objects according to implementation. A single maximum value or minimum value can alternatively be represented by an instance of the IC "Register" or "Extended register". |
| current and last average values | Instances of IC "Demand register". The logical name is the OBIS code of the current average value (D = 4, 14 or 24). For display purposes: Instances of IC "Register" or "Demand register". The logical name is the OBIS code of current average (D = 4, 14 or 24) or last average (D= 5, 15 or 25) as appropriate. |
| instantaneous values | Instances of IC "Register". |
| time integral values | Instances of IC "Register" or "Extended register". |
| occurrence counters | Instances of IC "Data" or "Register". |
| contracted values | Instances of IC "Register" or "Extended register". |

6.3.2 Electricity ID numbers

The different electricity ID numbers are held by instances of the IC "Data", with data type *unsigned*, *long-unsigned*, *double-long-unsigned*, *octet-string* or *visible-string*. If more than one of those is used, it is allowed to combine them into a "Profile generic" object. In this case, the captured objects are *value* attributes of electricity ID "Data" objects, the capture period is 1 to have just actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively, a "Register table" object can be used. See also Table 68.

| Electricity ID objects | IC | OBIS code | | | | | |
|------------------------------|----------------------|-----------|---|---|---|-------|-----|
| | | A | B | C | D | E | F |
| Electricity ID 1...10 object | 1, Data ^a | 1 | b | 0 | 0 | 0...9 | 255 |
| Electricity ID-s object | 7, Profile generic | 1 | b | 0 | 0 | 255 | 255 |
| Electricity ID-s object | 61, Register table | 1 | b | 0 | 0 | 255 | 255 |

^a If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

6.3.3 Billing period values / reset counter entries

These values are represented by instances of the IC "Data".

For billing period / reset counters and for number of available billing periods the data type shall be *unsigned*, *long-unsigned* or *double-long-unsigned*. For time stamps of billing periods, the data type shall be *double-long-unsigned* (in the case of UNIX time), *octet-string* or *date-time* formatted as specified in 4.1.6.1.

These objects may be related to channels.

When the values of historical periods are represented by "Profile generic" objects, the time stamp of the billing period objects shall be part of the captured objects.

| Billing period values / reset counter entries | IC | OBIS code | | | | | |
|--|----------------------|-----------|---|---|---|---|-----|
| | | A | B | C | D | E | F |
| For item names and OBIS codes see Table 68. | 1, Data ^a | 1 | 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. | | | | | | | |

6.3.4 Other electricity related general purpose objects

Program entries shall be represented by instances of the IC "Data" with data type *unsigned*, *long-unsigned*, *octet-string* or *visible-string*. For "Meter connection diagram ID" objects data type *enumerated* can be used as well. Program entries can also be related to a channel.

Output pulse constant, reading factor, CT/VT ratio, nominal value, input pulse constant, transformer and line loss coefficient values shall be represented by instances of the IC "Data", "Register" or "Extended register". For the *value* attribute, only simple data types are allowed.

Measurement period, recording interval and billing period duration values shall be represented by instances of IC "Data", "Register" or "Extended register" with the data type of the *value* attribute *unsigned*, *long-unsigned* or *double-long-unsigned*. The default unit is the second.

Time entry values shall be represented by instances of IC "Data", "Register" or "Extended register" with the data type of the *value* attribute *octet-string*, formatted as *date-time* in 4.1.6.1. The data types *unsigned*, *integer*, *long-unsigned* or *double-long-unsigned* can also be used where appropriate.

The *Clock synchronization method* shall be represented by an instance of an IC "Data" with data type *enum*.

| Synchronization method | enum: | (0) | no synchronization, |
|------------------------|-------|-----|-----------------------------|
| | | (1) | adjust to quarter, |
| | | (2) | adjust to measuring period, |
| | | (3) | adjust to minute, |
| | | (4) | reserved, |
| | | (5) | adjust to preset time, |
| | | (6) | shift time |

For the detailed OBIS codes, see Table 68.

| Electricity related general purpose objects | IC | OBIS code | | | | | |
|--|----------------------|-----------|---|---|----|---|-----|
| | | A | B | C | D | E | F |
| Program entries | 1, Data ^a | 1 | b | 0 | 2 | e | 255 |
| Output pulse values or constants | | 1 | b | 0 | 3 | e | 255 |
| Reading factor and CT/VT ratio | | 1 | b | 0 | 4 | e | 255 |
| Nominal values | | 1 | b | 0 | 6 | e | 255 |
| Input pulse values or constants | | 1 | b | 0 | 7 | e | 255 |
| Measurement period- / recording interval- / billing period duration | | 1 | b | 0 | 8 | e | 255 |
| Time entries | | 1 | b | 0 | 9 | e | 255 |
| Transformer and line loss coefficients | | 1 | b | 0 | 10 | e | 255 |
| ^a If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used. | | | | | | | |

6.3.5 Measurement algorithm

These values are represented by instances of the IC “Data”, with data type `enum`.

| Measurement algorithm objects | IC | OBIS code | | | | | |
|--|----------------------|-----------|---|---|----|---|-----|
| | | A | B | C | D | E | F |
| Measuring algorithm for active power | 1, Data ^a | 1 | 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.

The enumerated values are specified in the table below:

Table 39 – Measuring algorithms – enumerated values

| Measuring algorithm for active power and energy | |
|--|---|
| (0) | not specified |
| (1) | only the fundamentals of voltage and current are used |
| (2) | all harmonics of voltage and current are used |
| (3) | only the DC part of voltage and current is used |
| (4) | all harmonics and the DC part of voltage and current are used |
| Measuring algorithm for reactive power and energy | |
| (0) | not specified |
| (1) | (sum of) reactive power of each phase, calculated from the fundamental of the per phase voltage and the per phase current |
| (2) | polyphase reactive power calculated from polyphase apparent power and polyphase active power |
| (3) | (sum of) reactive power calculated from per phase apparent power and per phase active power |
| Measurement algorithm for apparent power and energy | |
| (0) | not specified |
| (1) | $S = U \times I$, with voltage: only fundamental, and current: only fundamental |
| (2) | $S = U \times I$, with voltage: only fundamental, and current: all harmonics |
| (3) | $S = U \times I$, with voltage: only fundamental, and current: all harmonics and DC part |
| (4) | $S = U \times I$, with voltage: all harmonics, and current: only fundamental |
| (5) | $S = U \times I$, with voltage: all harmonics, and current: all harmonics |
| (6) | $S = U \times I$, with voltage: all harmonics, and current: all harmonics and DC part |
| (7) | $S = U \times I$, with voltage: all harmonics and DC part, and current: only fundamental |
| (8) | $S = U \times I$, with voltage: all harmonics and DC part, and current: all harmonics |
| (9) | $S = U \times I$, with voltage: all harmonics and DC part, and current: all harmonics and DC part |

| | |
|---|---|
| (10) | $S = \sqrt{P^2 + Q^2}$, with P : only fundamental in U and I , and Q : only fundamental in U and I , where P and Q are polyphase quantities |
| (11) | $S = \sqrt{P^2 + Q^2}$, with P : all harmonics in U and I , and Q : only fundamental in U and I where P and Q are polyphase quantities |
| (12) | $S = \sqrt{P^2 + Q^2}$, with P : all harmonics and DC part in U and I , and Q : only fundamental in U and I where P and Q are polyphase quantities |
| (13) | $S = \sum \sqrt{P^2 + Q^2}$, with P : only fundamental in U and I , and Q : only fundamental in U and I where P and Q are single phase quantities |
| (14) | $S = \sum \sqrt{P^2 + Q^2}$, with P : all harmonics in U and I , and Q : only fundamental in U and I where P and Q are single phase quantities |
| (15) | $S = \sum \sqrt{P^2 + Q^2}$, with P : all harmonics and DC part in U and I , and Q : only fundamental in U and I where P and Q are single-phase quantities |
| Measurement algorithm for power factor calculation | |
| (0) | not specified |
| (1) | displacement power factor: the displacement between fundamental voltage and current vectors, which can be calculated directly from fundamental active power and apparent power, or another appropriate algorithm, |
| (2) | true power factor, the power factor produced by the voltage and current, including their harmonics . It may be calculated from apparent power and active power, including the harmonics. |

6.3.6 Metering point ID (electricity related)

A series of objects are available to hold electricity related metering point IDs. They are held by the *value* attribute of "Data" objects, with data type *unsigned*, *long-unsigned*, *double-long unsigned*, *octet-string* or *visible-string*. If more than one of those is used, it is allowed to combine them into one instance of the IC "Profile generic". In this case, the captured objects are the *value* attributes of the electricity related metering point ID "Data" objects, the capture period is 1 to have just actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively, an instance of the IC "Register table" can be used. For detailed OBIS codes, see Table 68.

| 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 | 64, 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.

6.3.7 Electricity related status objects

A number of electricity related objects are available to hold information about the internal operating status, the starting of the meter and the status of voltage and current circuits.

The status is held by the value attribute of a "Data" object, with data type *bit-string*, *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-unsigned* or *octet-string*.

Alternatively, the status is held by a "Status mapping" object, which holds both the status word and the mapping of its bits to the reference table.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 379/492 |
|-----------------------|------------|-------------------------|---------|

If there are several electricity related internal operating status objects used, it is allowed to combine them into an instance of the IC “Profile generic” or “Register table”, using the OBIS code of the global internal operating status. For detailed OBIS codes, see Table 68.

| 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.

6.3.8 List objects – Electricity (class_id = 7)

These COSEM objects are used to model lists of any kind of data, for example measurement values, constants, statuses, events. They are modelled by “Profile generic” objects.

One standard object per billing period scheme is defined. See also 7.5.5.3.

| List objects – Electricity | IC | OBIS code | | | | | |
|--|--------------------|-----------|---|----|---|---|------------------|
| | | A | B | C | D | E | F |
| For names and OBIS codes see Table 70. | 7, Profile generic | 1 | b | 98 | d | e | 255 ^a |

^a F = 255 means a wildcard here. See 7.11.3.

6.3.9 Threshold values

A number of objects are available for representing thresholds for instantaneous quantities. The thresholds may be “under limit”, “over limit”, “missing” and “time thresholds”. Time thresholds are used to detect “under limit”, “over limit” and “missing” conditions.

Objects are also available to represent the number of occurrences when these thresholds are exceeded, the duration of such events and the magnitude of the quantity during such events.

These values are represented by instances of IC “Data”, “Register” or “Extended register”.

All these quantities may be related to tariffs.

As defined in Clause 7.5.4.2, value group F may be used to identify multiple thresholds.

For OBIS codes, see Table 40 below and Table 62.

Table 40 – 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 | | 1 | b | 11, 12, 15, 31, 32, 35, 51, 52, 55, 71, 72, 75, 90...92 | | 0...120, 124... 127 | |

For monitoring the supply voltage, a more sophisticated functionality is also available, allowing to count the number of occurrences classified by the duration of the event and the depth of the voltage dip. For OBIS codes, see Clause 7.5.3.6, Table 67.

6.3.10 Register monitor objects (class_id = 21)

Further to 6.2.13, the following definitions apply:

- for monitoring thresholds of instantaneous values, the logical name of the “Register monitor” object may be the OBIS identifier of the threshold;
- for monitoring current average and last average values, the logical name of the “Register monitor” object may be the OBIS identifier of the demand value monitored.

Table 41 – 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 |
| Current average and last average values | | 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. | | | | | | | |

For the use of value group D, see Table 62.

For the use of value group E, see Table 63 and Table 64.

For the use of value group F, see 7.5.4.2.

6.4 Gas related COSEM objects

NOTE The reader is advised to refer to clause 7.8.1, General introduction to gas measurement before reading this section.

6.4.1 General

The use of interface classes to represent various data is described in the following tables.

Grouping the data by major categories supports the link between the data and the OBIS value groups associated with them.

6.4.2 Gas ID numbers

The different gas ID numbers are instances of the IC "Data", with data type *unsigned*, *long-unsigned*, *double-long-unsigned*, *octet-string* or *visible-string*.

If more than one of those is used, it is allowed to combine them into one instance of the IC "Profile generic". In this case, the captured objects are *value* attributes of the gas ID data objects, the capture period is 1 to have just actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively, an instance of the IC "Register table" can be used. See also Table 96.

| Gas ID | IC | OBIS code | | | | | |
|----------------------|----------------------|-----------|---|---|---|-------|-----|
| | | A | B | C | D | E | F |
| Gas ID 1...10 object | 1, Data ^a | 7 | b | 0 | 0 | 0...9 | 255 |
| Gas ID-s object | 7, Profile generic | 7 | b | 0 | 0 | 255 | 255 |
| Gas ID-s object | 61, Register table | 7 | b | 0 | 0 | 255 | 255 |

^a If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

6.4.3 Billing period values / reset counter entries

These values are represented by instances of the IC "Data".

For billing period / reset counters and for number of available billing periods the data type shall be *unsigned*, *long-unsigned* or *double-long-unsigned*. For time stamps of billing periods, the data type shall be *double-long-unsigned* (in the case of UNIX time), *octet-string* or *date-time* formatted as specified in 4.1.6.1.

These objects may be related to channels.

When the values of historical periods are represented by "Profile generic" objects, the time stamp of the billing period objects shall be part of the captured objects.

| Billing period values / reset counter entries objects | IC | OBIS code | | | | | |
|---|----------------------|-----------|---|---|---|---|-----|
| | | A | B | C | D | E | F |
| For item names and OBIS codes see Table 96 | 1, Data ^a | 7 | 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.

6.4.4 Other gas related general purpose objects

Configuration entries are represented by instances of the IC "Data" with data type *unsigned*, *long-unsigned* or *octet-string* or *enumerated*. Configuration entries can also be related to a channel.

For the digital and analogue output configuration objects the enumerated values are specified in Table 42 below.

Table 42 – Digital / Analogue output configurations – enumerated values

| Digital output configuration | |
|--------------------------------------|--|
| (0) | Output switched off (transistor blocking, "switch open") |
| (1) | Volume pulse output, logic active |
| (2) | Status output, logic active (signalling active => output switched on) |
| (3) | Time-synchronised output, logic active |
| (4) | Output switched on (transistor conducting, "switch closed") |
| (5) | Volume pulse output, logic inactive |
| (6) | Status output, logic inactive (signalling active => output switched off) |
| (7) | Time-synchronised output, logic inactive |
| (8) | High frequency pulse output |
| (9) | Event output, logic active (message active => output switched on) |
| (10) | Event output, logic inactive (message active => output switched off) |
| (99) | Continuous pulse (for test purposes) |
| Analogue output configuration | |
| (0) | inactive |
| (1) | 1= 4–20 mA |
| (2) | 2= 0–20 mA |
| (3) | 3= Voltage output |

The use of ICs shall be as specified below:

- Output pulse constant values are represented by instances of the IC "Data", "Register" or "Extended register". For the *value* attribute, only simple data types are allowed;
- Conversion factor values are represented by instances of the IC "Data", "Register" or "Extended register". For the *value* attribute, only simple data types are allowed;
- Threshold values are represented by instances of the IC "Register" or "Extended register". For the *value* attribute, only simple data types are allowed;
- Nominal values of volume sensors are represented by instances of the IC "Register" or "Extended register". For the *value* attribute, only simple data types are allowed;
- Input pulse constant values are represented by instances of the IC "Data", "Register" or "Extended register". For the *value* attribute, only simple data types are allowed;
- Interval and period values are represented by instances of IC "Data", "Register" or "Extended register" with the data type of the *value* attribute *unsigned*, *long-unsigned* or *double-long-unsigned*;
- Time entry values are represented by instances of IC "Data", "Register" or "Extended register" with the data type of the *value* attribute *octet-string*, formatted as *date-time* in 4.1.6.1. The data types *unsigned*, *integer*, *long-unsigned* or *double-long-unsigned* can also be used where appropriate;
- Station management information values are represented by instances of the IC "Register" or "Extended register". For the *value* attribute, only simple data types are allowed;
- Gas parameters for volume conversion currently used in compressibility calculation values are represented by instances of the IC "Data", "Register" or "Extended register". For the *value* attribute, only simple data types are allowed;
- Gas measuring method specific parameters, including gas parameters for Venturi measurement and for density measurement are represented by instances of the IC "Data", "Register" or "Extended register". For the *value* attribute, only simple data types are allowed.

- “Sensor manager” objects – see 4.5.11 – manage complex information related to sensors. See 4.5.11. This interface class will be used by the following (metrological) sensors e.g. in gas applications:
 - absolute temperature;
 - absolute pressure;
 - velocity of sound;
 - density of gas;
 - relative density;
 - gauge pressure;
 - differential pressure;
 - density of air.

Value group E of the OBIS code can be mapped to the value group C codes, identifying the various physical quantities, see Table 86. The possible values are 41, 42, 44, 45...49.

EXAMPLE Absolute pressure sensor manager object OBIS code 7.0.0.15.42.255.

If there is more than one sensor for the same physical quantity, then value group B shall be used to identify the sensors.

For detailed item names and OBIS codes see Table 96.

| Gas related general purpose objects | IC | OBIS code | | | | | |
|---|--|-----------|---|---|-----------|---|-----|
| | | A | B | C | D | E | F |
| Configuration objects | 1, Data ^a | 7 | b | 0 | 2 | e | 255 |
| Output pulse constants converted / unconverted | 1, Data ^a | 7 | b | 0 | 3 | e | 255 |
| Conversion factors | 1, Data ^a | 7 | b | 0 | 4 | e | 255 |
| Threshold values | 3, Register or 4, Extended register | 7 | b | 0 | 5 | e | 255 |
| Nominal values volume sensor | 3, Register or 4, Extended register | 7 | b | 0 | 6 | e | 255 |
| Input pulse constants | 1, Data ^a | 7 | b | 0 | 7 | e | 255 |
| Intervals and periods | 1, Data ^a | 7 | b | 0 | 8 | e | 255 |
| Time entries | 1, Data ^a | 7 | b | 0 | 9 | e | 255 |
| Station management information | 3, Register or 4, Extended register | 7 | b | 0 | 10, 11 | e | 255 |
| Gas parameters for volume conversion, currently used in compressibility calculation | 1, Data, 3, Register or 4, Extended register | 7 | b | 0 | 12 | e | 255 |
| Gas measuring method specific parameters | 1, Data, 3, Register or 4, Extended register | 7 | b | 0 | 13, 14 | e | 255 |
| Sensor manager objects | 67, Sensor manager | 7 | b | 0 | 15 | e | 255 |

^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

6.4.5 Gas related internal operating status objects

A number of gas related objects are available to hold information about the *internal operating status*. The status is held by the *value* attribute of a “Data” object, with data type *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-unsigned*, *octet-string* or *bit-string*. Alternatively, the status is held by a “Status mapping” object, which holds both the status word and the mapping of its bits to the reference table. If there are several gas related internal operating status objects used, it is allowed to combine them into an instance of the IC “Profile generic” or “Register table”, using the OBIS code of the global internal operating status.

| Gas related internal operating status objects | IC | OBIS code | | | | | |
|--|----------------------|-----------|---|----|---|-------|-----|
| | | A | B | C | D | E | F |
| Internal operating status signals, gas related, content manufacturer specific | 1, Data ^a | 7 | b | 96 | 5 | 0...9 | 255 |
| Internal operating status signals, gas related, content mapped to reference table | 63, Status mapping | 7 | b | 96 | 5 | 0...9 | 255 |
| ^a If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used. | | | | | | | |

6.4.6 Measured values

6.4.6.1 Indexes and index differences

Table 43 shows the objects to be used to represent indexes and index differences of volume, mass and energy.

Table 43 – Indexes and index differences

| Indexes and index differences | IC | OBIS code | | | | | |
|--|---------------------------------------|-----------|---|---|---|----------------|-----------------------------|
| | | A | B | C ^a | D ^b | E ^c | F |
| Index | | | | | 0...3 | | 255 |
| Index relative to billing periods | Register or Extended register | | | | 24...26 42...44 63...65 81...83 | | 0...99 101...126 |
| Index difference over measurement periods | 7 | | b | 1...8, 11...16, 21...26, 31...36, 61...66 | 6....23 | | 255 |
| Index difference over billing periods | | | | | 27...32, 45...50, 66...71, 84...89 | 0, 1...63 | 255 |
| Maximum of index differences over billing periods | Extended register, or Profile generic | | | | 33...41, 51...62, 72...80, 90...98 | | 0...99 101...126, 255 |
| ^a See Table 86 – Value group C codes – Gas | | | | | | | |
| ^b See Table 87 – Value group D codes – Gas – Indexes and index differences | | | | | | | |
| ^c See Table 92 – Value group E codes – Gas – Indexes and index differences – Tariff rates | | | | | | | |

6.4.6.2 Flow rate

Table 44 shows the objects to be used to represent flow rate values.

Table 44 – Flow rate

| Flow rate | IC | OBIS code | | | | | |
|--|--|-----------|---|----------------|---|---|----------------------------------|
| | | A | B | C ^a | D ^b | E | F |
| Flow rate, instantaneous values | Register or Extended register | 7 | b | 43 | 0, 1, 2, 13, | 0 | 255 |
| Flow rate current average and last average values over averaging periods | Register, Extended register or Demand register | | | | 15...18, 19...22, 35...38, 39...42, 55...58, 59...62, 63...66, 67...70 | 0 | 255 |
| Maximum of last averages of flow rates relative to measuring period | Extended register or Profile generic | | | | 23...26, 27...30, 43...46, 47...50 | | 255 |
| Maximum of last averages of flow rates relative to billing period 1 | Extended register or Profile generic | | | | 31...34, 51...54 | 0 | 0...99, 101... 126, 255 |

^a See Table 86 – Value group C codes – Gas
^b See Table 88 – Value group D codes – Gas – Flow rate

6.4.6.3 Process values

Table 45 shows the objects to be used to represent process values.

Table 45 – Process values

| Process values | IC | OBIS code | | | | | |
|--|-------------------------------|-----------|---|--------------------|---------------------------------------|---|-----|
| | | A | B | C ^a | D ^b | E | F |
| Measurands of the gas process in different conditions, average values, minima and maxima relative to different process intervals | Register or Extended register | 7 | b | 41, 42, 44...49 | 0, 2, 3, 10, 11, 13, 15...92 | 0 | 255 |

^a See Table 86 – Value group C codes – Gas
^b See Table 89 – Value group D codes – Gas – Process values

6.4.7 Conversion related factors and coefficients

Table 46 shows the objects available to represent correction, conversion, Compressibility, Superior calorific value and gas law deviation coefficient values. Various OBIS code allocations are made taking into consideration the specifics of the measuring process.

Table 46 – Conversion related factors and coefficients

| Conversion related factors and coefficients | IC | OBIS code | | | | | |
|---|-------------------------------|-----------|---|----------------|--------------------|----------------|-----|
| | | A | B | C ^a | D ^b | E ^c | F |
| Process independent current value or weighted value | Register or Extended register | 7 | b | 51...55 | 0, 2, 3, 10, 11 | 0,1 | 255 |
| Current average and last average values | | | | | | 11...28 | |

^a See Table 86 – Value group C codes – Gas
^b See Table 90 – Value group D codes – Gas – Conversion related factors and coefficients
^c See Table 93 – Value group E codes – Gas – Conversion related factors and coefficients

6.4.8 Calculation methods

Table 47 shows the OBIS codes of the objects used to identify calculation methods for correction, compression and compressibility calculation as well as for superior calorific value and gas law deviation coefficient.

Only one calculation method per calculation process can be in use, but gas measurement devices may have the capability to support various calculation methods. The choice of the methods may depend on regulation, national or company requirements etc. The use of these objects should be part of project specific companion specifications. The contents of the calculation methods listed are not defined in this document.

The calculation method is held by the *value* attribute with data types *octet-string*, *visible-string*, *unsigned*, *long-unsigned* or *enumerated*.

NOTE Calculation methods for compressibility factor Z can be found for example in ISO 12213-3, EN 12405-1, EN 12405-2 and AGA 8. EN 12405-1 also offers methods for volume conversion and EN 12405-2 for energy conversion.

Table 47 – Calculation methods

| Calculation methods | IC | OBIS code | | | | | |
|---------------------|-------------------------------------|-----------|---|----------------|----------------|----------------|-----|
| | | A | B | C ^a | D ^b | E ^c | F |
| Calculation methods | Data, Register or Extended register | 7 | b | 51...55 | 12 | 0...20 | 255 |

^a See Table 86 – Value group C codes – Gas
^b See Table 90 – Value group D codes – Gas – Conversion related factors and coefficients
^c See Table 94 – Value group E codes – Gas – Calculation methods

6.4.9 Natural gas analysis

Table 48 shows the objects to be used to represent values of natural gas analysis.

Table 48 – Natural gas analysis

| Natural gas analysis | IC | OBIS code | | | | | | |
|---|-------------------------------|-----------|---|----------------|---------------------|----------------|-----|--|
| | | A | B | C ^a | D ^b | E ^c | F | |
| Reference values of gas analysis process | Register or Extended register | 7 | b | 70 | 8,9 | 0 | 255 | |
| Gas characteristics | | | | | 10...20, 60...84 | 0,1 | | |
| Process independent current value, and weighted value | | | | | | 11...28 | | |
| Gas characteristics | | | | | | | | |
| Average values for current and last intervals | | | | | | | | |

^a See Table 86 – Value group C codes – Gas
^b See Table 91 – Value group D codes – Gas – Natural gas analysis values
^c See Table 95 – Value group E codes – Gas – Natural gas analysis values – Averages

6.4.10 List objects – Gas (class_id = 7)

These COSEM objects are used to model lists of any kind of data, for example measurement values, constants, statuses, events. They are modelled by “Profile generic” objects. See also 7.8.6.3.

One standard object per billing period scheme is defined. See also 7.8.6.3.

| List objects - Gas | IC | OBIS code | | | | | |
|--|--------------------|-----------|---|----|---|---|------------------|
| | | A | B | C | D | E | F |
| For names and OBIS codes see Table 98. | 7, Profile generic | 7 | b | 98 | d | e | 255 ^a |

^a F = 255 means a wildcard here. See 7.11.3.

6.5 Coding of OBIS identifications

To identify different instances of the same IC, their logical_name shall be different. In COSEM, the logical_name is taken from the OBIS definition (see Clauses 6.2, 6.3, 6.4 and 7).

OBIS codes are used within the COSEM environment as an *octet-string* [6]. Each octet contains the unsigned value of the corresponding OBIS value group, coded without tags.

If a data item is identified by less than six value groups, all unused value groups shall be filled with 255.

Octet 1 contains the binary coded value of A (A = 0, 1, 2 ...9) in the four rightmost bits. The four leftmost bits contain the information on the identification system. The four leftmost bits set to zero indicate that the OBIS identification system (version 1) is used as *logical_name*.

| Identification system used | Four leftmost bits of octet 1 (MSB left) |
|----------------------------|--|
| OBIS, see Clause 7. | 0 0 0 0 |
| Reserved for future use | 0 0 0 1 ... 1 1 1 1 |

Within all value groups, the usage of a certain selection is fully defined; others are reserved for future use. If in the value groups B to F a value belonging to the manufacturer specific range (see clause 7.2.2) is used, then the whole OBIS code shall be considered as manufacturer specific, and the value of the other groups does not necessarily carry a meaning defined neither by Clause 4 nor Clause 7 of this Technical Report.

7 COSEM Object Identification System (OBIS)

7.1 Scope

The OBject Identification System (OBIS) defines the identification codes for commonly used data items in metering equipment. This Clause 7 specifies the overall structure of the identification system and the mapping of all data items to their identification codes.

OBIS provides a unique identifier for all data within the metering equipment, including not only measurement values, but also abstract values used for configuration or obtaining information about the behaviour of the metering equipment. The ID codes defined in this standard are used for the identification of:

- logical names of the instances of the ICs, the objects, as defined in Clauses 4 and 6;
- data transmitted through communication lines, see Clause 7.11.1;
- data displayed on the metering equipment, see Clause 7.11.2.

This standard applies to all types of metering equipment, such as fully integrated meters, modular meters, tariff attachments, data concentrators etc.

To cover metering equipment measuring energy types other than electricity, combined metering equipment measuring more than one type of energy or metering equipment with several physical measurement channels, the concepts of medium and channels are introduced. This allows meter data originating from different sources to be identified.

7.2 OBIS code structure

7.2.1 Value groups and their use

OBIS codes identify data items used in energy metering equipment, in a hierarchical structure using six value groups A to F, see Table 49.

Table 49 – OBIS code structure and use of value groups

| Value group | Use of the value group |
|-------------|---|
| A | Identifies the media (energy type) to which the metering is related. Non-media related information is handled as abstract data. |
| B | Generally, identifies the measurement channel number, i.e. the number of the input of a metering equipment having several inputs for the measurement of energy of the same or different types (for example in data concentrators, registration units). Data from different sources can thus be identified. It may also identify the communication channel, and in some cases it may identify other elements. The definitions for this value group are independent from the value group A. |
| C | Identifies abstract or physical data items related to the information source concerned, for example current, voltage, power, volume, temperature. The definitions depend on the value in the value group A. Further processing, classification and storage methods are defined by value groups D, E and F. For abstract data, value groups D to F provide further classification of data identified by value groups A to C. |
| D | Identifies types, or the result of the processing of physical quantities identified by values in value groups A and C, according to various specific algorithms. The algorithms can deliver energy and demand quantities as well as other physical quantities. |
| E | Identifies further processing or classification of quantities identified by values in value groups A to D. |
| F | Identifies historical values of data, identified by values in value groups A to E, according to different billing periods. Where this is not relevant, this value group can be used for further classification. |

7.2.2 Manufacturer specific codes

In value groups B to F, the following ranges are available for manufacturer-specific purposes:

- group B: 128...199;
- group C: 128...199, 240;
- group D: 128...254;
- group E: 128...254;
- group F: 128...254.

If any of these value groups contain a value in the manufacturer specific range, then the whole OBIS code shall be considered as manufacturer specific, and the value of the other groups does not necessarily carry a meaning defined in Clause 6 or 7.

In addition, manufacturer specific ranges are defined in Table 56 with A = 0, C = 96, in Table 68 with A = 1, C = 96, in Table 75 with A = 4, C = 96, in Table 81 with A = 5/6, C = 96, in Table 96 with A = 7, C = 96 and in Table 102 with A = 8/9, C = 96.

7.2.3 Reserved ranges

By default, all codes not allocated are reserved.⁷

7.2.4 Summary of rules for manufacturer, utility, consortia and country specific codes

Table 50 summarizes the rules for manufacturer specific codes specified in 7.2.2, utility specific codes specified in 7.3.2, consortia specific codes specified in 7.3.4.2 and country specific codes specified in 7.3.4.3.

Table 50 – Rules for manufacturer, utility, consortia and country specific codes

| Code type | Value group | | | | | |
|---|----------------|-----------|----------------|---------------|-----------|-----------|
| | A | B | C | D | E | F |
| Manufacturer specific ¹ | 0, 1, 4...9, F | 128...199 | c | d | e | f |
| | | b | 128...199, 240 | d | e | f |
| | | b | c | 128...254 | e | f |
| | | b | c | d | 128...254 | f |
| | | b | c | d | e | 128...254 |
| Manufacturer specific abstract ² | 0 | 0...64 | 96 | 50...99 | 0...255 | 0...255 |
| Manufacturer specific, media related general purpose ² | 1, 4...9, F | 0...64 | 96 | 50...99 | 0...255 | 0...255 |
| Utility specific ³ | 0, 1, 4...9, F | 65...127 | 0...255 | 0...255 | 0...255 | 0...255 |
| Consortia specific ⁴ | 0, 1, 4...9, F | 0...64 | 93 | See Table 54. | | |
| Country specific ⁵ | | 0...64 | 94 | See Table 55. | | |

⁷ Administered by the DLMS User Association (see Foreword).

| | |
|--------|---|
| NOTE 1 | "b", "c", "d", "e", "f" means any value in the relevant value group. |
| NOTE 2 | The range D = 50...99 is available for identifying objects, which are not represented by another defined code, but need representation on the display as well. If this is not required, the range D = 128...254 should be used. |
| NOTE 3 | If the value in value group B is 65...127, the whole OBIS code should be considered as utility specific and the value of other groups does not necessarily carry a meaning defined neither in Clause 6 nor Clause 7. |
| NOTE 4 | The usage of value group E and F are defined in consortia specific documents. |
| NOTE 5 | The usage of value group E and F are defined in country specific documents. |

Objects for which this Technical Report defines standard identifiers shall not be re-identified by manufacturer, utility, consortia or country specific identifiers.

On the other hand, an object previously identified by a manufacturer-, utility-, consortia- or country- specific identifier may receive a standard identifier in the future, if its use is of common interest for the users of this standard.

7.2.5 Standard object codes

Standard object codes are meaningful combinations of defined values of the six value groups.

Notation: In the following tables, in the various value groups, "b", "c", "d", "e", "f" signifies any value in the respective value group. If only one object is instantiated, the value shall be 0. If a value group is shaded, then this value group is not used.

NOTE The DLMS UA maintains a list of standard COSEM object definitions at www.dlms.com. The validity of the combination of OBIS codes and class_id-s as well as the data types of the attributes are tested during conformance testing.

7.3 Value group definitions – overview

7.3.1 Value group A

The range for value group A is 0 to 15; see Table 51.

Table 51 – Value group A codes

| Value group A | |
|---------------|-------------------------------------|
| 0 | Abstract objects |
| 1 | Electricity related objects |
| ... | |
| 4 | Heat cost allocator related objects |
| 5, 6 | Thermal energy related objects |
| 7 | Gas related objects |
| 8 | Cold water related objects |
| 9 | Hot water related objects |
| ... | |
| 15 | Other media |
| All other | Reserved |

The following subclauses contain value group definitions B to F common for all values of value group A.

7.3.2 Value group B

The range for value group B is 0 to 255; see Table 52.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 391/492 |
|-----------------------|------------|-------------------------|---------|

Table 52 – Value group B codes

| Value group B | |
|------------------|-----------------------------|
| 0 | No channel specified |
| 1...64 | Channel 1..64 |
| | |
| 65...127 | Utility specific codes |
| 128...199 | Manufacturer specific codes |
| 200...255 | Reserved |

If channel information is not essential, the value 0 shall be assigned.

The range 65...127 is available for utility specific use. If the value of value group B is in this range, the whole OBIS code shall be considered as utility specific and the value of other groups does not necessarily carry a meaning defined neither in Clause 4 nor in Clause 7.

7.3.3 Value group C

7.3.3.1 General

The range for value group C is 0 to 255. The definitions depend on the value in value group A. The codes for abstract objects are specified in 7.3.3.2. See also:

- electricity related codes specified in 7.5.1;
- heat cost allocator related codes specified in 7.6.2;
- **thermal energy** related codes specified in 7.7.2;
- gas related codes specified in 7.8.2;
- water related codes specified in 7.9.2;
- other media related codes specified in 7.10.2.

7.3.3.2 Abstract objects

Abstract objects are data items, which are not related to a certain type of physical quantity. See Table 53.

Table 53 – Value group C codes – Abstract objects

| Value group C Abstract objects (A = 0) | |
|---|--|
| 0...89 | Context specific identifiers ^a |
| | |
| 93 | Consortia specific identifiers (See 7.3.4.2). |
| 94 | Country specific identifiers (See 7.3.4.3) |
| | |
| 96 | General and service entry objects – Abstract (See 7.4.1) |
| 97 | Error register objects – Abstract (See 7.4.2) |
| 98 | List objects – Abstract (See 7.4.3, 7.4.4) |
| 99 | Data profile objects – Abstract (See 7.4.5) |
| ... | |
| 127 | Inactive objects ^b |

| | |
|-----------------------|---|
| 128...199, 240 | Manufacturer specific codes |
| All other | Reserved |
| ^a | Context specific identifiers identify objects specific to a certain protocol and/or application. For the COSEM context, the identifiers are defined in 6.2. |
| ^b | An inactive object is an object, which is defined and present in a meter, but which has no assigned functionality. |

7.3.4 Value group D

7.3.4.1 General

The range for value group D is 0 to 255.

7.3.4.2 Consortia specific identifiers

Table 54 specifies the use of value group D for consortia specific applications. In this table, there are no reserved ranges for manufacturer specific codes. The usage of value group E and F are defined in consortia specific documents.

Objects that are already identified in this Technical Specification shall not be re-identified by consortia specific identifiers.

Table 54 – Value group D codes – Consortia specific identifiers

| Value group D Consortia specific identifiers (A = any, C = 93) | |
|--|----------|
| All values | Reserved |
| NOTE At the time of the publication of this document, no consortia specific identifiers are allocated. | |

7.3.4.3 Country specific identifiers

Table 55 specifies the use of value group D for country specific applications. Wherever possible, the country calling codes are used. In this table, there are no reserved ranges for manufacturer specific codes. The usage of value group E and F are defined in country specific documents.

Objects that are already identified in this standard shall not be re-identified by country specific identifiers.

Table 55 – Value group D codes – Country specific identifiers

| Value group D Country specific identifiers ^a (A = any, C = 94) | | | |
|--|---|-----------|---|
| 00 | Finland (Country calling code = 358) | 50 | |
| 01 | USA (= Country calling code) | 51 | Peru (= Country calling code) |
| 02 | Canada (Country calling code = 1) | 52 | South Korea (Country calling code = 82) |
| 03 | Serbia (Country calling code = 381) | 53 | Cuba (= Country calling code) |
| 04 | | 54 | Argentina (= Country calling code) |
| 05 | | 55 | Brazil (= Country calling code) |
| 06 | | 56 | Chile (= Country calling code) |
| 07 | Russia (Country calling code = 7) | 57 | Colombia (= Country calling code) |
| 08 | | 58 | Venezuela (= Country calling code) |
| 09 | | 59 | |
| 10 | Czech Republic (Country calling code = 420) | 60 | Malaysia (= Country calling code) |
| 11 | Bulgaria (Country calling code = 359) | 61 | Australia (= Country calling code) |
| 12 | Croatia (Country calling code = 385) | 62 | Indonesia (= Country calling code) |
| 13 | Ireland (Country calling code = 353) | 63 | Philippines (= Country calling code) |
| 14 | Israel (Country calling code = 972) | 64 | New Zealand (= Country calling code) |
| 15 | Ukraine (Country calling code = 380) | 65 | Singapore (= Country calling code) |
| 16 | Yugoslavia ^a | 66 | Thailand (= Country calling code) |
| 17 | | 67 | |
| 18 | | 68 | |
| 19 | | 69 | |
| 20 | Egypt (= Country calling code) | 70 | |
| 21 | | 71 | |
| 22 | | 72 | |
| 23 | | 73 | Moldova (Country calling code = 373) |
| 24 | | 74 | |
| 25 | | 75 | Belarus (Country calling code = 375) |
| 26 | | 76 | |
| 27 | South Africa (= Country calling code) | 77 | |
| 28 | | 78 | |
| 29 | | 79 | |
| 30 | Greece (= Country calling code) | 80 | |
| 31 | Netherlands (= Country calling code) | 81 | Japan (= Country calling code) |
| 32 | Belgium (= Country calling code) | 82 | |
| 33 | France (= Country calling code) | 83 | |
| 34 | Spain (= Country calling code) | 84 | |
| 35 | Portugal (Country calling code = 351) | 85 | Hong Kong (Country calling code = 852) |
| 36 | Hungary (= Country calling code) | 86 | China (= Country calling code) |
| 37 | Lithuania (Country calling code = 370) | 87 | Bosnia and Herzegovina (Country calling code = 387) |

| Value group D Country specific identifiers ^a (A = any, C = 94) | | | |
|--|--|-----------|---|
| 38 | Slovenia (Country calling code = 386) | 88 | |
| 39 | Italy (= Country calling code) | 89 | |
| 40 | Romania (= Country calling code) | 90 | Turkey (= Country calling code) |
| 41 | Switzerland (= Country calling code) | 91 | India (= Country calling code) |
| 42 | Slovakia (Country calling code = 421) | 92 | Pakistan (= Country calling code) |
| 43 | Austria (= Country calling code) | 93 | |
| 44 | United Kingdom (= Country calling code) | 94 | |
| 45 | Denmark (= Country calling code) | 95 | |
| 46 | Sweden (= Country calling code) | 96 | Saudi Arabia (Country calling code = 966) |
| 47 | Norway (= Country calling code) | 97 | United Arab Emirates (Country calling code = 971) |
| 48 | Poland (= Country calling code) | 98 | Iran (= Country calling code) |
| 49 | Germany (= Country calling code) | 99 | |
| All other codes are reserved | | | |
| ^a | With the dissolution of the former Yugoslavia into separate nations, country code 38 was decommissioned. | | |

7.3.4.4 Identification of general and service entry objects

For the use of value group D to identify:

- abstract general and service entry objects, see 7.4, Table 56;
- electricity related general and service entry objects, see Table 68;
- HCA related general and service entry objects, see Table 75;
- thermal energy related general and service entry objects, see Table 81;
- gas related general and service entry objects, see Table 96;
- water related general and service entry objects, see Table 102.

7.3.5 Value group E

The range for value group E is 0 to 255. It can be used for identifying further classification or processing of values defined by values in value groups A to D, as specified in the relevant energy type specific clauses. The various classifications and processing methods are exclusive.

For the use of value group E to identify:

- abstract general and service entry objects, see 7.4, Table 56;
- electricity related general and service entry objects, see Table 68;
- HCA related general and service entry objects, see Table 75;
- thermal energy related general and service entry objects, see Table 81;
- gas related general and service entry objects, see Table 96;
- water related general and service entry objects, see Table 102.

7.3.6 Value group F

7.3.6.1 General

The range for value group F is 0 to 255. In all cases, if value group F is not used, it is set to 255.

7.3.6.2 Identification of billing periods

Value group F specifies the allocation to different billing periods (sets of historical values) for the objects defined by value groups A to E, where storage of historical values is relevant. A billing period scheme is identified with its billing period counter, number of available billing periods, time stamp of the billing period and billing period length. Several billing period schemes may be possible. For more, see 6.2.2 and 7.11.3.

7.4 Abstract objects (Value group A = 0)

7.4.1 General and service entry objects – Abstract

Table 56 specifies OBIS codes for abstract objects. See also Table 37.

Table 56 – OBIS codes for general and service entry objects

| General and service entry objects | OBIS code | | | | | |
|--|-----------|-----|-----|-----|-----|------------------|
| | A | B | C | D | E | F |
| Billing period values/reset counter entries (First billing period scheme if there are two) | | | | | | |
| Billing period counter (1) | 0 | b | 0 | 1 | 0 | VZ or 255 |
| Number of available billing periods (1) | 0 | b | 0 | 1 | 1 | |
| Time stamp of the most recent billing period (1) | 0 | b | 0 | 1 | 2 | |
| Time stamp of the billing period (1) VZ (last reset) | 0 | b | 0 | 1 | 2 | VZ |
| Time stamp of the billing period (1) VZ ₋₁ | 0 | b | 0 | 1 | 2 | VZ ₋₁ |
| ... | ... | ... | ... | ... | ... | ... |
| Time stamp of the billing period (1) VZ _{-n} | 0 | b | 0 | 1 | 2 | VZ _{-n} |
| Billing period values/reset counter entries (Second billing period scheme) | | | | | | |
| Billing period counter (2) | 0 | b | 0 | 1 | 3 | VZ or 255 |
| Number of available billing periods (2) | 0 | b | 0 | 1 | 4 | |
| Time stamp of the most recent billing period (2) | 0 | b | 0 | 1 | 5 | |
| Time stamp of the billing period (2) VZ (last reset) | 0 | b | 0 | 1 | 5 | VZ |
| Time stamp of the billing period (2) VZ ₋₁ | 0 | b | 0 | 1 | 5 | VZ ₋₁ |
| ... | ... | ... | ... | ... | ... | ... |
| Time stamp of the billing period (2) VZ _{-n} | 0 | b | 0 | 1 | 5 | VZ _{-n} |
| Program entries | | | | | | |
| Active firmware identifier | 0 | b | 0 | 2 | 0 | |
| Active firmware version | 0 | b | 0 | 2 | 1 | |
| Active firmware signature | 0 | b | 0 | 2 | 8 | |
| Time entries | | | | | | |
| Local time | 0 | b | 0 | 9 | 1 | |
| Local date | 0 | b | 0 | 9 | 2 | |

| General and service entry objects | OBIS code | | | | | |
|---|-----------|---|-----|-----|------------|---|
| | A | B | C | D | E | F |
| Device IDs | | | | | | |
| Complete device ID | 0 | b | 96 | 1 | | |
| Device ID # 1 (manufacturing number) | 0 | b | 96 | 1 | 0 | |
| ... | | | ... | ... | ... | |
| Device ID # 10 | 0 | b | 96 | 1 | 9 | |
| Metering point ID (abstract) | 0 | 0 | 96 | 1 | 10 | |
| Parameter changes, calibration and access | | | | | | |
| Number of configuration program changes | 0 | b | 96 | 2 | 0 | |
| Date ^a of last configuration program change | 0 | b | 96 | 2 | 1 | |
| Date ^a of last time switch program change | 0 | b | 96 | 2 | 2 | |
| Date ^a of last ripple control receiver program change | 0 | b | 96 | 2 | 3 | |
| Status of security switches | 0 | b | 96 | 2 | 4 | |
| Date ^a of last calibration | 0 | b | 96 | 2 | 5 | |
| Date ^a of next configuration program change | 0 | b | 96 | 2 | 6 | |
| Date ^a of activation of the passive calendar | 0 | b | 96 | 2 | 7 | |
| Number of protected configuration program changes ^b | 0 | b | 96 | 2 | 10 | |
| Date ^a of last protected configuration program change ^b | 0 | b | 96 | 2 | 11 | |
| Date ^a (corrected) of last clock synchronization/setting | 0 | b | 96 | 2 | 12 | |
| Date of last firmware activation | 0 | b | 96 | 2 | 13 | |
| Input/output control signals | | | | | | |
| State of input/output control signals, global ^c | 0 | b | 96 | 3 | 0 | |
| State of input control signals (status word 1) | 0 | b | 96 | 3 | 1 | |
| State of output control signals (status word 2) | 0 | b | 96 | 3 | 2 | |
| State of input/output control signals (status word 3) | 0 | b | 96 | 3 | 3 | |
| State of input/output control signals (status word 4) | 0 | b | 96 | 3 | 4 | |
| Disconnect control | 0 | b | 96 | 3 | 10 | |
| Arbitrator | 0 | b | 96 | 3 | 20.. 29 | |
| Internal control signals | | | | | | |
| Internal control signals, global ^c | 0 | b | 96 | 4 | 0 | |
| Internal control signals (status word 1) | 0 | b | 96 | 4 | 1 | |
| Internal control signals (status word 2) | 0 | b | 96 | 4 | 2 | |
| Internal control signals (status word 3) | 0 | b | 96 | 4 | 3 | |
| Internal control signals (status word 4) | 0 | b | 96 | 4 | 4 | |
| Internal operating status | | | | | | |
| Internal operating status, global ^c | 0 | b | 96 | 5 | 0 | |
| Internal operating status (status word 1) | 0 | b | 96 | 5 | 1 | |
| Internal operating status (status word 2) | 0 | b | 96 | 5 | 2 | |
| Internal operating status (status word 3) | 0 | b | 96 | 5 | 3 | |
| Internal operating status (status word 4) | 0 | b | 96 | 5 | 4 | |
| Battery entries | | | | | | |
| Battery use time counter | 0 | b | 96 | 6 | 0 | |
| Battery charge display | 0 | b | 96 | 6 | 1 | |
| Date of next battery change | 0 | b | 96 | 6 | 2 | |
| Battery voltage | 0 | b | 96 | 6 | 3 | |
| Battery initial capacity | 0 | b | 96 | 6 | 4 | |
| Battery installation date and time | 0 | b | 96 | 6 | 5 | |

| General and service entry objects | OBIS code | | | | | |
|--|-----------|---|----|----|--------|---|
| | A | B | C | D | E | F |
| Battery estimated remaining use time | 0 | b | 96 | 6 | 6 | |
| Aux. supply use time counter | 0 | b | 96 | 6 | 10 | |
| Aux. voltage (measured) | 0 | b | 96 | 6 | 11 | |
| Power failure monitoring | | | | | | |
| Number of power failures | | | | | | |
| In all three phases | 0 | 0 | 96 | 7 | 0 | |
| In phase L1 | 0 | 0 | 96 | 7 | 1 | |
| In phase L2 | 0 | 0 | 96 | 7 | 2 | |
| In phase L3 | 0 | 0 | 96 | 7 | 3 | |
| In any phase [sic] | 0 | 0 | 96 | 7 | 21 | |
| Auxiliary supply | 0 | 0 | 96 | 7 | 4 | |
| Number of long power failures | | | | | | |
| In all three phases | 0 | 0 | 96 | 7 | 5 | |
| In phase L1 | 0 | 0 | 96 | 7 | 6 | |
| In phase L2 | 0 | 0 | 96 | 7 | 7 | |
| In phase L3 | 0 | 0 | 96 | 7 | 8 | |
| In any phase | 0 | 0 | 96 | 7 | 9 | |
| Time of power failure ^d | | | | | | |
| In all three phases | 0 | 0 | 96 | 7 | 10 | |
| In phase L1 | 0 | 0 | 96 | 7 | 11 | |
| In phase L2 | 0 | 0 | 96 | 7 | 12 | |
| In phase L3 | 0 | 0 | 96 | 7 | 13 | |
| In any phase | 0 | 0 | 96 | 7 | 14 | |
| Duration of long power failure ^e | | | | | | |
| In all three phases | 0 | 0 | 96 | 7 | 15 | |
| In phase L1 | 0 | 0 | 96 | 7 | 16 | |
| In phase L2 | 0 | 0 | 96 | 7 | 17 | |
| In phase L3 | 0 | 0 | 96 | 7 | 18 | |
| In any phase | 0 | 0 | 96 | 7 | 19 | |
| Time threshold for long power failure | | | | | | |
| Time threshold for long power failure | 0 | 0 | 96 | 7 | 20 | |
| NOTE 1 See Number of power failures in any phase above | 0 | b | 96 | 7 | 21 | |
| Operating time | | | | | | |
| Time of operation | 0 | b | 96 | 8 | 0 | |
| Time of operation rate 1...rate 63 | 0 | b | 96 | 8 | 1...63 | |
| Environment related parameters | | | | | | |
| Ambient temperature | 0 | b | 96 | 9 | 0 | |
| Ambient pressure | 0 | b | 96 | 9 | 1 | |
| Relative humidity | 0 | b | 96 | 9 | 2 | |
| Status register | | | | | | |
| Status register (Status register 1 if several status registers are used) | 0 | b | 96 | 10 | 1 | |
| Status register 2 | 0 | b | 96 | 10 | 2 | |
| ... | 0 | b | 96 | 10 | ... | |
| Status register 10 | 0 | b | 96 | 10 | 10 | |
| Event code | | | | | | |

| General and service entry objects | OBIS code | | | | | |
|--|------------------|----------|----------|----------|----------|----------|
| | A | B | C | D | E | F |
| Event code objects # 1...#100 | 0 | b | 96 | 11 | 0... | |
| | | | | | 99 | |
| Communication port log parameters | | | | | | |
| Reserved | 0 | b | 96 | 12 | 0 | |
| Number of connections | 0 | b | 96 | 12 | 1 | |
| Reserved | 0 | b | 96 | 12 | 2 | |
| Reserved | 0 | b | 96 | 12 | 3 | |
| Communication port parameter 1 | 0 | b | 96 | 12 | 4 | |
| GSM field strength | 0 | b | 96 | 12 | 5 | |
| Telephone number / Communication address of the physical device | 0 | b | 96 | 12 | 6 | |
| Consumer messages | | | | | | |
| Consumer message via local consumer information port | 0 | b | 96 | 13 | 0 | |
| Consumer message via the meter display and / or via consumer information port | 0 | b | 96 | 13 | 1 | |
| Currently active tariff | | | | | | |
| Currently active tariff objects # 1...#16 | 0 | b | 96 | 14 | 0... | |
| NOTE 2 Object #16 (E = 15) carries the name of register with the lowest tariff (default tariff register) | | | | | 15 | |
| Event counter objects | | | | | | |
| Event counter objects #1...#100 | 0 | b | 96 | 15 | 0... | |
| | | | | | 99 | |
| Profile entry digital signature objects | | | | | | |
| Profile entry digital signature objects #1...#10 | 0 | b | 96 | 16 | 0... | |
| | | | | | 9 | |
| Meter tamper event related objects | | | | | | |
| Meter open event counter | 0 | b | 96 | 20 | 0 | |
| Meter open event, time stamp of current event occurrence | 0 | b | 96 | 20 | 1 | |
| Meter open event, duration of current event | 0 | b | 96 | 20 | 2 | |
| Meter open event, cumulative duration | 0 | b | 96 | 20 | 3 | |
| Reserved | 0 | b | 96 | 20 | 4 | |
| Terminal cover open event counter | 0 | b | 96 | 20 | 5 | |
| Terminal cover open event, time stamp of current event occurrence | 0 | b | 96 | 20 | 6 | |
| Terminal cover open event, duration of current event | 0 | b | 96 | 20 | 7 | |
| Terminal cover open event, cumulative duration | 0 | b | 96 | 20 | 8 | |
| Reserved | 0 | b | 96 | 20 | 9 | |
| Tilt event counter | 0 | b | 96 | 20 | 10 | |
| Tilt event, time stamp of current event occurrence | 0 | b | 96 | 20 | 11 | |
| Tilt event, duration of current event | 0 | b | 96 | 20 | 12 | |
| Tilt event, cumulative duration | 0 | b | 96 | 20 | 13 | |
| Reserved | 0 | b | 96 | 20 | 14 | |
| Strong DC magnetic field event counter | 0 | b | 96 | 20 | 15 | |
| Strong DC magnetic field event, time stamp of current event occurrence | 0 | b | 96 | 20 | 16 | |
| Strong DC magnetic field event, duration of current event | 0 | b | 96 | 20 | 17 | |
| Strong DC magnetic field event, cumulative duration | 0 | b | 96 | 20 | 18 | |
| Reserved | 0 | b | 96 | 20 | 19 | |
| Supply control switch / valve tamper event counter | 0 | b | 96 | 20 | 20 | |
| Supply control switch / valve tamper event, time stamp of current event occurrence | 0 | b | 96 | 20 | 21 | |

| General and service entry objects | OBIS code | | | | | |
|---|---|---|----|----|----|---|
| | A | B | C | D | E | F |
| Supply control switch / valve tamper event, duration of current event | 0 | b | 96 | 20 | 22 | |
| Supply control switch / valve tamper event, cumulative duration | 0 | b | 96 | 20 | 23 | |
| Reserved | 0 | b | 96 | 20 | 24 | |
| Metrology tamper event counter | 0 | b | 96 | 20 | 25 | |
| Metrology tamper event, time stamp of current event occurrence | 0 | b | 96 | 20 | 26 | |
| Metrology tamper event, duration of current event | 0 | b | 96 | 20 | 27 | |
| Metrology tamper event, cumulative duration | 0 | b | 96 | 20 | 28 | |
| Reserved | 0 | b | 96 | 20 | 29 | |
| Communication tamper event counter | 0 | b | 96 | 20 | 30 | |
| Communication tamper event, time stamp of current event occurrence | 0 | b | 96 | 20 | 31 | |
| Communication tamper event, duration of current event | 0 | b | 96 | 20 | 32 | |
| Communication tamper event, cumulative duration | 0 | b | 96 | 20 | 33 | |
| Reserved | 0 | b | 96 | 20 | 34 | |
| Manufacturer specific ^f | 0 | b | 96 | 50 | e | f |
| ... | | | | | | |
| Manufacturer specific | 0 | b | 96 | 99 | e | f |
| All other codes are reserved | | | | | | |
| a | Date of the event may contain the date only, the time only or both, encoded as specified in 4.1.6.1. | | | | | |
| b | Protected configuration is characterized by the need to open the main meter cover to modify it, or to break a metrological seal. | | | | | |
| c | Global status words with E = 0 contain the individual status words E = 1...4. The contents of the status words are not defined in this standard. | | | | | |
| d | Time of power failure is recorded when either a short or long power failure occurs. | | | | | |
| e | Duration of long power failure holds the duration of the last long power failure. | | | | | |
| f | The range D = 50...99 is available for identifying objects, which are not represented by another defined code, but need representation on the display as well. If this is not required, the range D = 128...254 should be used. | | | | | |

7.4.2 Error registers, alarm registers / filters / descriptor objects – Abstract

The OBIS codes for abstract error registers, alarm registers and alarm filters are shown in Table 57.

Table 57 – OBIS codes for error registers, alarm registers and alarm filters – Abstract

| Error register, alarm register and alarm filter objects – Abstract | OBIS code | | | | | |
|--|-----------|---|----|----|---------|---|
| | A | B | C | D | E | F |
| Error register objects 1...10 | 0 | b | 97 | 97 | 0...9 | |
| Alarm register objects 1...10 | 0 | b | 97 | 98 | 0...9 | |
| Alarm filter objects 1...10 | 0 | b | 97 | 98 | 10...19 | |
| Alarm descriptor objects 1...10 | 0 | b | 97 | 98 | 20...29 | |

NOTE The information to be included in the error objects is not defined in this document.

7.4.3 List objects – Abstract

Lists – identified with a single OBIS code – are defined as a series of any kind of data (for example measurement value, constants, status, events). See Table 58.

Table 58 – OBIS codes for list objects – Abstract

| List objects – Abstract | OBIS code | | | | | |
|--|-----------|---|----|---|---|------------------|
| | A | B | C | D | E | F |
| Data of billing period (with billing period scheme 1 if there are more than one schemes available) | 0 | b | 98 | 1 | e | 255 ^a |
| Data of billing period (with billing period scheme 2) | 0 | b | 98 | 2 | e | 255 ^a |
| ^a F = 255 means a wildcard here. See 7.11.3. | | | | | | |

7.4.4 Register table objects – Abstract

Register tables are defined to hold a number of values of the same type. See Table 59.

Table 59 – OBIS codes for Register table objects – Abstract

| Register table objects – Abstract | OBIS code | | | | | |
|-----------------------------------|-----------|---|----|----|---|---|
| | A | B | C | D | E | F |
| General use, abstract | 0 | b | 98 | 10 | e | |

7.4.5 Data profile objects – Abstract

Abstract data profiles – instances of the “Profile generic IC” and identified with one single OBIS code as specified in Table 60 – are used to hold a series of measurement values of one or more similar quantities and/or to group various data.

Table 60 – OBIS codes for data profile objects – Abstract

| Data profile objects – Abstract | OBIS code | | | | | |
|---|-----------|---|----|----|---|---|
| | A | B | C | D | E | F |
| Load profile with recording period 1 ^a | 0 | b | 99 | 1 | e | |
| Load profile with recording period 2 ^a | 0 | b | 99 | 2 | e | |
| Load profile during test ^a | 0 | b | 99 | 3 | 0 | |
| Connection profile | 0 | b | 99 | 12 | e | |
| GSM diagnostic profile | 0 | b | 99 | 13 | e | |
| Charge collection history (Payment metering) | 0 | b | 99 | 14 | e | |
| Token credit history (Payment metering) | 0 | b | 99 | 15 | e | |
| Parameter monitor log | 0 | b | 99 | 16 | e | |
| Token transfer log (Payment metering) | 0 | b | 99 | 17 | e | |
| Event log ^a | 0 | b | 99 | 98 | e | |
| ^a These objects should be used if they (also) hold data not specific to the energy type. | | | | | | |

7.5 Electricity (Value group A = 1)

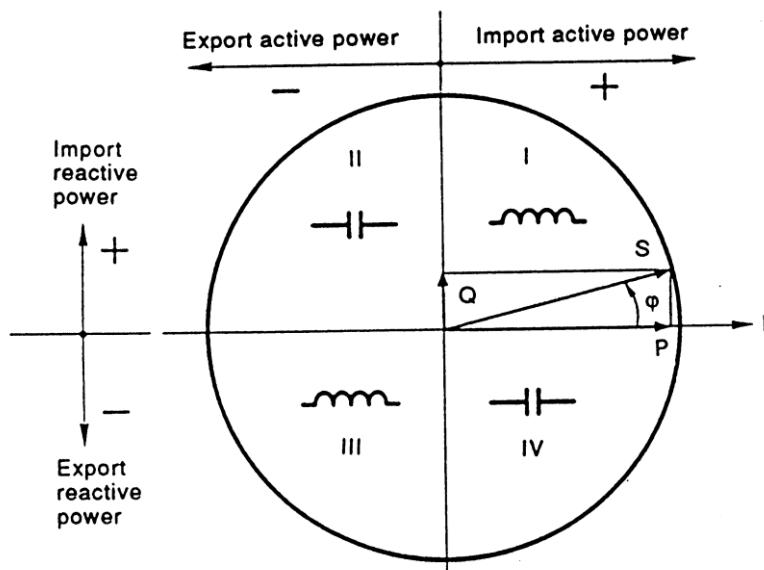
7.5.1 Value group C codes – Electricity

Table 61 specifies the use of value group C for electricity related objects.

Table 61 – Value group C codes – Electricity

| Value group C codes – Electricity (A = 1) | | | | |
|---|--|-------|-------|--|
| 0 | General purpose objects (See 7.5.5.1) | | | |
| ΣL_i | L_1 | L_2 | L_3 | (See also Note 2) |
| 1 | 21 | 41 | 61 | Active power+ (QI+QIV) |
| 2 | 22 | 42 | 62 | Active power- (QII+QIII) |
| 3 | 23 | 43 | 63 | Reactive power+ (QI+QII) |
| 4 | 24 | 44 | 64 | Reactive power- (QIII+QIV) |
| 5 | 25 | 45 | 65 | Reactive power QI |
| 6 | 26 | 46 | 66 | Reactive power QII |
| 7 | 27 | 47 | 67 | Reactive power QIII |
| 8 | 28 | 48 | 68 | Reactive power QIV |
| 9 | 29 | 49 | 69 | Apparent power+ (QI+QIV) (See also Note 3) |
| 10 | 30 | 50 | 70 | Apparent power- (QII+QIII) |
| 11 | 31 | 51 | 71 | Current: any phase (C = 11) / L_i phase ^a (C= 31, 51, 71) |
| 12 | 32 | 52 | 72 | Voltage: any phase (C = 12) / L_i phase ^a (C= 32, 52, 72) |
| 13 | 33 | 53 | 73 | Power factor (See also Note 4) |
| 14 | 34 | 54 | 74 | Supply frequency |
| 15 | 35 | 55 | 75 | Active power (abs(QI+QIV)+(abs(QII+QIII)) ^a |
| 16 | 36 | 56 | 76 | Active power (abs(QI+QIV)-abs(QII+QIII)) |
| 17 | 37 | 58 | 77 | Active power QI |
| 18 | 38 | 58 | 78 | Active power QII |
| 19 | 39 | 59 | 79 | Active power QIII |
| 20 | 40 | 60 | 80 | Active power QIV |
| | | | | |
| 81 | Angles ^b | | | |
| 82 | Unitless quantity (pulses or pieces) | | | |
| 83 | Transformer and line loss quantities ^c | | | |
| 84 | ΣL_i Power factor- (See also Note 4) | | | |
| 85 | L_1 Power factor- | | | |
| 86 | L_2 Power factor- | | | |
| 87 | L_3 Power factor- | | | |
| 88 | ΣL_i Ampere-squared hours (QI+QII+QIII+QIV) | | | |
| 89 | ΣL_i Volt-squared hours (QI+QII+QIII+QIV) | | | |
| 90 | ΣL_i current (algebraic sum of the – unsigned – value of the currents in all phases) | | | |
| 91 | L_0 current (neutral) ^a | | | |
| 92 | L_0 voltage (neutral) ^a | | | |

| Value group C codes – Electricity (A = 1) | |
|---|---|
| 93 | Consortia specific identifiers (See 7.3.4.2) |
| 94 | Country specific identifiers (See 7.3.4.3) |
| | |
| 96 | General and service entry objects – Electricity (See 7.5.5.1) |
| 97 | Error register objects – Electricity (See 7.5.5.2) |
| 98 | List objects – Electricity (See 7.5.5.3) |
| 99 | Data profile objects – Electricity (See 7.5.5.4) |
| 100...127 | Reserved |
| | |
| 128...199, 240 | Manufacturer specific codes |
| All other | Reserved |
| NOTE 1 | L_i , Quantity is the value (to be measured) of a measurement system connected between the phase i and a reference point. In 3-phase 4-wire systems, the reference point is the neutral. In 3-phase 3-wire systems, the reference point is the phase L_2 . |
| NOTE 2 | ΣL , Quantity is the total measurement value across all systems. |
| NOTE 3 | If just one apparent energy/demand value is calculated over the four quadrants, C = 9 shall be used. |
| NOTE 4 | Power factor quantities with C = 13, 33, 53, 73 are calculated either as PF = Active power+ (C = 1, 21, 41, 61) / Apparent power+ (C = 9, 29, 49, 69) or PF = Active power- (C = 2, 22, 42, 62) / Apparent power- (C = 10, 30, 50, 70). In the first case, the sign is positive (no sign), it means power factor in the import direction (PF+). In the second case, the sign is negative, it means power factor in the export direction (PF-). Power factor quantities C = 84, 85, 86 and 87 are always calculated as PF- = Active power- / Apparent power-. This quantity is the power factor in the export direction; it has no sign. |
| ^a | For details of extended codes, see 7.5.3.3. |
| ^b | For details of extended codes, see 7.5.3.4. |
| ^c | For details of extended codes, see 7.5.3.5. |



NOTE The quadrant definitions are according to IEC 62053-23:2003 Figure C1.

Figure 30 – Quadrant definitions for active and reactive power

7.5.2 Value group D codes – Electricity

7.5.2.1 Processing of measurement values

Table 62 specifies the use of value group D for electricity related objects.

Table 62 – Value group D codes – Electricity

| Value group D codes – Electricity (A = 1, C > 0, 93, 94, 96, 97, 98, 99) | |
|--|---|
| 0 | Billing period average (since last reset) |
| 1 | Cumulative minimum 1 |
| 2 | Cumulative maximum 1 |
| 3 | Minimum 1 |
| 4 | Current average 1 |
| 5 | Last average 1 |
| 6 | Maximum 1 |
| 7 | Instantaneous value |
| 8 | Time integral 1 |
| 9 | Time integral 2 |
| 10 | Time integral 3 |
| 11 | Cumulative minimum 2 |
| 12 | Cumulative maximum 2 |
| 13 | Minimum 2 |
| 14 | Current average 2 |
| 15 | Last average 2 |
| 16 | Maximum 2 |
| 17 | Time integral 7 |
| 18 | Time integral 8 |
| 19 | Time integral 9 |
| 20 | Time integral 10 |
| 21 | Cumulative minimum 3 |
| 22 | Cumulative maximum 3 |
| 23 | Minimum 3 |
| 24 | Current average 3 |
| 25 | Last average 3 |
| 26 | Maximum 3 |
| 27 | Current average 5 |
| 28 | Current average 6 |
| 29 | Time integral 5 |
| 30 | Time integral 6 |
| 31 | Under limit threshold |
| 32 | Under limit occurrence counter |
| 33 | Under limit duration |
| 34 | Under limit magnitude |
| 35 | Over limit threshold |

| Value group D codes – Electricity (A = 1, C > 0, 93, 94, 96, 97, 98, 99) | |
|--|--|
| 36 | Over limit occurrence counter |
| 37 | Over limit duration |
| 38 | Over limit magnitude |
| | |
| 39 | Missing threshold |
| 40 | Missing occurrence counter |
| 41 | Missing duration |
| 42 | Missing magnitude |
| | |
| 43 | Time threshold for under limit |
| 44 | Time threshold for over limit |
| 45 | Time threshold for missing magnitude |
| | |
| 46 | Contracted value |
| | |
| 51 | Minimum for recording interval 1 |
| 52 | Minimum for recording interval 2 |
| 53 | Maximum for recording interval 1 |
| 54 | Maximum for recording interval 2 |
| | |
| 55 | Test average |
| 56 | Current average 4 for harmonics measurement |
| | |
| 58 | Time integral 4 |
| | |
| 128...254 | Manufacturer specific codes |
| All other | Reserved |
| NOTES | |
| Averaging scheme 1 | Controlled by measurement period 1 (see Table 68), a set of registers is calculated by a metering device (codes 1...6). The typical usage is for billing purposes. |
| Averaging scheme 2 | Controlled by measurement period 2, a set of registers is calculated by a metering device (codes 11...16). The typical usage is for billing purposes. |
| Averaging scheme 3 | Controlled by measurement period 3, a set of registers is calculated by a metering device (codes 21...26). The typical usage is for instantaneous values. |
| Averaging scheme 4 | Controlled by measurement period 4, a test average value (code 55) is calculated by the metering device. |
| Current average 1, 2, 3 | See the definition of the "Demand register" IC in 4.3.4. The value is calculated using measurement period 1, 2 and/or 3 respectively. |
| Last average 1,2,3 | See the definition of the "Demand register" IC in 4.3.4. The value is calculated using measurement period 1, 2 or 3 respectively. |
| Minimum | The smallest of last average values during a billing period, see Table 68. |
| Maximum | The largest of last average values during a billing period. |
| Cumulative min. | The cumulative sum of minimum values over all the past billing periods. |
| Cumulative max. | The cumulative sum of maximum values over all the past billing periods. |
| Current average 4 | For harmonics measurement |
| Current average 5 | See the definition of the "Demand register" IC in 4.3.4. The value is calculated using recording interval 1; see Table 68. |
| Current average 6 | See the definition of the "Demand register" IC in 4.3.4. The value is calculated using recording interval 2. |

| Value group D codes – Electricity (A = 1, C <> 0, 93, 94, 96, 97, 98, 99) | |
|--|---|
| Time integral 1 | For a current billing period (F= 255): Time integral of the quantity calculated from the origin (first start of measurement) to the instantaneous time point. For a historical billing period (F= 0...99): Time integral of the quantity calculated from the origin to the end of the billing period given by the billing period code. |
| Time integral 2 | For a current billing period (F = 255): Time integral of the quantity calculated from the beginning of the current billing period to the instantaneous time point. For a historical billing period (F = 0...99): Time integral of the quantity calculated over the billing period given by the billing period code. |
| Time integral 3 | Time integral of the positive difference between the quantity and a prescribed threshold value. |
| Time integral 4 ("Test time integral") | Time integral of the quantity calculated over a time specific to the device or determined by test equipment. |
| Time integral 5 | Used as a base for load profile recording: Time integral of the quantity calculated from the beginning of the current recording interval to the instantaneous time point for recording period 1, see Table 68. |
| Time integral 6 | Used as a base for load profile recording: Time integral of the quantity calculated from the beginning of the current recording interval to the instantaneous time point for recording period 2, see Table 68. |
| Time integral 7 | Time integral of the quantity calculated from the origin (first start of measurement) up to the end of the last recording period with recording period 1, see Table 68. |
| Time integral 8 | Time integral of the quantity calculated from the origin (first start of measurement) up to the end of the last recording period with recording period 2, see Table 68. |
| Time integral 9 | Time integral of the quantity calculated from the beginning of the current billing period up to the end of the last recording period with recording period 1, see Table 68. |
| Time integral 10 | Time integral of the quantity calculated from the beginning of the current billing period up to the end of the last recording period with recording period 2, see Table 68. |
| Under limit values | Values under a certain threshold (for example dips). |
| Over limit values | Values above a certain threshold (for example swells). |
| Missing values | Values considered as missing (for example interruptions). |

7.5.2.2 Use of value group D for identification of other objects

For identifiers of electricity related general purpose objects see 7.5.5.1.

7.5.3 Value group E codes – Electricity

7.5.3.1 General

The following clauses define the use of value group E for identifying further classification or processing the measurement quantities defined by values in value groups A to D. The various classifications and processing methods are exclusive.

7.5.3.2 Tariff rates

Table 63 shows the use of value group E for identification of tariff rates typically used for energy (consumption) and demand quantities.

Table 63 – Value group E codes – Electricity – Tariff rates

| Value group E codes – Electricity – Tariff rates (A = 1) | |
|--|-----------------------------|
| 0 | Total |
| 1 | Rate 1 |
| 2 | Rate 2 |
| 3 | Rate 3 |
| ... | ... |
| 63 | Rate 63 |
| | |
| 128...254 | Manufacturer specific codes |
| All other | Reserved |

7.5.3.3 Harmonics

Table 64 shows the use of value group E for the identification of harmonics of instantaneous values of voltage, current or active power.

Table 64 – Value group E codes – Electricity – Harmonics

| Value group E codes – Electricity – Measurement of harmonics of voltage, current or active power (A = 1, C = 12, 32, 52, 72, 92, 11, 31, 51, 71, 90, 91, 15, 35, 55, 75, D = 7, 24) | |
|---|--|
| 0 | Total (fundamental + all harmonics) |
| 1 | 1 st harmonic (fundamental) |
| 2 | 2 nd harmonic |
| ... | <i>n</i> th harmonic |
| 120 | 120 th harmonic |
| | |
| 124 | Total Harmonic Distortion (THD) ^a |
| 125 | Total Demand Distortion (TDD) ^b |
| 126 | All harmonics ^c |
| 127 | All harmonics to nominal value ratio ^d |
| | |
| 128...254 | Manufacturer specific codes |
| All other | Reserved |
| ^a | THD is calculated as the ratio of the square root of the sum of the squares of each harmonic to the value of the fundamental quantity, expressed as a percent of the value of the fundamental. |
| ^b | TDD is calculated as the ratio of the square root of the sum of the squares of each harmonic to the maximum value of the fundamental quantity, expressed as percent of the maximum value of the fundamental. |
| ^c | Calculated as the square root of the sum of the squares of each harmonic. |
| ^d | This is calculated as ratio of the square root of the sum of the squares of each harmonic, to the nominal value of the fundamental quantity, expressed as percent of the nominal value of the fundamental. |

7.5.3.4 Phase angles

Table 65 shows the use of value group E for identification of phase angles.

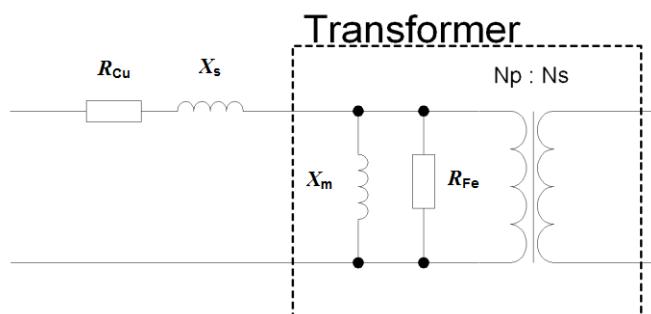
Table 65 – Value group E codes – Electricity – Extended phase angle measurement

| Value group E codes – Electricity – Extended phase angle measurement (A = 1, C = 81; D = 7) | | | | | | | | <= From |
|---|-------|-------|-------|-------|-------|-------|-------|---------|
| Angle | U(L1) | U(L2) | U(L3) | I(L1) | I(L2) | I(L3) | I(L0) | <= From |
| U(L1) | (00) | 01 | 02 | 04 | 05 | 06 | 07 | |
| U(L2) | 10 | (11) | 12 | 14 | 15 | 16 | 17 | |
| U(L3) | 20 | 21 | (22) | 24 | 25 | 26 | 27 | |
| I(L1) | 40 | 41 | 42 | (44) | 45 | 46 | 47 | |
| I(L2) | 50 | 51 | 52 | 54 | (55) | 56 | 57 | |
| I(L3) | 60 | 61 | 62 | 64 | 65 | (66) | 67 | |
| I(L0) | 70 | 71 | 72 | 74 | 75 | 76 | (77) | |
| ^ To (reference) | | | | | | | | |

7.5.3.5 Transformer and line loss quantities

Table 66 shows the meaning of value group E for the identification of transformer and line loss quantities. The use of value group D shall be according to Table 62, the use of value group F shall be according to Table 108. For these quantities, no tariffication is available.

The model of the line and the transformer used for loss calculation is shown on Figure 31.



Legend:

- R_{Cu} Line resistance losses, OBIS code 1.x.0.10.2.VZ
- X_s Line reactance losses, OBIS code 1.x.0.10.3.VZ
- X_m Transformer magnetic losses, OBIS code 1.x.0.10.0.VZ
- R_{Fe} Transformer iron losses, OBIS code 1.x.0.10.1.VZ
- N_p Number of turns on the primary side of the transformer
- N_s Number of turns on the secondary side of the transformer

NOTE Serial elements of the transformer are normally low compared to that of the line, therefore they are not considered here.

Figure 31 – Model of the line and the transformer for calculation of loss quantities

Table 66 – Value group E codes – Electricity – Transformer and line losses

| Value group E codes – Electricity – Transformer and line losses (A = 1, C = 83) | | | |
|---|---|---|---|
| E= | Quantity | Formula | Quadrant / comment |
| 1 | $\sum L_i$ Active line losses+ | On Load Active, positive $OLA+ = (CuA_1+) + (CuA_2+) + (CuA_3+)$ | QI+QIV |
| 2 | $\sum L_i$ Active line losses- | On Load Active, negative $OLA- = (CuA_1-) + (CuA_2-) + (CuA_3-)$ | QII+QIII |
| 3 | $\sum L_i$ Active line losses | On Load Active $OLA = (CuA_1) + (CuA_2) + (CuA_3)$ | QI+QII+QIII+QIV |
| 4 | $\sum L_i$ Active transformer losses+ | No Load Active, positive $NLA+ = (FeA_1+) + (FeA_2+) + (FeA_3+)$ | QI+QIV |
| 5 | $\sum L_i$ Active transformer losses- | No Load active, negative $NLA- = (FeA_1-) + (FeA_2-) + (FeA_3-)$ | QII+QIII |
| 6 | $\sum L_i$ Active transformer losses | No Load Active $NLA = (FeA_1) + (FeA_2) + (FeA_3)$ | QI+QII+QIII+QIV |
| 7 | $\sum L_i$ Active losses+ | Total Losses Active, positive $TLA+ = (OLA+) + (NLA+)$ | QI+QIV |
| 8 | $\sum L_i$ Active losses- | Total Losses Active, negative $TLA- = (OLA-) + (NLA-)$ | QII+QIII |
| 9 | $\sum L_i$ Active losses | Total Losses Active $TLA = OLA + NLA = TLA_1 + TLA_2 + TLA_3$ | QI+QII+QIII+QIV |
| 10 | $\sum L_i$ Reactive line losses+ | On Load Reactive, positive $OLR+ = (CuR_1+) + (CuR_2+) + (CuR_3+)$ | QI+QII |
| 11 | $\sum L_i$ Reactive line losses- | On Load Reactive, negative $OLR- = (CuR_1-) + (CuR_2-) + (CuR_3-)$ | QIII+QIV |
| 12 | $\sum L_i$ Reactive line losses | On Load Reactive $OLR = (CuR_1) + (CuR_2) + (CuR_3)$ | QI+QII+QIII+QIV |
| 13 | $\sum L_i$ Reactive transformer losses+ | No Load reactive, positive $NLR+ = (FeR_1+) + (FeR_2+) + (FeR_3+)$ | QI+QII |
| 14 | $\sum L_i$ Reactive transformer losses- | No Load Reactive, negative $NLR- = (FeR_1-) + (FeR_2-) + (FeR_3-)$ | QIII+QIV |
| 15 | $\sum L_i$ Reactive transformer losses | No Load Reactive $NLR = (FeR_1) + (FeR_2) + (FeR_3)$ | QI+QII+QIII+QIV |
| 16 | $\sum L_i$ Reactive losses+ | Total Losses Reactive, positive $TLR+ = (OLR+) + (NLR+)$ | QI+QII |
| 17 | $\sum L_i$ Reactive losses- | Total Losses Reactive, negative $TLR- = (OLR-) + (NLR-)$ | QIII+QIV |
| 18 | $\sum L_i$ Reactive losses | Total Losses Reactive $TLR = OLR + NLR = TLR_1 + TLR_2 + TLR_3$ | QI+QII+QIII+QIV |
| 19 | Total transformer losses with normalized $R_{Fe} = 1 \text{ M}\Omega$ | $U^2 h$ $1/R_{Fe} \times (U^2 h_{L1} + U^2 h_{L2} + U^2 h_{L3})$ | QI+QII+QIII+QIV |
| 20 | Total line losses with normalized $R_{Cu} = 1 \Omega$ | $I^2 h$ $R_{Cu} \times (I^2 h_{L1} + I^2 h_{L2} + I^2 h_{L3})$ | QI+QII+QIII+QIV |
| 21 | Compensated active gross+ | $CA+ = (A+) + (TLA+)$ | QI+QIV; A+ is the quantity A = 1, C = 1 |
| 22 | Compensated active net+ | $CA+ = (A+) - (TLA+)$ | QI+QIV |
| 23 | Compensated active gross- | $CA- = (A-) + (TLA-)$ | QII+QIII, A- is the quantity A = 1, C = 2 |
| 24 | Compensated active net- | $CA- = (A-) - (TLA-)$ | QII+QIII |
| 25 | Compensated reactive gross+ | $CR+ = (R+) + (TLR+)$ | QI+QII; R+ is the quantity A = 1, C = 3 |

| Value group E codes – Electricity – Transformer and line losses (A = 1, C = 83) | | | |
|---|------------------------------------|---------------------------------------|---|
| E= | Quantity | Formula | Quadrant / comment |
| 26 | Compensated reactive net+ | $CR+ = (R+) - (TLR+)$ | QI+QII |
| 27 | Compensated reactive gross- | $CR- = (R-) + (TLR-)$ | QIII+QIV; R- is the quantity A = 1, C = 4 |
| 28 | Compensated reactive net- | $CR- = (R-) - (TLR-)$ | QIII+QIV |
| 29 | Reserved | | |
| 30 | Reserved | | |
| 31 | L_1 Active line losses+ | $CuA_{1+} = I^2 h_{L1} \times R_{Cu}$ | QI+QIV R_{Cu} is the serial resistive element of the line loss, OBIS code 1.x.0.10.2.VZ |
| 32 | L_1 Active line losses- | $CuA_{1-} = I^2 h_{L1} \times R_{Cu}$ | QII+QIII |
| 33 | L_1 Active line losses | $CuA_1 = I^2 h_{L1} \times R_{Cu}$ | QI+QII+QIII+QIV |
| 34 | L_1 Active transformer losses+ | $FeA_{1+} = U^2 h_{L1} / R_{Fe}$ | QI+QIV R_{Fe} is the parallel resistive element of the transformer loss, OBIS code 1.x.0.10.1.VZ |
| 35 | L_1 Active transformer losses- | $FeA_{1-} = U^2 h_{L1} / R_{Fe}$ | QII+QIII |
| 36 | L_1 Active transformer losses | $FeA_1 = U^2 h_{L1} / R_{Fe}$ | QI+QII+QIII+QIV |
| 37 | L_1 Active losses+ | $TLA_{1+} = (CuA_{1+}) + (FeA_{1+})$ | QI+QIV |
| 38 | L_1 Active losses- | $TLA_{1-} = (CuA_{1-}) + (FeA_{1-})$ | QII+QIII |
| 39 | L_1 Active losses | $TLA_1 = CuA_1 + FeA_1$ | QI+QII+QIII+QIV |
| 40 | L_1 Reactive line losses+ | $CuR_{1+} = I^2 h_{L1} \times X_s$ | QI+QII X_s is the serial reactive element of the line loss, OBIS code 1.x.0.10.3.VZ |
| 41 | L_1 Reactive line losses- | $CuR_{1-} = I^2 h_{L1} \times X_s$ | QIII+QIV |
| 42 | L_1 Reactive line losses | $CuR_1 = I^2 h_{L1} \times X_s$ | QI+QII+QIII+QIV |
| 43 | L_1 Reactive transformer losses+ | $FeR_{1+} = U^2 h_{L1} / X_m$ | QI+QII X_m is the parallel reactive element of the transformer loss, OBIS code 1.x.0.10.0.VZ |
| 44 | L_1 Reactive transformer losses- | $FeR_{1-} = U^2 h_{L1} / X_m$ | QIII+QIV |
| 45 | L_1 Reactive transformer losses | $FeR_1 = U^2 h_{L1} / X_m$ | QI+QII+QIII+QIV |
| 46 | L_1 Reactive losses+ | $TLR_{1+} = (CuR_{1+}) + (FeR_{1+})$ | QI+QII |
| 47 | L_1 Reactive losses- | $TLR_{1-} = (CuR_{1-}) + (FeR_{1-})$ | QIII+QIV |
| 48 | L_1 Reactive losses | $TLR_1 = CuR_1 + FeR_1$ | QI+QII+QIII+QIV |
| 49 | L_1 Ampere-squared hours | $A^2 h_{L1}$ | QI+QII+QIII+QIV |
| 50 | L_1 Volt-squared hours | $V^2 h_{L1}$ | QI+QII+QIII+QIV |
| | | | |
| 51 | L_2 Active line losses+ | $CuA_{2+} = I^2 h_{L2} \times R_{Cu}$ | QI+QIV R_{Cu} is the serial resistive element of the line loss, OBIS code 1.x.0.10.2.VZ |
| 52 | L_2 Active line losses- | $CuA_{2-} = I^2 h_{L2} \times R_{Cu}$ | QII+QIII |
| 53...70 | L_2 quantities, (See 33...48) | | |
| | | | |

| Value group E codes – Electricity – Transformer and line losses (A = 1, C = 83) | | | |
|---|--------------------------------|---------------------------------------|--|
| E= | Quantity | Formula | Quadrant / comment |
| 71 | L_3 Active line losses + | $CuA_{3+} = I^2 h_{L3} \times R_{Cu}$ | QI+QIV R_{Cu} is the serial resistive element of the line loss, OBIS code 1.x.0.10.2.VZ |
| 72 | L_3 Active line losses - | $CuA_{3-} = I^2 h_{L3} \times R_{Cu}$ | QII+QIII |
| 73...90 | L_3 quantities (See 33...48) | | |
| | | | |
| 91...255 | Reserved | | |

NOTE In this table, no manufacturer specific range is available.

7.5.3.6 UNIPEDE voltage dips

Table 67 shows the use of value group E for the identification of voltage dips according to the UNIPEDE classification.

Table 67 – Value group E codes – Electricity – UNIPEDE voltage dips

| Value group E codes – Electricity – UNIPEDE voltage dips measurement (A = 1, C = 12, 32, 52, 72, D = 32) | | | | | | | |
|--|------------------------------------|-------------------------|------------------------|----------------------|--------------------|---------------------|----------------------|
| Depth in % of U_n | Residual voltage U in % of U_n | Duration Δt s | | | | | |
| | | 0,01 < Δt ≤ 0,1 | 0,1 < Δt ≤ 0,5 | 0,5 < Δt ≤ 1 | 1 < Δt ≤ 3 | 3 < Δt ≤ 20 | 20 < Δt ≤ 60 |
| 10%...<15% | 90 > $U \geq 85$ | 00 | 01 | 02 | 03 | 04 | 05 |
| 15%...<30% | 85 > $U \geq 70$ | 10 | 11 | 12 | 13 | 14 | 15 |
| 30%...<60% | 70 > $U \geq 40$ | 20 | 21 | 22 | 23 | 24 | 25 |
| 60%...<90% | 40 > $U \geq 10$ | 30 | 31 | 32 | 33 | 34 | 35 |
| 90%...<100% | 10 > $U \geq 0$ | 40 | 41 | 42 | 43 | 44 | 45 |

NOTE These *dip classes* form a subset of the classes defined in IEC/TR 61000-2-8, Table 2.

7.5.3.7 Use of value group E for the identification of other objects

For identifiers of electricity related general purpose objects see 7.5.5.1.

7.5.4 Value group F codes – Electricity

7.5.4.1 Billing periods

Value group F specifies the allocation to different billing periods (sets of historical values) for the objects with following codes:

- value group A: 1;
- value group C: as defined in Table 61;
- value group D:
 - 0: Billing period average (since last reset);
 - 1, 2, 3, 6: (Cumulative) minimum / maximum 1;
 - 8, 9, 10: Time integral 1 / 2 / 3;

- 11, 12, 13, 16: (Cumulative) minimum / maximum 2;
- 21, 22, 23, 26: (Cumulative) minimum / maximum 3;

There are two billing period schemes available (for example to store weekly and monthly values). For each billing period scheme, the following general purpose objects are available:

- billing period counter;
- number of available billing periods;
- time stamp of most recent and historical billing periods;
- billing period length.

For OBIS codes see Table 68. For additional information, see Clauses 6.2.2 and 7.11.3.

7.5.4.2 Multiple thresholds

Value group F is also used to identify several thresholds for the same quantity, identified with the following codes:

- value group A = 1;
- value group C = 1...20, 21...40, 41...60, 61...80, 82, 84...89, 90...92;
- value group D = 31, 35, 39 (under limit, over limit and missing thresholds);
- value group F = 0...99.

NOTE All quantities monitored are instantaneous values: D = 7 or D = 24.

When multiple thresholds are identified by value group F, then the Under limit / Over limit / Missing Occurrence counter / Duration / Magnitude quantities relative to a threshold are identified with the same value in value group F. In this case, value group F cannot be used to identify values relative to billing period. However, such values can be held by "Profile generic" objects.

Example:

- Over limit threshold #1 for current in any phase is identified with OBIS code 1-0:11.35.0*0;
- Over limit duration above threshold # 1 for current in any phase is identified with OBIS code 1-0:11.37.0*0.

To avoid ambiguity, value group F cannot be used to identify historical values of Under limit / Over limit / Missing Occurrence counter / Duration / Magnitude quantities. For historical values of these quantities "Profile generic" objects can be used and values related to previous billing periods can be accessed using selective access.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 412/492 |
|-----------------------|------------|-------------------------|---------|

7.5.5 OBIS codes – Electricity

7.5.5.1 General and service entry objects – Electricity

Table 68 specifies the OBIS codes for electricity related general and service entry objects.

Table 68 – OBIS codes for general and service entry objects – Electricity

| General and service entry objects – Electricity | OBIS code | | | | | |
|--|-----------|-----|-----|-----|-----|------------------|
| | A | B | C | D | E | F |
| Free ID-numbers for utilities | | | | | | |
| Complete combined electricity ID | 1 | b | 0 | 0 | | |
| Electricity ID 1 | 1 | b | 0 | 0 | 0 | |
| ... | ... | ... | ... | ... | ... | |
| Electricity ID 10 | 1 | b | 0 | 0 | 9 | |
| Billing period values/reset counter entries (First billing period scheme if there are more than one) | | | | | | |
| Billing period counter (1) | 1 | b | 0 | 1 | 0 | VZ or 255 |
| Number of available billing periods (1) | 1 | b | 0 | 1 | 1 | |
| Time stamp of the most recent billing period (1) | 1 | b | 0 | 1 | 2 | |
| Time stamp of the billing period (1) VZ (last reset) | 1 | b | 0 | 1 | 2 | VZ |
| Time stamp of the billing period (1) VZ ₋₁ | 1 | b | 0 | 1 | 2 | VZ ₋₁ |
| ... | ... | ... | ... | ... | ... | ... |
| Time stamp of the billing period (1) VZ _{-n} | 1 | b | 0 | 1 | 2 | VZ _{-n} |
| Billing period values/reset counter entries (Second billing period scheme) | | | | | | |
| Billing period counter (2) | 1 | b | 0 | 1 | 3 | VZ or 255 |
| Number of available billing periods (2) | 1 | b | 0 | 1 | 4 | |
| Time stamp of the most recent billing period (2) | 1 | b | 0 | 1 | 5 | |
| Time stamp of the billing period (2) VZ (last reset) | 1 | b | 0 | 1 | 5 | VZ |
| Time stamp of the billing period (2) VZ ₋₁ | 1 | b | 0 | 1 | 5 | VZ ₋₁ |
| ... | ... | ... | ... | ... | ... | ... |
| Time stamp of the billing period (2) VZ _{-n} | 1 | b | 0 | 1 | 5 | VZ _{-n} |
| Program entries | | | | | | |
| Active firmware identifier (Previously: Configuration program version number) | 1 | b | 0 | 2 | 0 | |
| Parameter record number | 1 | b | 0 | 2 | 1 | |
| Parameter record number, line 1 | 1 | b | 0 | 2 | 1 | 1 |
| Reserved for future use | 1 | b | 0 | 2 | 1 | 2... 127 |
| Manufacturer specific | 1 | b | 0 | 2 | 1 | 128... 254 |
| Time switch program number | 1 | b | 0 | 2 | 2 | |
| RCR program number | 1 | b | 0 | 2 | 3 | |
| Meter connection diagram ID | 1 | b | 0 | 2 | 4 | |
| Passive calendar name | 1 | b | 0 | 2 | 7 | |
| Active firmware signature | 1 | b | 0 | 2 | 8 | |
| Output pulse values or constants | | | | | | |
| NOTE For units, see 4.3.2. | | | | | | |
| Active energy, metrological LED | 1 | b | 0 | 3 | 0 | |
| Reactive energy, metrological LED | 1 | b | 0 | 3 | 1 | |

| General and service entry objects – Electricity | OBIS code | | | | | |
|--|-----------|---|---|---|----|----|
| | A | B | C | D | E | F |
| Apparent energy, metrological LED | 1 | b | 0 | 3 | 2 | |
| Active energy, output pulse | 1 | b | 0 | 3 | 3 | |
| Reactive energy, output pulse | 1 | b | 0 | 3 | 4 | |
| Apparent energy, output pulse | 1 | b | 0 | 3 | 5 | |
| Volt-squared hours, metrological LED | 1 | b | 0 | 3 | 6 | |
| Ampere-squared hours, metrological LED | 1 | b | 0 | 3 | 7 | |
| Volt-squared hours, output pulse | 1 | b | 0 | 3 | 8 | |
| Ampere-squared hours, output pulse | 1 | b | 0 | 3 | 9 | |
| Ratios | | | | | | |
| Reading factor for power | 1 | b | 0 | 4 | 0 | |
| Reading factor for energy | 1 | b | 0 | 4 | 1 | |
| Transformer ratio – current (numerator) ^a | 1 | b | 0 | 4 | 2 | VZ |
| Transformer ratio – voltage (numerator) ^a | 1 | b | 0 | 4 | 3 | VZ |
| Overall transformer ratio (numerator) ^a | 1 | b | 0 | 4 | 4 | VZ |
| Transformer ratio – current (denominator) ^a | 1 | b | 0 | 4 | 5 | VZ |
| Transformer ratio – voltage (denominator) ^a | 1 | b | 0 | 4 | 6 | VZ |
| Overall transformer ratio (denominator) ^a | 1 | b | 0 | 4 | 7 | VZ |
| Demand limits for excess consumption metering | | | | | | |
| Reserved for Germany | 1 | b | 0 | 5 | | |
| Nominal values | | | | | | |
| Voltage | 1 | b | 0 | 6 | 0 | |
| Basic/nominal current | 1 | b | 0 | 6 | 1 | |
| Frequency | 1 | b | 0 | 6 | 2 | |
| Maximum current | 1 | b | 0 | 6 | 3 | |
| Reference voltage for power quality measurement | 1 | b | 0 | 6 | 4 | |
| Reference voltage for aux. power supply | 1 | b | 0 | 6 | 5 | |
| Input pulse values or constants^b | | | | | | |
| NOTE For units, see 4.3.2. | | | | | | |
| Active energy | 1 | b | 0 | 7 | 0 | |
| Reactive energy | 1 | b | 0 | 7 | 1 | |
| Apparent energy | 1 | b | 0 | 7 | 2 | |
| Volt-squared hours | 1 | b | 0 | 7 | 3 | |
| Ampere-squared hours | 1 | b | 0 | 7 | 4 | |
| Unitless quantities | 1 | b | 0 | 7 | 5 | |
| Active energy, export | 1 | b | 0 | 7 | 10 | |
| Reactive energy, export | 1 | b | 0 | 7 | 11 | |
| Apparent energy, export | 1 | b | 0 | 7 | 12 | |
| Measurement period- / recording interval- / billing period duration | | | | | | |
| Measurement period 1, for averaging scheme 1 | 1 | b | 0 | 8 | 0 | VZ |
| Measurement period 2, for averaging scheme 2 | 1 | b | 0 | 8 | 1 | VZ |
| Measurement period 3, for instantaneous value | 1 | b | 0 | 8 | 2 | VZ |
| Measurement period 4, for test value | 1 | b | 0 | 8 | 3 | VZ |
| Recording interval 1, for load profile | 1 | b | 0 | 8 | 4 | VZ |
| Recording interval 2, for load profile | 1 | b | 0 | 8 | 5 | VZ |
| Billing period (Billing period 1 if there are two billing period schemes) | 1 | b | 0 | 8 | 6 | VZ |
| Billing period 2 | 1 | b | 0 | 8 | 7 | VZ |
| Measurement period 4, for harmonics measurement | 1 | b | 0 | 8 | 8 | VZ |

| General and service entry objects – Electricity | OBIS code | | | | | |
|---|------------------|----------|----------|----------|----------|----------|
| | A | B | C | D | E | F |
| Time entries | | | | | | |
| Time expired since last end of billing period (First billing period scheme if there are more than one) | 1 | b | 0 | 9 | 0 | |
| Local time | 1 | b | 0 | 9 | 1 | |
| Local date | 1 | b | 0 | 9 | 2 | |
| Reserved for Germany | 1 | b | 0 | 9 | 3 | |
| Reserved for Germany | 1 | b | 0 | 9 | 4 | |
| Week day (0...7) | 1 | b | 0 | 9 | 5 | |
| Time of last reset (First billing period scheme if there are more than one) | 1 | b | 0 | 9 | 6 | |
| Date of last reset (First billing period scheme if there are more than one) | 1 | b | 0 | 9 | 7 | |
| Output pulse duration | 1 | b | 0 | 9 | 8 | |
| Clock synchronization window | 1 | b | 0 | 9 | 9 | |
| Clock synchronization method | 1 | b | 0 | 9 | 10 | |
| Clock time shift limit (default value: s) | 1 | b | 0 | 9 | 11 | |
| Billing period reset lockout time (First billing period scheme if there are more than one) | 1 | b | 0 | 9 | 12 | |
| Second billing period scheme | | | | | | |
| Time expired since last end of billing period | 1 | b | 0 | 9 | 13 | |
| Time of last reset | 1 | b | 0 | 9 | 14 | |
| Date of last reset | 1 | b | 0 | 9 | 15 | |
| Billing period reset lockout time | 1 | b | 0 | 9 | 16 | |
| Coefficients | | | | | | |
| Transformer magnetic losses, X_m | 1 | b | 0 | 10 | 0 | VZ |
| Transformer iron losses, R_{Fe} | 1 | b | 0 | 10 | 1 | VZ |
| Line resistance losses, R_{Cu} | 1 | b | 0 | 10 | 2 | VZ |
| Line reactance losses, X_s | 1 | b | 0 | 10 | 3 | VZ |
| Measurement methods | | | | | | |
| Algorithm for active power measurement | 1 | b | 0 | 11 | 1 | |
| Algorithm for active energy measurement | 1 | b | 0 | 11 | 2 | |
| Algorithm for reactive power measurement | 1 | b | 0 | 11 | 3 | |
| Algorithm for reactive energy measurement | 1 | b | 0 | 11 | 4 | |
| Algorithm for apparent power measurement | 1 | b | 0 | 11 | 5 | |
| Algorithm for apparent energy measurement | 1 | b | 0 | 11 | 6 | |
| Algorithm for power factor calculation | 1 | b | 0 | 11 | 7 | |
| Metering point ID (electricity related) | | | | | | |
| Metering point ID 1 (electricity related) | 1 | 0 | 96 | 1 | 0 | |
| | | | | | | |
| Metering point ID 10 (electricity related) | 1 | 0 | 96 | 1 | 9 | |
| Internal operating status, electricity related | | | | | | |
| Internal operating status, global ^c | 1 | b | 96 | 5 | 0 | |
| Internal operating status (status word 1) | 1 | b | 96 | 5 | 1 | |
| Internal operating status (status word 2) | 1 | b | 96 | 5 | 2 | |
| Internal operating status (status word 3) | 1 | b | 96 | 5 | 3 | |
| Internal operating status (status word 4) | 1 | b | 96 | 5 | 4 | |

| General and service entry objects – Electricity | OBIS code | | | | | |
|---|---|-----|-----|-----|-----|-----|
| | A | B | C | D | E | F |
| Meter started status flag | 1 | b | 96 | 5 | 5 | |
| Electricity related status data | | | | | | |
| Status information missing voltage | 1 | 0 | 96 | 10 | 0 | |
| Status information missing current | 1 | 0 | 96 | 10 | 1 | |
| Status information current without voltage | 1 | 0 | 96 | 10 | 2 | |
| Status information auxiliary power supply | 1 | 0 | 96 | 10 | 3 | |
| Manufacturer specific ^d | 1 | b | 96 | 50 | e | f |
| | ... | ... | ... | ... | ... | ... |
| Manufacturer specific | 1 | b | 96 | 99 | e | f |
| ^a | If a transformer ratio is expressed as a fraction the ratio is numerator, divided by denominator. If the transformer ratio is expressed by an integer or real figure, only the numerator is used. | | | | | |
| ^b | The codes for export active, reactive and apparent energy shall be used only if meters measuring import energy and meters measuring export energy are connected to the pulse inputs. | | | | | |
| ^c | Global status words with E = 0 contain the individual status words E = 1...5. The contents of the status words are not defined in this Technical Report. | | | | | |
| ^d | The range D = 50...99 is available for identifying objects, which are not represented by another defined code, but need representation on the display as well. If this is not required, the range D = 128...254 should be used. | | | | | |

It should be noted, that some of the codes above are normally used for display purposes only, as the related data items are attributes of objects having their own OBIS name. See Clause 4.

7.5.5.2 Error register objects – Electricity

Table 69 specifies the OBIS codes for electricity related error register objects.

Table 69 – OBIS codes for error register objects – Electricity

| Error register objects – Electricity | OBIS code | | | | | |
|---|-----------|---|----|----|---|---|
| | A | B | C | D | E | F |
| Error register | 1 | b | 97 | 97 | e | |
| NOTE The information to be included in the error objects is not defined in this document. | | | | | | |

7.5.5.3 List objects – Electricity

Table 70 specifies the OBIS codes for electricity related list objects.

Table 70 – OBIS codes for list objects – Electricity

| List objects – Electricity | OBIS code | | | | | |
|--|-----------|---|----|---|---|------------------|
| | A | B | C | D | E | F |
| Electricity related data of billing period (with billing period scheme 1 if there are two schemes available) | 1 | b | 98 | 1 | e | 255 ^a |
| Electricity related data of billing period (with billing period scheme 2) | 1 | b | 98 | 2 | e | 255 ^a |
| ^a F = 255 means a wildcard here. See 7.11.3. | | | | | | |

7.5.5.4 Data profile objects – Electricity

Electricity related data profiles – identified with one single OBIS code – are used to hold a series of measurement values of one or more similar quantities and/or to group various data. The OBIS codes are specified in Table 71.

Table 71 – OBIS codes for data profile objects – Electricity

| Data profile objects – Electricity | OBIS code | | | | | |
|--------------------------------------|-----------|---|----|----|-----------------|---|
| | A | B | C | D | E | F |
| Load profile with recording period 1 | 1 | b | 99 | 1 | e | |
| Load profile with recording period 2 | 1 | b | 99 | 2 | e | |
| Load profile during test | 1 | b | 99 | 3 | 0 | |
| Dips voltage profile | 1 | b | 99 | 10 | 1 | |
| Swells voltage profile | 1 | b | 99 | 10 | 2 | |
| Cuts voltage profile | 1 | b | 99 | 10 | 3 | |
| Voltage harmonic profile | 1 | b | 99 | 11 | n th | |
| Current harmonic profile | 1 | b | 99 | 12 | n th | |
| Voltage unbalance profile | 1 | b | 99 | 13 | 0 | |
| Power failure event log | 1 | b | 99 | 97 | e | |
| Event log | 1 | b | 99 | 98 | e | |
| Certification data log | 1 | b | 99 | 99 | e | |

7.5.5.5 Register table objects – Electricity

Register tables - identified with a single OBIS code - are defined to hold a number of values of the same type. The OBIS codes are specified in Table 72.

Table 72 – OBIS codes for Register table objects – Electricity

| Register table objects – Electricity | OBIS code | | | | | |
|--------------------------------------|-----------|---|----|----|---|---|
| | A | B | C | D | E | F |
| UNIPEDE voltage dips, any phase | 1 | b | 12 | 32 | | |
| UNIPEDE voltage dips, L ₁ | 1 | b | 32 | 32 | | |
| UNIPEDE voltage dips, L ₂ | 1 | b | 52 | 32 | | |
| UNIPEDE voltage dips, L ₃ | 1 | b | 72 | 32 | | |
| Extended angle measurement | 1 | b | 81 | 7 | | |
| General use, electricity related | 1 | b | 98 | 10 | e | |

7.6 Heat Cost Allocators (Value group A = 4)

7.6.1 General

NOTE The following introductory text is from EN 13757-1:20 Clause 12.6.

Heat Cost Allocators (HCAs) are mounted on radiators in the area to be monitored. The HCA should be mounted with in free air and radiators should not be enclosed. There will normally also be multiple HCAs, even for a single customer. This makes at, the present, direct connection to all HCAs using a two way connections an infeasible solution. It is nevertheless important, that data coming from a (number of) HCAs (via a concentrator) can be handled in the same way as data from other meters for remote reading.

The current subsection describes the naming of objects carrying HCA information in a COSEM environment. The words used in this clause are those used in EN 834:1994 the corresponding media standard.

The output from an HCA is "the temperature integral with respect to time", and it is only a relative sum. The main parameter from a HCA is this integral. Time series of this integral may be stored in the HCA for later readout. Other media related information available from a HCA are temperature and rating factors.

7.6.2 Value group C codes – HCA

The name of the different objects in the table for HCA objects corresponds to the name used in the relevant standard, EN 834:1994. The OBIS codes are specified in Table 73.

Table 73 – Value group C codes – HCA

| Value group C codes – HCA (A = 4) | |
|-----------------------------------|--|
| 0 | General purpose objects ^a |
| 1 | Unrated integral ^b |
| 2 | Rated integral ^c |
| 3 | Radiator surface temperature ^d |
| 4 | Heating medium temperature, t_m |
| 5 | Flow (forward) temperature, t_v |
| 6 | Return temperature, t_R |
| 7 | Room temperature, t_L |
| 93 | Consortia specific identifiers, see Table 54. |
| 94 | Country specific identifiers, see Table 55. |
| 96 | General and service entry objects– HCA (See 7.6.4.1). |
| 97 | Error register objects – HCA (See 7.6.4.2). |
| 98 | List objects – HCA |
| 99 | Data profile objects – HCA (See 7.6.4.3) |
| 128...199, 240 | Manufacturer specific codes |
| All other | Reserved |
| ^a | Settings like time constant, thresholds etc. See the table of object codes in EN 13757-1:20 Clause 13.3.1. |
| ^b | Readout prior to compensation as specified in EN 834:1994. |

| Value group C codes – HCA (A = 4) | |
|-----------------------------------|--|
| c | Readout after compensation as specified in EN 834:1994. |
| d | Temperature measured prior to any rating |
| NOT E 1 | The radiator surface (C = 3) temperature and the heating media (C=4) temperature are mutually exclusive. |
| NOT E 2 | The forward flow (C = 5) and reverse flow (C = 6) temperatures are exclusive to the radiator surface (C = 3) temperature. |
| NOT E 3 | The room temperature measurement (C = 7) is always be accompanied by either a radiator surface (C = 3) temperature, a heating media (C = 4) temperature or a pair of forward / return flow (C = 5 / C = 6) temperatures. |

7.6.3 Value group D codes – HCA

This value group specifies the result of processing a *Quantity* according to a specific algorithm for Heat Cost Allocator related values. The OBIS codes are specified in Table 74.

Table 74 – Value group D codes – HCA

| Value group D codes – HCA (A = 4, C <> 0, 96...99) | |
|--|--|
| 0 | Current value |
| 1 | Periodical value ^a |
| 2 | Set date value |
| 3 | Billing date value |
| 4 | Minimum of value |
| 5 | Maximum of value |
| 6 | Test value ^b |
| All other | Reserved |
| ^a | A set of values periodically stored (this may be once or twice a month) |
| ^b | A value specially processed for test purpose. This may be due to an increased precision of the data, or to a faster (but less precise) processing of data. |

7.6.4 OBIS codes – HCA

7.6.4.1 General and service entry objects – HCA

Table 75 specifies OBIS codes for heat cost allocator related general and service entry objects.

Table 75 – OBIS codes for general and service entry objects – HCA

| General and service entry objects – HCA | OBIS code | | | | | |
|---|-----------|---|-----|-----|-----|---|
| | A | B | C | D | E | F |
| Free ID-numbers for utilities | | | | | | |
| Complete combined ID | 4 | b | 0 | 0 | | |
| ID 1 | 4 | b | 0 | 0 | 0 | |
| ... | | | ... | ... | ... | |
| ID 10 | 4 | b | 0 | 0 | 9 | |
| Storage information | | | | | | |
| Status (VZ) of the historical value counter | 4 | b | 0 | 1 | 1 | |
| Number of available historical values | 4 | b | 0 | 1 | 2 | |
| Target date | 4 | b | 0 | 1 | 10 | |
| Billing date | 4 | b | 0 | 1 | 11 | |
| Configuration | | | | | | |
| Program version no. | 4 | b | 0 | 2 | 0 | |
| Firmware version no. | 4 | b | 0 | 2 | 1 | |
| Software version no. | 4 | b | 0 | 2 | 2 | |
| Device measuring principle ^a | 4 | b | 0 | 2 | 3 | |
| Conversion factors | | | | | | |
| Resulting rating factor, K | 4 | b | 0 | 4 | 0 | |
| Thermal output rating factor, K _Q | 4 | b | 0 | 4 | 1 | |
| Thermal coupling rating factor overall, K _C | 4 | b | 0 | 4 | 2 | |
| Thermal coupling rating factor room side, K _{CR} | 4 | b | 0 | 4 | 3 | |
| Thermal coupling rating factor heater side, K _{CH} | 4 | b | 0 | 4 | 4 | |
| Low temperature rating factor, K _T | 4 | b | 0 | 4 | 5 | |
| Display output scaling factor | 4 | b | 0 | 4 | 6 | |
| Threshold values | | | | | | |
| Start temperature threshold | 4 | b | 0 | 5 | 10 | |
| Difference temperature threshold | 4 | b | 0 | 5 | 11 | |
| Period information | | | | | | |
| Measuring period for average value | 4 | b | 0 | 8 | 0 | |
| Recording interval for consumption profile | 4 | b | 0 | 8 | 4 | |
| Billing period | 4 | b | 0 | 8 | 6 | |
| Time entries | | | | | | |
| Local time | 4 | b | 0 | 9 | 1 | |
| Local date | 4 | b | 0 | 9 | 2 | |

| General and service entry objects – HCA | | OBIS code | | | | | |
|--|---|-----------|---|----|----|---|---|
| Time stamp (local time) of the most recent billing period ^b | | 4 | b | 0 | 9 | 3 | |
| Manufacturer specific ^c | | 4 | b | 96 | 50 | e | f |
| | | 4 | b | 96 | 99 | e | f |
| ^a | This is an object of the type 'Data' enumerated, (0) single sensor, (1) single sensor + start sensor, (2) dual sensor, (3) triple sensor. | | | | | | |
| ^b | In case of billing period schemes absence or event triggered, commonly calculated from local date and local time information. | | | | | | |
| ^c | The range D = 50...99 is available for identifying objects, which are not represented by another defined code, but need representation on the display as well. If this is not required, the range D = 128...254 should be used. | | | | | | |

7.6.4.2 Error register objects – HCA

Table 76 specifies OBIS codes for HCA related error register objects.

Table 76 – OBIS codes for error register objects – HCA

| Error register objects – HCA | OBIS code | | | | | |
|------------------------------|-----------|---|----|----|---|---|
| | A | B | C | D | E | F |
| Error registers | 4 | b | 97 | 97 | e | |

7.6.4.3 Data profile objects – HCA

HCA related data profiles – identified with one single OBIS code – are used to hold a series of measurement values of one or more similar quantities and/or to group various data. The OBIS codes are specified in Table 77.

Table 77 – OBIS codes for data profile objects – HCA

| Data profile objects – HCA | OBIS code | | | | | |
|----------------------------|-----------|---|----|---|---|---|
| | A | B | C | D | E | F |
| Data profile objects | 4 | b | 99 | 1 | e | |

7.6.4.4 OBIS codes for HCA related objects (examples)

Table 78 specifies examples for OBIS codes of HCA related objects.

Table 78 – OBIS codes for HCA related objects (examples)

| HCA related objects | OBIS code | | | | | |
|---|-----------|---|---|---|---|------------------|
| | A | B | C | D | E | F |
| Consumption | | | | | | |
| Current unrated integral | 4 | b | 1 | 0 | 0 | |
| Current rated integral | 4 | b | 2 | 0 | 0 | |
| Rated integral, last set date | 4 | b | 2 | 2 | 0 | V _Z |
| Unrated integral, previous billing date | 4 | b | 1 | 3 | 0 | V _{Z-1} |
| Rated integral, two most recent periodical values | 4 | b | 2 | 1 | 0 | 102 |
| Monitoring values | | | | | | |
| Radiator temperature, current value | 4 | b | 3 | 0 | | |
| Flow temperature, test value | 4 | b | 5 | 6 | | |
| Room temperature, minimum value | 4 | b | 7 | 4 | | |

7.7 Thermal energy (Value group A = 5 or A = 6)

7.7.1 General

This section describes the naming of objects carrying Thermal energy meter information in a COSEM environment. It covers the handling of heat, as well as cooling. The media specific terms used in this clause are those used in EN 1434-1:2015, EN 1434-2:2015 and parts of the corresponding media standard. The output from a Thermal energy meter is "the integral of power, i.e. the enthalpy difference times the mass flow-rate, with respect to time".

Thermal energy meters can be used for measurement in heating (A=6) or cooling (A=5) systems.

Value group A = 5 has been set aside for metering of cooling specific objects and value group A = 6 for the metering of heat specific objects. The other value groups are identical for heating and cooling.

7.7.2 Value group C codes – Thermal energy

The name of the different objects in the table for heat metering and cooling metering objects corresponds to the name used in EN 1434-1:2015. The OBIS codes are specified in Table 79.

Table 79 – Value group C codes – Thermal energy

| Value group C codes – Thermal energy related objects (A = 5 or A = 6) | |
|---|--|
| 0 | General and service entry objects – Thermal energy (See 7.7.4.1) |
| 1 | Energy |
| 2 | Volume |
| 3 | Mass ^b |
| 4 | Inlet (Flow) volume ^a |
| 5 | Inlet (Flow) mass ^a |
| 6 | Outlet (Return) volume ^a |
| 7 | Outlet (Return) mass ^a |
| 8 | Power |
| 9 | Flow rate |
| 10 | Inlet (Flow) temperature ^a |
| 11 | Outlet (Return) temperature ^a |
| 12 | Temperature difference ^c |
| 13 | Pressure ^d |
| 93 | Consortia specific identifiers, see Table 54 |
| 94 | Country specific identifiers, see Table 55 |
| 96 | General and service entry objects – Thermal energy (See 7.7.4.1) |
| 97 | Error register objects – Thermal energy (See 7.7.4.2) |
| 98 | List objects – Thermal energy |
| 99 | Data profile objects – Thermal energy (See 7.7.4.3) |
| 128...199, 240 | Manufacturer specific codes |
| All other | Reserved |

| Value group C codes – Thermal energy related objects (A = 5 or A = 6) | |
|---|---|
| a | In a heating system the term "flow" is equivalent to "inlet" and the term "return" is equivalent to "outlet" |
| b | Used when metering steam. |
| c | Will often be available with a higher precision and accuracy than inlet (flow) and outlet (return) temperature. |
| d | Pressure of the media, if measured. Pressure can be retrieved as backup value from a general and service entry object (C=0), if incapable of measurement. |

7.7.3 Value group D codes – Thermal energy

This value group specifies the result of processing a *Quantity* according to a specific algorithm for heat or cooling related values. See Table 80.

Table 80 – Value group D codes – Thermal energy

| Value group D codes – Thermal energy (A = 5 or A = 6), (C <> 0, 96...99) | |
|--|--|
| 0 | Current value |
| 1 | Periodical value 1 ^a |
| 2 | Set date value |
| 3 | Billing date value |
| 4 | Minimum of value 1 |
| 5 | Maximum of value 1 |
| 6 | Test value ^b |
| 7 | Instantaneous value ^c |
| 8 | Time integral 1 ^d |
| 9 | Time integral 2 ^e |
| 10 | Current average ^f |
| 11 | Last average ^g |
| 12 | Periodical value 2 ^a |
| 13 | Periodical value 3 ^a |
| 14 | Minimum of value 2 |
| 15 | Maximum of value 2 |
| | |
| 20 | Under limit occurrence counter |
| 21 | Under limit duration |
| 22 | Over limit occurrence counter |
| 23 | Over limit duration |
| 24 | Missing data occurrence counter ^h |
| 25 | Missing data duration ^h |
| All other | Reserved |
| ^a | A set of data that is collected periodically. Recording of data in this way is directly supported by 'profiles'. |
| ^b | A value specially processed for test purpose. This may be due to an increased precision of the data, or to a faster (but less precise) processing of data. |
| ^c | An immediate readout from the system, typically with a shorter measuring time than the current value. |

| Value group D codes – Thermal energy (A = 5 or A = 6), (C > 0, 96...99) | |
|---|---|
| ^d | For a current billing period (F = 255): Time integral of the <i>quantity</i> calculated from the origin (first start of measurement) to the instantaneous time point. For a historical billing period (F = 0...99): Time integral of the <i>quantity</i> calculated from the origin to the end of the billing period given by the billing period code. |
| ^e | For a current billing period (F = 255): Time integral of the <i>quantity</i> calculated from the beginning of the current billing period to the instantaneous time point. For a historical billing period (F = 0...99): Time integral of the <i>quantity</i> calculated over the billing period given by the billing period code. |
| ^f | The value of a current demand register. |
| ^g | The value of a demand register at the end of the last measurement period. |
| ^h | Values considered as missing (for instance due to sensor failure). |

7.7.4 OBIS codes – Thermal energy

7.7.4.1 General and service entry objects – Thermal energy

Table 81 specifies OBIS codes for thermal energy related general and service entry objects.

Table 81 – OBIS codes for general and service entry objects – Thermal energy

| General and service entry objects – Thermal energy | OBIS code | | | | | |
|---|-----------|---|-----|-----|-----|----------------|
| | A | B | C | D | E | F |
| Free ID-numbers for utilities | | | | | | |
| Complete combined ID | 5/6 | b | 0 | 0 | | |
| ID 1 | 5/6 | b | 0 | 0 | 0 | |
| ... | | | ... | ... | ... | |
| ID 10 | 5/6 | b | 0 | 0 | 9 | |
| Storage information | | | | | | |
| Status (VZ) of the historical /periodical value counter | 5/6 | b | 0 | 1 | 1 | ^f |
| Status (VZ) of the periodical value counter, period 1 | 5/6 | b | 0 | 1 | 1 | ^{1 f} |
| Number of available historical / periodical values | 5/6 | b | 0 | 1 | 2 | ^f |
| Number of available periodical values for period 2 | 5/6 | b | 0 | 1 | 2 | ^{2 f} |
| Set date | 5/6 | b | 0 | 1 | 10 | |
| Billing date | 5/6 | b | 0 | 1 | 11 | |
| Configuration | | | | | | |
| Program version | 5/6 | b | 0 | 2 | 0 | |
| Firmware version | 5/6 | b | 0 | 2 | 1 | |
| Software version | 5/6 | b | 0 | 2 | 2 | |
| Meter location (flow or return) ^a | 5/6 | b | 0 | 2 | 3 | |
| Device version | 5/6 | b | 0 | 2 | 4 | |
| Serial number of inlet (flow) temperature transducer | 5/6 | b | 0 | 2 | 10 | |
| Serial number of outlet (return) temperature transducer | 5/6 | b | 0 | 2 | 11 | |
| Serial number of forward flow transducer | 5/6 | b | 0 | 2 | 12 | |
| Serial number of return flow transducer | 5/6 | b | 0 | 2 | 13 | |
| Conversion factors | | | | | | |
| Heat coefficient, k | 5/6 | b | 0 | 4 | 1 | |

| General and service entry objects – Thermal energy | OBIS code | | | | | |
|--|--|---|-----|-----|-----|---|
| | A | B | C | D | E | F |
| | 5/6 | b | 0 | 4 | 2 | |
| Pressure (backup value) ^b | 5/6 | b | 0 | 4 | 2 | |
| Enthalpy ^c | 5/6 | b | 0 | 4 | 3 | |
| Threshold values | | | | | | |
| Threshold value limit for rate 1 ^d | 5/6 | b | 0 | 5 | 1 | |
| ... | | | ... | ... | ... | |
| Threshold value limit for rate 9 ^d | 5/6 | b | 0 | 5 | 9 | |
| Maximum contracted flow rate ^e | 5/6 | b | 0 | 5 | 21 | |
| Maximum contracted power ^e | 5/6 | b | 0 | 5 | 22 | |
| Maximum contracted $\Delta\theta$ ^e | 5/6 | b | 0 | 5 | 23 | |
| Minimum contracted return temperature ^e | 5/6 | b | 0 | 5 | 24 | |
| Timing information | | | | | | |
| Averaging period for measurements, generic | 5/6 | b | 0 | 8 | 0 | |
| Averaging period for instantaneous measurements | 5/6 | b | 0 | 8 | 1 | |
| Averaging period for volume / flow measurements | 5/6 | b | 0 | 8 | 2 | |
| Averaging period for temperature measurements | 5/6 | b | 0 | 8 | 3 | |
| Averaging period for pressure measurements | 5/6 | b | 0 | 8 | 4 | |
| Averaging period, power | 5/6 | b | 0 | 8 | 5 | |
| Averaging period, flow rate | 5/6 | b | 0 | 8 | 6 | |
| Averaging period, test values | 5/6 | b | 0 | 8 | 7 | |
| Measurement period, peak values, period 1(short) ^g | 5/6 | b | 0 | 8 | 11 | |
| Measurement period, peak values, period 2 ^g | 5/6 | b | 0 | 8 | 12 | |
| Measurement period, peak values, period 3 ^g | 5/6 | b | 0 | 8 | 13 | |
| Measurement period, peak values, period 4 ^g | 5/6 | b | 0 | 8 | 14 | |
| Measurement period, periodical values, period 1(short) ^g | 5/6 | b | 0 | 8 | 21 | |
| Measurement period, periodical values, period 2 ^g | 5/6 | b | 0 | 8 | 22 | |
| Measurement period, periodical values, period 3 ^g | 5/6 | b | 0 | 8 | 23 | |
| Measurement period, periodical values, period 4 ^g | 5/6 | b | 0 | 8 | 24 | |
| Measurement period, test values | 5/6 | b | 0 | 8 | 25 | |
| Recording interval 1 for profiles ^h | 5/6 | b | 0 | 8 | 31 | |
| Recording interval 2 for profiles ^h | 5/6 | b | 0 | 8 | 32 | |
| Recording interval 3 for profiles ^h | 5/6 | b | 0 | 8 | 33 | |
| Billing period | 5/6 | b | 0 | 8 | 34 | |
| Time entries | | | | | | |
| Local time | 5/6 | b | 0 | 9 | 1 | |
| Local date | 5/6 | b | 0 | 9 | 2 | |
| Time stamp (local time) of the most recent billing period ⁱ | 5/6 | b | 0 | 9 | 3 | |
| Manufacturer specific ^j | 5/6 | b | 96 | 50 | E | f |
| | | | | | | |
| Manufacturer specific | 5/6 | b | 96 | 99 | E | f |
| ^a | Information about where the (single) flow meter is inserted. A non-zero value is used when the flow meter is located in the flow path. | | | | | |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 426/492 |
|-----------------------|------------|-------------------------|---------|

| General and service entry objects – Thermal energy | OBIS code | | | | | |
|--|---|---|---|---|---|---|
| | A | B | C | D | E | F |
| b | Defines the pressure of the media, if not measured. The default value is 16 bar according to EN 1434-2:2015. | | | | | |
| c | The enthalpy of the thermal conveying liquid. This will be necessary when using media other than pure water. The enthalpy is a part of the calculations when converting from mass to power. | | | | | |
| d | Part of the contract between the customer and the supplier. The threshold defines when to switch rate, and can be used for diagnostic purposes, or to control limiting valves as well. | | | | | |
| e | Part of the contract between the customer and the supplier. The threshold may be used to set a 'flag', for diagnostic purposes, or to control limiting valves. | | | | | |
| f | Value group 'F' may be left unused, if there is only one set of historical / periodical values in the meter. | | | | | |
| g | The instantiation of periods in a meter shall always start at period 1. | | | | | |
| h | If only one recording interval is implemented, then it shall be recording interval 1. If multiple recording intervals are implemented, the recording interval 1 shall be the interval with the shorter period. | | | | | |
| i | In case of billing period schemes absence or event triggered, commonly calculated from local date and local time information | | | | | |
| j | The range D = 50...99 is available for identifying objects, which are not represented by another defined code, but need representation on the display as well. If this is not required, the range D = 128...254 should be used. | | | | | |

7.7.4.2 Error register objects – Thermal energy

Table 82 specifies OBIS codes for thermal energy related error register objects.

Table 82 – OBIS codes for error register objects – Thermal energy

| Error register objects – Thermal energy | OBIS code | | | | | |
|---|-----------|---|----|----|---|---|
| | A | B | C | D | E | F |
| Error registers | 5/6 | B | 97 | 97 | e | |
| NOTE The information to be included in the error objects is not defined in this document. | | | | | | |

7.7.4.3 Data profile objects – Thermal energy

Thermal energy related data profiles – identified with one single OBIS code – are used to hold a series of measurement values of one or more similar quantities and/or to group various data. The OBIS codes are specified in Table 83.

Table 83 – OBIS codes for data profile objects – Thermal energy

| Data profile objects – Thermal energy | OBIS code | | | | | |
|--|-----------|---|----|----|---|---|
| | A | B | C | D | E | F |
| Consumption / load profile with recording interval 1 | 5/6 | b | 99 | 1 | 1 | |
| Consumption / load profile with recording interval 2 | 5/6 | b | 99 | 1 | 2 | |
| Consumption / load profile with recording interval 3 | 5/6 | b | 99 | 1 | 3 | |
| Profile of maxima with recording interval 1 | 5/6 | b | 99 | 2 | 1 | |
| Profile of maxima with recording interval 2 | 5/6 | b | 99 | 2 | 2 | |
| Profile of maxima with recording interval 3 | 5/6 | b | 99 | 2 | 3 | |
| Consumption / load profile during test | 5/6 | b | 99 | 3 | 1 | |
| | | | | | | |
| Certification data log | 5/6 | b | 99 | 99 | e | |

7.7.4.4 OBIS codes for Thermal energy related objects (examples)

Table 84 shows examples for OBIS codes of Thermal energy related objects.

Table 84 – OBIS codes for Thermal energy related objects (examples)

| Thermal energy related objects (examples) | OBIS code | | | | | |
|---|--|---|----|----|----|------------------|
| | A | B | C | D | E | F |
| Consumption | | | | | | |
| Energy, current value, total | 5/6 | b | 1 | 0 | 0 | |
| Energy, current value, rate 1 | 5/6 | b | 1 | 0 | 1 | |
| Energy, periodical, total, the second last storage | 5/6 | b | 1 | 1 | 0 | 102 |
| Energy, billing date value, total, last storage, rate 1 | 5/6 | b | 1 | 3 | 1 | V _Z |
| Monitoring values | | | | | | |
| Energy, maximum value (current period) | 5/6 | b | 1 | 5 | | |
| Flow rate, Period value 2, previous storage | 5/6 | b | 9 | 12 | | V _{Z-1} |
| Power, Max value, previous period | 5/6 | b | 8 | 5 | | V _{Z-1} |
| Energy, Missing duration c | 5/6 | b | 1 | 25 | | |
| Differential temperature, Test value | 5/6 | b | 12 | 6 | | |
| Flow path, temperature transducers serial no. | 5/6 | b | 0 | 2 | 10 | |
| Error handling | | | | | | |
| Overall error status ^a | 5/6 | b | 97 | 97 | 0 | |
| Subsystem where error has occurred ^b | 5/6 | b | 97 | 97 | 1 | |
| Duration of error condition ^c | 5/6 | b | 97 | 97 | 2 | |
| ^a | This object is a 'mirror' of the object 0.x.97.97.0. | | | | | |
| ^b | This is the time during which the meter has not been able to calculate energy. | | | | | |
| ^c | A further subdivision of error information. | | | | | |

7.8 Gas (Value group A = 7)

7.8.1 General introduction to gas measurement

7.8.1.1 Overview

Measurement of the energy supplied in the form of gas to customers is a complex process. It has to take into account the characteristics of the measuring site, the gas measurement technology, the conditions and the properties of the gas and the characteristics of the billing process.

Energy measurement is generally a multi-step process:

- The first step is to measure either the amount of the volume or the mass of gas based on various measuring principles, like volume, flow, density or mass measurement. Accuracy can be improved by correcting the measurement error of the meter;
- In the case of volume measurement, the next step is to convert the volume measured at metering conditions to volume at base conditions;
- In the final step, the energy is calculated from the volume at base conditions or the mass, and the calorific value. The calorific value – either per volume unit or per mass unit – is determined using gas analysis techniques.

The measurement technology and the implementation of the volume conversion and energy calculation process depend on the application segment.

Conversion and calculation steps can take place at the measuring site by electronic devices, or in the IT system.

For measurement of larger volumes, there are several devices involved in the process, depending on installation and hazardous area requirements. Not only the final results, but also interim values in the conversion and calculation process are of interest for checking and controlling purposes.

7.8.1.2 Typical gas metering installations

7.8.1.2.1 Residential application

A typical residential gas metering installation is shown in Figure 32.

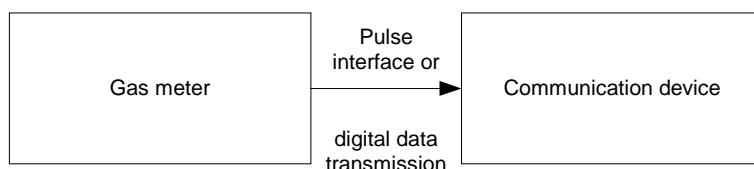


Figure 32 – Residential gas metering installation

The meter is typically a diaphragm (positive displacement) meter, which may perform mechanical temperature correction.

The information from the gas meter to the communication device may be transferred in the form of pulses. Alternatively, the meter may be equipped with a digital interface, e.g. an encoder turning the index reading to digital information.

Volume conversion and energy calculation takes place in the IT system.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 429/492 |
|-----------------------|------------|-------------------------|---------|

7.8.1.2.2 Industrial application

A typical industrial gas metering installation is shown in Figure 33.

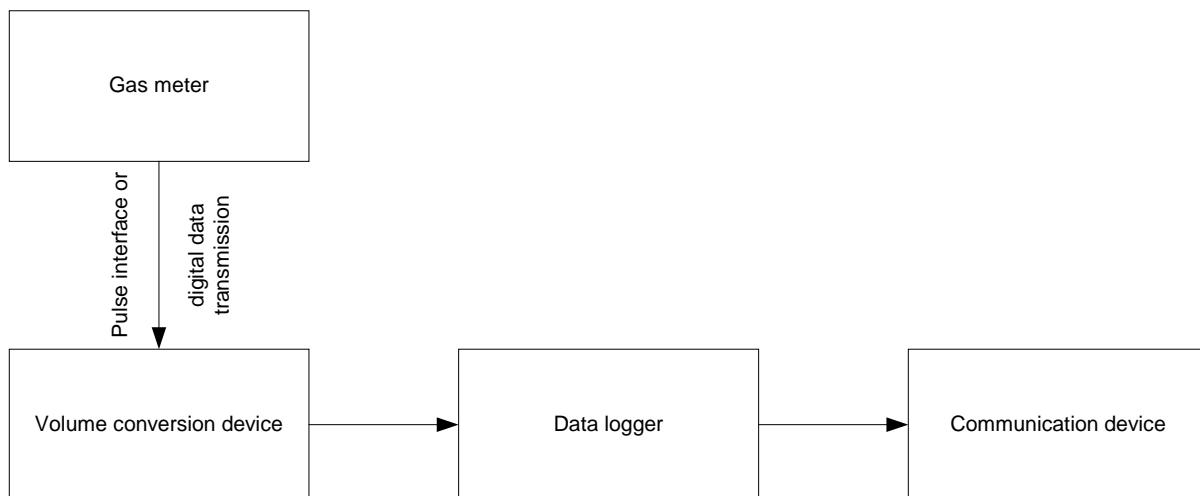


Figure 33 – Industrial gas metering installation (single stream)

In industrial applications, typically more functions are implemented at the measuring site than in residential applications. This may include the calculation of the volume at base conditions, and, if the calorific value is available (e.g. via remote communication), the calculation of the energy.

The data logger stores data relevant for billing, data validation and process control.

The functions may be integrated in fewer devices, depending on the hazardous zone restrictions and the level of integration of electronics.

7.8.1.2.3 Gas transport application

A typical gas transport metering “city gate” installation – also used for very large consumers – is shown in Figure 34.

Such gas stations are equipped with more than one pipe for the gas flow (multi stream). Typically, volume conversion devices are installed on each pipe, because the measurement is closely pipe related. Generally, there is one data logger and a device used to determine the calorific value (e.g. gas chromatograph).

All devices are connected via a bus system.

Depending on the design of these devices, selected functions may be implemented in a single cabinet or physical device.

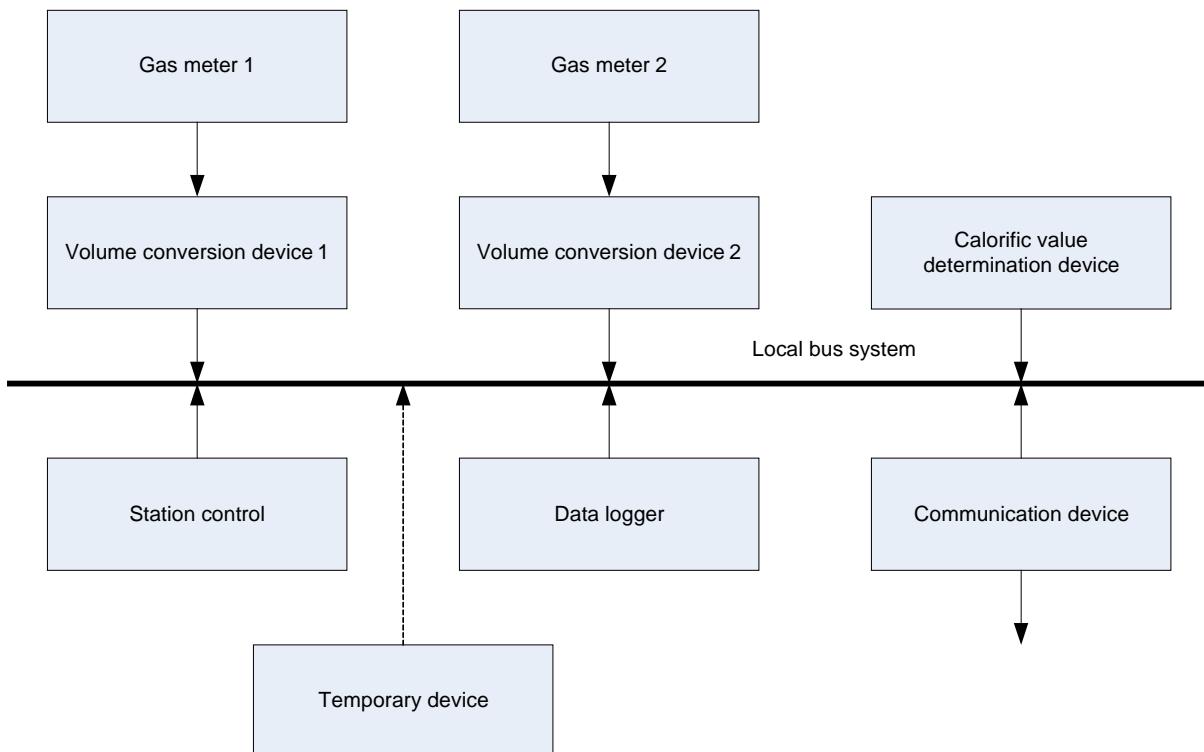


Figure 34 – City gate or border crossing installation (multi stream)

7.8.1.3 Gas volume conversion

7.8.1.3.1 General

The gas volume conversion process needs the following inputs:

- the volume information from a gas meter;
- the temperature of the gas measured;
- the pressure of the gas measured: this may be replaced by a constant;
- the compressibility, this may be replaced by a constant.

When the process is implemented in a gas conversion device, it is assumed to be capable of:

- performing error correction (optionally);
- measuring the temperature;
- measuring the pressure of the gas (optionally); and
- calculating the compressibility according to agreed algorithms, in function of temperature, pressure and gas composition (optionally).

The volume conversion device may handle bidirectional gas flows. The main direction of flow is *forward*.

It may be equipped with *disturbance registers* used when the value of temperature, pressure or compressibility is outside permissible metrological limits of plausibility, leading to an *alert condition*. When such alert condition occurs, the gas conversion process switches to store results in disturbance registers, until the alert conditions disappears.

7.8.1.3.2 Step 1: Error correction (optional)

The error curve of the gas meter is corrected by a correction factor:

$$V_c = C_f * V_m$$

where:

- V_c is the corrected volume;
- C_f is the correction factor given by an equation $C_f = f(q)$ or $C_f = f(Re)$; where q is the flow and Re is the Reynolds number;
- V_m is the volume at metering conditions.

The error correction method depends on station construction and operating conditions and its selection is made generally by manufacturer, utility or market specific.

7.8.1.3.3 Step 2: Volume conversion to base conditions

Volume at base conditions is calculated using the equation:

$$V_b = C \times V$$

Where:

- V_b is the volume at base conditions,
- V may be V_m or V_c (Volume at metering conditions or corrected volume);
- C is the conversion factor given by the relationship:

$$C = (P / P_b) \times (T_b / T) \times (Z_b / Z)$$

where Z is the compressibility factor allowing to take into account the difference in compressibility between the gas measured and the ideal gas. It is a function of the pressure and the temperature:

$$Z = f(P, T)$$

Settable gas properties and components are used for the compressibility calculation, combined into one of several existing calculation methods. If the compressibility factor is not calculated, it may be included as a fixed value in the calculation of the conversion factor. Below 1,5 bar, the value of Z is usually set to 1.

If the pressure is not measured, it may be included as a fixed value in the calculation of the conversion factor.

7.8.1.3.4 Step 3: Energy conversion

The final step is to calculate the energy, using the equation:

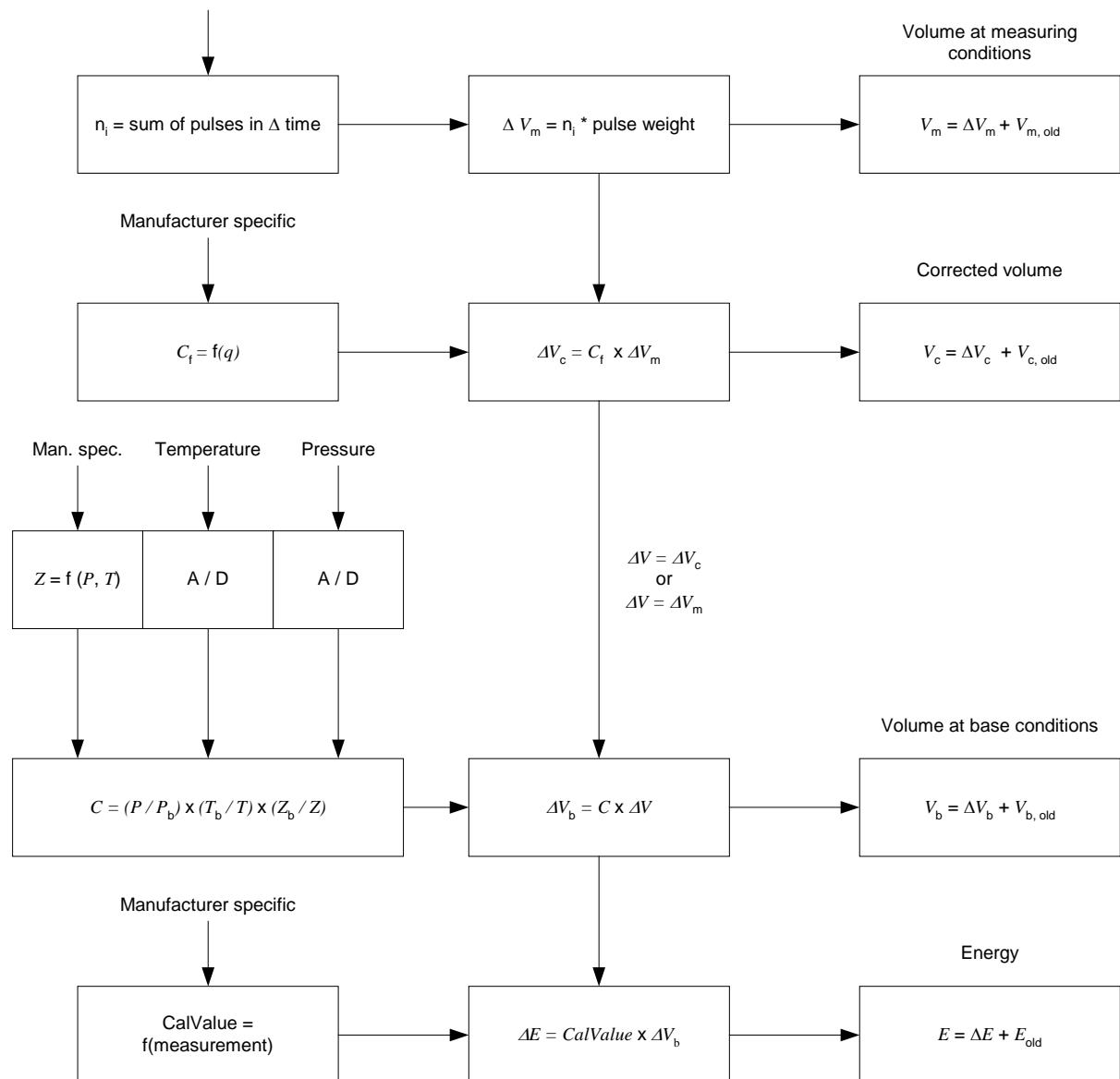
$$E = \text{CalValue} \times V_b$$

where CalValue is the calorific value, expressed in J/m^3 . Typically, it is measured by calorimeter or gas chromatograph devices.

7.8.1.3.5 Model of data flow for volume conversion and energy calculation

The model of data flow for volume conversion and energy calculation is shown in Figure 35.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 432/492 |
|-----------------------|------------|-------------------------|---------|

**Figure 35 – Data flow of volume conversion and energy calculation**

The OBIS codes of the main objects in the data flow are shown in Table 85, with the following assumptions:

- the conversion process passes through all four functions from metering to energy;
- the device has one single channel;
- the direction of the gas flow is forward;
- energy is the result of the conversion process from volume at base conditions to energy, by applying the calorific value as factor;
- the data of interest are current values of absolute indexes and the gas process data.

Table 85 – OBIS codes of the main objects in the gas conversion process data flow

| Name | Symbol | OBIS code |
|---|--|-----------------|
| Indexes | | |
| Forward absolute meter volume, index, at metering conditions | V_m | 7.0.3.0.0.255 |
| Forward absolute converter volume, index, at metering conditions | V_m | 7.0.13.0.0.255 |
| Forward absolute converter volume, index, corrected value | V_c | 7.0.13.1.0.255 |
| Forward absolute converter volume, index, at base conditions | V_b | 7.0.13.2.0.255 |
| Forward absolute energy, index, at base conditions | E | 7.0.33.2.0.255 |
| Compressibility, correction and conversion values | | |
| Correction factor ^a | C_f | 7.0.51.0.0.255 |
| Conversion factor ^b | C | 7.0.52.0.0.255 |
| Compressibility factor, current value at metering conditions ^c | Z | 7.0.53.0.0.255 |
| Compressibility factor, current value at base conditions ^c | Z_b | 7.0.53.2.0.255 |
| Compressibility factor, preset value ^c | Z_b | 7.0.53.11.0.255 |
| Compressibility factor, calculation method ^c | | 7.0.53.12.0.255 |
| Superior calorific value ^d | CalVal | 7.0.54.0.0.255 |
| Metering site condition information | | |
| Gas temperature (absolute), value at metering conditions ^e | T | 7.0.41.0.0.255 |
| Gas temperature (absolute), value at base conditions ^e | T_b | 7.0.41.2.0.255 |
| Gas temperature (absolute), backup value ^e | T | 7.0.41.3.0.255 |
| Gas pressure (absolute), value at metering conditions ^f | P | 7.0.42.0.0.255 |
| Gas pressure (absolute), value at base conditions ^f | P_b | 7.0.42.2.0.255 |
| Gas pressure (absolute), backup value ^f | P | 7.0.42.3.0.255 |
| ^a | A fixed value used to correct a scalar error on a meter: for example, if a meter under-registers volume by 0,5 %, then a correction factor value of 1,005 will compensate for the error. | |
| ^b | See 7.8.1.3.3. | |
| ^c | Compressibility, Z: effectively, the “difference” in compressibility between the gas being measured and “noble” gas. For example, EN 12405, SGERG-88, AGA 8 give full information on this, though below 1,5 Bar (a) this is usually set to 1. | |
| ^d | The superior (or gross) calorific value can be seen as a conversion factor for converting volume to energy although it is also used for the conversion algorithm. | |
| ^e | Temperature of the gas, expressed in Kelvin. Volume conversion depends on Kelvin temperature measurement. This may represent a measured value or a base condition, or a backup value, used if the temperature sensor fails, as identified by the value of value group D. | |
| ^f | Pressure of the gas, expressed in a suitable unit, in absolute terms, for example Bar (a). This means that the value is referenced to a perfect vacuum, as opposed to “Gauge” pressure, which is referenced to current atmospheric conditions. This may represent a measured value or a base condition, or a backup value, used if the pressure sensor fails, as identified by the value of value group D. | |

7.8.1.4 Data logging

7.8.1.4.1 General

The data logging process captures, generates and makes available the data necessary for billing, as well as the data necessary for managing the measurement process and the gas grid.

7.8.1.4.2 Time bound processing

Quantities measured by the gas meter, calculated in the data logger or in the IT system may be:

- indexes, index differences and maxima of index differences; and
- average, minimum and maximum values

related to various intervals and periods. A distinction is made between:

- recording intervals for profiles;
- measurement periods for average values;
- process intervals;
- measurement periods for index differences;
- billing periods for indexes, index differences and maxima of index differences;
- averaging periods.

Some of these periods and intervals may have a default length, or otherwise their length can be held by specific objects. See 7.8.6.1, Table 96.

The processing methods depend on the kind of the quantity:

- indexes and index differences; see 7.8.3.2;
- flow rate, see 7.8.3.3;
- process values, see 7.8.3.4;
- conversion related factors and coefficients, see 7.8.3.5; and
- natural gas analysis, see 7.8.3.6.

7.8.1.4.3 Gas day

One specific element in gas metering is that the start of a gas day may be different from the start of a calendar day.

NOTE 1 For example the gas day starts at 6:00 in Germany.

NOTE 2 In some countries, the gas day start time retains its value when DST starts and ends, causing a 25 hour and 23 hour day in each year.

Therefore, taking the example above, a gas month lasts from 6:00 of the first day of a calendar month to 6:00 of the first day of the next calendar month. Similarly, a gas year starts at 6:00 on 1st of January and ends at 6:00 on 1st January of the next year.

7.8.1.4.4 Data profiles

COSEM “Profile generic” objects may capture one or several values – attributes of COSEM objects – in their buffer.

For gas metering, both *general purpose* and *dedicated* profiles are available:

- a general purpose “Profile generic” object captures one or several values. Such objects have a general OBIS code / logical name that do not provide specific information on the values captured. These profiles are also available with some fixed recording intervals;
- a dedicated “Profile generic” object captures only one value. The OBIS code / logical name of such a dedicated “Profile generic” object is “self-explanatory”, i.e. it reflects the OBIS code of the object the value attribute of which is captured.

NOTE A time stamp and a status attribute may be captured in addition to the value(s) of interest.

In any case, the values captured are identified by the *capture_objects* attribute. See 7.8.6.4.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 435/492 |
|-----------------------|------------|-------------------------|---------|

7.8.2 Value group C codes – Gas

The allocations in the value group C – see Table 86 – take into account the different combinations of measuring and calculating devices located at a metering point, to allow identifying the source where the data are generated.

For the purposes of volume / mass / energy measurement, value group C identifies:

- the location of the device in the measurement chain: meter (encoder), converter, logger;
- the direction of the gas flow: forward or reverse;
- the qualifier of the measurement: undisturbed, disturbed, or absolute, where absolute value is the sum of the values calculated under undisturbed and disturbed conditions.
- Value group C is also used for identifying process data.

For the purposes of gas analysis, a distinction is made between measured values generated by gas analysing systems (C = 70) and parameters used for calculation (C = 0, D = 12).

Table 86 – Value group C codes – Gas

| Value group C codes – Gas (A = 7) | |
|-----------------------------------|---|
| 0 | General purpose objects |
| 1 | Forward undisturbed meter volume |
| 2 | Forward disturbed meter volume |
| 3 | Forward absolute meter volume |
| 4 | Reverse undisturbed meter volume |
| 5 | Reverse disturbed meter volume |
| 6 | Reverse absolute meter volume |
| 7 | Forward absolute meter volume (encoder) |
| 8 | Reverse absolute meter volume (encoder) |
| 11 | Forward undisturbed converter volume |
| 12 | Forward disturbed converter volume |
| 13 | Forward absolute converter volume |
| 14 | Reverse undisturbed converter volume |
| 15 | Reverse disturbed converter volume |
| 16 | Reverse absolute converter volume |
| 21 | Forward undisturbed logger volume |
| 22 | Forward disturbed logger volume |
| 23 | Forward absolute logger volume |
| 24 | Reverse undisturbed logger volume |
| 25 | Reverse disturbed logger volume |
| 26 | Reverse absolute logger volume |
| 31 | Forward undisturbed energy |
| 32 | Forward disturbed energy |

| Value group C codes – Gas (A = 7) | |
|-----------------------------------|--|
| 33 | Forward absolute energy |
| 34 | Reverse undisturbed energy |
| 35 | Reverse disturbed energy |
| 36 | Reverse absolute energy |
| | |
| 41 | Absolute temperature |
| 42 | Absolute pressure |
| 43 | Flow rate |
| 44 | Velocity of sound |
| 45 | Density (of gas) |
| 46 | Relative density |
| 47 | Gauge pressure |
| 48 | Differential pressure |
| 49 | Density of air |
| | |
| 51 | Correction factor |
| 52 | Conversion factor |
| 53 | Compressibility factor |
| 54 | Superior calorific value ^a |
| 55 | Gas law deviation coefficient (= compressibility factor ratio) |
| | |
| 61 | Forward undisturbed mass |
| 62 | Forward disturbed mass |
| 63 | Forward absolute mass |
| 64 | Reverse undisturbed mass |
| 65 | Reverse disturbed mass |
| 66 | Reverse absolute mass |
| | |
| 70 | Natural gas analysis |
| | |
| 93 | Consortia specific identifiers |
| 94 | Country specific identifiers |
| | |
| 96 | General and service entry objects – Gas (See 7.8.6.1) |
| 97 | Error register objects – Gas (See 7.8.6.2) |
| 98 | List objects – Gas (See 7.8.6.3) |
| 99 | Data profiles – Gas (See 7.8.6.4) |
| 128...199, 240 | Manufacturer specific codes |
| All other | Reserved |
| Notes | |

| |
|--|
| Value group C codes – Gas (A = 7) |
|--|

| |
|--|
| ^a The superior (or gross) calorific value can be seen as a conversion factor for converting volume to energy although it is also used for the conversion algorithm. |
|--|

7.8.3 Value group D codes – Gas

7.8.3.1 General

Allocations in value group D allow to further classify quantities identified by codes in value group A to C. The allocations depend on the kind of quantity:

- indexes and index differences; see 7.8.3.2;
- flow rate, see 7.8.3.3;
- process values, see 7.8.3.4;
- conversion related factors and coefficients, see 7.8.3.5; and
- natural gas analysis values, see 7.8.3.6.

7.8.3.2 Gas indexes and index differences

The allocations allow identifying the various volume, mass and energy quantities measured along the measuring chain and the gas volume conversion process, relative to various measurement and billing periods:

- indexes: current values and historical values relative to various billing periods;
 - index differences: current and last values relative to measurement periods and billing periods;
- NOTE Index difference over a certain measurement or billing period is also known as consumption. For consumption, thresholds may be defined, see Table 96.
- maximum of index differences over various measurement periods, relative to various billing periods;

A distinction is made between *value at metering conditions, corrected value* and *value at base conditions (converted value)*. The applicability of these qualifiers depends on the location in the measuring chain and in the gas volume conversion process.

Three measurement periods are available:

- measurement period 1: default value 15 min;
- measurement period 2: default value 1 hour;
- measurement period 3: no default value specified.

Four billing periods are available:

- billing period 1: default value 1 day;
- billing period 2: default value 1 month;
- billing period 3: default value 1 year;
- billing period 4: no default value specified.

The default values specified reflect the most common applications. If other values are used, they may be held by COSEM objects specified for this purpose; see 6.4.4 and Table 96.

In addition to the current values of the indexes, the following values are available:

For measurement periods 1 to 3:

- index differences for the current and the last measurement period (6 values each).

| | | |
|-----------------------|------------|-------------------------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 |
| | | 438/492 |

For billing periods 1, 3 and 4:

- historical indexes (3 values each);
- index differences for the current and the last billing period (6 values each);
- maximum of index differences over measurement periods 1, 2 and 3 (9 values each);
- in total, 18 values each.

For billing period 2:

- historical indexes (3 values);
- index differences for the current and the last billing period (6 values);
- maximum of index differences over measurement periods 1, 2 and 3, as well as over billing period 1 (12 values);
- in total, 21 values.

For all these values, tariffs may be applied. See 7.8.4.2.

Table 87 specifies the use of value group D to identify gas related indexes and index differences.

Table 87 – Value group D codes – Gas – Indexes and index differences

| Value group D codes – Gas – Indexes and index differences (A= 7, C = 1...8, 11...16, 21...26, 31...36, 61...66) | | | |
|--|------------------|---|-----------------------|
| | Quantity | Qualifier | Period |
| 0 | Index | Value at metering conditions | Current ^{a)} |
| 1 | Index | Corrected value ^a | Current ^{c)} |
| 2 | Index | Value at base conditions / "Converted value" | Current ^{c)} |
| 3 | Index | Current redundant value at metering conditions ^b | Current ^{c)} |
| Values relative to measurement period 1 (default value = 15 minutes) | | | |
| 6 | Index difference | Value at metering conditions | Current |
| 7 | Index difference | Corrected value | Current |
| 8 | Index difference | Value at base conditions | Current |
| 9 | Index difference | Value at metering conditions | Last |
| 10 | Index difference | Corrected value | Last |
| 11 | Index difference | Value at base conditions | Last |
| Values relative to measurement period 2 (default value = 1 hour) | | | |
| 12 | Index difference | Value at metering conditions | Current |
| 13 | Index difference | Corrected value | Current |
| 14 | Index difference | Value at base conditions | Current |
| 15 | Index difference | Value at metering conditions | Last |
| 16 | Index difference | Corrected value | Last |
| 17 | Index difference | Value at base conditions | Last |
| Values relative to measurement period 3 (no default value) | | | |
| 18 | Index difference | Value at metering conditions | Current |
| 19 | Index difference | Corrected value | Current |

| Value group D codes – Gas – Indexes and index differences (A= 7, C = 1...8, 11...16, 21...26, 31...36, 61...66) | | | |
|--|---|------------------------------|-------------------------|
| | Quantity | Qualifier | Period |
| 20 | Index difference | Value at base conditions | Current |
| 21 | Index difference | Value at metering conditions | Last |
| 22 | Index difference | Corrected value | Last |
| 23 | Index difference | Value at base conditions | Last |
| Values relative to billing period 1 (default value = 1 day) | | | |
| 24 | Index | Value at metering conditions | Historical ^c |
| 25 | Index | Corrected value | Historical ^c |
| 26 | Index | Value at base conditions | Historical ^c |
| 27 | Index difference | Value at metering conditions | Current |
| 28 | Index difference | Corrected value | Current |
| 29 | Index difference | Value at base conditions | Current |
| 30 | Index difference | Value at metering conditions | Last |
| 31 | Index difference | Corrected value | Last |
| 32 | Index difference | Value at base conditions | Last |
| 33 | Maximum of Index differences over measurement period 1 ^c | Value at metering conditions | |
| 34 | Maximum of Index differences over measurement period 1 ^c | Corrected value | |
| 35 | Maximum of Index differences over measurement period 1 ^c | Value at base conditions | |
| 36 | Maximum of Index differences over measurement period 2 ^c | Value at metering conditions | |
| 37 | Maximum of Index differences over measurement period 2 ^c | Corrected value | |
| 38 | Maximum of Index differences over measurement period 2 ^c | Value at base conditions | |
| 39 | Maximum of Index differences over measurement period 3 ^c | Value at metering conditions | |
| 40 | Maximum of Index differences over measurement period 3 ^c | Corrected value | |
| 41 | Maximum of Index differences over measurement period 3 ^c | Value at base conditions | |
| Values relative to billing period 2 (default value = 1 month) | | | |
| 42 | Index | Value at metering conditions | Historical ^c |
| 43 | Index | Corrected value | Historical ^c |
| 44 | Index | Value at base conditions | Historical ^c |
| 45 | Index difference | Value at metering conditions | Current |
| 46 | Index difference | Corrected value | Current |
| 47 | Index difference | Value at base conditions | Current |
| 48 | Index difference | Value at metering conditions | Last |
| 49 | Index difference | Corrected value | Last |
| 50 | Index difference | Value at base conditions | Last |
| 51 | Maximum of Index differences over measurement | Value at metering conditions | |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 440/492 |
|-----------------------|------------|-------------------------|---------|

| Value group D codes – Gas – Indexes and index differences (A= 7, C = 1...8, 11...16, 21...26, 31...36, 61...66) | | | |
|--|---|------------------------------|---------------|
| | Quantity | Qualifier | Period |
| | period 1 °C | | |
| 52 | Maximum of Index differences over measurement period 1 °C | Corrected value | |
| 53 | Maximum of Index differences over measurement period 1 °C | Value at base conditions | |
| 54 | Maximum of Index differences over measurement period 2 °C | Value at metering conditions | |
| 55 | Maximum of Index differences over measurement period 2 °C | Corrected value | |
| 56 | Maximum of Index differences over measurement period 2 °C | Value at base conditions | |
| 57 | Maximum of Index differences over measurement period 3 °C | Value at metering conditions | |
| 58 | Maximum of Index differences over measurement period 3 °C | Corrected value | |
| 59 | Maximum of Index differences over measurement period 3 °C | Value at base conditions | |
| 60 | Maximum of Index differences over billing period 1 °C | Value at metering conditions | |
| 61 | Maximum of Index differences over billing period 1 °C | Corrected value | |
| 62 | Maximum of Index differences over billing period 1 °C | Value at base conditions | |
| Values relative to billing period 3 (default value = 1 year) | | | |
| 63 | Index | Value at metering conditions | Historical °C |
| 64 | Index | Corrected value | Historical °C |
| 65 | Index | Value at base conditions | Historical °C |
| 66 | Index difference | Value at metering conditions | Current |
| 67 | Index difference | Corrected value | Current |
| 68 | Index difference | Value at base conditions | Current |
| 69 | Index difference | Value at metering conditions | Last |
| 70 | Index difference | Corrected value | Last |
| 71 | Index difference | Value at base conditions | Last |
| 72 | Maximum of Index differences over measurement period 1 °C | Value at metering conditions | |
| 73 | Maximum of Index differences over measurement period 1 °C | Corrected value | |
| 74 | Maximum of Index differences over measurement period 1 °C | Value at base conditions | |
| 75 | Maximum of Index differences over measurement period 2 °C | Value at metering conditions | |
| 76 | Maximum of Index differences over measurement period 2 °C | Corrected value | |
| 77 | Maximum of Index differences over measurement period 2 °C | Value at base conditions | |
| 78 | Maximum of Index differences over measurement period 3 °C | Value at metering conditions | |
| 79 | Maximum of Index differences over measurement period 3 °C | Corrected value | |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 441/492 |
|-----------------------|------------|-------------------------|---------|

| Value group D codes – Gas – Indexes and index differences (A= 7, C = 1...8, 11...16, 21...26, 31...36, 61...66) | | | |
|--|---|------------------------------|-------------------------|
| | Quantity | Qualifier | Period |
| 80 | Maximum of Index differences over measurement period 3 ^c | Value at base conditions | |
| Values relative to billing period 4 (no default value) | | | |
| 81 | Index | Value at metering conditions | Historical ^c |
| 82 | Index | Corrected value | Historical ^c |
| 83 | Index | Value at base conditions | Historical ^c |
| 84 | Index difference | Value at metering conditions | Current |
| 85 | Index difference | Corrected value | Current |
| 86 | Index difference | Value at base conditions | Current |
| 87 | Index difference | Value at metering conditions | Last |
| 88 | Index difference | Corrected value | Last |
| 89 | Index difference | Value at base conditions | Last |
| 90 | Maximum of Index differences over measurement period 1 ^c | Value at metering conditions | |
| 91 | Maximum of Index differences over measurement period 1 ^c | Corrected value | |
| 92 | Maximum of Index differences over measurement period 1 ^c | Value at base conditions | |
| 93 | Maximum of Index differences over measurement period 2 ^c | Value at metering conditions | |
| 94 | Maximum of Index differences over measurement period 2 ^c | Corrected value | |
| 95 | Maximum of Index differences over measurement period 2 ^c | Value at base conditions | |
| 96 | Maximum of Index differences over measurement period 3 ^c | Value at metering conditions | |
| 97 | Maximum of Index differences over measurement period 3 ^c | Corrected value | |
| 98 | Maximum of Index differences over measurement period 3 ^c | Value at base conditions | |
| All other | Reserved | | |
| ^a | Error correction of meter curves can be allocated to meters (e.g. temperature compensation of a diaphragm gas meter) or subsequent connected devices (e.g. high pressure correction curve of a turbine meter implemented in an associated volume conversion device). | | |
| ^b | From data logger (parallel recording) for use in case of a measurement device fails. | | |
| ^c | Current value: F = 255 Historical values (F ≠ 255): - With F = 1...12, 0...99 value(s) of (a) previous billing period, relative to the billing period counter. - With F = 101...126 value(s) of (a) previous billing period(s) relative to the current billing period. | | |

7.8.3.3 Flow rate

The allocations allow identifying values associated with the flow rate of the gas. The flow rate is a process information. It is not linked to a physical device. No tariffication is applicable.

A distinction is made between:

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 442/492 |
|-----------------------|------------|-------------------------|---------|

- current average, last average, and maximum of last average values measured over various averaging periods, relative to various measurement and billing periods. Measurement period 2 and 3 shall be multiple of the averaging period of block demand / sliding demand measurement.
- values at metering conditions, corrected value, value at base conditions (converted value) and value at standard conditions;

NOTE Standard conditions refer to national regulations, which may differ from ISO standards reference values for base conditions.

EXAMPLE Gas reference temperature at standard conditions is 0 °C, gas reference temperature at base conditions is +15 °C.

For averaging period 2, block demand (default) or sliding demand is available. In the case of sliding demand, the averaging period is split to sub-periods. The number of sub-periods is carried by the object 7.b.0.8.35.255; see Table 96.

The last average values of the various flow rate quantities can be captured to load profiles, with self-explanatory OBIS codes, see 7.8.6.4.

Table 88 specifies the use of value group D to identify gas related flow rate values.

Table 88 – Value group D codes – Gas – Flow rate

| Value group D codes – Gas – Flow rate (A = 7, C = 43) | | |
|---|---|--|
| | Quantity | Qualifier |
| 0 | Instantaneous | Current value at metering conditions |
| 1 | Instantaneous | Corrected value |
| 2 | Instantaneous | Value at base conditions / "Converted value" |
| 13 | Instantaneous | Value at standard conditions |
| Averaging period 1, default value = 5 minutes | | |
| 15 | Current average for averaging period 1 | Value at metering conditions |
| 16 | | Corrected value |
| 17 | | Value at base conditions |
| 18 | | Value at standard conditions |
| 19 | Last average for averaging period 1 | Value at metering conditions |
| 20 | | Corrected value |
| 21 | | Value at base conditions |
| 22 | | Value at standard conditions |
| 23 | Maximum of last averages for averaging period 1 relative to measurement period 2 (default value = 1 hour) | Value at metering conditions |
| 24 | | Corrected value |
| 25 | | Value at base conditions |
| 26 | | Value at standard conditions |
| 27 | Maximum of last averages for averaging period 1 relative to measurement period 3 (no default value) | Value at metering conditions |
| 28 | | Corrected value |
| 29 | | Value at base conditions |
| 30 | | Value at standard conditions |

| Value group D codes – Gas – Flow rate (A = 7, C = 43) | | | |
|---|---|------------------------------|--|
| 31 | Maximum of last averages for averaging period 1 relative to billing period 1 (default value = 1 day) | Value at metering conditions | |
| 32 | | Corrected value | |
| 33 | | Value at base conditions | |
| 34 | | Value at standard conditions | |
| Averaging period 2, default value = 15 minutes (block demand or sliding demand) | | | |
| 35 | Current average for averaging period 2 | Value at metering conditions | |
| 35 | | Corrected value | |
| 37 | | Value at base conditions | |
| 38 | | Value at standard conditions | |
| 39 | Last average for averaging period 2 | Value at metering conditions | |
| 40 | | Corrected value | |
| 41 | | Value at base conditions | |
| 42 | | Value at standard conditions | |
| 43 | Maximum of last averages for averaging period 2 relative to measurement period 2 (default value = 1 hour) | Value at metering conditions | |
| 44 | | Corrected value | |
| 45 | | Value at base conditions | |
| 46 | | Value at standard conditions | |
| 47 | Maximum of last averages for averaging period 2 relative to measurement period 3 (no default value) | Value at metering conditions | |
| 48 | | Corrected value | |
| 49 | | Value at base conditions | |
| 50 | | Value at standard conditions | |
| 51 | Maximum of last averages for averaging period 2 relative to billing period 1 (default value = 1 day) | Value at metering conditions | |
| 52 | | Corrected value | |
| 53 | | Value at base conditions | |
| 54 | | Value at standard conditions | |
| Averaging period 3, default value = 1 hour | | | |
| 55 | Current average for averaging period 3 | Value at metering conditions | |
| 56 | | Corrected value | |
| 57 | | Value at base conditions | |
| 58 | | Value at standard conditions | |
| 59 | Last average for averaging period 3 | Value at metering conditions | |
| 60 | | Corrected value | |
| 61 | | Value at base conditions | |
| 62 | | Value at standard conditions | |

| Value group D codes – Gas – Flow rate (A = 7, C = 43) | | |
|---|--|------------------------------|
| Averaging period 4, (no default value) | | |
| 63 | Current average for averaging period 4 | Value at metering conditions |
| 64 | | Corrected value |
| 65 | | Value at base conditions |
| 66 | | Value at standard conditions |
| 67 | Last average for averaging period 4 | Value at metering conditions |
| 68 | | Corrected value |
| 69 | | Value at base conditions |
| 70 | | Value at standard conditions |
| All other | Reserved | |

7.8.3.4 Process values

For process values, a distinction is made between:

- instantaneous values;
- average, minimum and maximum values over various process intervals;
- value at metering conditions, value at base conditions; and value at standard conditions;

NOTE Standard conditions refer to national regulations, which may differ from ISO standards reference values for base conditions.

EXAMPLE Gas reference temperature at standard conditions is 0 °C, gas reference temperature at base conditions is +15 °C.

- for some quantities, backup, actual and preset values are available.

Table 89 specifies the use of value group D to identify gas related process values.

Table 89 – Value group D codes – Gas – Process values

| Value group D codes – Gas – Process values (A = 7, C = 41, 42, 44...49) | | |
|---|---|---|
| | Quantity | Qualifier |
| 0 | Instantaneous | Current value at metering conditions ^a |
| 2 | Instantaneous | Value at base conditions / “Converted value” ^b |
| 3 | Instantaneous | Backup value |
| 10 | Instantaneous | Actual value |
| 11 | Instantaneous | Preset value |
| 13 | Instantaneous | Value at standard conditions |
| Process interval 1 (default value = 15 minutes) | | |
| 15 | Average, current interval, process interval 1 | Value at metering conditions |
| 16 | | Value at base conditions |
| 17 | | Value at standard conditions |
| 18 | Minimum, current interval, process interval 1 | Value at metering conditions |
| 19 | | Value at base conditions |

| Value group D codes – Gas – Process values (A = 7, C = 41, 42, 44...49) | | |
|---|---|------------------------------|
| 20 | | Value at standard conditions |
| 21 | | Value at metering conditions |
| 22 | Maximum, current interval, process interval 1 | Value at base conditions |
| 23 | | Value at standard conditions |
| 24 | | Value at metering conditions |
| 25 | Average, last interval, process interval 1 | Value at base conditions |
| 26 | | Value at standard conditions |
| 27 | | Value at metering conditions |
| 28 | Minimum, last interval, process interval 1 | Value at base conditions |
| 29 | | Value at standard conditions |
| 30 | | Value at metering conditions |
| 31 | Maximum, last interval, process interval 1 | Value at base conditions |
| 32 | | Value at standard conditions |
| Process interval 2 (default value = 1 hour) | | |
| 33 | | Value at metering conditions |
| 34 | Average, current interval, process interval 2 | Value at base conditions |
| 35 | | Value at standard conditions |
| 36 | | Value at metering conditions |
| 37 | Minimum, current interval, process interval 2 | Value at base conditions |
| 38 | | Value at standard conditions |
| 39 | | Value at metering conditions |
| 40 | Maximum, current interval, process interval 2 | Value at base conditions |
| 41 | | Value at standard conditions |
| 42 | | Value at metering conditions |
| 43 | Average, last interval, process interval 2 | Value at base conditions |
| 44 | | Value at standard conditions |
| 45 | | Value at metering conditions |
| 46 | Minimum, last interval, process interval 2 | Value at base conditions |
| 47 | | Value at standard conditions |
| 48 | | Value at metering conditions |
| 49 | Maximum, last interval, process interval 2 | Value at base conditions |
| 50 | | Value at standard conditions |
| Process interval 3 (default value = 1 day) | | |
| 51 | | Value at metering conditions |
| 52 | Average, current interval, process interval 3 | Value at base conditions |

| Value group D codes – Gas – Process values (A = 7, C = 41, 42, 44...49) | | |
|---|---|------------------------------|
| 53 | | Value at standard conditions |
| 54 | | Value at metering conditions |
| 55 | Minimum, current interval, process interval 3 | Value at base conditions |
| 56 | | Value at standard conditions |
| 57 | | Value at metering conditions |
| 58 | Maximum, current interval, process interval 3 | Value at base conditions |
| 59 | | Value at standard conditions |
| 60 | | Value at metering conditions |
| 61 | Average, last interval, process interval 3 | Value at base conditions |
| 62 | | Value at standard conditions |
| 63 | | Value at metering conditions |
| 64 | Minimum, last interval, process interval 3 | Value at base conditions |
| 65 | | Value at standard conditions |
| 66 | | Value at metering conditions |
| 67 | Maximum, last interval, process interval 3 | Value at base conditions |
| 68 | | Value at standard conditions |
| Process interval 4 (default value = 1 month) | | |
| 69 | | Value at metering conditions |
| 70 | Average, current interval, process interval 4 | Value at base conditions |
| 71 | | Value at standard conditions |
| 72 | | Value at metering conditions |
| 73 | Minimum, current interval, process interval 4 | Value at base conditions |
| 74 | | Value at standard conditions |
| 75 | | Value at metering conditions |
| 76 | Maximum, current interval, process interval 4 | Value at base conditions |
| 77 | | Value at standard conditions |
| 78 | | Value at metering conditions |
| 79 | Average, last interval, process interval 4 | Value at base conditions |
| 80 | | Value at standard conditions |
| 81 | | Value at metering conditions |
| 82 | Minimum, last interval, process interval 4 | Value at base conditions |
| 83 | | Value at standard conditions |
| 84 | | Value at metering conditions |
| 85 | Maximum, last interval, process interval 4 | Value at base conditions |
| 86 | | Value at standard conditions |

| Value group D codes – Gas – Process values (A = 7, C = 41, 42, 44...49) | | |
|---|---|------------------------------|
| Process interval 5, since last event | | |
| 87 | | Value at metering conditions |
| 88 | Average, process interval 5, interval since last event | Value at base conditions |
| 89 | | Value at standard conditions |
| 90 | | Value at metering conditions |
| 91 | Average, process interval 6, interval between last two events | Value at base conditions |
| 92 | | Value at standard conditions |
| All other | Reserved | |
| ^a | To be used for e.g. velocity of sound. | |
| ^b | Value of the base conditions is associated with reference values for volume conversion: C = 41, 42. | |

7.8.3.5 Conversion related factors and coefficients

For correction, conversion, compressibility, superior calorific value and gas law deviation coefficient values, various OBIS code allocations are made taking into consideration the specifics of the measuring process. See Table 90.

For these values, average values over various averaging periods are also defined; see 7.8.4.5.

Table 90 specifies the use of value group D to identify gas conversion related factors and coefficients values.

Table 90 – Value group D codes – Gas – Conversion related factors and coefficients

| Value group D codes – Gas – Conversion related factors and coefficients (A = 7, C = 51...55) | |
|---|--|
| 0 | Current value at metering conditions |
| 2 | Current value at base conditions / "Converted Value" |
| 3 | Backup |
| 10 | Actual |
| 11 | Preset |
| 12 | Method |
| All other | Reserved |

7.8.3.6 Natural gas analysis values

For natural gas analysis, allocations in value group D identify the key parameters and the components of the natural gas. For these values, average values over various averaging periods are also defined; see 7.8.4.6. Table 91 specifies the use of value group D to identify natural gas analysis values.

Table 91 – Value group D codes – Gas – Natural gas analysis values

| Value group D codes – Gas – Natural gas analysis values (A = 7, C = 70) | |
|---|---|
| 8 | Reference pressure of gas analysis |
| 9 | Reference temperature of gas analysis |
| 10 | Superior ^a Wobbe index 0 °C |
| 11 | Inferior ^b Wobbe index 0 °C |
| 12 | Methane number |
| 13 | Total sulphur |
| 14 | Hydrogen sulphide H ₂ S |
| 15 | Mercaptans |
| 16 | Water dew point (DP H ₂ O) |
| 17 | Water (H ₂ O) dew point outlet / normalised |
| 18 | Hydrocarbon dew point (DP C _X H _Y) |
| 19 | Inferior ^c calorific value H _{i,n} |
| 20 | Water H ₂ O |
| | |
| 60 | Nitrogen N ₂ |
| 61 | Hydrogen H ₂ |
| 62 | Oxygen O ₂ |
| 63 | Helium He |
| 64 | Argon Ar |
| 65 | Carbon monoxide CO |
| 66 | Carbon dioxide CO ₂ |
| 67 | Methane CH ₄ |
| 68 | Ethene C ₂ H ₄ |
| 69 | Ethane C ₂ H ₆ |
| 70 | Propene C ₃ H ₆ |
| 71 | Propane C ₃ H ₈ |
| 72 | i-butane i-C ₄ H ₁₀ |
| 73 | n-butane n-C ₄ H ₁₀ |
| 74 | neo-pentane neo-C ₅ H ₁₂ |
| 75 | i-pentane i-C ₅ H ₁₂ |
| 76 | n-pentane n-C ₅ H ₁₂ |
| 77 | Hexane C ₆ H ₁₄ |
| 78 | Hexane share higher hydrocarbons C ₆ H ₁₄ % |
| 79 | Hexane+ C ₆ H ₁₄₊ |

| Value group D codes – Gas – Natural gas analysis values (A = 7, C = 70) | |
|---|---|
| 80 | Heptane C ₇ H ₁₆ |
| 81 | Octane C ₈ H ₁₈ |
| 82 | Nonane C ₉ H ₂₀ |
| 83 | Decane C ₁₀ H ₂₂ |
| 84 | Tetrahydrothiophene C ₄ H ₈ S |
| All other | Reserved |
| a | Superior (gross) Wobbe index |
| b | Inferior (net) Wobbe index |
| c | Inferior (net) calorific value |

7.8.4 Value group E codes – Gas

7.8.4.1 General

The following clauses define the use of value group E for identifying further classification or processing the measurement quantities defined by value groups A to D. The various classifications and processing methods are exclusive.

7.8.4.2 Indexes and index differences – Tariff rates

Table 92 shows the use of value group E for identification of tariff rates typically used for indexes and index differences of volume, mass and energy, specified in Table 87.

Table 92 – Value group E codes – Gas – Indexes and index differences – Tariff rates

| Value group E codes – Gas – Indexes and index differences – Tariff rates (A = 7, C = 1...8, 11...16, 21...26, 31...36, 61...66, D = 0...3, 6...98) | |
|---|-----------------------------|
| 0 | Total |
| 1 | Rate 1 |
| ... | |
| 63 | Rate 63 |
| | |
| 128...254 | Manufacturer specific codes |
| All other | Reserved |

7.8.4.3 Flow rate

No further classification in value group E are made. Therefore E shall be 0.

7.8.4.4 Process values

No further classification in value group E is made. Therefore, E shall be 0.

7.8.4.5 Conversion related factors and coefficients – Averages

Table 93 shows the use of value group E for the identification of average values of conversion related factors and coefficients – as specified in 7.8.3.5 – over various averaging periods.

Table 93 – Value group E codes – Gas – Conversion related factors and coefficients

| Value group E codes – Gas – Conversion related factors and coefficients– Averages (A = 7, C = 51...55, D = 0, 2, 3, 10, 11) | |
|---|--|
| 0 | Process independent current value ^a |
| 1 | Weighted value (e.g. Superior calorific value) ^b |
| 11 | Average, current interval, averaging period 1 (default 5 minutes) |
| 12 | Average, last interval, averaging period 1 (default 5 minutes) |
| 13 | Average, current interval, averaging period 2 (default 15 minutes) |
| 14 | Average, last interval, averaging period 2 (default 15 minutes) |
| 15 | Average, current interval, averaging period 3 (default 1 hour) |
| 16 | Average, last interval, averaging period 3 (default 1 hour) |
| 17 | Average, current interval, averaging period 4 (no default value) |
| 18 | Average, last interval, averaging period 4 (no default value) |
| 19 | Average, current interval, averaging period 5 (default 1 day) |
| 20 | Average, last interval, averaging period 5 (default 1 day) |
| 21 | Average, current interval, averaging period 6 (default 1 month) |
| 22 | Average, last interval, averaging period 6 (default 1 month) |
| 23 | Average, current interval, averaging period 7 (default 1 year) |
| 24 | Average, last interval, averaging period 7 (default 1 year) |
| 25 | Average, current interval, averaging period 8 (no default value) |
| 26 | Average, last interval, averaging period 8 (no default value) |
| 27 | Average, averaging period 9, interval since last event |
| 28 | Average, averaging period 10, interval between last two events |
| All other | Reserved |
| ^a | Process independent current value is a gas analysis technology independent value, which is generated asynchronous to processing cycles, but used for further calculations. |
| ^b | Weighted value is the result of specific algorithms taking into account different values by weighting their influence on the algorithm result. |

7.8.4.6 Calculation methods

Table 94 shows the use of value group E for the identification of calculation methods. See also 6.4.8.

Table 94 – Value group E codes – Gas – Calculation methods

| Value group E codes – Calculation methods (A = 7, C = 51...55, D = 12) | |
|---|---------------------------------|
| 0 | Calculation method in use |
| 1 | Calculation method 1 supported |
| ... | ... |
| 20 | Calculation method 20 supported |
| All other | Reserved |

7.8.4.7 Natural gas analysis values – Averages

Table 95 shows the use of value group E for the identification of natural gas analysis values – as specified in 7.8.3.6 – over various averaging periods.

Table 95 – Value group E codes – Gas – Natural gas analysis values – Averages

| Value group E codes – Gas – Natural gas analysis values – Averages (A = 7, C = 70, D = 8...20, 60...84) | |
|--|--|
| 0 | Process independent current value ^a |
| 1 | Weighted value (e.g. CO ₂ in [GJ / t]) ^b |
| | |
| 11 | Average, current interval, averaging period 1 (default 5 minutes) |
| 12 | Average, last interval, averaging period 1 (default 5 minutes) |
| 13 | Average, current interval, averaging period 2 (default 15 minutes) |
| 14 | Average, last interval, averaging period 2 (default 15 minutes) |
| 15 | Average, current interval, averaging period 3 (default 1 hour) |
| 16 | Average, last interval, averaging period 3 (default 1 hour) |
| 17 | Average, current interval, averaging period 4 (no default value) |
| 18 | Average, last interval, averaging period 4 (no default value) |
| 19 | Average, current interval, averaging period 5 (default 1 day) |
| 20 | Average, last interval, averaging period 5 (default 1 day) |
| 21 | Average, current interval, averaging period 6 (default 1 month) |
| 22 | Average, last interval, averaging period 6 (default 1 month) |
| 23 | Average, current interval, averaging period 7 (default 1 year) |
| 24 | Average, last interval, averaging period 7 (default 1 year) |
| 25 | Average, current interval, averaging period 8 (no default value) |
| 26 | Average, last interval, averaging period 8 (no default value) |
| 27 | Average, averaging period 9, interval since last event |
| 28 | Average, averaging period 10, interval between last two events |
| All other | Reserved |
| ^a | Process independent current value is a gas analysis technology independent value, which is generated asynchronous to processing cycles, but used for further calculations. |
| ^b | Weighted value is the result of specific algorithms taking into account different values by weighting their influence on the algorithm result. |

7.8.5 Value group F codes – Gas

Value group F identifies current (with F = 255) or historical values of quantities identified by value groups A to E, where appropriate.

There are four billing period schemes available (for example to store daily, monthly, yearly and weekly values). For each billing period scheme, the following general purpose objects are available:

- billing period counter;
- number of available billing periods;
- time stamp of most recent and historical billing periods;
- billing period length.

For OBIS codes see Table 96. For additional information, see Clauses 6.2.2 and 7.11.3.

7.8.6 OBIS codes – Gas

7.8.6.1 General and service entry objects – Gas

Table 96 specifies the OBIS codes for gas related general and service entry objects.

Table 96 – OBIS codes for general and service entry objects – Gas

| General and shows service entry objects – Gas | OBIS code | | | | | |
|--|-----------|-----|-----|-----|-----|-----------|
| | A | B | C | D | E | F |
| Free ID-numbers for utilities | | | | | | |
| Complete combined gas ID | 7 | b | 0 | 0 | | |
| Gas ID 1 | 7 | b | 0 | 0 | 0 | |
| ... | ... | ... | ... | ... | ... | |
| Gas ID 10 | 7 | b | 0 | 0 | 9 | |
| Billing period values / reset counter entries (First billing period scheme if there are more than one) | | | | | | |
| Billing period counter (1) | 7 | b | 0 | 1 | 0 | VZ or 255 |
| Number of available billing periods (1) | 7 | b | 0 | 1 | 1 | |
| Time stamp of the most recent billing period (1) | 7 | b | 0 | 1 | 2 | |
| Time stamp of the billing period (1) VZ (last reset) | 7 | b | 0 | 1 | 2 | VZ |
| Time stamp of the billing period (1) VZ-1 | 7 | b | 0 | 1 | 2 | VZ-1 |
| ... | | | ... | ... | ... | |
| Time stamp of the billing period (1) VZ-n | 7 | b | 0 | 1 | 2 | VZ-n |
| Billing period values / reset counter entries (Second billing period scheme) | | | | | | |
| Billing period counter (2) | 7 | b | 0 | 1 | 3 | VZ or 255 |
| Number of available billing periods (2) | 7 | b | 0 | 1 | 4 | |
| Time stamp of the most recent billing period (2) | 7 | b | 0 | 1 | 5 | |
| Time stamp of the billing period (2) VZ (last reset) | 7 | b | 0 | 1 | 5 | VZ |
| Time stamp of the billing period (2) VZ-1 | 7 | b | 0 | 1 | 5 | VZ-1 |
| ... | | | ... | ... | ... | |

| General and shows service entry objects – Gas | OBIS code | | | | | |
|--|------------------|----------|----------|----------|----------|-----------|
| | A | B | C | D | E | F |
| Time stamp of the billing period (2) VZ-n | 7 | b | 0 | 1 | 5 | VZ-n |
| Billing period values / reset counter entries (Third billing period scheme) | | | | | | |
| Billing period counter (3) | 7 | b | 0 | 1 | 6 | VZ or 255 |
| Number of available billing periods (3) | 7 | b | 0 | 1 | 7 | |
| Time stamp of the most recent billing period (3) | 7 | b | 0 | 1 | 8 | |
| Time stamp of the billing period (3) VZ (last reset) | 7 | b | 0 | 1 | 8 | VZ |
| Time stamp of the billing period (3) VZ-1 | 7 | b | 0 | 1 | 8 | VZ-1 |
| ... | | | ... | ... | ... | |
| Time stamp of the billing period (3) VZ-n | 7 | b | 0 | 1 | 8 | VZ-n |
| Billing period values / reset counter entries (Fourth billing period scheme) | | | | | | |
| Billing period counter (4) | 7 | b | 0 | 1 | 9 | VZ or 255 |
| Number of available billing periods (4) | 7 | b | 0 | 1 | 10 | |
| Time stamp of the most recent billing period (4) | 7 | b | 0 | 1 | 11 | |
| Time stamp of the billing period (4) VZ (last reset) | 7 | b | 0 | 1 | 11 | VZ |
| Time stamp of the billing period (4) VZ-1 | 7 | b | 0 | 1 | 11 | VZ-1 |
| ... | | | ... | ... | ... | |
| Time stamp of the billing period (4) VZ-n | 7 | b | 0 | 1 | 11 | VZ-n |
| Configuration | | | | | | |
| Program version | 7 | b | 0 | 2 | 0 | |
| Firmware version | 7 | b | 0 | 2 | 1 | |
| Software version | 7 | b | 0 | 2 | 2 | |
| Device version | 7 | b | 0 | 2 | 3 | |
| Active firmware signature | 7 | b | 0 | 2 | 8 | |
| Number of device channels | 7 | b | 0 | 2 | 10 | |
| Pressure sensor, serial no. | 7 | b | 0 | 2 | 11 | |
| Temperature sensor, serial no. | 7 | b | 0 | 2 | 12 | |
| Calculator, serial no. | 7 | b | 0 | 2 | 13 | |
| Volume sensor ^a , serial no. | 7 | b | 0 | 2 | 14 | |
| Density sensor, serial no. | 7 | b | 0 | 2 | 15 | |
| Sensor (medium irrespective), serial no. | 7 | b | 0 | 2 | 16 | |
| Digital output configuration | 7 | b | 0 | 2 | 17 | |
| Analogue output configuration | 7 | b | 0 | 2 | 18 | |
| Output pulse constants converted / unconverted | | | | | | |
| Volume forward at metering conditions | 7 | b | 0 | 3 | 0 | |
| Volume reverse at metering conditions | 7 | b | 0 | 3 | 1 | |
| Volume absolute ^b at metering conditions | 7 | b | 0 | 3 | 2 | |
| Volume forward at base conditions | 7 | b | 0 | 3 | 3 | |
| Volume reverse at base conditions | 7 | b | 0 | 3 | 4 | |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 454/492 |
|-----------------------|------------|-------------------------|---------|

| General and shows service entry objects – Gas | OBIS code | | | | | |
|---|------------------|----------|----------|----------|----------|----------|
| | A | B | C | D | E | F |
| Volume absolute ^b at base conditions | 7 | b | 0 | 3 | 5 | |
| Conversion factors | | | | | | |
| | 7 | b | 0 | 4 | 0 | |
| {This area is to be used for polynomials, constants for conversion, and similar} | 7 | b | 0 | 4 | 1 | |
| ... | 7 | b | 0 | 4 | 2 | |
| | 7 | b | 0 | 4 | 3 | |
| | 7 | b | 0 | 4 | 4 | |
| Threshold values | | | | | | |
| Threshold power for over-consumption relative to measurement period 2 for indexes and index differences | | | | | | |
| limit 1 | 7 | b | 0 | 5 | 1 | 1 |
| ... | | | ... | ... | ... | ... |
| limit 4 | 7 | b | 0 | 5 | 1 | 4 |
| Threshold power for over-consumption relative to measurement period 3 for indexes and index differences | | | | | | |
| limit 1 | 7 | b | 0 | 5 | 1 | 11 |
| ... | | | ... | ... | ... | ... |
| limit 4 | 7 | b | 0 | 5 | 1 | 14 |
| Threshold limit for rate 1 for over-consumption relative to measurement period 2 for indexes and index differences | 7 | b | 0 | 5 | 2 | 1 |
| ... | | | ... | ... | ... | ... |
| limit for rate 9 | 7 | b | 0 | 5 | 2 | 9 |
| Threshold limit for rate 1 for over-consumption relative to measurement period 3 for indexes and index differences | 7 | b | 0 | 5 | 2 | 11 |
| ... | | | ... | ... | ... | ... |
| limit for rate 9 | 7 | b | 0 | 5 | 2 | 19 |
| | | | | | | |
| Maximum contracted consumption for rec. interval 1 | 7 | b | 0 | 5 | 3 | |
| Maximum contracted consumption for rec. interval 2 | 7 | b | 0 | 5 | 4 | |
| Absolute temperature, minimum limit setting ^c | 7 | b | 0 | 5 | 11 | |
| Absolute temperature, maximum limit setting ^c | 7 | b | 0 | 5 | 12 | |
| Absolute pressure, minimum limit setting ^c | 7 | b | 0 | 5 | 13 | |
| Absolute pressure, maximum limit setting ^c | 7 | b | 0 | 5 | 14 | |
| Nominal values volume sensor | | | | | | |
| Pressure | 7 | b | 0 | 6 | 1 | |
| Temperature | 7 | b | 0 | 6 | 2 | |
| Q_{\min} | 7 | b | 0 | 6 | 3 | |
| Q_{\max} | 7 | b | 0 | 6 | 4 | |
| Input pulse constants | | | | | | |
| Volume forward at metering conditions | 7 | b | 0 | 7 | 0 | |
| Volume reverse metering conditions | 7 | b | 0 | 7 | 1 | |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 455/492 |
|-----------------------|------------|-------------------------|---------|

| General and shows service entry objects – Gas | OBIS code | | | | | |
|---|------------------|----------|----------|----------|----------|----------|
| | A | B | C | D | E | F |
| Volume absolute ^b at metering conditions | 7 | <i>b</i> | 0 | 7 | 2 | |
| Volume forward at base conditions | 7 | <i>b</i> | 0 | 7 | 3 | |
| Volume reverse at base conditions | 7 | <i>b</i> | 0 | 7 | 4 | |
| Volume absolute ^b at base conditions | 7 | <i>b</i> | 0 | 7 | 5 | |
| Intervals and periods | | | | | | |
| Recording interval 1, for profile ^d | 7 | <i>b</i> | 0 | 8 | 1 | |
| Recording interval 2, for profile ^d | 7 | <i>b</i> | 0 | 8 | 2 | |
| Measurement period 1, for average value 1 | 7 | <i>b</i> | 0 | 8 | 3 | |
| Measurement period 2, for average value 2 | 7 | <i>b</i> | 0 | 8 | 4 | |
| Measurement period 3, for instantaneous value | 7 | <i>b</i> | 0 | 8 | 5 | |
| Measurement period 4, for test value | 7 | <i>b</i> | 0 | 8 | 6 | |
| Billing period | 7 | <i>b</i> | 0 | 8 | 10 | |
| NOTE Codes 7.b.0.8.11...35 are newly defined in Blue Book Edition 9. | | | | | | |
| Process interval 1, default value 15 minutes | 7 | <i>b</i> | 0 | 8 | 11 | |
| Process interval 2, default value 1 hour | 7 | <i>b</i> | 0 | 8 | 12 | |
| Process interval 3, default value 1 day | 7 | <i>b</i> | 0 | 8 | 13 | |
| Process interval 4, default value 1 month | 7 | <i>b</i> | 0 | 8 | 14 | |
| Process interval 5, for process value, since last event | 7 | <i>b</i> | 0 | 8 | 15 | |
| Process interval 6, between last two events | 7 | <i>b</i> | 0 | 8 | 16 | |
| Measurement period 1, for indexes and index differences, default value 15 minutes | 7 | <i>b</i> | 0 | 8 | 17 | |
| Measurement period 2, for indexes and index differences, default value 1 hour | 7 | <i>b</i> | 0 | 8 | 18 | |
| Measurement period 3, for indexes and index differences, no default value | 7 | <i>b</i> | 0 | 8 | 19 | |
| Billing period 1, for indexes and index differences, default value 1 day | 7 | <i>b</i> | 0 | 8 | 20 | |
| Billing period 2, for indexes and index differences, default value 1 month | 7 | <i>b</i> | 0 | 8 | 21 | |
| Billing period 3, for indexes and index differences, default value 1 year, | 7 | <i>b</i> | 0 | 8 | 22 | |
| Billing period 4, for indexes and index differences, no default value | 7 | <i>b</i> | 0 | 8 | 23 | |
| Averaging period 1, default value 5 minutes | 7 | <i>b</i> | 0 | 8 | 25 | |
| Averaging period 2, default value 15 minutes | 7 | <i>b</i> | 0 | 8 | 26 | |
| Averaging period 3, default value 1 hour | 7 | <i>b</i> | 0 | 8 | 27 | |
| Averaging period 4, no default value | 7 | <i>b</i> | 0 | 8 | 28 | |
| Averaging period 5, default value 1 day | 7 | <i>b</i> | 0 | 8 | 29 | |
| Averaging period 6, default value 1 month | 7 | <i>b</i> | 0 | 8 | 30 | |
| Averaging period 7, default value 1 year | 7 | <i>b</i> | 0 | 8 | 31 | |
| Averaging period 8, no default value | 7 | <i>b</i> | 0 | 8 | 32 | |
| Averaging period 9, since last event | 7 | <i>b</i> | 0 | 8 | 33 | |
| Averaging period 10, between two last events | 7 | <i>b</i> | 0 | 8 | 34 | |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 456/492 |
|-----------------------|------------|-------------------------|---------|

| General and shows service entry objects – Gas | OBIS code | | | | | |
|---|------------------|----------|------------|----------|----------|----------|
| | A | B | C | D | E | F |
| Number of sub-periods for averaging period 2 | 7 | b | 0 | 8 | 35 | |
| Time entries | | | | | | |
| Number of days (time expired) since last reset (First billing period scheme if there are more than one) | 7 | b | 0 | 9 | 0 | |
| Local time | 7 | b | 0 | 9 | 1 | |
| Local date | 7 | b | 0 | 9 | 2 | |
| Start of conventional gas day | 7 | b | 0 | 9 | 3 | |
| Residual time shift ^e | 7 | b | 0 | 9 | 4 | |
| Time of last reset (First billing period scheme if there are more than one) | 7 | b | 0 | 9 | 6 | |
| Date of last reset (First billing period scheme if there are more than one) | 7 | b | 0 | 9 | 7 | |
| Clock time shift limit | 7 | b | 0 | 9 | 11 | |
| First billing period scheme | | | | | | |
| <i>Number of days (time expired) since last reset (end of billing period)</i> | | | See above. | | | |
| <i>Time of last reset</i> | | | See above. | | | |
| <i>Date of last reset</i> | | | See above. | | | |
| Billing period reset lockout time (First billing period scheme if there are more than one) | 7 | b | 0 | 9 | 12 | |
| Second billing period scheme | | | | | | |
| Number of days (time expired) since last end of billing period | 7 | b | 0 | 9 | 13 | |
| Time of last reset | 7 | b | 0 | 9 | 14 | |
| Date of last reset | 7 | b | 0 | 9 | 15 | |
| Billing period reset lockout time | 7 | b | 0 | 9 | 16 | |
| Third billing period scheme | | | | | | |
| Number of days (Time expired) since last end of billing period | 7 | b | 0 | 9 | 17 | |
| Time of last reset | 7 | b | 0 | 9 | 18 | |
| Date of last reset | 7 | b | 0 | 9 | 19 | |
| Billing period reset lockout time | 7 | b | 0 | 9 | 20 | |
| Fourth billing period scheme | | | | | | |
| Number of days (time expired) since last end of billing period | 7 | b | 0 | 9 | 21 | |
| Time of last reset | 7 | b | 0 | 9 | 22 | |
| Date of last reset | 7 | b | 0 | 9 | 23 | |
| Billing period reset lockout time | 7 | b | 0 | 9 | 24 | |
| Station management information objects | | | | | | |
| Heating temperature ^f , current value | 7 | b | 0 | 10 | 0 | |
| Heating temperature, average 15 minutes | 7 | b | 0 | 10 | 1 | |
| Heating temperature, average 60 minutes | 7 | b | 0 | 10 | 11 | |
| Heating temperature, average day | 7 | b | 0 | 10 | 21 | |
| Heating temperature, average month | 7 | b | 0 | 10 | 31 | |

| General and shows service entry objects – Gas | OBIS code | | | | | |
|--|------------------|----------|----------|----------|----------|----------|
| | A | B | C | D | E | F |
| Ambient device temperature ⁹ , current value | 7 | b | 0 | 11 | 0 | |
| Ambient device temperature, average 15 minutes | 7 | b | 0 | 11 | 1 | |
| Ambient device temperature, average 60 minutes | 7 | b | 0 | 11 | 11 | |
| Ambient device temperature, average day | 7 | b | 0 | 11 | 21 | |
| Ambient device temperature, average month | 7 | b | 0 | 11 | 31 | |
| Gas parameters for volume conversion, currently used in compressibility calculation | | | | | | |
| Reference pressure of gas analysis | 7 | b | 0 | 12 | 8 | |
| Reference temperature of gas analysis | 7 | b | 0 | 12 | 9 | |
| Superior Wobbe number 0 °C | 7 | b | 0 | 12 | 10 | |
| Inferior Wobbe number 0 °C | 7 | b | 0 | 12 | 11 | |
| Methane number | 7 | b | 0 | 12 | 12 | |
| Total sulphur | 7 | b | 0 | 12 | 13 | |
| Hydrogen sulphide H ₂ S | 7 | b | 0 | 12 | 14 | |
| Mercaptans | 7 | b | 0 | 12 | 15 | |
| Water dew point (DP H ₂ O) | 7 | b | 0 | 12 | 16 | |
| Water (H ₂ O) dew point outlet / normalised | 7 | b | 0 | 12 | 17 | |
| Hydrocarbon dew point (DP C _x H _y) | 7 | b | 0 | 12 | 18 | |
| Inferior calorific value H _{i,n} | 7 | b | 0 | 12 | 19 | |
| Water H ₂ O | 7 | b | 0 | 12 | 20 | |
| Density (of gas), base conditions | 7 | b | 0 | 12 | 45 | |
| Relative density | 7 | b | 0 | 12 | 46 | |
| Superior calorific value H _{s,n} | 7 | b | 0 | 12 | 54 | |
| Nitrogen N ₂ | 7 | b | 0 | 12 | 60 | |
| Hydrogen H ₂ | 7 | b | 0 | 12 | 61 | |
| Oxygen O ₂ | 7 | b | 0 | 12 | 62 | |
| Helium He | 7 | b | 0 | 12 | 63 | |
| Argon Ar | 7 | b | 0 | 12 | 64 | |
| Carbon monoxide CO | 7 | b | 0 | 12 | 65 | |
| Carbon dioxide CO ₂ | 7 | b | 0 | 12 | 66 | |
| Methane CH ₄ | 7 | b | 0 | 12 | 67 | |
| Ethene C ₂ H ₄ | 7 | b | 0 | 12 | 68 | |
| Ethane C ₂ H ₆ | 7 | b | 0 | 12 | 69 | |
| Propene C ₃ H ₆ | 7 | b | 0 | 12 | 70 | |
| Propane C ₃ H ₈ | 7 | b | 0 | 12 | 71 | |
| i-butane i-C ₄ H ₁₀ | 7 | b | 0 | 12 | 72 | |
| n-butane n-C ₄ H ₁₀ | 7 | b | 0 | 12 | 73 | |
| neo-pentane neo-C ₅ H ₁₂ | 7 | b | 0 | 12 | 74 | |
| i-pentane i-C ₅ H ₁₂ | 7 | b | 0 | 12 | 75 | |
| n-pentane n-C ₅ H ₁₂ | 7 | b | 0 | 12 | 76 | |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 458/492 |
|-----------------------|------------|-------------------------|---------|

| General and shows service entry objects – Gas | OBIS code | | | | | |
|---|------------------|----------|----------|----------|----------|----------|
| | A | B | C | D | E | F |
| Hexane C ₆ H ₁₄ | 7 | b | 0 | 12 | 77 | |
| Hexane share higher hydrocarbons C ₆ H ₁₄ % | 7 | b | 0 | 12 | 78 | |
| Hexane+ C ₆ H ₁₄₊ | 7 | b | 0 | 12 | 79 | |
| Heptane C ₇ H ₁₆ | 7 | b | 0 | 12 | 80 | |
| Octane C ₈ H ₁₈ | 7 | b | 0 | 12 | 81 | |
| Nonane C ₉ H ₂₀ | 7 | b | 0 | 12 | 82 | |
| Decane C ₁₀ H ₂₂ | 7 | b | 0 | 12 | 83 | |
| Tetrahydrothiophene | 7 | b | 0 | 12 | 84 | |
| Gas parameters for Venturi measurement | | | | | | |
| Internal pipe diameter | 7 | b | 0 | 13 | 1 | |
| Orifice diameter | 7 | b | 0 | 13 | 2 | |
| Pressure type (orifice fitting) | 7 | b | 0 | 13 | 3 | |
| Flow coefficient (alfa) | 7 | b | 0 | 13 | 4 | |
| Expansion coefficient (epsilon) | 7 | b | 0 | 13 | 5 | |
| Reflux coefficient | 7 | b | 0 | 13 | 6 | |
| Isoentropic coefficient | 7 | b | 0 | 13 | 7 | |
| Dynamic viscosity | 7 | b | 0 | 13 | 8 | |
| Differential pressure dp for cut off | 7 | b | 0 | 13 | 9 | |
| Reynold number | 7 | b | 0 | 13 | 10 | |
| Gas parameters for density measurement | | | | | | |
| K0 Densimeter Coefficient | 7 | b | 0 | 14 | 1 | |
| K2 Densimeter Coefficient | 7 | b | 0 | 14 | 2 | |
| Densimeter period for instantaneous measurement | 7 | b | 0 | 14 | 10 | |
| Densimeter period for measurement period 15 minutes | 7 | b | 0 | 14 | 11 | |
| Sensor manager | | | | | | |
| Sensor manager objects | 7 | b | 0 | 15 | e | |
| Internal operating status, gas related | | | | | | |
| Internal operating status, global ^h | 7 | b | 96 | 5 | 0 | |
| Internal operating status (status word 1) ^h | 7 | b | 96 | 5 | 1 | |
| Internal operating status (status word 2) ^h | 7 | b | 96 | 5 | 2 | |
| Internal operating status (status word 3) ^h | 7 | b | 96 | 5 | 3 | |
| Internal operating status (status word 4) ^h | 7 | b | 96 | 5 | 4 | |
| Internal operating status (status word 5) ^h | 7 | b | 96 | 5 | 5 | |
| Internal operating status (status word 6) ^h | 7 | b | 96 | 5 | 6 | |
| Internal operating status (status word 7) ^h | 7 | b | 96 | 5 | 7 | |
| Internal operating status (status word 8) ^h | 7 | b | 96 | 5 | 8 | |
| Internal operating status (status word 9) ^h | 7 | b | 96 | 5 | 9 | |
| Manufacturer specific ⁱ⁾ | 7 | b | 96 | 50 | e | |
| | | | | | | |
| Manufacturer specific | 7 | b | 96 | 99 | e | |

| General and shows service entry objects – Gas | | OBIS code | | | | | |
|--|--|------------------|---|---|---|---|---|
| | | A | B | C | D | E | F |
| a | A volume sensor could be an external mechanical meter / encoder / electronic index. | | | | | | |
| b | Absolute in the sense that negative volume is summed as positive ABS(). | | | | | | |
| c | An absolute temperature or absolute pressure outside these limits may affect the error status of the device. | | | | | | |
| d | If multiple recording intervals are implemented, then recording interval 1 shall be the shorter. | | | | | | |
| e | This value indicates the remaining time interval for soft time setting, where the clock is corrected in small steps (equivalent to Clock object method 6). | | | | | | |
| f | Temperature heating is applied by stations with gas heating systems. | | | | | | |
| g | Application for control of battery environment or volume conversion device environmental control. | | | | | | |
| h | Status words referring to a status table with fix status words or to any status table bits using mapped status (class_id = 63). | | | | | | |
| i | The range D = 50..99 is available for identifying objects, which are not represented by another defined code, but need representation on the display as well. If this is not required, the range D = 128...254 should be used. | | | | | | |

7.8.6.2 Error register objects – Gas

Table 97 specifies the OBIS codes for gas related error register objects.

Table 97 – OBIS codes for error register objects – Gas

| Error register objects – Gas | OBIS code | | | | | |
|---|------------------|---|----|----|---|---|
| | A | B | C | D | E | F |
| Error registers | 7 | b | 97 | 97 | e | |
| NOTE The information to be included in the error objects is not defined in this document. | | | | | | |

7.8.6.3 List objects – Gas

Table 98 specifies the OBIS codes for gas related list objects.

Table 98 – OBIS codes for list objects – Gas

| List objects – Gas | OBIS code | | | | | |
|--|--|---|----|----|---|------------------|
| | A | B | C | D | E | F |
| Gas related data of billing period (with billing period scheme 1 if there are more than one schemes available) | 7 | b | 98 | 1 | e | 255 ^a |
| Gas related data of billing period (with billing period scheme 2) | 7 | b | 98 | 2 | e | 255 ^a |
| Gas related data of billing period (with billing period scheme 3) | 7 | b | 98 | 3 | e | 255 ^a |
| Gas related data of billing period (with billing period scheme 4) | 7 | b | 98 | 4 | e | 255 ^a |
| Gas related data of event triggered billing profile ^b | 7 | b | 98 | 11 | e | 255 ^a |
| ^a | F = 255 means a wildcard here. See 7.11.3. | | | | | |
| ^b | Event triggered means the termination of a billing period by events, e.g. by commands. (Therefore, the profile entries are not equidistant in time). | | | | | |

7.8.6.4 Data profile objects – Gas

Gas related data profiles – identified with one single OBIS code – are used to hold a series of measurement values of one or more similar quantities and/or to group various data. The OBIS codes are specified in Table 99.

Table 99 – OBIS codes for data profile objects – Gas

| Data profile objects – Gas | OBIS code | | | | | |
|---|---|---|----|----------------|----------------|---|
| | A | B | C | D | E | F |
| Load profile with recording interval 1 | 7 | b | 99 | 1 | 4 ^a | |
| Load profile with recording interval 2 | 7 | b | 99 | 2 | 4 ^a | |
| Profile of maxima with recording interval 1 | 7 | b | 99 | 3 | 4 ^a | |
| Profile of maxima with recording interval 2 | 7 | b | 99 | 4 | 4 ^a | |
| Load profiles for indexes and index differences of volume, mass and energy ^b | 7 | b | 99 | d ^c | e ^d | |
| Load profiles for process values | 7 | b | 99 | d ^e | e ^f | |
| Load profiles for flow rate | 7 | b | 99 | 43 | e ^g | |
| Power failure event log | 7 | b | 99 | 97 | e | |
| Event log | 7 | b | 99 | 98 | e | |
| Certification data log | 7 | b | 99 | 99 | 0 | |
| Load profile with recording interval 15 minutes | 7 | b | 99 | 99 | 1 | |
| Load profile with recording interval 60 minutes | 7 | b | 99 | 99 | 2 | |
| Load profile with recording interval day | 7 | b | 99 | 99 | 3 | |
| Load profile with recording interval month | 7 | b | 99 | 99 | 4 | |
| ^a | The value in value group E has been changed from 0 to 4 to avoid overlaps with the self-descriptive profile OBIS codes. The use of the value 0 is deprecated. | | | | | |
| ^b | Value group D and E identify the value captured in these profiles. Value group D and E of the OBIS code of the load profile is mapped to value group C and D of the OBIS code identifying the value captured. The value captured in the buffer is always attribute 2 (value) of the respective Register / Extended register object. | | | | | |
| ^c | The possible values are 1...8, 11...16, 21...26, 31...36, 61...66. See Table 86. | | | | | |
| ^d | The possible values are 0...3, 6...98. See Table 87. EXAMPLE A load profile with OBIS code 7.b.99.11.17.255 contains the logged values from a volume conversion device: Forward undisturbed converter volume, index difference, value at base conditions, relative to measurement period 2. The values are captured at the end of each measurement period (last values). | | | | | |
| ^e | The possible values are 41, 42, 44...49. See Table 86. | | | | | |
| ^f | The possible values are 0, 2, 13, 24...32, 42...50, 60...68, 78...86, 90...92. See Table 89. EXAMPLE A load profile with OBIS code 7.b.99.41.43.255 contains the logged values of absolute gas temperature, average, last interval, (relative to) process interval 2. | | | | | |
| ^g | The possible values are 0, 1, 2, 13, 19...22, 39...42, 59...62, 67...70. See Table 88. EXAMPLE A load profile with OBIS code 7.b.99.43.19.255 contains the logged values of the flow rate, last average for averaging period 1, value at metering conditions. | | | | | |

7.9 Water (Value group A = 8 and A = 9)

7.9.1 General

This subclause 7.9 specifies the naming of objects carrying water meter information in a COSEM environment. It covers the handling of hot, as well as the handling of cold water.

7.9.2 Value group C codes – Water

Table 100 specifies the use of value group C for hot and cold water.

Table 100 – Value group C codes – Water

| Value group C codes – Water (A=8 or A=9) | |
|--|---|
| 0 | General purpose objects |
| 1 | Accumulated volume |
| 2 | Flow rate |
| 3 | Forward temperature |
| 93 | Consortia specific identifiers, see Table 54. |
| 94 | Country specific identifiers, see Table 55. |
| 96 | General and service entry objects – Water (See 7.9.4.1) |
| 97 | Error register objects – Water (See 7.9.4.2) |
| 98 | List objects – Water |
| 99 | Data profile objects – Water (See 7.9.4.3) |
| 128...199, 240 | Manufacturer specific codes |
| All other | Reserved |

7.9.3 Value group D codes – Water

This value group specifies the result of processing a *Quantity* according to a specific algorithm for water related values. See Table 101

Table 101 – Value group D codes – Water

| Value group D codes – Water (A = 8 or A = 9, C <> 0, 96...99) | |
|---|--------------------|
| 0 | Current value |
| 1 | Periodical value |
| 2 | Set date value |
| 3 | Billing date value |
| 4 | Minimum of value |
| 5 | Maximum of value |
| 6 | Test value |
| All other | Reserved |

7.9.4 OBIS codes – Water

7.9.4.1 General and service entry objects – Water

Table 102 specifies the OBIS codes for water related general and service entry objects.

Table 102 – OBIS codes for general and service entry objects – Water

| General and service entry objects – Water | OBIS code | | | | | |
|--|---|---|-----|-----|-----|---|
| | A | B | C | D | E | F |
| Free ID-numbers for utilities | | | | | | |
| Complete combined ID | 8/9 | b | 0 | 0 | | |
| ID 1 | 8/9 | b | 0 | 0 | 0 | |
| ... | | | ... | ... | ... | |
| ID 10 | 8/9 | b | 0 | 0 | 9 | |
| Storage information | | | | | | |
| Status (VZ) of the historical value counter | 8/9 | b | 0 | 1 | 1 | |
| Number of available historical values | 8/9 | b | 0 | 1 | 2 | |
| Due date | 8/9 | b | 0 | 1 | 10 | |
| Billing date | 8/9 | b | 0 | 1 | 11 | |
| Billing date period | 8/9 | b | 0 | 1 | 12 | |
| Program Entries | | | | | | |
| Program version no. | 8/9 | b | 0 | 2 | 0 | |
| Device version no. | 8/9 | b | 0 | 2 | 3 | |
| Threshold values | | | | | | |
| Contracted maximum consumption | 8/9 | b | 0 | 5 | 1 | |
| Input pulse constants | | | | | | |
| Volume forward | 8/9 | b | 0 | 7 | 1 | |
| Measurement-/registration-period duration | | | | | | |
| Recording interval for load profile | 8/9 | b | 0 | 8 | 1 | |
| Time integral, averaging period for actual flow rate value | 8/9 | b | 0 | 8 | 6 | |
| Time entries | | | | | | |
| Local time | 8/9 | b | 0 | 9 | 1 | |
| Local date | 8/9 | b | 0 | 9 | 2 | |
| Time stamp (local time) of the most recent billing period ^a | 8/9 | b | 0 | 9 | 3 | |
| Manufacturer specific ^b | 8/9 | b | 96 | 50 | e | f |
| | | | | | | |
| Manufacturer specific | 8/9 | b | 96 | 99 | e | f |
| ^a | In case of billing period schemes absence or event triggered, commonly calculated from local date and local time information. | | | | | |
| ^b | The range D = 50...99 is available for identifying objects, which are not represented by another defined code, but need representation on the display as well. If this is not required, the range D = 128...254 should be used. | | | | | |

7.9.4.2 Error register objects – Water

Table 103 specifies the OBIS codes for water related error register objects.

Table 103 – OBIS codes for error register objects – Water

| Error register objects – Water | OBIS code | | | | | |
|---|-----------|---|----|----|---|---|
| | A | B | C | D | E | F |
| Error registers | 8/9 | b | 97 | 97 | e | |
| NOTE The information to be included in the error objects is not defined in this document. | | | | | | |

7.9.4.3 Data profile objects – Water

Water related data profiles – identified with one single OBIS code – are used to hold a series of measurement values of one or more similar quantities and/or to group various data. The OBIS codes are specified in Table 104.

Table 104 – OBIS codes for data profile objects – Water

| Data profile objects – Water | OBIS code | | | | | |
|------------------------------|-----------|---|----|---|---|---|
| | A | B | C | D | E | F |
| Consumption/load profile | 8/9 | b | 99 | 1 | e | |

7.9.4.4 OBIS codes for water related objects (examples)

Table 105 – OBIS codes for water related objects (examples)

Table 105 specifies examples for OBIS codes of water related objects.

| Water related objects | OBIS code | | | | | |
|--|-----------|---|---|---|---|------------------|
| | A | B | C | D | E | F |
| Consumption | | | | | | |
| Current index, total | 8/9 | b | 1 | 0 | 0 | |
| Current index, tariff 1 | 8/9 | b | 1 | 0 | 1 | |
| Current index, periodical, total, the two last periods | 8/9 | b | 1 | 1 | 0 | 102 |
| Monitoring values | | | | | | |
| Flow rate, maximum value, previous period | 8/9 | b | 2 | 5 | 0 | V _{Z-1} |
| Forward temperature, billing date value, last billing period | 8/9 | b | 3 | 3 | 0 | 101 |

7.10 Other media (Value group A= 15)

7.10.1 General

This subclause 7.10 specifies naming of objects related to other media than what is defined with values A = 1, 4...9. Typical application is distributed energy generation using renewable energy sources.

NOTE The details of OBIS codes will be specified as application of DLMS/COSEM in this area grows.

7.10.2 Value group C codes – Other media

Table 106 specifies the use of value group C for other media.

Table 106 – Value group C codes – Other media

| Value group C codes – Other media | |
|-----------------------------------|-----------------------------|
| 0 | General purpose objects |
| 1...10 | Solar |
| 11...20 | Wind |
| | |
| 128...254 | Manufacturer specific codes |
| All other | Reserved |

7.10.3 Value group D codes – Other media

To be specified later.

7.10.4 Value group E codes – Other media

To be specified later.

7.10.5 Value group F codes – Other media

To be specified later.

7.11 Code presentation

7.11.1 Reduced ID codes (e.g. for IEC 62056-21)

To comply with the syntax defined for protocol modes A to D of IEC 62056-21:2002, the range of ID codes is reduced to fulfil the limitations which usually apply to the number of digits and their ASCII representation. Values in all value groups are limited to a range of 0...99 and within that range, to the values specified in the clauses specifying the use of the value groups.

Some value groups may be suppressed, if they are not relevant to an application:

- optional value groups: A, B, E, F;
- mandatory value groups: C, D.

To allow the interpretation of shortened codes delimiters are inserted between all value groups, see Figure 36:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| A | - | B | : | C | . | D | . | E | * | F |
|---|---|---|---|---|---|---|---|---|---|---|

IEC 304/02

Figure 36 – Reduced ID code presentation

The delimiter between value groups E and F can be modified to carry some information about the source of a reset (& instead of * if the reset was performed manually).

The manufacturer shall ensure that the combination of the OBIS code and the class_id (see Clause 4) uniquely identifies each COSEM object.

7.11.2 Display

The usage of OBIS codes to display values is normally limited in a similar way as for data transfer, for example according to IEC 62056-21:2002.

Some codes in value group C and D may be replaced by letters to clearly indicate the differences from other data items:

Table 107 – Example of display code replacement

| Value group C and D | |
|---------------------|--------------|
| OBIS code | Display code |
| 96 | C |
| 97 | F |
| 98 | L |
| 99 | P |

NOTE The letter codes may also be used in protocol modes A to D.

7.11.3 Special handling of value group F

Unless otherwise specified, the value group F is used for the identification of values of billing periods.

The billing periods can be identified relative to the status of the billing period counter or relative to the current billing period.

For electricity, there are two billing period schemes available in Table 68, each scheme defined by the length of the billing period, the billing period counter, the number of available billing periods and the time stamps of the billing period. See also 6.2.2 and 7.5.4.1.

For gas, there are four billing period schemes available, see Table 96.

With $0 \leq F \leq 99$, a single billing period is identified relative to the value of the billing period counter, VZ. If the value of the value group of any OBIS code is equal to VZ, this identifies the most recent (youngest) billing period. VZ₋₁ identifies the second youngest, etc. The billing period counter may have different operating modes, for example modulo-12 or modulo-100. The value after reaching the limit of the billing period counter is 0 for the operating mode modulo-100 and 1 for other operating modes (for example modulo-12).

With $101 \leq F \leq 125$, a single billing period or a set of billing periods are identified relative to the current billing period. F=101 identifies the last billing period, F = 102 the second last / two last billing periods, etc., F = 125 identifies the 25th last / 25 last billing periods.

F = 126 identifies an unspecified number of last billing periods, therefore it can be used as a wildcard.

F=255 means that the value group F is not used, or identifies the current billing period value(s).

For use of ICs for representing values of historical billing periods, see 6.2.2.

Table 108 – Value group F – Billing periods

| Value group F | |
|------------------|------------------------------------|
| VZ | Most recent value |
| VZ ₋₁ | Second most recent value |
| VZ ₋₂ | Third most recent value |
| VZ ₋₃ | Fourth most recent value |
| VZ ₋₄ | ... |
| etc. | |
| 101 | Last value |
| 102 | Second / two last value(s) |
| | |
| 125 | 25 th /25 last value(s) |
| 126 | Unspecified number of last values |

7.11.4 COSEM

The usage of OBIS codes in the COSEM environment shall be as defined in Clause 6.

Annex A (Informative)

Additional information on Auto answer and Auto connect ICs

NOTE This information is related to the "Auto answer" (class_id = 28, version = 2, see 4.7.5) and "Auto connect" (class_id = 29, version = 2, see 4.7.6) interface classes.

Since the capabilities (e.g. connection time, number of parallel connections) of communication networks (e.g. GPRS) are limited, devices e.g. meters are not permanently connected to the communication network.

Devices may connect to the network in regular intervals or on special events either to send unsolicited data or just to become accessible.

If a DLMS/COSEM client e.g. a Head End System needs to access a server e.g. a meter that is not connected to the communication network a wake-up request can be sent. This may be a wake-up call or a wake-up message, e.g. an SMS message. After successfully processing the wake-up request the device connects to network.

Figure A. 1 below shows an example for a GSM/GPRS communication network. Please note that the dashed lines represent the network services, the solid lines refer to possible application layer services.

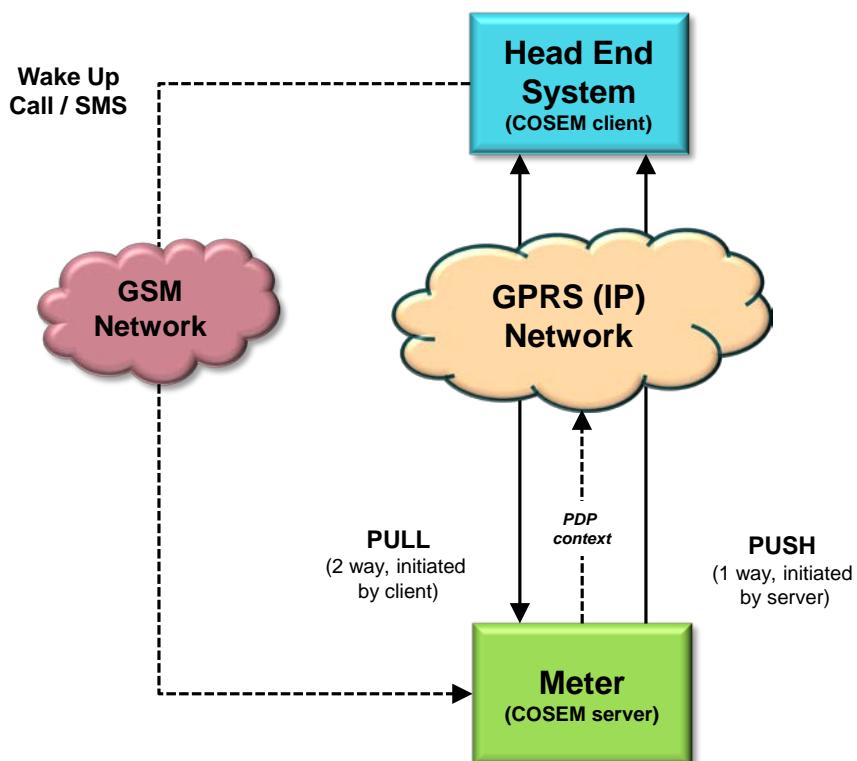


Figure A. 1 – Network connectivity example for a GSM/GPRS network

The basic network connectivity in the case of a mobile network (GPRS or equivalent service) is modelled by the "Auto connect" IC. Depending on the mode the connection can be 'always on', 'always on in a time window' or 'only on after a wake-up'. If the device is connected to the network it has the PDP context attached and it is accessible from by the HES via its IP address. If necessary the current IP address of the server (meter) can be sent to the client (HES) using the DataNotification xDLMS service.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 468/492 |
|-----------------------|------------|-------------------------|---------|

The wake-up process is modelled using an instance of the “Auto answer” IC which provides additional security (check calling number) compared to today's solution.

Please also note that the “Auto answer” class is fully decoupled from the xDLMS application layer services. The main reason is to have a clear separation between the communication layers as well as to avoid creating an unsecured backdoor to execute application layer services with almost no protection. The execution of xDLMS services via SMS should be handled by sending ciphered xDLMS APDUs in a pre-established AA.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 469/492 |
|-----------------------|------------|-------------------------|---------|

**Annex B
(Informative)**
Additional information to M-Bus client (class_id = 72, version 1)

State transitions of the *encryption_key_status* attribute for different use cases are shown in Figure B. 1.



Figure B. 1 – Encryption key status diagram

State transitions of the *encryption_key_status* attribute for different use cases are given below. At the time of installation of the slave four cases are possible:

- 1) Encryption key is preset in the slave and cannot be changed, see Table B. 1;
- 2) Encryption key is preset in the slave and new key is set after installation, see Table B. 2;
- 3) Encryption key is not preset in the slave, but can be set, see Table B. 3 and Table B. 4;
- 4) The slave is used without encryption. In this case, the *encryption_key_status* stays in state (0).

Table B. 1 – Encryption key is preset in the slave and cannot be changed

| Step | State | Condition for state transition (event or successfully invoked method) |
|------|-------|--|
| 1 | (2) | set_encryption_key |
| 2 | (3) | encrypted telegrams successfully exchanged |
| 3 | (4) | – |

Table B. 2 – Encryption key is preset in the slave and new key is set after installation

| Step | State | Condition for state transition (event or successfully invoked method) |
|------|-------|--|
| 1 | (2) | set_encryption_key |
| 2 | (3) | transfer_key |
| 3 | (2) | set_encryption_key |
| 4 | (3) | encrypted telegrams successfully exchanged |
| 5 | (4) | – |

Table B. 3 – Encryption key is not preset in the slave, but can be set, case a)

| Step | State | Condition for state transition (event or successfully invoked method) |
|------|-------|--|
| 1a | (0) | set_encryption_key |
| 2a | (1) | transfer_key |
| 3a | (3) | encrypted telegrams successfully exchanged |
| 4a | (4) | – |

Table B. 4 – Encryption key is not preset in the slave, but can be set, case b)

| Step | State | Condition for state transition (event or successfully invoked method) |
|------|-------|--|
| 1b | (0) | transfer_key |
| 2b | (2) | set_encryption_key |
| 3b | (3) | encrypted telegrams successfully exchanged |
| 4b | (4) | – |

Annex C (Informative)

Additional information on IPv6 setup class (class_id = 48, version = 0)

C.1 Introduction

In most regards, IPv6 is a conservative extension of IPv4. Most transport and application-layer protocols need little or no change to operate over IPv6; exceptions are application protocols that embed internet-layer addresses, such as FTP or NTPv3.

IPv6 specifies a new packet format, designed to minimize packet-header processing. Since the headers of IPv4 packets and IPv6 packets are significantly different, the two protocols are not interoperable.

C.2 IPv6 addressing

The most important feature of IPv6 is a much larger address space than that of IPv4: addresses in IPv6 are 128 bits long, compared to 32-bit addresses in IPv4. Furthermore, compared to IPv4, IPv6 supports multi-addressing on one physical interface (global, unique or link local IPv6 addresses).

IPv6 addresses are typically composed of two logical parts: a 64-bit (sub-)network prefix used for routing, and a 64-bit host part used to identify a host within the network.

The formats allowed for an IPv6 address are shown in Figure C. 1 (see <http://www.iana.org/assignments/ipv6-address-space/>). Note that to facilitate the IPv6 address writing, a specific notation defined in RFC 4291 has been specified by IETF (e.g. FF00::/8).

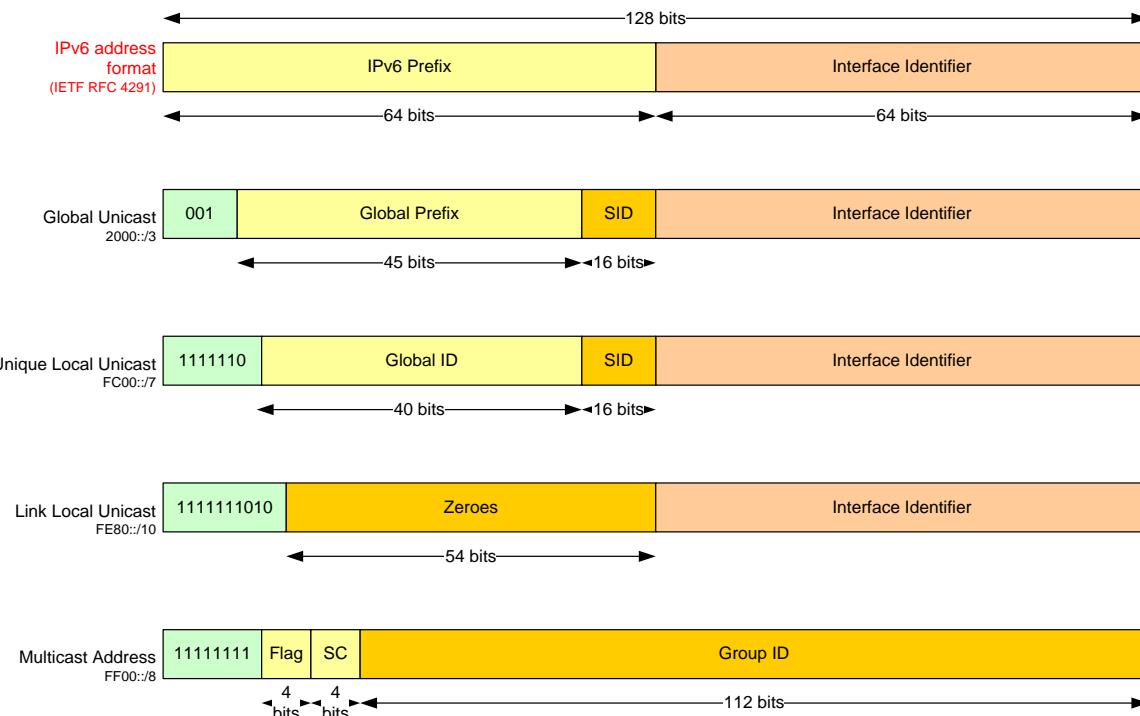


Figure C. 1 – IPv6 address formats

Where:

- *Global Unicast* is a routable address in the whole internet network and is composed as follows:

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 472/492 |
|-----------------------|------------|-------------------------|---------|

- Global prefix assigned by IANA (see <http://www.iana.org/assignments/ipv6-unicast-address-assignments/>);
- Subnet ID (SID) allocated by the network administrator; and
- Interface Identifier either generated from the interface's MAC address (using modified EUI-64 format), or obtained from a DHCPv6 server, or assigned manually;
- *Unique Local Unicast* is an address only applicable to local network. This type of address is not routable outside the local network. The Global ID and the Subnet ID (SID) are allocated by the network administrator;
- *Link Local Unicast* is a unicast address allowed for a link local (without router). This type of address is not routable outside a local link;
- *Multicast* is an address assigned to different devices of the network. Following the scope (SC) of the address, the multicast group may be either Interface-local, Link-local, Admin-local, Site-local, Organization-local or global. For more information about Flag and SC (scope) parameters, see RFC 4291, 2.7.

It is important to note that there is no broadcast address defined in IPv6.

For more information concerning IPv6 addressing, see RFC 4291.

C.3 IPv6 header format

As defined in RFC 2460 clause 3, the header of one IPv6 packet is composed as shown in Figure C. 2:

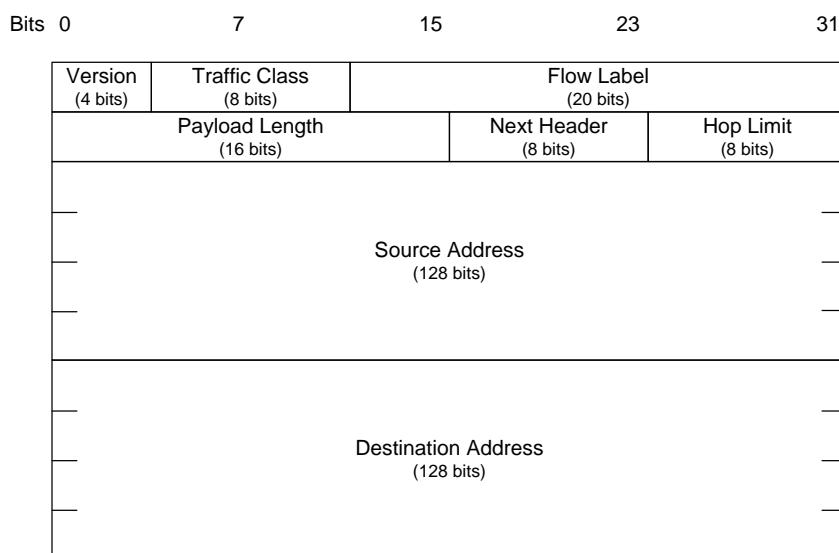
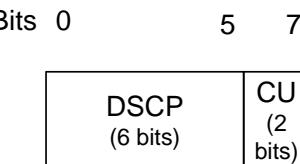


Figure C. 2 – IPv6 header format

Where:

- *Version* specifies the version of the protocol. For IPv6, the value is fixed and equals to 6;
- *Traffic class* is used by originating nodes and/or forwarding routers to identify and distinguish between different classes or priorities of IPv6 packets (see RFC 2474 Clause 3). Note that the traffic class value is only a notion of prioritization used to distribute the IPv6 frame and do not secure the transmission (the philosophy of the IP network is to do its best).

Figure C. 3 shows the content of the traffic class parameter:

**Figure C. 3 – Traffic class parameter format**

Where:

- DSCP - Differentiated services code point contains the prioritization of the IPv6 packet in the network (see RFC 2474 for details);
- CU – Currently unused;
- *Flow label* may be used by a source to label sequences of packets for which it requests special handling by the IPv6 routers, such as non-default quality of service or "real-time" service. Currently, no fully definition of this field is given by IETF and may be seen as reserved for future use;
- *Payload length* indicates the size of the upper layer payload carried by the IPv6 frame;
- *Next header* identifies the type of header immediately following the IPv6 header of the current frame. It may indicate an upper applicative header (ICMP, UDP, TCP...) or extensions;
- *Hop limit* defines the time of life of the IPv6 packet in term of number of hop. The usage is similar to the one defined for IPv4 (decremented by 1 by each node that forwards the packet, the packet is discarded if Hop Limit is decremented to zero);
- *Source and destination addresses* indicate the originator of the IPv6 packet and the intended recipient.

Table C. 1 summarizes the IPv6 headers managed / not managed by the IPv6 objects.

Table C. 1 – IPv6 header vs. IPv6 IC

| Parameters | IPv6 setup IC |
|----------------------------------|---------------|
| Version | Not managed |
| Traffic class | Managed |
| Flow label | Not managed |
| Payload length | Not managed |
| Next header | Not managed |
| Hop limit | Not managed |
| Source and destination addresses | Not managed |

C.4 IPv6 header extensions

C.4.1 Overview

Contrary to IPv4, IPv6 provides an extensible header by adding optional elements one by one. Currently, the following list of optional headers is defined by RFC 2460, see Table C. 1.

Table C. 2 summarizes the options managed / not managed by the IPv6 setup objects.

Table C. 2 – Optional IPv6 header extensions vs. IPv6 IC

| Extensions | IPv6 setup IC | See clause |
|--|---------------|------------|
| Hop-by-Hop options | Not managed | C.4.2 |
| Destination options | Not managed | C.4.3 |
| Routing options | Not managed | C.4.4 |
| Fragment options | Not managed | C.4.5 |
| Security options (Authentication and Encapsulating Security Payload) | Not managed | C.4.6 |
| NOTE The options are listed in the order as they appear in the packet. | | |

C.4.2 Hop-by-Hop options

The Hop-by-Hop options field must be examined by all devices on the path. Four elements currently defined may compose this header (see RFC 2460, 4.3):

- A padding form Pad1 which introduces one byte of padding (see RFC 2460, 4.2);
- A padding form PadN which introduces more than 2 bytes of padding (see RFC 2460, 4.2);
- A Jumbogram field to indicate the length of the IPv6 datagram in case of jumbo payload (higher than the maximum size of length field of IPv6 header) (see RFC 2460 and RFC 2675); and
- A router alert: The option indicates that the contents of the datagram may be interesting to the router (see RFC 2711).

These optional elements are directly managed by the IPv6 protocol stack. Therefore, they are not managed by the COSEM IPv6 setup class.

C.4.3 Destination options

The Destination options field must be examined only by the target device of the IP datagram. Four elements are currently defined by IETF (see RFC 2460, 4.6):

- A padding form Pad1 which introduces one byte of padding (see RFC 2460, 4.2);
- A padding form PadN which introduces more than 2 bytes of padding (see RFC 2460, 4.2);
- Mobility parameters (linked to new IP wireless networks – UMTS, LTE...) (see RFC 3775); and
- Tunnelling options which permit to communicate between two IPv6 networks separated by an IPv4 network (6To4 mechanism) (see RFC 2473).

NOTE The Mobility parameters and the Tunnelling options have been defined separately from RFC 2460 and are not mentioned in this document. See appropriate RFC's for more details.

These optional elements are directly managed by the IPv6 protocol stack. These optional fields are not managed by the COSEM IPv6 setup class.

C.4.4 Routing options

Routing options element is used to specify some routing parameters (see RFC 2460, 4.4). Currently, only one type is defined by IETF (type 2), which is used in case of mobility IPv6.

This notion is out of scope of the COSEM IPv6 setup class.

For information, due to an important security issue (see RFC 5095), the type 0 previously defined has been deprecated and is forbidden. Furthermore, the type 1 has been temporarily defined for experimental Nimrod and is obsolete since 1996.

C.4.5 Fragment options

Fragment options field is used in the case of fragmentation of the applicative payload if its size is higher than the MTU of the network (see RFC 2460, 4.5).

This element is directly managed by the IPv6 protocol stack. These optional fields are not managed by the COSEM IPv6 setup class.

C.4.6 Security options

Contrary to IPv4, IPv6 protocol layer includes natively the IPsec protocol. These extensions are fully defined in the RFC 4302 and RFC 4303.

The security options are composed of two extensible headers:

- Authentication Header (AH): Contains information used to verify the authenticity of most parts of the packet (see RFC 4302, Clause 2);
- Encapsulating Security Payload (ESP): Carries encrypted data for secure communication (see RFC 4303, Clause 2).

Due to the complexity of the IPsec protocol, the configuration of IPsec is out of scope of this document and must be treated separately.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 476/492 |
|-----------------------|------------|-------------------------|---------|

Annex D (Informative)

Overview of the narrow-band OFDM PLC technology for PRIME networks

For the specification of the PRIME narrow-band OFDM PLC setup classes, see 4.12.

NOTE This technology is supported by the PRIME Alliance, <http://www.prime-alliance.org>.

ITU-T G.9904:2012 specifies a physical layer, a medium access control layer and convergence layers for cost-effective narrowband (<200 kbps) data transmission over electrical power lines, intended for use in smart metering and smart grid applications. It is based on Orthogonal Frequency Division Multiplexing (OFDM).

The specification currently describes the following:

- a low-cost PHY capable of achieving rates of encoded 128 kbps;
- a Master-Slave MAC optimised for the power line environment;
- a convergence layer for the LLC layer specified in IEC 61334-4-32;
- a convergence layer for IPv4;
- a convergence layer for IPv6;

CLC/TS 52056-8-4:2015 specifies the DLMS/COSEM narrowband OFDM PLC profile for PRIME networks.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 477/492 |
|-----------------------|------------|-------------------------|---------|

Annex E (informative)

Overview of the narrow-band OFDM PLC technology for G3-PLC networks

For the specification of the G3 narrow-band OFDM PLC setup classes, see 4.13.

NOTE This specification is supported by the G3-PLC Alliance, <http://www.G3-PLC.com>.

ITU-T G.9903:2014 specifies the physical, MAC and 6LoWPAN Adaptation layers of the G3-PLC technology while ITU-T G.9901:2014 deals with frequency bandplan allocation and associated transmission level limitations.

Power line communication has been used for many decades, but a variety of new services and applications require more reliability and higher data rates. However, the power line channel is very hostile. Channel characteristics and parameters vary with frequency, location, time and the type of equipment connected to it. The lower frequency regions from 10 kHz to 200 kHz are especially susceptible to interference. Furthermore, the power line is a very frequency selective channel. Besides background noise, it is subject to impulsive noise often occurring at 50/60 Hz and group delays up to several hundred microseconds.

G3-PLC uses advanced modulation and channel coding techniques, which enables efficient use of the limited bandwidth of the CENELEC bands and facilitates communication over the power line channel. This combination enables a very robust communication in the presence of narrow-band interference, impulsive noise, and frequency selective attenuation. The specification addresses the following main objectives:

- provide robust communication on extremely harsh power line channels;
- provide a minimum of 20 kbps effective data rate in the normal mode of operation;
- ability to notch selected frequencies, to allow the cohabitation with other Narrow-band PLC communication technologies (e.g. IEC 61334-5-1:2001 S-FSK) or to be compliant with specific regulatory requirements;
- dynamic tone adaptation capability to select frequencies on the channel that do not have major interference, thereby ensuring a robust communication;
- access control, authentication, confidentiality and integrity to ensure high level of security.

To this end, the G3-PLC protocol stack aggregates several layers and sub-layers that form the G3-PLC profile:

- a robust high-performance PHY layer based on OFDM and adapted to the narrow-band PLC environment;
- a MAC layer of the IEEE 802.15.4 type (extended), well suited to low data rates;
- IPv6, the new generation of IP (Internet Protocol), which widely opens the range of potential applications and services; and
- to allow good IPv6 and MAC interoperability, an Adaptation sublayer taken from the Internet world (IETF) and called 6LoWPAN (RFC 4944 extended and RFC 6282). The adaptation sub layer also embeds the LOADng routing algorithm to allow multi-hop mesh connectivity.

For more information about the extensions of IEEE 802.15.4, RFC 4944 and RFC 6282 (AKA 6LoWPAN) standards, and LOADng, see ITU-T G.9903:2014.

CLC/TS 52056-8-5:2015 specifies the DLMS/COSEM narrowband OFDM PLC profile for G3-PLC networks.

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 478/492 |
|-----------------------|------------|-------------------------|---------|

Annex F (informative) Bibliography

| | |
|-------------------------------|---|
| IEC 61334-6:2000 | <i>Distribution automation using distribution line carrier systems – Part 6: A-XDR encoding rule</i> |
| IEC 62056-7-6:2013 | <i>Electricity metering data exchange – The DLMS/COSEM suite – Part 7 6: The 3-layer, connection-oriented HDLC based communication profile</i> |
| IEC 62056-8-3:2013 | <i>Electricity metering data exchange – The DLMS/COSEM suite – Part 8 3: Communication profile for PLC S-FSK neighbourhood networks</i> |
| IEC 62056-9-7:2013 | <i>Electricity metering data exchange – The DLMS/COSEM suite – Part 9-7: Communication profile for TCP-UDP/IP networks</i> |
| ISO 12213-3 | <i>Natural gas – Calculation of compression factor – Part 3: Calculation using physical properties</i> |
| EN 12405-1 | <i>Gas meters - Conversion devices - Part 1: Volume conversion</i> |
| EN 12405-2 | <i>Gas meters - Conversion devices - Part 2: Energy conversion</i> |
| ITU Recommendation X.217:1995 | <i>Information technology – Open Systems Interconnection – Service definition for the association control service element</i> |
| ITU Recommendation X.227:1995 | <i>Information technology – Open Systems Interconnection Connection-oriented protocol for the association control service element: Protocol specification</i> |
| DIN 43863-3:1997 | <i>Electricity meters – Part 3: Tariff metering device as additional equipment for electricity meters – EDIS – Energy Data Identification System</i> |
| RFC 793 | <i>Transmission Control Protocol (Also IETF STD 0007), 1981, Updated by: RFC 3168. http://www.ietf.org/rfc/rfc793.txt</i> |
| RFC 940 | <i>Toward an Internet Standard Scheme for Subnetting, 1985. http://www.ietf.org/rfc/rfc940.txt</i> |
| RFC 950 | <i>Internet Standard Subnetting Procedure, 1985. http://www.ietf.org/rfc/rfc950.txt</i> |
| RFC 2460 | <i>Internet Protocol, Version 6 (IPv6), 1998. http://www.ietf.org/rfc/rfc2460.txt</i> |
| RFC 2473 | <i>Generic Packet Tunneling in IPv6, 1998. http://www.ietf.org/rfc/rfc2473.txt</i> |
| RFC 2675 | <i>IPv6 Jumbograms, 1999. http://www.ietf.org/rfc/rfc2675.txt</i> |
| RFC 2711 | <i>IPv6 Router Alert Option, 1999. http://www.ietf.org/rfc/rfc2711.txt</i> |
| RFC 3775 | <i>Mobility Support in IPv6, 2004. http://www.ietf.org/rfc/rfc3775.txt</i> |
| RFC 4291 | <i>IP Version 6 Addressing Architecture, 2006. http://www.ietf.org/rfc/rfc4291.txt</i> |
| RFC 4302 | <i>IP Authentication Header, 2005. http://www.ietf.org/rfc/rfc4302.txt</i> |
| RFC 4303 | <i>IP Encapsulating Security Payload (ESP), 2005. http://www.ietf.org/rfc/rfc4303.txt</i> |
| RFC 4944 | <i>Transmission of IPv6 Packets over IEEE 802.15.4 Networks, 2007. http://www.ietf.org/rfc/rfc4944.txt</i> |
| RFC 5095 | <i>Deprecation of Type 0 Routing Headers in IPv6, 2007. http://www.ietf.org/rfc/rfc5095.txt</i> |
| RFC 6282 | <i>Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks, 2011. http://www.ietf.org/rfc/rfc6282.txt</i> |

Index

| | |
|--|--|
| 61334-4-32 LLC SSCS setup | 253 |
| Abstract object | 367, 391, 392 |
| ac_max_ctl_re_tx | 255 |
| Access | 397 |
| Access rights | 49 |
| Access selector | 298 |
| access_number | 209 |
| access_right | 307 |
| access_rights_list | 84, 86, 300, 303 |
| Account | 29, 161, 163, 353, 359 |
| account_activation_time | 170 |
| account_closure_time | 170 |
| account_mode_and_status | 165 |
| acknowledgement_time_t1 | 248 |
| acknowledgement_timer | 247 |
| actions | 143, 149, 156 |
| activate_account | 172 |
| activate_passive | 142 |
| activate_passive_unit_charge | 188 |
| Active energy | 413, 414, 415 |
| Active initiator | 26, 239 |
| Active power | 378, 402, 415 |
| active_devices | 284 |
| active_initiator | 237 |
| active_mask | 65 |
| Activity calendar | 139, 140, 353, 357 |
| Actor, Arbitrator | 155 |
| add_mask | 65 |
| add_object | 93, 311, 316, 322 |
| add_register | 65 |
| addr_state | 207, 212 |
| address_config_mode | 222 |
| address_mask | 212 |
| adjacent_cells | 205 |
| adjust_to_measuring_period | 134 |
| adjust_to_minute | 134 |
| adjust_to_preset_time | 134 |
| adjust_to_quarter | 134 |
| adjustment_method | 153 |
| adp_broadcast_log_table | 276 |
| adp_broadcast_log_table_entry_TTL | 274 |
| adp_max_hops | 272 |
| adp_max_join_wait_time | 277 |
| adp_path_discovery_time | 277 |
| adp_routing_table | 275 |
| adp_weak_LQI_value | 272 |
| Advanced power failure monitoring | 370 |
| AES-GCM-128 | 103 |
| AES-GCM-256 | 103 |
| aggregated_debt | 167 |
| Agreed_key | 121 |
| alarm | 340 |
| <i>Alarm descriptor</i> | 109, 374, 400 |
| Alarm filter | 374, 400 |
| Alarm monitor | 109, 358 |
| <i>Alarm register</i> | 109, 374, 400 |
| Alert condition | 431 |
| Algorithm | 378, 379 |
| All configured | 236 |
| All Physical | 236 |
| always repeater | 235, 336 |
| amount_to_clear | 166 |
| Ampere-squared hours | 402, 409, 414 |
| Angles | 402 |
| APN | 203 |
| Apparent energy | 414, 415 |
| Apparent power | 378, 402, 415 |
| Application context | 49 |
| application_context_name | 90, 308, 314, 319 |
| aps_interframe_delay | 283 |
| aps_max_window_size | 283 |
| Arbitrator | 18, 34, 155, 369 |
| associated_partners_id | 89, 308, 313, 318 |
| Association | 353, 365 |
| Association LN | 41, 82, 307, 312, 317 |
| Association object | 250 |
| Association SN | 40, 41, 82, 295, 297, 299, 302 |
| Association view | 49 |
| association_status | 91, 310, 315, 320 |
| Attribute | 21, 39, 41, 42, 93 |
| Authenticated request | 102, 123 |
| Authenticated response | 102, 123 |
| Authentication | 297 |
| Authentication key | 105 |
| Authentication mechanism | 49 |
| authentication_mechanism_name | 91, 310, 315, 320 |
| Average value | 414, 445, 448 |
| Averaging period | 443, 456 |
| Averaging scheme | 405 |
| A-XDR | 479 |
| backup_HAN | 287 |
| Base conditions, gas | 429 |
| Base node | 27, 251 |
| base_name | 40, 83, 297, 300, 303 |
| base_node_address | 253 |
| Basic/nominal current | 414 |
| Battery | 370, 397 |
| Beacon slot | 27 |
| Billing period | 354, 357, 376, 382, 396, 401, 406, 411, 413, 414, 425, 435, 438, 440, 460, 466 |
| Billing period counter | 355, 396, 413, 466 |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 480/492 |
|-----------------------|------------|-------------------------|---------|

| | |
|---|------------------------------|
| Billing period for indexes and index differences | 456 |
| Billing period scheme, Gas..... | 457 |
| Billing period values / reset counter entries | 453 |
| Billing period, end of | 358 |
| Block demand | 61, 64 |
| <i>Broadcast</i> | 357 |
| Broadcast address..... | 193, 239, 327 |
| broadcast_frames | 216, 243 |
| buffer | 67, 71, 292 |
| bus_address | 207 |
| busy_state_timer | 247 |
| CAD | 28 |
| Calculation methods, Gas | 387, 451 |
| Calendar | 140 |
| Calendar day | 435 |
| Calendar month | 435 |
| calendar_name | 141 |
| Calibration | 397 |
| CALLING device address, | 193 |
| Calling line identification | 200 |
| calling_window | 202, 332, 333 |
| Calorific value | 429, 430, 432 |
| capture | 70, 73, 293, 341 |
| Capture period..... | 368, 373, 376, 379, 382 |
| capture_definition | 208, 339 |
| capture_method | 77 |
| capture_objects | 68, 292 |
| capture_period | 68, 293, 339 |
| capture_time | 61, 63, 217 |
| Captured object | 376, 379, 382 |
| Cardinality | 41 |
| cell_info | 205 |
| Certificate | 103, 116 |
| Certificate Signing Request..... | 107 |
| Certificate, export | 107 |
| Certificate, import | 107 |
| Certification data..... | 417 |
| Certification data log, Gas | 461 |
| Challenge..... | 299, 311 |
| change_HLS_secret 93, 296, 299, 311, 316, 322 | |
| change_LLS_secret | 296, 299 |
| change_secret | 86, 302, 306 |
| changed_parameter | 150 |
| Channel | 392 |
| channel_Id | 216 |
| CHAP protocol..... | 229 |
| CHAP_algorithm..... | 228 |
| Charge | 30, 161, 183, 353, 359 |
| Charge collection history | 401 |
| charge_configuration | 186 |
| charge_reference_list | 167 |
| charge_type | 184 |
| Christmas..... | 139 |
| CIASE | 234, 335 |
| class_id | 41, 88, 307, 312, 317 |
| clearance_threshold | 167 |
| Client user identification..... | 82 |
| client_SAP | 89, 308, 313, 318 |
| client_system_title | 103, 323 |
| Clock | 51, 132, 292, 353, 356 |
| <i>Clock synchronization method</i> | 377 |
| Clock time shift limit | 415 |
| clock_base | 134 |
| close_tunnel_timeout | 290 |
| Coefficient | 415 |
| Cold water | 391 |
| collect | 30 |
| comm_speed | 193, 195, 197, 212, 326, 329 |
| Commodity | 30, 32 |
| Communication channel | 389 |
| Communication port | 399 |
| Communication port log parameter | 372 |
| Communication tamper event..... | 373 |
| communication_window | 113 |
| Compact data..... | 18, 75, 353, 367 |
| compact_buffer | 76 |
| Compressibility..... | 386, 431, 434, 448 |
| Compressibility calculation | 458 |
| Compressibility factor..... | 432 |
| Compressibility, correction and conversion values – Averages | 450 |
| Configuration | 209, 368, 454 |
| Configuration data..... | 41 |
| Configuration program | 397, 413 |
| Conformance block | 90, 309, 314, 319 |
| connect_logical_device | 94 |
| Consortia specific390, 392, 393, 403, 418, 423, 437, 462 | |
| Consumer message..... | 372, 399 |
| consumption_based_collection | 162 |
| consumption_based_credit | 161 |
| Context specific..... | 392 |
| Contracted value | 376, 405 |
| control_mode | 147 |
| control_state | 147 |
| Convergence layer, PRIME NB OFDM PLC | 477 |
| Conversion | 386 |
| Conversion factor | 383, 432, 455 |
| Converted value | 438, 443 |
| Converter | 436 |
| Corrected value | 438, 443 |
| Corrected volume | 432 |
| Correction | 386 |
| Correction factor | 432 |
| COSEM Conformance Test Process | 21 |
| COSEM Glossary of Terms | 21 |
| COSEM logical device name | 48, 49 |
| COSEM Object Identification System (OBIS) | 389 |
| COSEM objects, Abstract | 352 |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 481/492 |
|-----------------------|------------|-------------------------|---------|

| | |
|--|--|
| COSEM objects, Electricity..... | 376 |
| COSEM objects, Gas | 382 |
| COSEM physical device | 230, 250 |
| Country specific | 390, 392, 393, 403, 418, 423, 437, 462 |
| CRC_NOK_frames_ | 217, 243 |
| CRC_OK_frames_ | 217, 243 |
| create_HAN | 289 |
| Credit | 161, 172, 177, 353, 359 |
| credit mode | 30 |
| Credit states | 173 |
| credit token | 33 |
| Credit, emergency | 30 |
| Credit, enabled | 173 |
| Credit, exhausted | 173 |
| Credit, in use | 173 |
| Credit, priority | 173 |
| Credit, selectable | 173 |
| Credit, selected/invoked | 173 |
| credit_available_threshold | 180 |
| credit_charge_configuration | 168 |
| credit_configuration | 178 |
| credit_reference_list | 167 |
| credit_status | 179 |
| credit_type | 177 |
| cs_attachment | 204 |
| CT/VT ratio | 377 |
| Cumulative maximum | 404 |
| Cumulative minimum | 404 |
| Cumulative value | 376 |
| currency | 170 |
| Current | 402 |
| Current and last average values | 376 |
| Current association..... | 49 |
| Current average..... | 404, 405, 443 |
| current_average_value..... | 61, 63 |
| current_credit_amount | 177 |
| current_credit_in_use | 165 |
| current_credit_status | 165 |
| current_user | 85, 92, 305, 321 |
| Currently active tariff..... | 372 |
| Cuts | 417 |
| Data51, 56, 73, 142, 368, 370, 373, 377, 378, 380, 383 | |
| Data access error | 297 |
| Data format..... | 42, 44 |
| Data logger | 430 |
| Data model | 21 |
| Data profile | 435 |
| Data profile objects – Abstract | 401 |
| Data profile objects – Electricity | 416 |
| Data profile objects – HCA | 421 |
| Data profile objects – Water | 464 |
| Data protection , COSEM | 115, 118, 353, 366 |
| Data security, access | 50 |
| Data security, transport | 50 |
| Data type | 42, 43 |
| data_index | 68, 111, 114 |
| data_send | 341 |
| Date and time format | 44 |
| day_profile_table | 142 |
| Daylight saving..... | 132, 139 |
| daylight_savings_deviation | 134 |
| daylight_savings_begin | 133 |
| daylight_savings_end | 133 |
| Default encryption key | 342 |
| default_baud | 191, 206, 325 |
| default_mode | 191, 324 |
| delete | 138, 140 |
| delete_mask | 65 |
| Delimiters | 466 |
| delta_electrical_phase | 233, 334 |
| Demand register | 51, 61, 64, 65, 376 |
| Demotion..... | 28 |
| Density measurement | 459 |
| destination_list | 203, 333 |
| Desynchronization | 239 |
| desynchronization-listing | 242 |
| Device ID | 367, 397 |
| Device start-up..... | 254 |
| device_addr | 192, 325 |
| device_address | 193, 212, 327 |
| device_serial_number..... | 196 |
| device_type | 209, 214, 340 |
| DHCP | 221 |
| Digital signature, profile entry | 19, 372 |
| Digitally signed request..... | 102, 123 |
| Digitally signed response | 102, 123 |
| Dips | 417 |
| Direction of the gas flow | 436 |
| Disconnect control | 145, 356, 358, 369 |
| Disconnected state | 27 |
| DiscoverReport request | 245 |
| Display | 466 |
| Display code | 466 |
| Disturbance registers | 431 |
| DL_reference | 219, 222 |
| DLMS version | 90, 309, 314, 319 |
| Domain Name Server | 221 |
| Duration | 404 |
| dynamic repeater | 235, 336 |
| ECDH P-384 | 103 |
| ECDSA P-256 | 103 |
| Electrical port | 360 |
| Electricity..... | 391, 404, 406 |
| Electricity breaker | 369 |
| Electricity ID | 413 |
| Emergency activation time | 148 |
| Emergency credit | 30 |
| Emergency duration | 148 |
| Emergency profile | 148 |
| Emergency profile, Limiter | 149 |
| Emergency threshold | 148 |
| emergency_credit | 161 |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 482/492 |
|-----------------------|------------|-------------------------|---------|

| | | | |
|------------------------------------|---|--|-------------------------|
| emergency_profile | 149 | Frequency notching, G3-PLC | 478 |
| emergency_profile_active | 149 | Friendly credit | 30 |
| emergency_profile_group | 149 | Full Function Device | 263 |
| enable/disable | 137 | Fundamental | 378, 379 |
| enable_disable_ | 283 | Fundamental component | 378 |
| Enabled | 30 | G3-PLC | 478 |
| Encoder | 436 | G3-PLC 6LoWPAN adaptation layer | 365 |
| Encrypted request | 102, 123 | G3-PLC 6LoWPAN adaptation layer setup | 264, 272 |
| Encrypted response | 102, 123 | G3-PLC MAC layer counters | 263, 265, 365 |
| Encryption key | 93, 209, 299, 311, 316, 322, 338, 342 | G3-PLC MAC setup | 264, 266, 365 |
| encryption_key_status | 209 | Gas | 391, 429 |
| End of billing period | 415 | Gas analysis | 429, 436 |
| Energy calculation | 429, 432 | Gas day | 435 |
| Energy conversion | 432 | Gas ID number | 382 |
| Energy measurement | 429 | Gas law deviation | 386 |
| Enterprise Resource Planning | 30 | Gas month | 435 |
| Entries, schedule | 137 | Gas station management | 383 |
| Entries, special days | 139 | Gas transport | 430 |
| entries_in_use | 69, 293 | Gas valve | 369 |
| Environment | 398 | Gas year | 435 |
| Environmental related parameters | 371 | gateway_IP_address | 221 |
| Ephemeral Unified Model | 106 | General and service entry objects | 396 |
| equipment_id | 196 | General and service entry objects – Electricity | 413 |
| Error correction | 431 | General and service entry objects – Gas | 453 |
| Error register | 360, 373, 374, 392, 400, 403, 416, 418, 421, 423, 427, 437, 462, 464 | General and service entry objects – HCA | 420 |
| Error registers – Abstract | 400 | General and service entry objects – Thermal energy | 425 |
| Error registers – Electricity | 416 | General and service entry objects – Water | 463 |
| Error registers – Gas | 460 | General purpose object | 402, 406 |
| Error registers – HCA | 421 | generate_certificate_request | 107 |
| Error registers – Water | 464 | generate_key_pair | 106 |
| Event code | 371, 398 | get_attributes&methods | 298 |
| Event counter | 372, 399 | get_attributes&services | 296 |
| Event log | 375, 401, 417 | get_buffer_by_index | 294 |
| Event log, Gas | 461 | get_buffer_by_range | 294 |
| Excess consumption metering | 414 | get_HLS_challenge | 297 |
| execute | 136 | get_protected_attributes | 117, 123, 124 |
| executed_script | 144 | getlist_by_classid | 295 |
| execution_time | 144 | getobj_by_logicalname | 295 |
| export_certificate | 107, 108 | Global broadcast encryption key | 105 |
| Extended register | 60, 64, 65, 73, 142, 152, 361, 370, 371, 376, 377, 380, 383, 384, 385 | Global Positioning System | 134 |
| extended_pan_ID | 280 | Global unicast encryption key | 105 |
| Fatal error register | 196 | global_broadcast_encryption_key | 122 |
| fatal_error | 328 | global_key_transfer | 324 |
| Firmware | 355 | global_unicast_encryption_key | 122 |
| Firmware identifier | 396 | GPRS modem setup | 203, 353, 362 |
| Firmware signature | 396 | GSM diagnostic | 204, 353, 362 |
| Firmware version | 396 | GSM diagnostic profile | 401 |
| Floating point format | 46 | GSM field strength | 399 |
| Flow rate | 386, 450 | Harmonics | 378, 379, 381, 407, 417 |
| Flow rate, gas | 442 | Hayes standard | 198, 329 |
| Forgotten system | 239 | HDLC mode | 193, 326 |
| Frame synchronization | 239 | HDLC setup | 353, 360 |
| frequencies | 234, 335 | HDLC setup class | 192, 325 |
| Frequency | 402, 414 | Head End System | 30 |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 483/492 |
|-----------------------|------------|-------------------------|---------|

| | |
|---|------------------------------|
| Heat / cooling related objects | 428 |
| Heat cost allocator | 391, 418 |
| High resolution clock | 356 |
| Historical index | 439 |
| Historical values | 389 |
| HLS secret | 93, 299, 311, 316, 322 |
| Hot water | 391 |
| I/O control signal..... | 368 |
| ID numbers, Electricity..... | 376 |
| Identification number | 338 |
| Identification system | 388 |
| identification_number | 209, 214, 340 |
| Identified_key | 121 |
| identify_device | 288 |
| IEC 61334-4-32 LLC setup..... | 363 |
| IEC local port setup | 353 |
| IEC twisted pair (1) Fatal Error register..... | 361 |
| IEC twisted pair (1) MAC address setup | 361 |
| IEC twisted pair (1) setup | 18, 194, 195, 327, 360, 361 |
| IEEE address..... | 28 |
| IHD | 28 |
| Image..... | 25, 98 |
| <i>Image activation</i> | 356, 358 |
| Image transfer | 82, 94, 95, 353, 366 |
| <u>Image, activation</u> | 101 |
| <u>Image, completeness</u> | 98 |
| <u>Image, initiate transfer</u> | 98 |
| image_activate | 97 |
| image_block_size | 95 |
| image_block_transfer | 97 |
| image_first_not_transferred_block_number | 95 |
| image_to_activate_infos | 96 |
| image_transfer_enabled | 95 |
| image_transfer_initiate | 96 |
| image_transfer_status | 96 |
| image_transferred_blocks_status | 95 |
| image_verify | 97 |
| ImageBlock | 25 |
| ImageBlockNumber | 25 |
| <u>ImageBlocks, transfer</u> | 98 |
| ImageBlockSize | 25, 97 |
| ImageSize | 25 |
| import_certificate | 107 |
| In Home Display | 28 |
| Inactive objects | 375, 392 |
| inactivity_time_out | 193, 219, 327 |
| Index difference, Gas | 385, 438, 450 |
| Index reading | 429 |
| Index, Gas | 385, 438, 450 |
| Information security | 50 |
| Information security, COSEM object | 50 |
| Information, measured..... | 39 |
| Information, static | 39 |
| initialization_string | 197, 329 |
| Initiator | 26, 236 |
| initiator_electrical_phase | 233, 334 |
| initiator_mac_address | 236, 337 |
| Input control signals | 397 |
| Input pulse constant | 383, 414, 455 |
| Input pulse values | 414 |
| Input/output control signals | 397 |
| insert | 138, 140 |
| Install code | 28 |
| Instances | 41 |
| Instantaneous value | 376, 381, 404, 414, 445 |
| Instantiation | 39, 41, 56, 57, 142, 291 |
| Intelligent search initiator | 234, 238, 335 |
| inter_octet_time_out | 193, 326 |
| Interface class | 39, 51, 53 |
| Interface object | 39 |
| Internal control signals | 369, 397 |
| Internal operating status | 369, 380, 385, 397, 415, 459 |
| Internet | 362 |
| Interval | 383 |
| Invocation counter | 366 |
| invoke_credit | 181 |
| <i>invoke_protected_method</i> | 117, 124 |
| IP_address | 220 |
| IP_options | 220 |
| IP_reference | 218 |
| IPCP_options | 227 |
| IPv4 setup | 219, 353, 362 |
| IPv6 addressing | 472 |
| IPv6 header | 473 |
| IPv6 setup | 353, 362 |
| ISO/IEC 8802-2 LLC layer setup | 353 |
| ISO/IEC 8802-2 LLC Type 1 setup | 246, 363 |
| ISO/IEC 8802-2 LLC Type 2 setup | 246, 364 |
| ISO/IEC 8802-2 LLC Type 3 setup | 248, 364 |
| Key agreement | 103 |
| Key information | 126 |
| key_agreement | 106 |
| key_info | 121 |
| key_transfer | 105 |
| Last average | 404, 405, 443 |
| Last value | 467 |
| <i>last_average_value</i> | 61, 63 |
| last_collection_amount | 187 |
| last_collection_time | 187 |
| last_outcome | 158 |
| LCP_options | 225 |
| Letter codes | 466 |
| Limit | 467 |
| Limiter | 147, 353, 359 |
| Line loss | 402 |
| Line reactance losses | 415 |
| Line resistance losses | 415 |
| Link key | 28 |
| link_key | 281 |
| List objects – Abstract | 392, 400 |
| List objects – Electricity | 380, 416 |

| | | | |
|--|--|--|--|
| List objects – Gas | 388, 460 | Max credit limit | 359 |
| List objects – General | 374 | Max vend limit | 359 |
| list_of_ | 200 | max_frame_length | 244, 337 |
| listening_window | 199, 215, 330 | max_info_length_transmit | 193 |
| LLC layer | 244, 337 | max_info_length_receive | 193, 326 |
| LLC sub-layer | 230 | max_info_length_transmit | 326 |
| LLS secret | 86, 299, 302, 306 | max_number_transmissions_n2 | 247 |
| LLS_secret | 310 | max_number_transmissions_n4 | 248 |
| Load limiting | 31 | max_octets_acn_pdu_n3 | 248 |
| Load profile | 401, 406, 414, 417 | max_octets_i_pdu_n1 | 247 |
| Local date | 396, 415 | max_octets_ui_pdu | 246 |
| Local port setup | 191, 324, 359 | max_pdu_size | 214 |
| Local time | 396, 415 | max_provision | 171 |
| local_disconnect | 146 | max_receiving_gain | 234, 335 |
| local_reconnect | 146 | max_transmitting_gain | 234, 335 |
| Logger | 436 | Maximum | 293, 438, 445 |
| Logical device | 41, 48, 49, 82, 87, 94, 297, 299, | Maximum and minimum values | 376 |
| 302, 312, 317 | | Maximum current | 414 |
| Logical Device Name.... | 40, 230, 250, 353, 366 | Maximum of index differences | 439 |
| Logical name | 39, 40, 41 | Maximum of last averages | 443 |
| Logical name referencing | 87, 312, 317 | Maximum Segment Size | 218 |
| logical_name | 39, 56, 95 | maximum_incoming_transfer_size | 290 |
| login_password | 229 | maximum_outcoming_transfer_size | 290 |
| low_credit_threshold | 171 | M-Bus | 353 |
| MAC address | 28, 196, 237 | M-Bus client | 338, 342, 344, 347, 361, 470 |
| MAC address setup | 353, 362, 364 | M-Bus control log object | 361 |
| MAC sub-layer | 230 | M-Bus diagnostics | 361 |
| mac_address | 225, 234, 335 | M-Bus disconnect control object | 361 |
| mac_coord_short_address | 344 | M-Bus master port setup | 212, 361 |
| mac_freq_notching | 345 | M-Bus port setup | 338 |
| mac_get_neighbour_table_entry | 347 | M-Bus profile generic object | 338, 361 |
| mac_group_addresses | 235, 336 | M-Bus slave | 338, 361 |
| mac_max_frame_retries | 345 | M-Bus slave port setup | 206, 361 |
| mac_max_orphan_timer | 344 | M-Bus value object | 338 |
| mac_neighbour_table | 345 | M-Bus_Data_Header_Type | 214 |
| mac_neighbour_table_ | 345 | mbus_port | 208 |
| mac_PAN_id | 344 | M-Bus_port_communication_state | 213 |
| mac_security_enabled | 344 | mbus_port_reference | 339 |
| mac_short_address | 344 | M-Bus_profile | 213 |
| mac_TMR_TTL | 345 | Measurement channel | 389 |
| Magnitude | 404 | Measurement methods | 415 |
| Maintenance, interface classes | 291 | Measurement period | 377, 405, 414, 435, 438, 439, 443, 456 |
| Managed payment mode | 31 | Measurement period for indexes and index differences | 456 |
| Management Logical Device | 48, 49, 250 | Measurement, absolute | 436 |
| Mandatory | 41, 42 | Measurement, disturbed | 436 |
| manual_disconnect | 146 | Measurement, undisturbed | 436 |
| manual_reconnect | 146 | Medium access control layer | 477 |
| Manufacturer ID | 338 | Messaging | 21 |
| Manufacturer specific | 41, 390, 392, 393, 400, 403, 405, 407, 413, 416, 421, 426, 437, 459, 463 | Meter | 436 |
| Manufacturer specific codes | 407, 418, 423, 462 | Meter connection diagram | 413 |
| manufacturer_id | 196, 214, 340 | Meter open event | 373 |
| Manufacturing number | 397 | Meter reset | 357 |
| mask_list | 65 | Meter tamper | 399 |
| Master key | 105, 116 | Meter tamper event | 373 |
| master_key | 122 | Metering conditions | 429 |

| | | | |
|--|------------------------------|---|-----------------------------|
| Metering point ID | 368, 379 | OBIS codes, water related objects | 464 |
| Metering point ID (abstract) | 397 | OBIS, Reserved ranges | 390 |
| Metering point ID (electricity related) | 415 | Object codes | 396 |
| Method | 21, 39, 42 | Object Identification System | 352 |
| Metrological LED | 413 | <i>object_list</i> 49, 83, 86, 88, 92, 93, 295, 297, 298, | |
| metrological | 152 | 300, 303, 307, 310, 312, 315, 318, 321 | |
| Metrology tamper event | 373 | Occurrence counter | 376, 404 |
| MIB variables | 230 | One-Pass Diffie-Hellman | 126 |
| min_delta_credit | 336 | Operating time | 371, 398 |
| min_over_threshold_duration | 148 | Operator | 204 |
| min_under_threshold_duration | 148 | Optical port | 360 |
| Minimum | 404, 445 | Optical test output | 357 |
| mode | 195, 199, 201, 330, 331, 333 | Optional | 41, 42 |
| Modelling | 21 | originator_system_title | 121 |
| Modem | 197, 198, 329, 331, 356 | Orthogonal Frequency Division Multiplexing | 477 |
| Modem configuration | 197, 353, 356 | Other media | 391 |
| modem_profile | 198, 329 | other_information | 121 |
| Modulo-100 | 467 | Output control | 358 |
| Modulo-12 | 467 | Output control signals | 397 |
| monitored_value | 143, 148, 280, 283 | Output pulse | 414 |
| Most recent value | 467 | Output pulse constant | 377, 383 |
| most_recent_requests_table | 157 | Output pulse constants converted / unconverted | 454 |
| MSS | 218 | Output signals | 357 |
| multicast_IP_address | 220 | output_state | 147 |
| Natural gas analysis | 387, 448 | output_type | 153 |
| Natural gas analysis values – Averages | 452 | Over limit | 406 |
| NB OFDM PLC for G3-PLC networks | 353 | p_bit_timer | 247 |
| NB OFDM PLC for PRIME networks | 353 | Packet switched network | 201 |
| NB OFDM PLC profile for PRIME networks | 250 | PAN coordinator | 263 |
| nb_of_sim_conn | 219 | pan_ID | 280 |
| neighbor_discovery_setup | 223 | Parameter | 397 |
| Network Control Protocol | 227 | Parameter changes | 368 |
| network_key | 281 | Parameter monitor | 150, 353, 358 |
| Neutral current | 402 | Parameter monitor log | 150, 401 |
| Neutral voltage | 402 | Parameter record | 413 |
| Never repeater | 235, 336 | parameter_list | 151 |
| NEW | 236 | pass_p1 | 192, 325 |
| New system | 26 | pass_p2 | 192, 325 |
| New system title | 26 | pass_w5 | 192, 325 |
| next_credit_available_threshold | 171 | Passive calendar | 142 |
| next_period | 64 | Password | 86, 296, 299, 302, 306, 325 |
| NO-BODY | 236 | Payment metering | 18, 161, 359 |
| Node | 27 | payment_event_based_collection | 162 |
| Nominal value | 383, 414 | Period | 61 |
| Nominal values volume sensor | 455 | Permission | 155 |
| Non configured state | 239 | permissions_table | 157 |
| Nonce | 126 | Personal Area Network | 278 |
| Normal mode | 357 | Phase angle | 408 |
| Normal threshold | 148 | phone_list | 332 |
| Notation | 40 | PHY_reference | 225 |
| number_of_calls | 200, 331 | Physical device | 48, 94, 365 |
| number_of_periods | 61, 64 | Physical layer, PRIME NB OFDM PLC | 477 |
| number_of_retries | 113 | PIN_code | 203 |
| number_of_rings | 200, 331 | PLC OFDM Type 1 Application identification | 364 |
| OBIS | 39, 41, 388 | | |
| OBIS code structure | 389 | | |
| OBIS codes, HCA related objects | 422 | | |

| | |
|--|-------------------------|
| PLC OFDM Type 1 MAC counters | 364 |
| PLC OFDM Type 1 MAC functional parameters..... | 364 |
| PLC OFDM Type 1 MAC network administration data..... | 364 |
| PLC OFDM Type 1 MAC setup | 364 |
| PLC OFDM Type 1 physical layer counters | 364 |
| Positive displacement..... | 429 |
| Post-payment | 31 |
| Power factor | 379, 402, 415 |
| Power factor, displacement | 379 |
| Power factor, true | 379 |
| Power failure | 138, 139, 219, 370, 398 |
| Power failure duration..... | 370 |
| Power failure event log | 417 |
| Power quality | 414 |
| Pperiod | 63 |
| PPP setup..... | 225, 353, 362 |
| PPP_authentication | 228 |
| Preferred readout | 70 |
| preset_adjusting_time | 134 |
| preset_credit_amount | 180 |
| Pressure | 431 |
| Primary address | 338 |
| primary_ | 214 |
| primary_address | 209, 340 |
| primary_address_list | 195, 327 |
| primary_DNS_address | 221, 223 |
| PRIME NB OFDM PLC Applications identification | 262 |
| PRIME NB OFDM PLC MAC counters..... | 258 |
| PRIME NB OFDM PLC MAC functional parameters..... | 256 |
| PRIME NB OFDM PLC MAC setup..... | 255 |
| PRIME NB OFDM PLC Physical layer counters | 254 |
| priority | 173, 178, 184 |
| Process data | 436 |
| Process interval | 435, 445, 456 |
| Process value | 41, 57, 386, 450 |
| Process value, gas | 445 |
| processed_value | 154 |
| processed_value_actions | 154 |
| processed_value_thresholds | 154 |
| Profile..... | 292 |
| Profile entries..... | 368, 373, 376, 379, 382 |
| Profile generic51, 66, 354, 355, 368, 373, 376, 379, 382 | |
| profile_entries | 69, 293 |
| Program entries | 377, 396, 413 |
| Promotion | 28 |
| prop_baud | 192, 325 |
| proportion | 187 |
| Proprietary | 39 |
| Protection parameters | 115 |
| protection_buffer | 116, 117, 119 |
| protection_object_list | 119 |
| protection_parameters_get | 121 |
| protection_parameters_set | 122 |
| protocol_address | 290 |
| protocol_version | 280 |
| ps_status | 205 |
| PSTN modem configuration..... | 328 |
| Public client..... | 49 |
| Publish/subscribe | 109 |
| Pulse constant | 413 |
| Pulse duration | 415 |
| Pulse value | 413 |
| Pulses | 402 |
| Push | 108, 357, 358 |
| Push setup | 110, 353, 362 |
| Push window | 110 |
| push_object_list | 111 |
| Quadrant | 402, 409 |
| quality_of_service | 203 |
| random delay | 110 |
| randomisation_start_interval | 113 |
| Rate..... | 398, 407 |
| raw_value | 153 |
| raw_value_actions | 153 |
| raw_value_thresholds | 153 |
| RCR program number..... | 413 |
| Reactive energy | 413, 414, 415 |
| Reactive power | 378, 402, 415 |
| read_by_logicalname .86, 296, 298, 301, 306 | |
| Reading factor..... | 377, 414 |
| Readout | 70, 353 |
| receive_lifetime_var_t2 | 249 |
| receive_window_size_rw | 247 |
| received_signal_strength | 216 |
| recipient_system_title | 121 |
| Recording interval | 414, 435, 456 |
| Recording period..... | 401, 406, 417 |
| Reduced Function Device..... | 263 |
| Reduced ID codes | 466 |
| Reference voltage..... | 414 |
| Register39, 57, 60, 64, 65, 73, 142, 370, 376, 377, 380, 383 | |
| Register activation | 64, 353, 358 |
| Register monitor..... | 142, 152, 353, 358, 381 |
| Register table | 72 |
| Register table objects – Abstract | 401 |
| Register table objects – Electricity | 417 |
| register_assignment | 64, 65 |
| register_device | 285 |
| Registered system | 26 |
| Registration..... | 27 |
| reject_timer | 247 |
| rejoin_interval | 282 |
| rejoin_retry_interval | 282 |
| remote_disconnect..... | 146, 147 |
| remote_reconnect | 146, 147 |
| remove_certificate | 108 |
| remove_HAN | 289 |

| | | | |
|------------------------------------|-------------------------------------|---------------------------------------|---|
| remove_mirror | 288 | security_suite | 103, 323 |
| remove_object | 311 | Selectable | 32 |
| remove_object (data) | 93, 316, 322 | Selected/Invoked | 32 |
| repayable | 32 | Selective access | 42, 67, 69, 71, 76, 83, 84, 88, 92, 111, 119, 297, 300, 303, 307, 310, 312, 315, 318, 321 |
| repeater | 235, 336 | Selective access parameters | 42 |
| repeater_status | 235, 336 | send | 112 |
| Repetition phase | 240, 243 | sender_address | 229 |
| repetition_delay | 113, 202, 332, 333 | Sensor manager | 151, 152, 384, 459 |
| repetitions | 202, 332, 333 | Sensor, pressure | 154 |
| repetitions_counter | 243 | Serial number, sensor | 152 |
| reply_status_list | 244, 338 | serial_number | 152 |
| reply_to_HLS_86 | 93, 302, 306, 311, 316, 322 | Server model | 48 |
| reply_to_HLS_authentication | 299 | server_address | 229 |
| reply_to_HLS_challenge | 297 | server_port | 229 |
| Reporting system | 26 | server_SAP | 89, 308, 313, 318 |
| Reporting system list | 245 | server_system_title | 103, 323 |
| required_protection | 123 | Service node | 27 |
| Reserved base_names | 40 | Service Specific Convergence Sublayer | 253 |
| Reserved credit | 32 | service_node_address | 253 |
| reserved_credit | 161 | set_amount_to_value | 181 |
| reset | 58, 61, 64, 70, 73, 154, 293 | set_encryption_key | 342 |
| Reset | 415, 457 | set_protected_attributes | 117, 124 |
| Reset counter | 382 | set_total_amount_remaining | 188 |
| reset_account | 172 | S-FSK Active initiator | 363 |
| reset_alarm | 341 | S-FSK MAC counters | 363 |
| reset_NEW_not | 237 | S-FSK MAC synchronization timeouts | 363 |
| response_time | 192, 325 | S-FSK Phy&MAC setup | 363 |
| restore_HAN | 287 | S-FSK Physical layer | 230 |
| Ripple control receiver program | 397 | S-FSK PLC setup | 353 |
| Robust Header Compression | 228 | S-FSK Reporting system list | 363 |
| SAP | 94 | Shared line | 199, 330 |
| SAP assignment | 40, 48, 49, 82, 93, 353, 365 | shift_time | 134 |
| SAP_assignment_list | 94 | Short name | 40, 42 |
| scaler_unit | 57, 73, 153 | short_address | 280 |
| scan_attempts | 282 | Single action schedule | 144, 353, 358 |
| Schedule | 18, 136, 139, 140, 353, 357 | slave_deinstall | 210, 341 |
| Script | 135, 137, 138, 144 | slave_install | 210, 340 |
| Script table | 40, 135, 140, 142, 144, 353, 356 | Sliding demand | 61, 64 |
| Script, null | 135 | Smart Energy Profile | 281 |
| scripts | 135 | SMTP setup | 229, 362 |
| sealing_method | 153 | Social credit | 32 |
| search_initiator | 239 | Solar | 465 |
| season_profile | 141 | Sort method | 368, 373, 376, 379, 382 |
| secondary_address | 327 | sort_method | 68, 293 |
| secondary_DNS_address | 221, 223 | sort_object | 68, 293 |
| Secret | 86, 91, 93, 302, 306, 311, 315, 320 | Source of reset | 466 |
| Secret, new | 299 | Special days | 136 |
| Security mechanisms | 50 | Special days table | 139, 140, 353, 357 |
| Security setup | 18, 82, 102, 353, 366 | stack_profile | 280 |
| Security suite | 115 | Standard object codes | 391 |
| Security switches | 397 | Standard readout profile | 360 |
| security_activate | 104, 323 | start_time_current | 63 |
| security_policy | 102, 323 | start_up_control | 280 |
| security_setup_reference | 85, 91, 301, 304, 315, 320 | Static Unified Model | 126 |

| | |
|---|-----------------------------------|
| Station management information objects | 457 |
| status | 60, 63, 133, 199, 331, 340 |
| Status information, Electricity | 416 |
| Status mapping | 74 |
| Status register | 371, 398 |
| Status value | 57 |
| status_word | 74 |
| Strong DC magnetic field event | 373 |
| subnet_mask | 221 |
| Subnetwork | 27 |
| Sub-slot | 26 |
| superior calorific value | 386 |
| Swells | 417 |
| Switch node | 251 |
| Switch state | 28 |
| SYNCHRO_FOUND | 241 |
| Synchronization | 138, 234, 335 |
| Synchronization method | 415 |
| Synchronization process | 241 |
| Synchronization window | 415 |
| synchronization_confirmation_timeout | 239 |
| synchronization_locked | 236, 337 |
| synchronization_register | 242 |
| synchronization-locked | 241 |
| Synchronization-locked state | 236, 337 |
| Synchronization-unlocked state | 236, 337 |
| synchronize_clock | 210, 341 |
| System Title | 237 |
| tabi_list | 195, 328 |
| table_cell_definition | 73 |
| table_cell_values | 72 |
| table_ID | 71 |
| Tamper | 373 |
| Tariff rates | 406, 450 |
| Tarification | 64, 357, 358 |
| TCP-UDP setup | 218, 353, 362 |
| TCP-UDP_port | 218 |
| Telephone number | 399 |
| Temperature | 431 |
| template_description | 77 |
| template_id | 76 |
| Temporary debt | 32 |
| Terminal cover open event | 373 |
| Terminal node | 251 |
| Terminal state | 27 |
| Test mode | 357 |
| Test time integral | 406 |
| Test value | 414 |
| Thermal energy | 19, 391, 423 |
| threshold | 143 |
| Threshold | 406, 412 |
| Threshold value | 148, 380, 383, 455 |
| Threshold, missing | 380, 405 |
| Threshold, over limit | 380, 404 |
| Threshold, under limit | 380, 404 |
| threshold_active | 148 |
| threshold_emergency | 148 |
| threshold_normal | 148 |
| Tilt event | 373 |
| time | 133 |
| Time changes | 138 |
| Time entries | 377, 383, 396, 415, 420, 426, 457 |
| Time integral | 376, 404, 405, 406 |
| Time of operation | 398 |
| Time setting backward | 139 |
| Time setting forward | 139 |
| Time stamp | 396, 413 |
| Time switch program | 397, 413 |
| Time synchronization | 139 |
| time_based_collection | 162 |
| time_based_credit | 161 |
| time_between_scans | 282 |
| time_out_frame_not | 240 |
| time_zone | 133 |
| Timeslot | 26 |
| token | 189 |
| Token | 33 |
| Token carrier | 33 |
| Token Carrier Interface | 189 |
| Token credit history | 401 |
| Token gateway | 189, 353, 359 |
| Token transfer log | 401 |
| token_credit | 161 |
| token_delivery_method | 189 |
| token_description | 189 |
| token_gateway_configuration | 169 |
| token_status | 189 |
| token_time | 189 |
| Top-up | 33 |
| Total | 407 |
| Total Demand Distortion | 407 |
| Total Harmonic Distortion | 407 |
| total_amount_paid | 184 |
| total_amount_remaining | 187 |
| transaction_id | 121 |
| transfer_key | 211, 342 |
| Transformer and line loss | 408 |
| Transformer loss | 402 |
| Transformer magnetic losses | 415 |
| Transformer ratio – current (numerator) | 414 |
| Transformer ratio – voltage | 414 |
| Transformer thermal losses | 415 |
| Transmission phase | 240, 243 |
| transmissions_counter | 216, 243 |
| transmit_lifetime_var_t3 | 249 |
| transmit_window_size_k | 247 |
| Transporting | 21 |
| trust_center_address, ZigBee ® | 281 |
| Twisted pair | 194, 327 |
| Twisted pair setup | 353 |
| Unbalance | 417 |
| Unconfigured | 340 |
| Unconfigured state | 237 |

| | | | |
|--|--------------------|--|-----------------------------|
| Under limit..... | 406 | Value groups, optional | 466 |
| unicast_IPv6_addresses | 222 | Value, default | 42 |
| UNIPEDE..... | 411 | Value, maximum | 42 |
| UNIPEDE voltage dips | 417 | Value, minimum | 42 |
| Unit | 58 | Van Jacobson compression..... | 228 |
| unit_charge_activation_time | 186 | Vend..... | 33 |
| unit_charge_active | 184 | Venturi measurement | 459 |
| unit_charge_passive | 186 | Version ...41, 88, 291, 307, 312, 317, 340, 388 | |
| Unitless quantities | 414 | Version, previous | 18, 291 |
| UNIX clock..... | 356 | Voltage | 402, 414 |
| unregister_all_devices | 286 | Voltage dips | 411 |
| unregister_device | 286 | Volt-squared hours..... | 402, 409, 414 |
| update_amount | 181 | Volume | 431 |
| update_link_key | 289 | Volume at base conditions..... | 430, 432 |
| update_network_key | 289 | Volume at metering conditions | 432 |
| update_total_amount_remaining | 188 | Volume conversion | 429, 430, 432, 458 |
| update_unit_charge | 188 | Volume measurement..... | 429 |
| use_DHCP_flag | 221 | Wake-up..... | 468 |
| use_insecure_join | 281 | Wake-up request..... | 200 |
| User group specific..... | 41 | warning_threshold | 178 |
| user_list | 85, 92, 305, 321 | Water..... | 462, 463, 465 |
| user_name | 229 | Week day | 415 |
| UTC | 132, 133 | week_profile_table | 141 |
| Utility specific..... | 390, 392 | Weighting | 155 |
| Utility tables | 71, 353, 367 | weighting_table | 157 |
| V.44 compression..... | 103 | Wind | 465 |
| Value..... | 39, 56, 57 | window_size_receive | 193, 326 |
| Value at base conditions | 438, 443, 445 | window_size_transmit | 193, 326 |
| Value at metering conditions | 438, 443, 445 | Wireless Mode Q | 353, 361 |
| Value at standard conditions | 443, 445 | Wireless Mode Q channel..... | 212 |
| Value group A | 391 | Wrapped_key..... | 121 |
| Value group B | 391 | write | 294 |
| Value group C | 353, 392, 466 | xDLMS context..... | 49 |
| Value group C, Electricity | 402 | xDLMS_context_info | 90, 309, 314, 319 |
| Value group C, Gas | 436 | year_of_manufacture | 196 |
| Value group C, HCA | 418 | ZigBee® | 28, 112, 278, 282, 353, 365 |
| Value group C, Other media..... | 465 | ZigBee® 053474 | 23 |
| Value group C, Thermal energy | 423 | ZigBee® 2007 | 279 |
| Value group C, Water | 462 | ZigBee® client | 29 |
| Value group D | 376, 393, 411, 419 | ZigBee® cluster | 29 |
| Value group D, Electricity | 404 | ZigBee® Communications hub | 278 |
| Value group D, Gas | 438 | ZigBee® coordinator | 29, 278 |
| Value group D, HCA | 419 | ZigBee® mirror | 29, 288 |
| Value group D, Other media..... | 465 | ZigBee® network control | 283, 365 |
| Value group D, Thermal energy | 424 | ZigBee® PAN | 284 |
| Value group D, Water | 462 | ZigBee® PAN creation | 278 |
| Value group E395, 407, 408, 411, 450, 451, 452 | | ZigBee® Pro | 29, 279 |
| Value group E, Electricity | 406 | ZigBee® router | 29 |
| Value group E, Gas | 450 | ZigBee® SAS APS fragmentation | 282, 365 |
| Value group E, Other media..... | 465 | ZigBee® SAS join | 281, 365 |
| Value group F | 380, 396, 466 | ZigBee® SAS startup | 279, 365 |
| Value group F, Electricity..... | 411 | ZigBee® server | 29, 278 |
| Value group F, Gas | 453 | ZigBee® Smart Energy Specification | 288 |
| Value group F, Other media | 465 | ZigBee® Trust Center | 29, 278, 281 |
| Value groups, mandatory | 466 | ZigBee® tunnel setup | 290, 365 |

| | | | |
|-----------------------|------------|-------------------------|---------|
| DLMS User Association | 2015-12-21 | DLMS UA 1000-1 Ed. 12.1 | 491/492 |
|-----------------------|------------|-------------------------|---------|

NOTES