

**TECHNICAL REPORT**  
**Comprehensive Semantic Model**  
**for Energy Management**

**COSEM Interface Classes**

DLMS UA 1000-1 Ed. 15 Part 2

Version 1.0

21 December 2021

DLMS User Association

© Copyright 1997-2021 DLMS User Association



## CONTENTS

|   |    |
|---|----|
| FOREWORD .....  | 13 |
| List of main technical changes in Edition 15.....                                       | 16 |
| INTRODUCTION .....  | 17 |
| 1 Scope .....   | 18 |
| 2 Referenced Documents .....  | 19 |
| 3 Terms, definitions and abbreviated terms.....   | 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 the ISO/IEC 14908 setup ICs (see 4.19) .....       | 29 |
| 3.5 Terms and definitions related to ZigBee® (see 4.15) .....                           | 29 |
| 3.6 Terms and definitions related to Payment metering interface classes (see 4.6) ..... | 31 |
| 3.7 Terms and definitions related to the Arbitrator IC (see 4.5.12) .....               | 35 |
| 3.8 Abbreviated terms .....   | 35 |
| 4 The COSEM interface classes .....   | 40 |
| 4.1 Basic principles .....  | 40 |
| 4.1.1 General.....  | 40 |
| 4.1.2 Referencing methods.....  | 41 |
| 4.1.3 Reserved base_names for special COSEM objects.....                                | 42 |
| 4.1.4 Class description notation.....   | 42 |
| 4.1.5 Common data types.....  | 45 |
| 4.1.6 Data formats .....  | 47 |
| 4.1.7 The COSEM server model .....  | 52 |
| 4.1.8 The COSEM logical device .....  | 53 |
| 4.1.9 Information security .....  | 55 |
| 4.2 Overview of the COSEM interface classes .....                                       | 55 |
| 4.3 Interface classes for parameters and measurement data.....                          | 65 |
| 4.3.1 Data (class_id = 1, version = 0).....   | 65 |
| 4.3.2 Register (class_id = 3, version = 0) .....  | 66 |
| 4.3.3 Extended register (class_id = 4, version = 0) .....                               | 71 |
| 4.3.4 Demand register (class_id = 5, version = 0) .....                                 | 73 |
| 4.3.5 Register activation (class_id = 6, version = 0).....                              | 77 |
| 4.3.6 Profile generic (class_id = 7, version = 1) .....                                 | 78 |
| 4.3.7 Utility tables (class_id = 26, version = 0) .....                                 | 84 |
| 4.3.8 Register table (class_id = 61, version = 0) .....                                 | 86 |
| 4.3.9 Status mapping (class_id = 63, version = 0) .....                                 | 88 |
| 4.3.10 Compact data (class_id: 62, version = 1) .....                                   | 89 |
| 4.4 Interface classes for access control and management.....                            | 99 |
| 4.4.1 Overview.....   | 99 |

|                       |            |                             |       |
|-----------------------|------------|-----------------------------|-------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 1/668 |
|-----------------------|------------|-----------------------------|-------|

## COSEM Interface Classes

|        |   |     |
|--------|---|-----|
| 4.4.2  | Client user identification .....  | 99  |
| 4.4.3  | Association SN (class_id = 12, version = 4) .....                               | 100 |
| 4.4.4  | Association LN (class_id = 15, version = 3) .....                               | 105 |
| 4.4.5  | SAP assignment (class_id = 17, version = 0) .....                               | 112 |
| 4.4.6  | Image transfer (class_id = 18, version = 0) .....                               | 113 |
| 4.4.7  | Security setup (class_id = 64, version = 1) .....                               | 125 |
| 4.4.8  | Push interface class .....  | 131 |
| 4.4.9  | COSEM data protection (class_id = 30, version = 0) .....                        | 143 |
| 4.4.10 | Function control (class_id: 122, version: 0).....                               | 161 |
| 4.4.11 | Array manager (class_id = 123, version = 0).....                                | 163 |
| 4.4.12 | Communication port protection (class_id = 124, version = 0).....                | 169 |
| 4.5    | Interface classes for time- and event bound control .....                       | 172 |
| 4.5.1  | Clock (class_id = 8, version = 0) .....   | 172 |
| 4.5.2  | Script table (class_id = 9, version = 0).....                                   | 175 |
| 4.5.3  | Schedule (class_id = 10, version = 0).....                                      | 177 |
| 4.5.4  | Special days table (class_id = 11, version = 0) .....                           | 180 |
| 4.5.5  | Activity calendar (class_id = 20, version = 0) .....                            | 182 |
| 4.5.6  | Register monitor (class_id = 21, version = 0) .....                             | 184 |
| 4.5.7  | Single action schedule (class_id = 22, version = 0).....                        | 186 |
| 4.5.8  | Disconnect control (class_id = 70, version = 0) .....                           | 187 |
| 4.5.9  | Limiter (class_id = 71, version = 0).....                                       | 191 |
| 4.5.10 | Parameter monitor (class_id = 65, version = 1).....                             | 194 |
| 4.5.11 | Sensor manager (class_id = 67, version = 0) .....                               | 197 |
| 4.5.12 | Arbitrator (class_id = 68, version = 0).....                                    | 202 |
| 4.5.13 | Modelling examples: tariffication and billing .....                             | 206 |
| 4.6    | Payment metering related interface classes.....                                 | 208 |
| 4.6.1  | Overview of the COSEM accounting model.....                                     | 208 |
| 4.6.2  | Account (class_id = 111, version = 0).....                                      | 210 |
| 4.6.3  | Credit (class_id = 112, version = 0) .....                                      | 220 |
| 4.6.4  | Charge (class_id = 113, version = 0) .....                                      | 231 |
| 4.6.5  | Token gateway (class_id = 115, version = 0) .....                               | 237 |
| 4.6.6  | <b>IEC 62055-41 Attributes (class_id = 116, version =0)</b> .....               | 239 |
| 4.7    | Interface classes for setting up data exchange via local ports and modems ..... | 243 |
| 4.7.1  | IEC local port setup (class_id = 19, version = 1) .....                         | 243 |
| 4.7.2  | IEC HDLC setup (class_id = 23, version = 1) .....                               | 245 |
| 4.7.3  | IEC twisted pair (1) setup (class_id = 24, version = 1) .....                   | 247 |
| 4.7.4  | Modem configuration (class_id = 27, version = 1) .....                          | 249 |
| 4.7.5  | Auto answer (class_id = 28, version = 2) .....                                  | 251 |
| 4.7.6  | Auto connect (class_id = 29, version = 2) .....                                 | 254 |
| 4.7.7  | GPRS modem setup (class_id = 45, version = 0) .....                             | 256 |
| 4.7.8  | GSM diagnostic (class_id: 47, version: 2) .....                                 | 257 |
| 4.7.9  | LTE monitoring (class_id: 151, version = 1) .....                               | 260 |
| 4.8    | Interface classes for setting up data exchange via M-Bus .....                  | 264 |
| 4.8.1  | Overview.....   | 264 |
| 4.8.2  | M-Bus slave port setup (class_id = 25, version = 0) .....                       | 264 |
| 4.8.3  | M-Bus client (class_id = 72, version = <b>2</b> ) .....                         | 265 |
| 4.8.4  | Wireless Mode Q channel (class_id = 73, version = 1) .....                      | 273 |

|       |            |                              |                       |
|-------|------------|------------------------------|-----------------------|
| 2/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|-------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

|         |  |     |
|---------|--|-----|
| 4.8.5   | M-Bus master port setup (class_id = 74, version = 0).....  | 274 |
| 4.8.6   | DLMS server M-Bus port setup (class_id = 76, version = 0) .....  | 274 |
| 4.8.7   | M-Bus diagnostic (class_id = 77, version = 0).....   | 277 |
| 4.9     | Interface classes for setting up data exchange over the Internet.....  | 281 |
| 4.9.1   | TCP-UDP setup (class_id = 41, version = 0).....  | 281 |
| 4.9.2   | IPv4 setup (class_id = 42, version = 0).....   | 282 |
| 4.9.3   | IPv6 setup (class_id = 48, version = 0).....   | 286 |
| 4.9.4   | MAC address setup (class_id = 43, version = 0) .....   | 289 |
| 4.9.5   | PPP setup (class_id = 44, version = 0) .....   | 290 |
| 4.9.6   | SMTP setup (class_id = 46, version = 0) .....  | 294 |
| 4.9.7   | NTP setup (class_id = 100, version = 0) .....  | 295 |
| 4.9.8   | CoAP setup (class_id = 152 version = 0) .....  | 298 |
| 4.9.9   | CoAP diagnostic (class_id = 153, version = 0) .....  | 300 |
| 4.10    | Interface classes for setting up data exchange using S-FSK PLC .....   | 303 |
| 4.10.1  | General.....   | 303 |
| 4.10.2  | Overview.....  | 303 |
| 4.10.3  | S-FSK Phy&MAC set-up (class_id = 50, version = 1) .....  | 305 |
| 4.10.4  | S-FSK Active initiator (class_id = 51, version = 0).....   | 311 |
| 4.10.5  | S-FSK MAC synchronization timeouts (class_id = 52, version = 0) .....  | 312 |
| 4.10.6  | S-FSK MAC counters (class_id = 53, version = 0).....   | 314 |
| 4.10.7  | IEC 61334-4-32 LLC setup (class_id = 55, version = 1).....   | 318 |
| 4.10.8  | S-FSK Reporting system list (class_id = 56, version = 0) .....   | 319 |
| 4.11    | Interface classes for setting up the LLC layer for ISO/IEC 8802-2 .....  | 320 |
| 4.11.1  | General.....   | 320 |
| 4.11.2  | ISO/IEC 8802-2 LLC Type 1 setup (class_id = 57, version = 0) .....   | 320 |
| 4.11.3  | ISO/IEC 8802-2 LLC Type 2 setup (class_id = 58, version = 0) .....   | 321 |
| 4.11.4  | ISO/IEC 8802-2 LLC Type 3 setup (class_id = 59, version = 0) .....   | 322 |
| 4.12    | Interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for PRIME networks .....      | 324 |
| 4.12.1  | Overview.....  | 324 |
| 4.12.2  | Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes.....  | 325 |
| 4.12.3  | 61334-4-32 LLC SSCS setup (class_id = 80, version = 0) .....   | 327 |
| 4.12.4  | PRIME NB OFDM PLC Physical layer parameters .....  | 328 |
| 4.12.5  | PRIME NB OFDM PLC Physical layer counters (class_id = 81, version = 0).....  | 328 |
| 4.12.6  | PRIME NB OFDM PLC MAC setup (class_id = 82, version = 0) .....   | 330 |
| 4.12.7  | PRIME NB OFDM PLC MAC functional parameters (class_id = 83 version = 0).....                                       | 331 |
| 4.12.8  | PRIME NB OFDM PLC MAC counters (class_id = 84, version = 0) .....  | 334 |
| 4.12.9  | PRIME NB OFDM PLC MAC network administration data (class_id = 85, version = 0) .....                               | 335 |
| 4.12.10 | PRIME NB OFDM PLC MAC address setup (class_id = 43, version = 0) .....   | 338 |
| 4.12.11 | PRIME NB OFDM PLC Application identification (class_id = 86, version = 0) .....                                    | 338 |
| 4.13    | Interface classes for setting up and managing the DLMS/COSEM narrowband OFDM PLC profile for G3-PLC networks ..... | 340 |
| 4.13.1  | Overview.....  | 340 |

|                       |            |                             |       |
|-----------------------|------------|-----------------------------|-------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 3/668 |
|-----------------------|------------|-----------------------------|-------|

## COSEM Interface Classes

|               |  |            |
|---------------|--|------------|
| 4.13.2        | Mapping of G3-PLC PIB attributes to COSEM IC attributes .....  | 340        |
| 4.13.3        | G3-PLC MAC layer counters (class_id = 90, version = 1) .....   | 344        |
| 4.13.4        | G3-PLC MAC setup (class_id = 91, version = 3) .....  | 345        |
| 4.13.5        | G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 3) .....                                     | 358        |
| <b>4.13.6</b> | <b>G3-PLC Hybrid RF MAC layer counters (class_id = 160, version = 0)</b> .....                               | <b>365</b> |
| <b>4.13.7</b> | <b>G3-PLC Hybrid RF MAC setup (class_id = 161, version = 0)</b> .....  | <b>367</b> |
| <b>4.13.8</b> | <b>G3-PLC Hybrid 6LoWPAN adaptation layer setup (class_id = 162, version = 0)</b> .....                      | <b>370</b> |
| 4.14          | Interface classes for setting up and managing DLMS/COSEM HS-PLC ISO/IEC 12139-1 neighbourhood networks ..... | 373        |
| 4.14.1        | Overview.....  | 373        |
| 4.14.2        | HS-PLC ISO/IEC 12139-1 MAC setup (class_id = 140, version = 0) .....   | 373        |
| 4.14.3        | HS-PLC ISO/IEC 12139-1 CPAS setup (class_id = 141, version = 0) .....  | 375        |
| 4.14.4        | HS-PLC ISO/IEC 12139-1 IP SSAS setup (class_id = 142, version = 0).....                                      | 375        |
| 4.14.5        | HS-PLC ISO/IEC 12139-1 HDLC SSAS setup (class_id = 143, version = 0).....                                    | 376        |
| 4.15          | ZigBee® setup classes .....  | 378        |
| 4.15.1        | Overview.....  | 378        |
| 4.15.2        | ZigBee® SAS startup (class_id = 101, version = 0) .....  | 380        |
| 4.15.3        | ZigBee® SAS join (class_id = 102, version = 0) .....   | 382        |
| 4.15.4        | ZigBee® SAS APS fragmentation (class_id = 103, version = 0) .....  | 383        |
| 4.15.5        | ZigBee® network control (class_id = 104, version = 0).....   | 384        |
| 4.15.6        | ZigBee® tunnel setup (class_id = 105, version = 0) .....   | 390        |
| 4.16          | Interface classes for setting up and managing the DLMS/COSEM profile for LPWAN networks.....                 | 392        |
| 4.16.1        | General.....   | 392        |
| 4.16.2        | Generic interface classes.....   | 392        |
| 4.17          | LPWAN specific interface classes .....   | 397        |
| 4.17.1        | General.....   | 397        |
| 4.17.2        | LoRaWAN® interface classes .....   | 398        |
| 4.18          | Interface classes for setting up and managing the DLMS/COSEM profile for Wi-SUN networks .....               | 404        |
| 4.18.1        | Wi-SUN setup (class_id = 95, version 0) .....  | 404        |
| 4.18.2        | Wi-SUN diagnostic (class_id = 96, version 0) .....   | 409        |
| 4.18.3        | RPL diagnostic (class_id = 97, version 0) .....  | 412        |
| 4.18.4        | MPL diagnostic (class_id = 98, version 0).....   | 414        |
| 4.19          | Interface classes for setting up and managing the DLMS/COSEM profile for ISO/IEC 14908 PLC networks .....    | 417        |
| 4.19.1        | General.....   | 417        |
| 4.19.2        | ISO/IEC 14908 identification (class_id = 130, version = 0) .....   | 417        |
| 4.19.3        | ISO/IEC 14908 protocol setup (class_id = 131, version = 0) .....   | 418        |
| 4.19.4        | ISO/IEC 14908 protocol status (class_id = 132, version = 0) .....  | 418        |
| 4.19.5        | ISO/IEC 14908 diagnostic (class_id = 133, version = 0) .....   | 421        |
| 5             | Previous versions of interface classes .....   | 424        |
| 5.1           | General.....   | 424        |
| 5.1.1         | New versions of interface classes .....  | 424        |
| 5.1.2         | New interface classes.....   | 424        |
| 5.1.3         | Removal of interface classes .....   | 424        |

|       |            |                              |                       |
|-------|------------|------------------------------|-----------------------|
| 4/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|-------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

|        |  |     |
|--------|--|-----|
| 5.2    | Previous versions of interface classes – general.....  | 424 |
| 5.3    | Previous versions of interface classes for parameters and measurement data .....   | 424 |
| 5.3.1  | Profile generic (class_id = 7, version = 0).....   | 424 |
| 5.3.2  | Compact data (class_id = 62, version = 0).....   | 429 |
| 5.4    | Previous versions of interface classes for access control and management .....   | 432 |
| 5.4.1  | Association SN (class_id = 12, version = 0).....   | 432 |
| 5.4.2  | Association SN (class_id = 12, version = 1).....   | 435 |
| 5.4.3  | Association SN (class_id = 12, version = 2).....   | 437 |
| 5.4.4  | Association SN (Class_id = 12, version =3) .....   | 441 |
| 5.4.5  | Association LN (class_id = 15, version = 0) .....  | 446 |
| 5.4.6  | Association LN (class_id = 15, version = 1) .....  | 451 |
| 5.4.7  | Association LN (class_id = 15, version = 2) .....  | 457 |
| 5.4.8  | Security setup (class_id = 64, version = 0) .....  | 464 |
| 5.4.9  | Push Setup (class_id = 40, version = 0) .....  | 466 |
| 5.4.10 | Push Setup (class_id = 40, version = 1) .....  | 470 |
| 5.4.11 | <b>Push setup (class_id = 40, version = 2)</b> .....   | 476 |
| 5.5    | Previous versions of interface classes for time- and event-bound control.....  | 486 |
| 5.5.1  | Parameter monitor (class_id = 65, version = 0).....  | 486 |
| 5.6    | Previous versions of payment metering related interface classes .....  | 488 |
| 5.7    | Previous versions of interface classes for setting up data exchange via local ports and modems .....                               | 488 |
| 5.7.1  | IEC local port setup (class_id = 19, version = 0) .....  | 488 |
| 5.7.2  | IEC HDLC setup, (class_id = 23, version = 0) .....   | 490 |
| 5.7.3  | IEC twisted pair (1) setup (class_id = 24, version = 0) .....  | 492 |
| 5.7.4  | PSTN modem configuration (class_id = 27, version = 0) .....  | 494 |
| 5.7.5  | Auto answer (class_id = 28, version = 0) .....   | 495 |
| 5.7.6  | PSTN auto dial (class_id = 29, version = 0) .....  | 497 |
| 5.7.7  | Auto connect (class_id = 29, version = 1) .....  | 499 |
| 5.7.8  | GSM diagnostic (class_id = 47, version = 0) .....  | 500 |
| 5.7.9  | GSM diagnostic (class_id: 47, version = 1) .....   | 503 |
| 5.7.10 | LTE monitoring (class_id: 151, version: 0) .....   | 505 |
| 5.8    | Previous versions of interface classes for setting up data exchange via M-Bus .....  | 507 |
| 5.8.1  | M-Bus client (class_id = 72, version = 0) .....  | 507 |
| 5.8.2  | <b>M-Bus client (class_id = 72, version = 1)</b> .....   | 512 |
| 5.9    | <b>Previous versions of interface classes for setting up data exchange over the internet</b> .....                                 | 518 |
| 5.10   | <b>Previous versions of interface classes for data exchange using S-FSK PLC</b> .....  | 518 |
| 5.10.1 | <b>S-FSK Phy&amp;MAC setup (class_id = 50, version = 0)</b> .....  | 518 |
| 5.10.2 | S-FSK IEC 61334-4-32 LLC setup (class_id = 55, version = 0) .....  | 523 |
| 5.11   | Previous versions of interface classes for setting up the LLC layer for ISO/IEC 8802-2 .....                                       | 524 |
| 5.12   | Previous versions of interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for PRIME networks ..... | 524 |
| 5.13   | Previous versions of interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for G3-PLC networks..... | 524 |
| 5.13.1 | <b>Overview</b> .....  | 524 |

|                       |            |                             |       |
|-----------------------|------------|-----------------------------|-------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 5/668 |
|-----------------------|------------|-----------------------------|-------|

## COSEM Interface Classes

|         |  |     |
|---------|--|-----|
| 5.13.2  | Mapping of G3-PLC IB attributes to COSEM IC attributes (ITU-T G9903:2013 Amd. 1 version).....                                    | 524 |
| 5.13.3  | Mapping of G3-PLC IB attributes to COSEM IC attributes (ITU-T G9903:2017 version) .....  | 526 |
| 5.13.4  | G3 NB OFDM PLC MAC layer counters (class_id = 90, version = 0) .....   | 528 |
| 5.13.5  | G3 NB OFDM PLC MAC setup (class_id = 91, version = 0) .....  | 529 |
| 5.13.6  | G3-PLC MAC setup (class_id = 91, version = 1) .....  | 534 |
| 5.13.7  | <b>G3-PLC MAC setup (class_id = 91, version = 2)</b> .....   | 539 |
| 5.13.8  | G3 NB OFDM PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 0).....  | 553 |
| 5.13.9  | G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 1) .....   | 558 |
| 5.13.10 | <b>G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 2)</b> .....  | 565 |
| 5.14    | Previous versions of interface classes for setting up and managing DLMS/COSEM HS-PLC ISO/IEC 12139-1 neighbourhood networks..... | 571 |
| 5.15    | Previous versions of ZigBee® setup classes.....  | 572 |
| 6       | Relation to OBIS .....   | 573 |
| 6.1     | General.....   | 573 |
| 6.2     | Abstract COSEM objects.....  | 573 |
| 6.2.1   | Use of value group C .....   | 573 |
| 6.2.2   | Data of historical billing periods .....   | 575 |
| 6.2.3   | Billing period values / reset counter entries .....  | 577 |
| 6.2.4   | Other abstract general purpose OBIS codes .....  | 577 |
| 6.2.5   | Clock objects (class_id = 8) .....   | 578 |
| 6.2.6   | Modem configuration and related objects .....  | 579 |
| 6.2.7   | Script table objects (class_id = 9) .....  | 579 |
| 6.2.8   | Special days table objects (class_id = 11) .....   | 580 |
| 6.2.9   | Schedule objects (class_id = 10).....  | 580 |
| 6.2.10  | Activity calendar objects (class_id = 20) .....  | 581 |
| 6.2.11  | Register activation objects (class_id = 6).....  | 581 |
| 6.2.12  | Single action schedule objects (class_id = 22).....  | 581 |
| 6.2.13  | Register monitor objects (class_id = 21).....  | 582 |
| 6.2.14  | Parameter monitor objects (class_id = 65).....   | 582 |
| 6.2.15  | Limiter objects (class_id = 71) .....  | 582 |
| 6.2.16  | Array manager objects (class_id = 123).....  | 582 |
| 6.2.17  | Payment metering related objects .....   | 582 |
| 6.2.18  | IEC local port setup objects (class_id = 19) .....   | 583 |
| 6.2.19  | Standard readout profile objects (class_id = 7) .....  | 584 |
| 6.2.20  | IEC HDLC setup objects (class_id = 23).....  | 584 |
| 6.2.21  | IEC twisted pair (1) setup objects (class_id = 24) .....   | 584 |
| 6.2.22  | Objects related to data exchange over M-Bus.....   | 585 |
| 6.2.23  | Objects to set up data exchange over the Internet .....  | 586 |
| 6.2.24  | Push Setup objects (class_id = 40) .....   | 588 |
| 6.2.25  | Objects for setting up data exchange using S-FSK PLC .....   | 588 |
| 6.2.26  | Objects for setting up the ISO/IEC 8802-2 LLC layer .....  | 589 |
| 6.2.27  | Objects for data exchange using narrowband OFDM PLC for PRIME networks .....   | 589 |
| 6.2.28  | Objects for data exchange using narrow-band OFDM PLC for G3-PLC networks .....   | 590 |

|       |            |                              |                       |
|-------|------------|------------------------------|-----------------------|
| 6/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|-------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

|        |   |     |
|--------|---|-----|
| 6.2.29 | ZigBee® setup objects.....  | 591 |
| 6.2.30 | Objects for setting up and managing data exchange using ISO/IEC 14908 PLC networks..... | 591 |
| 6.2.31 | Objects for data exchange using HS-PLC ISO/IEC 12139-1 ISO/EC 12139-1 networks .....    | 592 |
| 6.2.32 | Objects for data exchange using Wi-SUN networks.....                                    | 592 |
| 6.2.33 | Association objects (class_id = 12, 15).....  | 593 |
| 6.2.34 | SAP assignment object (class_id = 17).....  | 593 |
| 6.2.35 | COSEM logical device name object.....   | 593 |
| 6.2.36 | Information security related objects .....  | 593 |
| 6.2.37 | Image transfer objects (class_id = 18).....   | 594 |
| 6.2.38 | Function control objects (class_id = 122).....  | 594 |
| 6.2.39 | Communication port protection objects (class_id = 124).....                             | 594 |
| 6.2.40 | Utility table objects (class_id = 26) .....   | 595 |
| 6.2.41 | Compact data objects (class_id = 62).....   | 595 |
| 6.2.42 | Device ID objects .....   | 595 |
| 6.2.43 | Metering point ID objects .....   | 596 |
| 6.2.44 | Parameter changes and calibration objects .....   | 596 |
| 6.2.45 | I/O control signal objects .....  | 596 |
| 6.2.46 | Disconnect control objects (class_id = 70).....   | 597 |
| 6.2.47 | Arbitrator objects (class_id = 68).....   | 597 |
| 6.2.48 | Status of internal control signals objects .....  | 597 |
| 6.2.49 | Internal operating status objects .....   | 598 |
| 6.2.50 | Battery entries objects .....   | 598 |
| 6.2.51 | Power failure monitoring objects .....  | 598 |
| 6.2.52 | Operating time objects.....   | 599 |
| 6.2.53 | Environment related parameters objects.....   | 599 |
| 6.2.54 | Status register objects.....  | 599 |
| 6.2.55 | Event code objects .....  | 600 |
| 6.2.56 | Communication port log parameter objects .....  | 600 |
| 6.2.57 | Consumer message objects .....  | 600 |
| 6.2.58 | Currently active tariff objects .....   | 601 |
| 6.2.59 | Event counter objects .....   | 601 |
| 6.2.60 | Profile entry digital signature objects.....  | 601 |
| 6.2.61 | Profile entry counter objects .....   | 601 |
| 6.2.62 | Meter tamper event related objects .....  | 602 |
| 6.2.63 | Error register objects .....  | 602 |
| 6.2.64 | Alarm register, Alarm filter and Alarm descriptor objects .....                         | 603 |
| 6.2.65 | General list objects.....   | 604 |
| 6.2.66 | Event log objects (class_id 7) .....  | 604 |
| 6.2.67 | Inactive objects .....  | 604 |
| 6.3    | AC Electricity related COSEM objects .....  | 605 |
| 6.3.1  | Value group D definitions.....  | 605 |
| 6.3.2  | Electricity ID numbers.....   | 605 |
| 6.3.3  | Billing period values / reset counter entries .....                                     | 606 |
| 6.3.4  | Other electricity related general purpose objects .....                                 | 606 |
| 6.3.5  | Measurement algorithm .....   | 607 |

|                       |            |                             |       |
|-----------------------|------------|-----------------------------|-------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 7/668 |
|-----------------------|------------|-----------------------------|-------|

## COSEM Interface Classes

|            |  |            |
|------------|--|------------|
| 6.3.6      | Metering point ID (electricity related).....                               | 609        |
| 6.3.7      | Electricity related status objects .....                                   | 609        |
| 6.3.8      | List objects – Electricity (class_id = 7).....                             | 610        |
| 6.3.9      | Threshold values .....   | 610        |
| 6.3.10     | Register monitor objects (class_id = 21).....                              | 611        |
| <b>6.4</b> | <b>DC electricity related COSEM objects .....</b>                          | <b>612</b> |
| 6.4.1      | Value group D definitions.....   | 612        |
| 6.4.2      | ID numbers – DC electricity .....  | 612        |
| 6.4.3      | Billing period values / reset counter entries .....                        | 613        |
| 6.4.4      | Other DC electricity related general purpose objects.....                  | 613        |
| 6.4.5      | Measurement algorithm .....  | 614        |
| 6.4.6      | Metering point ID (DC electricity related).....                            | 615        |
| 6.4.7      | DC electricity related status objects .....                                | 615        |
| 6.4.8      | List objects – DC electricity (class_id = 7) .....                         | 616        |
| 6.4.9      | Threshold values .....   | 616        |
| 6.4.10     | Register monitor objects (class_id = 21).....                              | 617        |
| 6.5        | HCA related COSEM objects.....   | 618        |
| 6.5.1      | General.....   | 618        |
| 6.5.2      | ID numbers – HCA.....  | 618        |
| 6.5.3      | Billing period values / reset counter entries - HCA .....                  | 618        |
| 6.5.4      | General purpose objects – HCA .....  | 619        |
| 6.5.5      | Measured Values – HCA.....   | 619        |
| 6.5.6      | Error register objects – HCA .....   | 621        |
| 6.5.7      | List objects – HCA.....  | 621        |
| 6.5.8      | Data profile objects – HCA.....  | 621        |
| 6.6        | Thermal energy meter related COSEM objects .....                           | 622        |
| 6.6.1      | General.....   | 622        |
| 6.6.2      | ID numbers – Thermal energy meter .....                                    | 622        |
| 6.6.3      | Billing period values / reset counter entries - Thermal energy meter ..... | 622        |
| 6.6.4      | General purpose objects – Thermal energy meter .....                       | 623        |
| 6.6.5      | Measured values - Thermal energy meter.....                                | 623        |
| 6.6.6      | Error register objects – Thermal energy meter .....                        | 625        |
| 6.6.7      | List objects – Thermal energy meter.....                                   | 626        |
| 6.6.8      | Data profile objects – Thermal energy meter .....                          | 626        |
| 6.7        | Gas related COSEM objects.....   | 627        |
| 6.7.1      | General.....   | 627        |
| 6.7.2      | ID numbers – Gas .....   | 627        |
| 6.7.3      | Billing period values / reset counter entries – Gas .....                  | 627        |
| 6.7.4      | Other general purpose objects – Gas .....                                  | 627        |
| 6.7.5      | Internal operating status objects – Gas .....                              | 630        |
| 6.7.6      | Measured values – Gas .....  | 630        |
| 6.7.7      | Conversion related factors and coefficients – Gas .....                    | 631        |
| 6.7.8      | Calculation methods – Gas .....  | 632        |
| 6.7.9      | Natural gas analysis .....   | 632        |
| 6.7.10     | List objects – Gas.....  | 633        |
| 6.8        | Water meter related COSEM objects .....                                    | 634        |
| 6.8.1      | General.....   | 634        |

## COSEM Interface Classes

|                       |   |     |
|-----------------------|---|-----|
| 6.8.2                 | ID numbers – water meter.....   | 634 |
| 6.8.3                 | Billing period values / reset counter entries – water meter.....              | 634 |
| 6.8.4                 | General purpose objects – water meter .....                                   | 635 |
| 6.8.5                 | Measured values – water meter .....   | 635 |
| 6.8.6                 | Error register objects – water meter .....                                    | 637 |
| 6.8.7                 | List objects – water meter .....  | 637 |
| 6.8.8                 | Data profile objects – water meter.....                                       | 638 |
| 6.9                   | Coding of OBIS identifications.....   | 638 |
| Annex A (informative) | Additional information on Auto answer and Auto connect ICs .....              | 639 |
| Annex B (informative) | Additional information to M-Bus client (class_id = 72, version 1 & 2) .....   | 641 |
| Annex C (informative) | Additional information on IPv6 setup class (class_id = 48, version = 0) ..... | 645 |
| C.1                   | General.....  | 645 |
| C.2                   | IPv6 addressing.....  | 645 |
| C.3                   | IPv6 header format .....  | 647 |
| C.4                   | IPv6 header extensions .....  | 648 |
| C.4.1                 | Overview.....   | 648 |
| C.4.2                 | Hop-by-Hop options.....   | 649 |
| C.4.3                 | Destination options.....  | 649 |
| C.4.4                 | Routing options .....   | 649 |
| C.4.5                 | Fragment options .....  | 650 |
| C.4.6                 | Security options.....   | 650 |
| Annex D (informative) | Overview of the narrow-band OFDM PLC technology for PRIME networks .....      | 651 |
| Annex E (informative) | Overview of the narrow-band OFDM PLC technology for G3-PLC networks .....     | 652 |
| Index                 | .....   | 660 |

|   |            |
|---|------------|
| Figure 1 – The three steps approach of DLMS/COSEM: Modelling – Messaging – Transporting ..... | 18         |
| Figure 2 – The meaning of the definitions concerning the Image .....                          | 26         |
| Figure 3 – An interface class and its instances .....   | 41         |
| Figure 4 – The COSEM server model.....  | 53         |
| Figure 5 – Combined metering device .....   | 53         |
| Figure 6 – Overview of the interface classes – Part 1 .....                                   | 56         |
| <b>Figure 7 – Overview of the interface classes – Part 2 .....</b>                            | <b>57</b>  |
| <b>Figure 8 – Overview of the interface classes – Part 3 .....</b>                            | <b>58</b>  |
| Figure 9 – Overview of the interface classes - Part 4.....                                    | 59         |
| Figure 10 – The time attributes when measuring sliding demand.....                            | 73         |
| Figure 11 – The attributes in the case of block demand .....                                  | 73         |
| Figure 12 – The attributes in the case of sliding demand (number of periods = 3) .....        | 74         |
| Figure 13 – Image transfer process flow chart.....  | 119        |
| <b>Figure 14 – Image transfer with M-Bus transfer diagram .....</b>                           | <b>124</b> |
| Figure 15 – COSEM model of push operation.....  | 132        |

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
|                       |            | 9/668                       |

## COSEM Interface Classes

|   |            |
|---|------------|
| Figure 16 – Push windows, delays and retries .....  | 134        |
| Figure 17 – COSEM model of data protection.....   | 145        |
| Figure 18 – Example: Read <i>protection_buffer</i> attribute .....  | 147        |
| Figure 19 – Example of managing an array .....  | 164        |
| Figure 20 – The generalized time concept.....   | 172        |
| Figure 21 – State diagram of the Disconnect control IC .....  | 188        |
| Figure 22 – Definition of upper and lower thresholds .....  | 201        |
| Figure 23 – COSEM tariffication model (example) .....   | 206        |
| Figure 24 – COSEM billing model (example) .....   | 207        |
| Figure 25 – Outline Account model .....   | 209        |
| Figure 26 – Diagram of attribute relationships .....  | 210        |
| Figure 27 – Credit States when priority >0 .....  | 221        |
| Figure 28 – Operation of current_credit_status flags.....   | 223        |
| Figure 29 – Interaction of current_credit_amount and available_credit with Token<br>“Credit” and Emergency “Credit” ..... | 230        |
| Figure 30 – Object model of DLMS servers .....  | 303        |
| Figure 31 – Object model of DLMS servers .....  | 325        |
| Figure 32 – Example of a ZigBee® network.....   | 379        |
| Figure 33 – Push windows and delays .....   | 466        |
| <b>Figure 34 – Push windows and delays .....</b>  | <b>477</b> |
| Figure 35 – Data of historical billing periods – example with module 12, VZ = 5 .....                                     | 576        |
| Figure A.1 – Network connectivity example for a GSM/GPRS network.....   | 639        |
| Figure B.1 – Encryption key status diagram .....  | 641        |
| Figure B.2 – Example of SITP transfer with single M-Bus transfer.....   | 643        |
| Figure C.1 – IPv6 address formats.....  | 646        |
| <br>  |            |
| Table 1 – Reserved base_names for SN referencing .....  | 42         |
| Table 2 – Interface class overview .....  | 42         |
| Table 3 – Common data types .....   | 46         |
| Table 4 – List of interface classes by class_id .....   | 59         |
| Table 5 – Enumerated values for physical units.....   | 68         |
| Table 6 – Examples for scaler_unit .....  | 71         |
| Table 7 – Parameters for selective access to the buffer attribute .....   | 82         |
| Table 8 – Parameters for selective access to the buffer attribute .....   | 85         |
| Table 9 – Encoding of selective access parameters with data_index .....   | 92         |
| Table 10 – Example daily billing data captured to <i>compact_buffer</i> .....   | 94         |
| Table 11 – “Compact data” object attributes – Daily billing data example .....  | 95         |
| Table 12 – Example daily billing data read using GET-WITH LIST.....   | 95         |
| Table 13 – Example diagnostic and alarm data captured to <i>compact_buffer</i> .....                                      | 96         |
| Table 14 – “Compact data” object attributes – Diagnostic and Alarm data example .....                                     | 96         |
| Table 15 – Example diagnostic and alarm data read from “Profile generic” <i>buffer</i> .....                              | 96         |
| Table 16 – Example logbook data entries in “Profile generic” <i>buffer</i> .....  | 97         |

## COSEM Interface Classes

|  |     |
|--|-----|
| Table 17 – Example logbook data captured to <i>compact_buffer</i> .....  | 97  |
| Table 18 – “Compact data” object attributes – Logbook data example .....   | 98  |
| Table 19 – Example logbook data read from “Profile generic” <i>buffer</i> .....                                    | 98  |
| Table 20 – Parameters for selective access to the <i>object_list</i> and <i>access_rights_list</i> attribute ..... | 103 |
| Table 21 – Parameters for selective access to the <i>object_list</i> attribute .....                               | 108 |
| Table 22 – <b>Mapping of M-Bus State to image_transfer_status</b> .....  | 122 |
| Table 23 – Key information required to establish data protection keys.....   | 155 |
| Table 24 – Protection parameters of <i>protection_parameters_get</i> attribute .....                               | 156 |
| Table 25 – Protection parameters of <i>protection_parameters_set</i> attribute.....                                | 157 |
| Table 26 – Protection parameters of <i>get_protected_attributes</i> method .....                                   | 158 |
| Table 27 – Protection parameters of <i>set_protected_attributes</i> method .....                                   | 159 |
| Table 28 – Protection parameters of <i>invoke_protected_method</i> method .....                                    | 160 |
| Table 29 – Example values for NCA and CLT .....  | 170 |
| Table 30 – Schedule .....  | 177 |
| Table 31 – Special days table.....   | 177 |
| Table 32 – Disconnect control IC – states and state transitions.....   | 189 |
| Table 33 – Explicit presentation of threshold value arrays .....   | 202 |
| Table 34 – Explicit presentation of action_sets .....  | 202 |
| Table 35 – Credit states .....   | 220 |
| Table 36 – Credit state transitions .....  | 221 |
| <b>Table 37 – Mapping of commodity attribute to Account IC’s currency attribute</b> .....                          | 241 |
| Table 38 – Fatal error register .....  | 249 |
| Table 39 – Mapping IEC 61334-4-512:2001 MIB variables to COSEM IC attributes / methods.....                        | 304 |
| Table 40 – MAC addresses in the S-FSK profile .....  | 306 |
| Table 41 – Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes.....                                 | 326 |
| Table 42 – Mapping of G3-PLC IB attributes to COSEM IC attributes .....  | 341 |
| <b>Table 43 – Mapping of G3-PLC Hybrid PLC &amp; RF IB attributes to COSEM IC attributes</b> .....                 | 343 |
| Table 44 – Bandplans.....  | 348 |
| Table 45 – Use of ZigBee® setup COSEM interface classes.....   | 379 |
| Table 46 – C/D Rule 1 .....  | 396 |
| Table 47 – Parameters for selective access to the <i>object_list</i> attribute .....                               | 436 |
| Table 48 – Parameters for selective access to the <i>object_list</i> and <i>access_rights_list</i> attribute ..... | 440 |
| Table 49 – Parameters for selective access to the <i>object_list</i> and <i>access_rights_list</i> attribute ..... | 442 |
| Table 50 – Parameters for selective access to the <i>object_list</i> attribute .....                               | 450 |
| Table 51 – Parameters for selective access to the <i>object_list</i> attribute .....                               | 456 |
| Table 52 – Parameters for selective access to the <i>object_list</i> attribute .....                               | 462 |
| Table 53 – ADS address elements .....  | 493 |
| Table 54 – Mapping of G3-PLC IB attributes specified in ITU-T G.9903:2013 Amd. 1 to COSEM IC attributes .....      | 525 |

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
|                       |            | 11/668                      |

## COSEM Interface Classes

|  |     |
|--|-----|
| Table 55 – Mapping of G3-PLC IB attributes specified in ITU-T G.9903:2017 to COSEM IC attributes ..... | 526 |
| Table 56 – Use of value group C for abstract objects in the COSEM context .....                        | 574 |
| Table 57 – Representation of various values by appropriate ICs .....                                   | 605 |
| Table 58 – Measuring algorithms – enumerated values .....  | 608 |
| Table 59 – Threshold objects, electricity .....  | 610 |
| Table 60 – Register monitor objects, electricity .....   | 611 |
| Table 61 – Representation of various values by appropriate ICs .....                                   | 612 |
| Table 62 – Measuring algorithms – enumerated values .....  | 615 |
| Table 63 – Threshold objects, DC electricity .....   | 616 |
| Table 64 – Register monitor objects, DC electricity .....  | 617 |
| Table 65 – Digital / Analogue output configurations – enumerated values .....                          | 628 |
| Table 66 – Indexes and index differences .....   | 630 |
| Table 67 – Flow rate .....   | 631 |
| Table 68 – Process values .....  | 631 |
| Table 69 – Conversion related factors and coefficients .....   | 632 |
| Table 70 – Calculation methods .....   | 632 |
| Table 71 – Natural gas analysis .....  | 633 |
| Table B.1 – Encryption key is preset in the slave and cannot be changed .....                          | 642 |
| Table B.2 – Encryption key is preset in the slave and new key is set after installation .....          | 642 |
| Table B.3 – Encryption key is not preset in the slave, but can be set, case a) .....                   | 642 |
| Table B.4 – Encryption key is not preset in the slave, but can be set, case b) .....                   | 642 |
| Table C.1 – IPv6 header vs. IPv6 IC .....  | 648 |
| Table C.2 – Optional IPv6 header extensions vs. IPv6 IC .....  | 649 |

## FOREWORD

### **Copyright**

© Copyright 1997-2021 DLMS User Association

This document is confidential. It may not be copied, nor handed over to persons outside the DLMS User Association.

The copyright is enforced by national and international law. The "Berne Convention for the Protection of Literary and Artistic Works", which is signed by 176 countries world-wide, and other treaties apply.

### **Liability**

DLMS User Association Publications have the form of recommendations for international use. While all reasonable efforts are made to ensure that the technical content of DLMS User Association Publications is accurate, the DLMS User Association cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

No liability shall attach to DLMS User Association or its directors, employees, servants or agents including individual experts and members of its technical committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this DLMS User Association Publication or any other DLMS User Association Publications.

### **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 document has been written by members of DLMS UA Maintenance Working Group.

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 13/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

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 standardisation

The contents of this edition is the basis of future revisions to:

- IEC 62056-6-2, Electricity Metering Data Exchange – The DLMS/COSEM suite – Part 6-2: COSEM interface classes.

### 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· draft IEC 62056-62 (CD versions) and EN 13757-1:2002   |
| Seventh Edition               | 12 <sup>th</sup> 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 <sup>th</sup> September 2007 | DLMS UA | Document restructured, editorial errors in Ed. 7.0 corrected.   |
| Ninth Edition                 | 9 <sup>th</sup> February 2009   | DLMS UA | New elements for smart metering and advanced gas volume and energy conversion added.  |
| Tenth Edition                 | 26 <sup>th</sup> August 2010    | DLMS UA | Brought in line with Green Book.<br>New interface classes and new versions of interface classes added.<br>New OBIS codes added.   |
| Eleventh Edition              | 26th August 2013                | DLMS UA | New interface classes and new versions of existing interface classes added. New OBIS codes added.   |
| Twelfth Edition               | 10 <sup>th</sup> September 2014 | DLMS UA | New interface classes and new versions of existing interface classes added in support of Security management, Payment Metering, Arbitrator, Compact data, Data protection, Push setup, G3-PLC, M-Bus, New OBIS codes added. |
| Twelfth Edition Corrigendum 1 | 21 <sup>st</sup> December 2015  | DLMS UA | Technical and editorial corrections   |
| Edition 12.1                  | 21 <sup>st</sup> December 2015  | DLMS UA | Consolidated edition including the Corrigendum 1.   |
| Edition 12.2                  | 21 <sup>st</sup> December 2016  | DLMS UA | New or updated interface classes for Function Control, Array Manager, NTP setup, Compact data, GSM-related and HS-PLC networks added. New OBIS codes added  |
| Edition 13                    | 8 <sup>th</sup> May 2019        | DLMS UA | SI units, communication port protection, heat cost allocator, thermal energy, water OBI codes added. Additional codes for reactive power added.   |
| Edition 14                    | 31 <sup>st</sup> Aug 2020       | DLMS UA | WiSUN, LPWAN, LTE monitoring, Parameter monitor, support for ISO/IEC 14908 PLC networks added; Push IC now V2; Delta values introduced; new clock objects added   |

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 14/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

| Version    | Date | Author  | Comment   |
|------------|------|---------|-----------|
| Edition 15 |      | DLMS UA | See below |

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 15/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

### List of main technical changes in Edition 15

| <b>Item</b> | <b>Clause</b>  | <b>Description</b>   |
|-------------|--|--|
| 1           | 3.8  | Abbreviation list updated  |
| 2           | 4.4.4.2.7  | Note removed from the description of secret attribute. (Also removed from 5.4.6.2.7, 5.4.7.2.7)  |
| 3           | 4.4.8.2  | Push IC – new version 3  |
| 4           | 4.6.6<br>4.8.3<br>4.13.4<br>4.13.5<br>4.13.6<br>4.13.7<br>4.13.8 | IEC 62055-41 Attributes V0<br>M-Bus client now V2, V1 moved to Clause 5<br>G3-PLC MAC setup now V2, V1 moved to Clause 5<br>G3-PLC 6LoWPAN adaptation layer setup now V3, V2 moved to Clause 5<br>G3-PLC Hybrid RF MAC layer counters V0<br>G3-PLC Hybrid RF MAC setup V0<br>G3-PLC Hybrid 6LoWPAN adaptation layer setup V0 |
| 5           | 5.4.11   | Push IC version 2 moved to here  |
| 6           | 6.3.6  | Meter point ID object was 64 corrected to 61   |
| 7           | 4.3.4  | Figure references corrected  |
| 8           | 4.9  | Classes added to support CoAP protocol   |

## INTRODUCTION

### **Object modelling and data identification**

COSEM was originally developed to address the requirements for interoperability and data security requirements in metering and control applications. The specification is not however limited to metering and control, it can be used to model any type of device that is designed to be connected to a communications network. (The COSEM acronym originally was *Companion Specification for Energy Metering* this no longer applies and has been modified to *Comprehensive Semantic Model for Energy Management* to reflect its wider application.)

COSEM uses *object modelling* techniques to model all functions of devices, without making any assumptions about which functions need to be supported, how those functions are implemented and how the data is 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 standard 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 requirements.

COSEM models the device as a *server* application – see 4.1.7 – used by *client* applications that retrieve data from, provide control information to, and instigate defined actions within the device 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 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 17/668 |
|-----------------------|------------|-----------------------------|--------|

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

## Part 2: COSEM interface classes

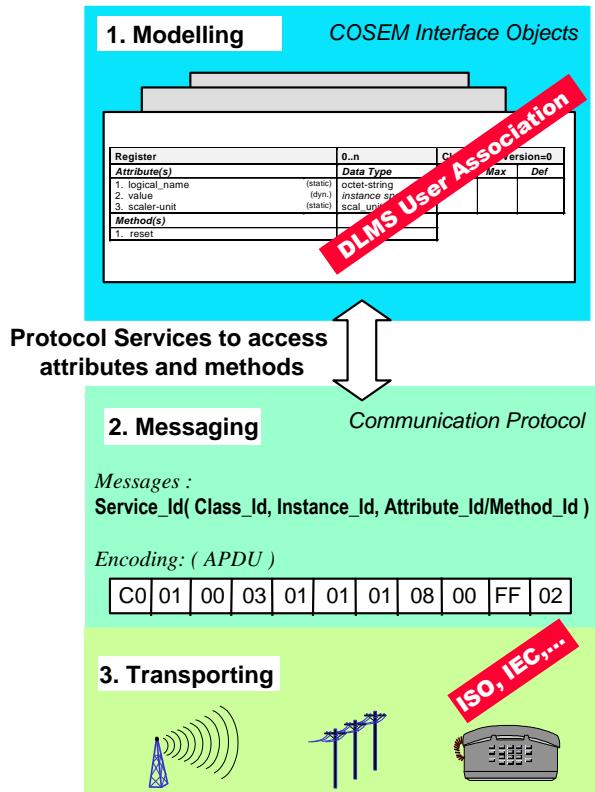
### 1 Scope

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 the two parts of this document. The OBIS object identification system is specified in Part 1 (DLMS UA 1000-1 Ed 15 Part 1:2021). This Part 2 specifies the COSEM interface classes (ICs) and the use of interface objects for modelling the various functions of the devices.

Steps 2 and 3 are specified in the Green Book, DLMS UA 1000-2 Ed.11:2021. 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”.

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 18/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

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

## 2 Referenced Documents

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

DIN 43863-3:1997, *Electricity meters – Part 3: Tariff metering device as additional equipment for electricity meters – EDIS – Energy Data Identification System*

DLMS UA 1000-1 Ed 15 Part 1:2021, OBIS Codes, Blue Book Part 1

DLMS UA 1000-2 Ed. 10:2020, DLMS/COSEM Architecture and Protocols, the “Green Book” Edition 10

DLMS UA 1000-2 Ed.11:2021, Comprehensive Semantic Model for Energy Management, Green Book

DLMS UA 1001-1 DLMS/COSEM Conformance test and certification process, the “Yellow Book”

DLMS UA 1002 COSEM Glossary of terms, the “White Book”

IEC TR 61000-2-8:2002 Electromagnetic compatibility (EMC) – Part 2-8: Environment - Voltage dips and short interruptions on public electric power supply systems with statistical measurement results

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

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

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

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

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

IEC 62053-23:2003 Electricity metering equipment (a.c.) – Particular requirements – Part 23: Static meters for reactive energy (classes 2 and 3)

IEC TR 62055-21:2005 Technical report Electricity metering - Payment systems- Part 21: Framework for standardization. 2005-8

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 19/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

IEC 62055-41:2018 Electricity metering - Payment systems - Part 41: Standard transfer specification (STS) - Application layer protocol for one-way token carrier systems

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

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

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

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

IEC 62056-3-1:2021 Electricity metering data exchange – The DLMS/COSEM suite – Part 3-1: Use of local area networks on twisted pair with carrier signalling

IEC 62056-46:2002/AMD1:2006 ,Electricity metering – Data exchange for meter reading, tariff and load control – Part 46: Data link layer using HDLC protocol

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

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

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

IEC 62056-8-6: 2017 Electricity metering data exchange – The DLMS/COSEM suite – Part 8-6: High speed PLC ISO/IEC 12139-1 profile for neighbourhood networks

IEC 62056-8-8:2020 Electricity metering data exchange - The DLMS/COSEM suite - Part 8-8: Communication profile for ISO/IEC 14908 series networks

ISO/IEC 8802-2:1998 IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 2: Logical Link Control

ISO/IEC 12139-1:2009 Information technology —Telecommunications and information exchange between systems — Powerline communication (PLC) — High speed PLC medium access control (MAC) and physical layer (PHY) — Part 1: General requirements

ISO/IEC 14908-1:2012 Interconnection of information technology equipment – Control network protocol Part Protocol stack

ISO/IEC/IEEE 60559:2011 Information technology – Microprocessor Systems – Floating-Point arithmetic

ISO 4217 Codes for the representation of currencies and funds

ITU-T E.212 (05.2008) SERIES E: OVERALL NETWORK OPERATION,TELEPHONE SERVICE, SERVICE OPERATION AND HUMAN FACTORS - International operation –

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 20/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

Maritime mobile service and public land mobile service - The international identification plan for public networks and subscriptions

3GPP TS 24.008 V13.7.0 (2016-10) Technical Specification Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3

3GPP TS 24.301 V13.4.0 (2016-01) Technical Specification Group Core Network and Terminals; Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3

3GPP TS 24.301 V13.11.0 (2018-01) Technical Specification Group Core Network and Terminals; Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3

3GPP TS 27.007 V16.1.0 (2019-06) Technical Specification Group Core Network and Terminals; AT command set for User Equipment (UE)

3GPP TS 36.101 V15.4.0 (2019-01) Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception

3GPP TS 36.133 V13.11.0 (2018-03) Technical Specification LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Requirements for support of radio resource management

3GPP TS 36.133 V14.4.0 (2017-06) Technical Specification LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Requirements for support of radio resource management

3GPP TS 36.213 V15.5.0 (2019-05) Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures

3GPP TS 36.304 V13.8.0 (2018-01) Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) procedures in idle mode

3GPP TS 36.321 V15.5.0 (2019-05) Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification

3GPP TS 36.331 V15.5.1 (2019-05) Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification

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

NOTE This Recommendation is referenced in version 0 of the G3-PLC setup classes.

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

NOTE This Recommendation is referenced in version 1 of the G3-PLC setup classes.

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 21/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

ITU-T G.9903:2017 SERIES G: TRANSMISSION SYSTEMS AND MEDIA, DIGITAL SYSTEMS AND NETWORKS - Access networks – In premises networks - Narrowband orthogonal frequency division multiplexing power line communication transceivers for G3-PLC networks

NOTE This Recommendation is referenced in current version of the G3-PLC setup classes.

ITU-T G.9903 Amd. 1:2021, 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

NOTE This Recommendation is referenced in version 2 of the G3-PLC setup classes.

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

EN 834:1994 Heat cost allocators for the determination of the consumption of room heating radiators – Appliances with electrical energy supply

EN 1434-1:2015 Heat meters – Part 1: General requirements

EN 1434-2:2015 Heat meters – Part 2: Constructional requirements

EN 13757-1:2014 Communication system for meters – Part 1: Data exchange

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

EN 13757-2:2018, Communication system for and remote reading of meters – Part 2: Physical and link layer

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

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

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

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

EN 13757-3:2018, Communication systems for meters – Part 3: Dedicated application layer

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

EN 13757-4:2013, Communication system for and remote reading of meters – Part 4: Wireless meter (Radio meter reading for operation in SRD bands)

EN 13757-4:2019, Communication system for and remote reading of meters – Part 4: Wireless meter (Radio meter reading for operation in SRD bands)

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

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 22/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

EN 13757-7:2018 Communication system for meters – Part 7: Transport and security services

IEEE 802.15.4: 2006 also available as ISO/IEC/IEEE 8802-15-4 Ed 1.0 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.

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

ETSI EN 303 204 V2.1.2 (2016-09), Network Based Short Range Devices (SRD); Radio equipment to be used in the 870 MHz to 876 MHz frequency range with power levels ranging up to 500 mW; Harmonised Standard covering the essential requirements of article 3.2 of the Directive 2014/53/EU

ANSI C12.19:2012 American National Standard For Utility Industry End Device Data Tables

ZigBee® 053474 ZigBee® Specification. The specification can be downloaded free of charge from <https://www.zigbee.org/zigbee-for-developers/zigbee-pro/>

[FANSPEC] Wi-SUN Alliance: Field Area Network Working Group (FANWG):Technical Profile Specification:Field Area Network:Version 1v26.

[PHYSPEC] Wi-SUN Alliance: PHY Working Group (PHYWG) Wi-SUN PHY Specification Revision 1V02

LoRaWAN 1.0.3 LoRaWAN® Specification v1.0.3 <https://lora-alliance.org/resource-hub/lorawanr-specification-v103>

The following RFCs are available online from the Internet Engineering Task Force (IETF):

<http://www.ietf.org/rfc/std-index.txt>, <http://www.ietf.org/rfc/>

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

RFC 768 User Datagram Protocol

RFC 791 Internet Protocol (Also: IETF STD 0005), 1981

RFC 793 Transmission Control Protocol

RFC 1144 Compressing TCP/IP Headers for Low-Speed Serial Links, 1990

RFC 1213 Management Information Base for Network Management of TCP/IP-based internets: MIB-II

RFC 1332 The PPP Internet Protocol Control Protocol (IPCP), 1992, Updated by: RFC 3241. Obsoletes: RFC 1172

RFC 1570 PPP LCP Extensions, 1994

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

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 23/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

- RFC 1662 PPP in HDLC-like Framing, (Also: IETF STD 0051), 1994, Obsoletes: RFC 1549
- RFC 1994 PPP Challenge Handshake Authentication Protocol (CHAP), 1996. Obsoletes: RFC 1334
- RFC 2433 PPP CHAP Extension, 1998
- RFC 2460 Internet Protocol, Version 6
- RFC 2474 Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, 1998
- RFC 2507 IP Header Compression, 1999
- RFC 2508 Compressing IP/UDP/RTP Headers for Low-Speed Serial Links, 1999
- RFC 2759 Microsoft PPP CHAP Extensions, Version 2, 2000
- RFC 2986 PKCS #10 v1.7: Certification Request Syntax Standard
- RFC 3095 ROBust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed, 2001
- RFC 3241 Robust Header Compression (ROHC) over PPP, 2002. Updates: RFC1332
- RFC 3315 Dynamic Host Configuration Protocol for IPv6 (DHCPv6)
- RFC 3513 Internet Protocol Version 6 (IPv6) Addressing Architecture, 2003
- RFC 3544 IP Header Compression over PPP, 2003
- RFC 3748 Extensible Authentication Protocol (EAP), 2004
- RFC 4291 IP Version 6 Addressing Architecture
- RFC 4443, Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification
- RFC 4944 Transmission of IPv6 Packets over IEEE 802.15.4 Networks
- RFC 4861 Neighbor Discovery for IP version 6 (IPv6), 2007
- RFC 4944 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
- RFC 5216 The EAP-TLS Authentication Protocol
- RFC 5280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 2008
- RFC 5905 Network Time Protocol Version 4: Protocol and Algorithms Specification, 2010.

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 24/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

RFC 6206 The Trickle Algorithm

RFC 6282 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

RFC 6550 RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks

RFC 6775 Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs), 2012

RFC 7731 Multicast Protocol for Low-Power and Lossy Networks (MPL)

RFC 7774 Multicast Protocol for Low-Power and Lossy Networks (MPL), Parameter Configuration Option for DHCPv6

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

### 3 Terms, definitions and abbreviated terms

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

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at <http://www.electropedia.org/>
- ISO Online browsing platform: available at <http://www.iso.org/obp>

#### 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

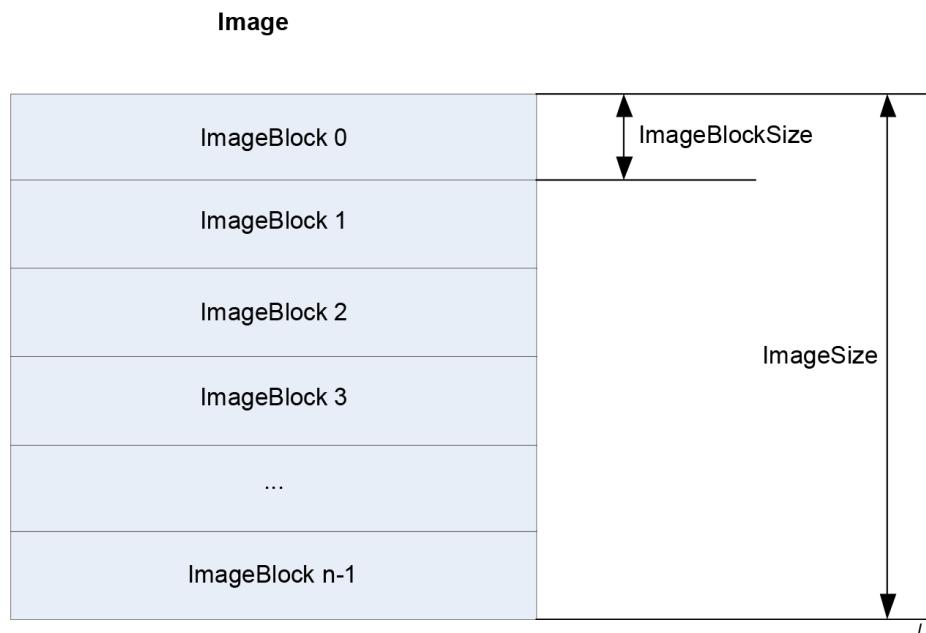
size of ImageBlock expressed in octets

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 25/668 |
|-----------------------|------------|-----------------------------|--------|

**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

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 26/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

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

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

### 3.2.5

#### **registered system**

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

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

### 3.2.6

#### **reporting system**

server system which issues a DiscoverReport

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

### 3.2.7

#### **sub-slot**

time needed to transmit two bytes by the physical layer

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

### 3.2.8

#### **timeslot**

time needed to transmit a physical frame

Note 1 to entry: As specified in IEC 61334-5-1:2001, 3.3.1, a physical frame comprises 2 bytes preamble, 2 bytes start subframe delimiter, 38 bytes PSDU and 3 bytes pause.

## 3.3 Terms and definitions related to the PRIME NB OFDM PLC setup ICs (see 4.12)

### Definitions related to the physical layer

#### 3.3.1

##### **base node**

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

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

#### 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]

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 27/668 |
|-----------------------|------------|-----------------------------|--------|

**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]

**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]

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 28/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

**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 the ISO/IEC 14908 setup ICs (see 4.19)**

**3.4.1****domain**

logical network that is a unit for addressing

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

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

**3.4.2****node**

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

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

**3.4.3****subnet**

set of nodes accessible through the same link layer protocol

**3.4.4****transaction**

sequence of messages that are correlated together

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

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

**3.5.1****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)

**3.5.2****IHD**

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

**3.5.3****install code**

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

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 29/668 |
|-----------------------|------------|-----------------------------|--------|

**3.5.4**

**link key \***

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

**3.5.5**

**MAC address/IEEE address**

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

**3.5.6**

**ZigBee®**

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

**3.5.7**

**ZigBee® client**

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

**3.5.8**

**ZigBee® coordinator \***

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

**3.5.9**

**ZigBee® cluster**

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

**3.5.10**

**ZigBee® mirror**

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

**3.5.11**

**ZigBee® PRO**

an alternative name for the ZigBee® 2007 protocol. ZigBee® 2007, now the current stack release, contains two stack profiles, stack profile 1 (simply called ZigBee®), for home and light commercial use, and stack profile 2 (called ZigBee® PRO). ZigBee® PRO offers more features, such as multi-casting, many-to-one routing and high security with Symmetric-Key Key Exchange (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

**3.5.12**

**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

**3.5.13**

**ZigBee® server**

this is similar to the role of the DLMS server

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 30/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

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

### 3.5.14

#### 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

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

#### 3.6.1

##### account

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

#### 3.6.2

##### available

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

#### 3.6.3

##### charge

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.

#### 3.6.4

##### credit mode

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

#### 3.6.5

##### 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

#### 3.6.6

##### 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

#### 3.6.7

##### 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

#### 3.6.8

##### emergency credit

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

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 31/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

Note 1 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.

### 3.6.9

#### **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

### 3.6.10

#### **friendly credit**

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).

### 3.6.11

#### **Head End System**

##### **HES**

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

### 3.6.12

#### **Home Area Network**

##### **HAN**

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

### 3.6.13

#### **in use**

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.

### 3.6.14

#### **load limiting**

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.

### 3.6.15

#### **local communications**

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

### 3.6.16

#### **manual entry**

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

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 32/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

**3.6.17****managed payment mode**

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

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

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

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

**3.6.18****payment metering installation**

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

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

**3.6.19****prepayment mode**

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

**3.6.20****post-payment**

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

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

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

**3.6.21****remote communications**

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

**3.6.22****repayable**

credit\_types such as emergency credit where an amount added to *current\_credit\_amount* of a "Credit" object has to be repaid before the Credit is selectable again

**3.6.23****reserved credit**

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

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

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 33/668 |
|-----------------------|------------|-----------------------------|--------|

**3.6.24****selectable**

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

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

**3.6.25****selected/invoked**

specific state of a “Credit” object where the value of *current\_credit\_amount* is included in the calculation of *available\_credit* in the related “Account”

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

**3.6.26****service**

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

**3.6.27****social credit**

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

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

Note 2 to entry: Social credits are modelled of “Credit” objects of type *emergency\_credit*, *time\_based\_credit*, *consumption\_based\_credit*.

**3.6.28****temporary debt**

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

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

**3.6.29****token**

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

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

**3.6.30****token carrier**

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

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 34/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

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.

### 3.6.31

#### **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.

### 3.6.32

#### **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

### 3.6.33

#### **vend**

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.

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

### 3.7.1

#### **action**

operation that can be requested locally or remotely from the server

### 3.7.2

#### **actor**

entity requesting an action

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

### 3.7.3

#### **arbitrator**

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

## 3.8 Abbreviated terms

| Abbreviation | Explanation  |
|--------------|--|
| 3GPP         | 3 <sup>rd</sup> Generation Partnership Project     |
| 6LoWPAN      | IPv6 over Low-Power Wireless Personal Area Network |
| AA           | Application Association                            |
| ABP          | Activation by Personalisation                      |
| AARE         | A-Associate Response – an APDU of the ACSE         |
| AARQ         | A-Associate Request – an APDU of the ACSE          |
| ACSE         | Association Control Service Element                |
| <b>ADD</b>   | <b>Automated Device Discovery</b>                  |

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 35/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

| <b>Abbreviation</b> | <b>Explanation</b>   |
|---------------------|--|
| APDU                | Application layer protocol data unit   |
| 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)  |
| ARFCN               | Absolute radio-frequency channel number  |
| ASE                 | Application Service Element  |
| ATM                 | Automated Topology Management  |
| A-XDR               | Adapted Extended Data Representation (IEC 61334-6)                                     |
| base_name           | The short_name corresponding to the first attribute ("logical_name") of a COSEM object |
| BCD                 | Binary Coded Decimal   |
| BER                 | Bit Error Rate   |
| CBCP                | CallBack Control Protocol (PPP)  |
| CC                  | Current Credit (S-FSK PLC profile)   |
| CDMA                | Code Division Multiple Access  |
| CENELEC             | European Committee for Electrotechnical Standardization                                |
| CHAP                | Challenge Handshake Authentication Protocol  |
| CIASE               | Configuration Initiation Application Service Element (S-FSK PLC profile)               |
| class_id            | Interface class identification code  |
| CLI                 | Calling Line Identity  |
| COSEM               | Comprehensive Semantic Model for Energy Management                                     |
| COSEM object        | An instance of a COSEM interface class   |
| CPAS                | Common Part Adaptation Sublayer  |
| CRC                 | Cyclic Redundancy Check  |
| CSAP                | Client Service Access Point  |
| CSD                 | Circuit Switched Data  |
| CSMA                | Carrier Sense Multiple Access  |
| CtoS                | Client to Server challenge   |
| CU                  | Currently Unused   |
| DC                  | Delta credit (S-FSK PLC profile)   |
| DC                  | Data concentrator  |
| DHCP                | Dynamic Host Configuration Protocol  |
| DIB                 | Data Information Block (M-Bus)   |
| DIF                 | Data Information Field (M-Bus)   |
| DL                  | Data Link  |
| DLMS                | Device Language Message Specification  |
| DLMS UA             | DLMS User Association  |

## COSEM Interface Classes

| <b>Abbreviation</b> | <b>Explanation</b>  |
|---------------------|---|
| DLPDU               | Data Link Protocol Data Unit  |
| DNS                 | Domain Name Server  |
| DSCP                | Differentiated Services Code Point  |
| DSSID               | Direct Switch ID  |
| EAP                 | Extensible Authentication Protocol  |
| EARFCN              | Enhanced Absolute radio-frequency channel number  |
| EDGE                | Enhanced Data rates for GSM Evolution   |
| EMC                 | Emergency Credit (in relation to payment metering)  |
| ERP                 | Enterprise Resource Planning  |
| EUI-48              | 48-bit Extended Unique Identifier   |
| EUI-64              | 64-bit Extended Unique Identifier   |
| E-UTRA              | Evolved UMTS Terrestrial Radio Access   |
| FCC                 | Federal Communications Commission   |
| FFD                 | Full-Function Device  |
| FIFO                | First-In-First-Out  |
| F/R                 | Fragmentation and Reassembly  |
| 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 <a href="http://www.hartcomm.org/">http://www.hartcomm.org/</a> (in relation with the Sensor manager interface class) |
| HDLC                | High-level Data Link Control  |
| HES                 | Head End System   |
| HHT                 | Hand Held Terminal  |
| HLS                 | High Level Security Authentication  |
| HSDPA               | High-Speed Downlink Packet Access   |
| HS-PLC              | High-Speed Power Line Carrier   |
| IANA                | Internet Assigned Numbers Authority   |
| IB                  | Information Base  |
| IC                  | Interface Class (COSEM)   |
| IC                  | Initial credit (S-FSK PLC profile)  |
| IEC                 | International Electrotechnical Commission   |
| IEEE                | Institute of Electrical and Electronics Engineers   |
| IETF                | Internet Engineering Task Force   |
| IPCP                | Internet Protocol Control Protocol  |
| IPv4                | Internet Protocol version 4   |
| IPv6                | Internet Protocol version 6   |
| ISO                 | International Organization for Standardization  |

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 37/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

| <b>Abbreviation</b> | <b>Explanation</b>  |
|---------------------|---|
| 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   |
| M2M                 | Machine to Machine  |
| MAC                 | Medium Access Control   |
| M-Bus               | Meter Bus   |
| MCC                 | Mobile Country Code   |
| MD5                 | Message Digest Algorithm 5  |
| MIB                 | Management Information Base (S-FSK PLC profile)   |
| MID                 | Measuring Instruments Directive 2004/22/EC of the European Parliament and of the Council  |
| MMO                 | Matyas-Meyer-Oseas hash (ZigBee ® term)   |
| MNC                 | Mobile Network Code   |
| MPAN                | (UK term) Meter Point Access Number – reference of the location of the Electricity meter on the electricity distribution network. |
| MPDU                | MAC Protocol Data Unit  |
| MSB                 | Most Significant Bit  |
| MSDU                | MAC Service Data Unit   |
| MT                  | Mobile Termination  |
| NB                  | Narrow-band   |
| ND                  | Neighbour Discovery   |
| <b>NRSRP</b>        | <b>Narrowband Reference Signal Received Power</b>   |
| <b>NRSRQ</b>        | <b>Narrowband Reference Signal Received Quality</b>   |
| NTP                 | Network Time Protocol   |
| o                   | optional  |
| OBIS                | OBject Identification System  |

## COSEM Interface Classes

| <b>Abbreviation</b> | <b>Explanation</b>   |
|---------------------|--|
| 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 <sup>1</sup> ) and ZigBee ® |
| Pad                 | Padding  |
| PAP                 | Password Authentication Protocol   |
| PCLK                | Pre-Configured Link Key (ZigBee® term)   |
| PDU                 | Protocol Data Unit   |
| PhL, PHY            | Physical Layer   |
| PIB                 | PLC Information Base   |
| <b>PID</b>          | <b>Program Identifier</b>  |
| PIN                 | Personal Identity Number   |
| PLC                 | Power Line Carrier   |
| PLMN                | Public Land Mobile Network   |
| PNPDU               | Promotion Needed PDU   |
| POS                 | Point Of Sale (Payment metering)   |
| POS                 | Personal Operating Space (ZigBee ®)  |
| PPDU                | Physical Protocol Data Unit  |
| PPP                 | Point-to-Point Protocol  |
| PSTN                | Public Switched Telephone Network  |
| QoS                 | Quality of Service   |
| RB                  | Radio Band   |
| REJ PDU             | Reject Protocol Data Unit  |
| RFC                 | Request for Comments; a document published by the Internet Engineering Task Force  |
| RFD                 | Reduced Function Device  |
| ROHC                | Robust Header Compression  |
| RREP                | Route Reply  |
| RREQ                | Route Request  |
| RRER                | Route Error  |
| RSRQ                | Reference Signal Received Quality  |
| RSRP                | Reference Signal Received Power  |
| RSSI                | Received Signal Strength Indication (ZigBee® term)                                 |
| SAP                 | Service Access Point   |
| SAS                 | Startup Attribute Set (ZigBee® term )  |
| SCP                 | Shared Contention Period   |
| <b>SDU</b>          | <b>Service Data Unit</b>   |
| SE                  | Smart Energy   |
| SEP                 | Smart Energy Profile (ZigBee® term )   |
| S-FSK               | Spread – Frequency Shift Keying  |
| SHA                 | Secure Hash Algorithm  |
| SI                  | International System of Units ( <i>Système International d'Unités</i> )            |
| SID                 | Switch identifier  |
| SITP                | Secure Information Transfer Protocol   |

## COSEM Interface Classes

| Abbreviation   | Explanation   |
|--|---|
| SMS  | Short Message Service   |
| SMTP   | Simple Mail Transfer Protocol   |
| SN   | Short Name  |
| SNA  | Subnetwork Address  |
| SSAS   | Service Specific Adaptation Sublayer  |
| SSCS   | Service Specific Convergence Layer  |
| StoC   | Server to Client Challenge  |
| TAB  | In the case of the EURIDIS profiles without DLMS and without DLMS/COSEM: data code.<br>In the case of profiles using DLMS or DLMS/COSEM: value at which the equipment is programmed for Discovery |
| TABi   | List of TAB field   |
| TCC  | Transmission Control Code (IPv4)  |
| TCP  | Transmission Control Protocol   |
| TFTP   | Trivial File Transfer Protocol  |
| TOU  | Time of use   |
| TTL  | Time To Live  |
| UDP  | User Datagram Protocol  |
| UMTS   | Universal Mobile Telecommunications System  |
| UNC  | Unconfigured (S-FSK PLC profile)  |
| UTC  | Coordinated Universal Time  |
| VIB  | Value Information Block (M-Bus)   |
| VIF  | Value Information Field (M-Bus)   |
| VZ   | Billing period counter (Form <i>Vorwertzähler</i> in German, see DIN 43863-3)   |
| wake-up  | trigger the meter to connect to the communication network to be available to a client (e.g. HES)  |
| WAN  | Wide Area Network   |
| wM-Bus   | Wireless M-Bus  |
| ZTC  | ZigBee® Trust Center  |
| 1) In the case of the G3-PLC technology, PAN may be defined as PLC Area Network. |   |

## 4 The COSEM interface classes

### 4.1 Basic principles

#### 4.1.1 General

This Clause 4.1 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.

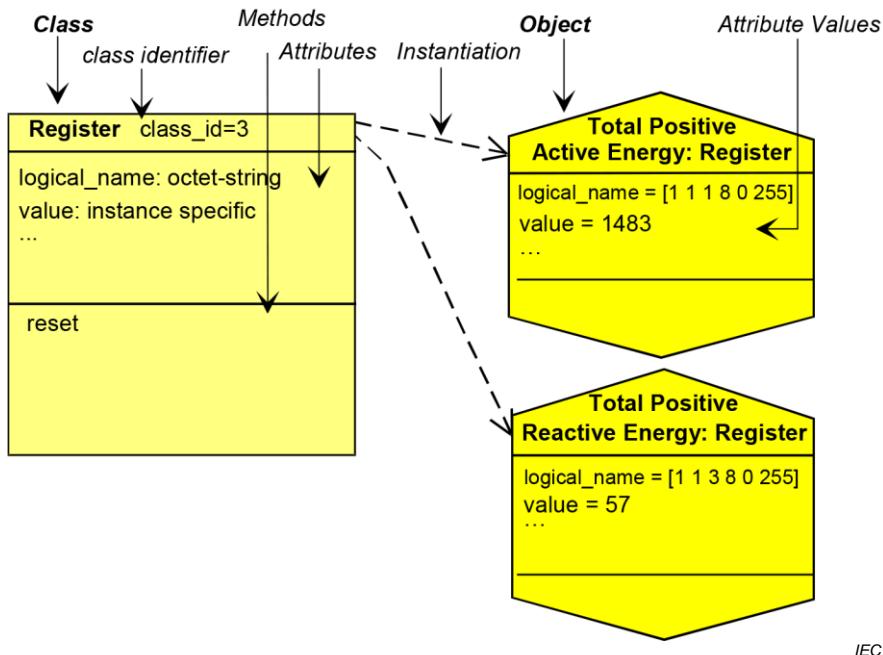
|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 40/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

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:



IEC

**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 DLMS UA 1000-1 Ed 15 Part 1:2021). 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.

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 41/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

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 DLMS UA 1000-2 Ed.11:2021, 9.5.

### 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

#### 4.1.4.1 Overview

This subclause 4.1.4 describes the notation used to define the ICs.

A short text describes the functionality and application of the IC. Table 2 provides 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 in Table 2.

**Table 2 – Interface class overview**

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

**4.1.4.2 Class name**

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

NOTE Interface classes names are mentioned in quotation marks.

**4.1.4.3 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.

**4.1.4.4 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.

**class\_id-s 32768 to 32778 are allocated to the STS Association.**

**4.1.4.5 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.**

Version numbers are to be allocated by the DLMS User Association.

**4.1.4.6 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).

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

NOTE Attribute names use the underscore notation. When mentioned in the text they are in *italic*. Example: *logical\_name*.

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 43/668 |
|-----------------------|------------|-----------------------------|--------|

**4.1.4.7 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 DLMS UA 1000-1 Ed 15 Part 1:2021.

**4.1.4.8 Data type**

Defines the data type of an attribute; see 4.1.5.

**4.1.4.9 Min.**

Specifies if the attribute has a minimum value.

*X* The attribute has a minimum value.

<empty> The attribute has no minimum value.

**4.1.4.10 Max.**

Defines if the attribute has a maximum value.

*X* The attribute has a maximum value.

<empty> The attribute has no maximum value.

**4.1.4.11 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.

**4.1.4.12 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.

**4.1.4.13 Specific methods**

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 Method names use the underscore notation. When mentioned in the text they are in *italic*. Example: *add\_object*.

**4.1.4.14 m/o**

Defines if the method is mandatory or optional.

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 44/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

*m (mandatory)* The method is mandatory.

*o (optional)* The method is optional.

### 4.1.4.15 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).

### 4.1.4.16 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 DLMS UA 1000-2 Ed.11:2021, Clause 9.

### 4.1.4.17 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.

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

- “Profile generic” objects, **buffer** attribute;
- “Association SN” objects, **object\_list** and **access\_rights\_list** attribute;
- “Association LN” objects, **object\_list** attribute;
- “Compact data”, objects, **compact\_buffer** attribute;
- “Push” objects, **push\_object\_list** attribute;
- “Data protection” objects, **protection\_object\_list** attribute **get\_protected\_attributes** method and **set\_protected\_attributes** method.

## 4.1.5 Common data types

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

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 45/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

**Table 3 – Common data types**

| Type description           | Tag <sup>a</sup> | 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...<br>2 147 483 647      |
| double-long-unsigned       | [6]              | Unsigned32   | 0...4 294 967<br>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 <sup>63</sup> ...2 <sup>63</sup> -1 |
| long64-unsigned            | [21]             | Unsigned64   | 0...2 <sup>64</sup> -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,<br>see 4.1.6.2.         |
| float64                    | [24]             | OCTET STRING (SIZE(8))   |   |
| date-time <sup>b</sup>     | [25]             | OCTET STRING SIZE(12))   | For formatting,<br>see 4.1.6.1.         |
| date                       | [26]             | OCTET STRING (SIZE(5))   |   |
| time                       | [27]             | OCTET STRING (SIZE(4))   |   |
| delta-integer              | [28]             | Integer8   | -128...127                              |
| delta-long                 | [29]             | Integer16  | -32 768...32 767                        |
| delta-double-long          | [30]             | Integer32  | -2 147 483 648...<br>2 147 483 647      |
| delta-unsigned             | [31]             | Unsigned8  | 0...255                                 |
| delta-long-unsigned        | [32]             | Unsigned16   | 0...65 535                              |
| delta-double-long-unsigned | [33]             | Unsigned32   | 0...4 294 967<br>295                    |
| -- complex data types      |                  |  |   |
| array                      | [1]              | The elements of the array are defined in the <i>Attribute</i> or <i>Method description</i> section of a COSEM IC specification.                        |   |
| structure                  | [2]              | The elements of the structure are defined in the <i>Attribute</i> or <i>Method description</i> section of a COSEM IC specification.                    |   |

## COSEM Interface Classes

| Type description  | Tag <sup>a</sup> | Definition   | Value range |
|---|------------------|--|-------------|
| 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. |             |
| <sup>a</sup> The tags are as defined in DLMS UA 1000-2 Ed.11:2021, 9.5                                    |                  |  |             |
| <sup>b</sup> The type-description for date-time has been harmonised as date-time throughout the document. |                  |  |             |

### 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          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

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 47/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

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”.

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

The elements dayOfMonth and dayOfWeek shall be interpreted together:

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

Examples:

- 1) year = 0xFFFF, month = 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: **first Friday after 22nd** March in every year (**fourth Friday**);
- 6) year = 0xFFFF, month = 0x0A, dayOfMonth = 0x16, dayofWeek = 0x07: **first Sunday after 22nd** October in every year (**fourth Sunday**);
- 7) year = 0x07DE, month = 0x08, dayOfMonth = 0x13, (2014.08.13, Wednesday) dayofWeek = 0x02 (Tuesday): error, as the dayOfMonth and dayOfWeek in the given year and month do not match.

*time* OCTET STRING (SIZE(4))

```
{  
    hour,  
    minute,  
    second,  
    hundredths  
}
```

## COSEM Interface Classes

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".

```
date-time      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  
}
```

The elements of *date* and *time* are encoded as defined above. Some may be set to "not specified" as defined above.

In addition:

deviation: interpreted as long  
range -720...+720 in minutes of local time to UTC  
0x8000 = not specified

Deviation highbyte and deviation lowbyte represent the 2 bytes of the long.

clock\_status interpreted as unsigned. The bits are defined as follows:

|              |                                     |
|--------------|-------------------------------------|
| bit 0 (LSB): | invalid <sup>a</sup> value,         |
| bit 1:       | doubtful <sup>b</sup> value,        |
| bit 2:       | different clock base <sup>c</sup> , |
| bit 3:       | invalid clock status <sup>d</sup> , |
| bit 4:       | reserved,                           |
| bit 5:       | reserved,                           |
| bit 6:       | reserved,                           |
| bit 7 (MSB): | daylight saving active <sup>e</sup> |

0xFF = not specified

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 49/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

- NOTE 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.
- NOTE 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.
- NOTE 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`.
- NOTE 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.
- NOTE e Flag set to true: the transmitted time contains the daylight saving deviation (summer time).  
Flag set to false: the transmitted time does not contain daylight saving deviation (normal time).

### 4.1.6.2 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                        |           |
| <i>msb</i> | <i>lsb</i> | <i>msb</i><br><i>lsb</i> | ...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                        |           |
| <i>msb</i> | <i>lsb</i> | <i>msb</i><br><i>lsb</i> | ...order  |

where:

- *s* is the sign bit;

## COSEM Interface Classes

- $e$  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   | Bits 30-23                      | Bits 22-0                                   |
| Sign bit | Exponent field: <b>01111111</b> | Significand                                 |
| <b>0</b> | Decimal value of exponent field | <b>1.000000000000000000000000000000</b>     |
| 0: +     | and exponent: $127 - 127 = 0$   | Decimal value of the significand: 1.0000000 |
| 1: -     |                                 |   |

**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 F0 00 00 00 00 00 00.

## COSEM Interface Classes

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

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

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.

#### 4.1.6.3 Null-data and delta-value encoding

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

With null-data encoding, the value may be replaced by “null-data” if it can be unambiguously recovered from the previous value (for example for time: if it can be calculated from the previous value and capture\_period; or for a value: if it is equal to the previous value).

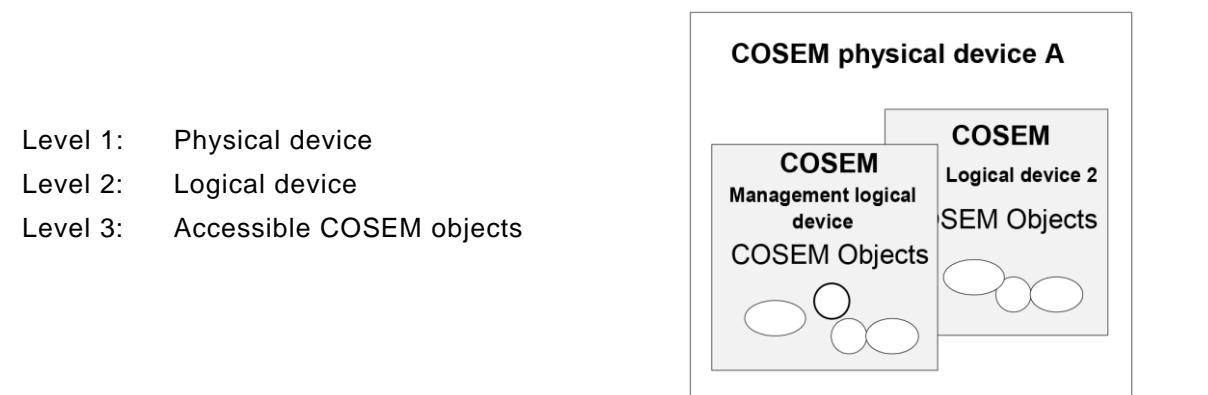
With delta-value encoding the original value may be replaced by a delta-value as follows:

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

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

#### 4.1.7 The COSEM server model

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



**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.

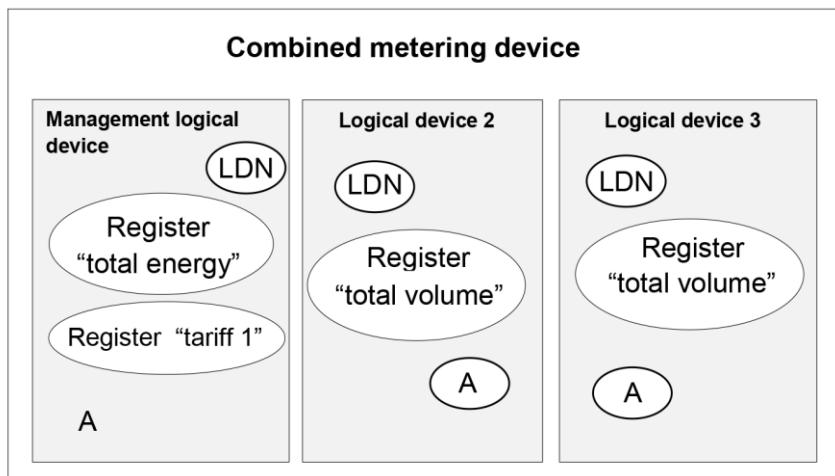
Physical device

Logical device

Objects

LDN: COSEM logical device name object

A: Association object

**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)

Each COSEM logical device can be identified by its unique COSEM LDN. The LDN is defined as a string of up to 16 octets. For the data types see 4.4.5 and 6.2.35.

The first three octets shall carry the manufacturer identifier<sup>1</sup>. The manufacturer shall ensure that the LDN, starting with the three octets identifying the manufacturer and followed by up to 13 octets, is unique for each and every LD manufactured.

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

##### 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:

---

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 53/668 |
|-----------------------|------------|-----------------------------|--------|

- 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.

For identifying the firmware the following objects are mandatory:

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

NOTE The digital signature algorithm is not specified here.

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

The following objects may be optionally present:

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

#### **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

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 54/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

logical devices within the physical device. The SAP assignment object shall be readable at least by the public client.

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

### 4.1.9 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 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.

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.

For a description of these security mechanisms, see DLMS UA 1000-2 Ed.11:2021DLMS UA 1000-2 Ed.11:2021DLMS UA 1000-2 Ed.11:2021 DLMS UA 1000-2 Ed.11:2021, 9.2.

## 4.2 Overview of the COSEM interface classes

The ICs defined currently and the relations between them are shown in Figure 6,Figure 7, Figure 8 and Figure 9.

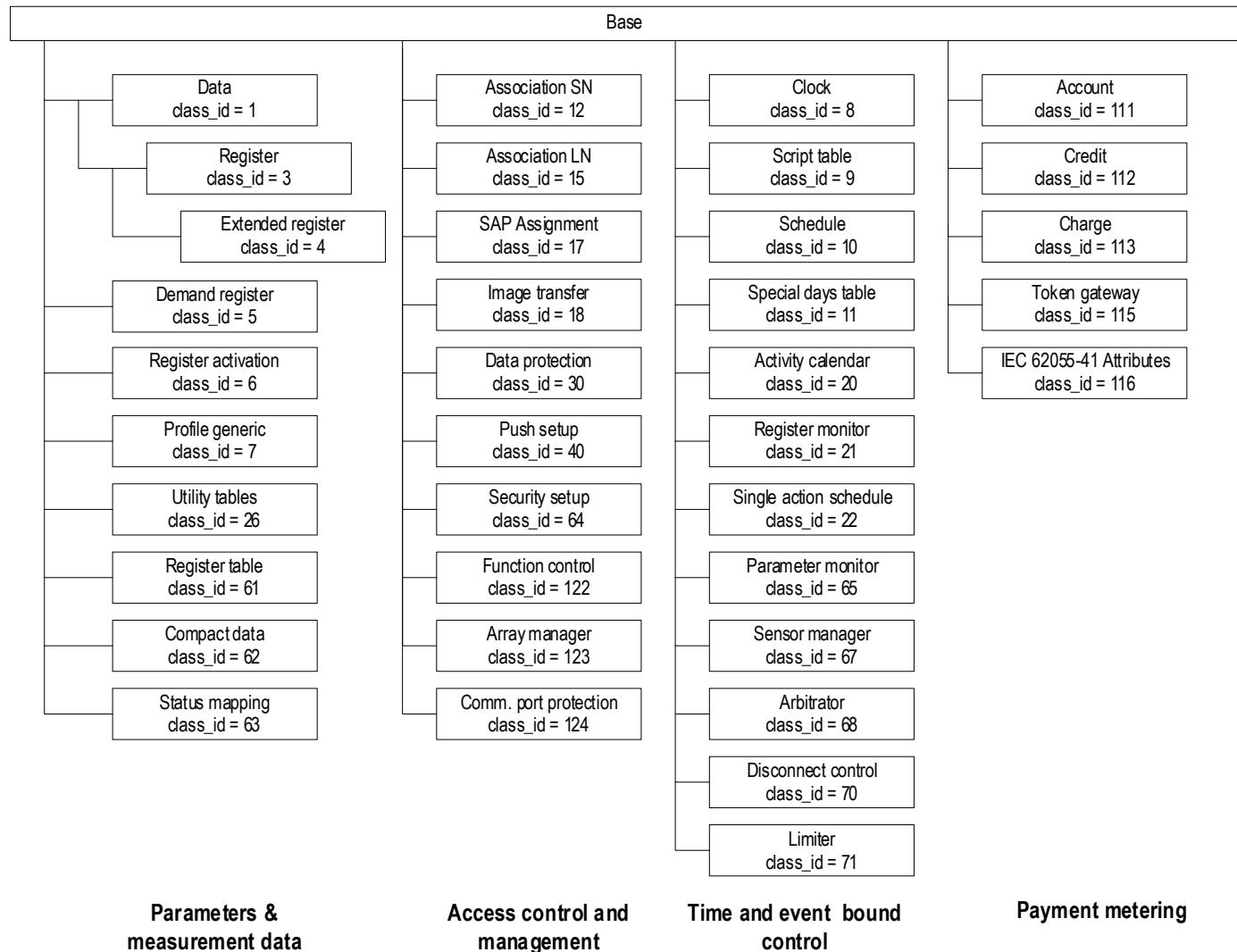
NOTE 1 The IC “base” itself is not specified explicitly. It contains only one attribute *logical\_name*.

NOTE 2 In the description of the “Demand register”, “Clock” and “Profile generic” ICs, the 2<sup>nd</sup> attributes are labelled differently from that of the 2<sup>nd</sup> 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.

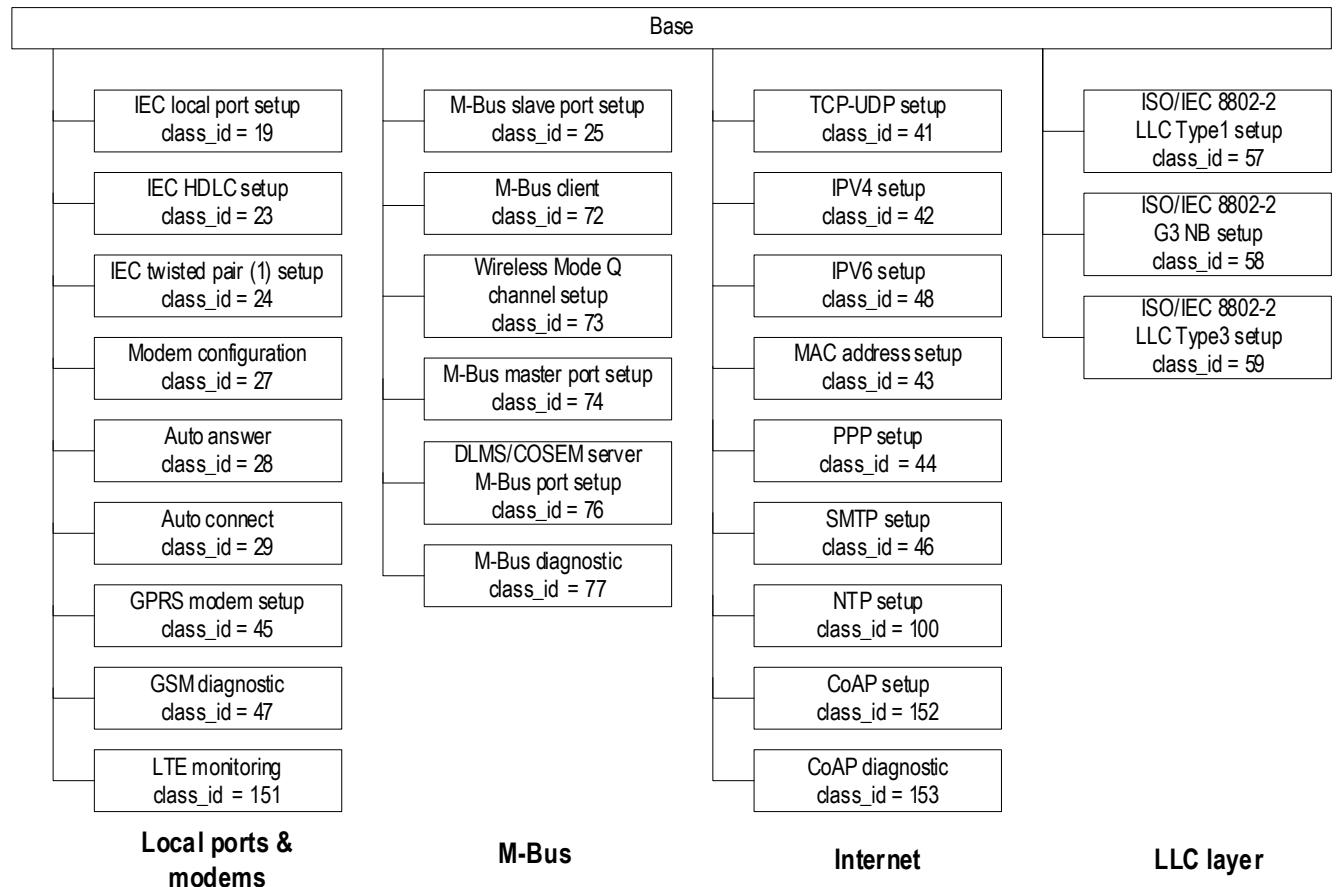
|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 55/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes



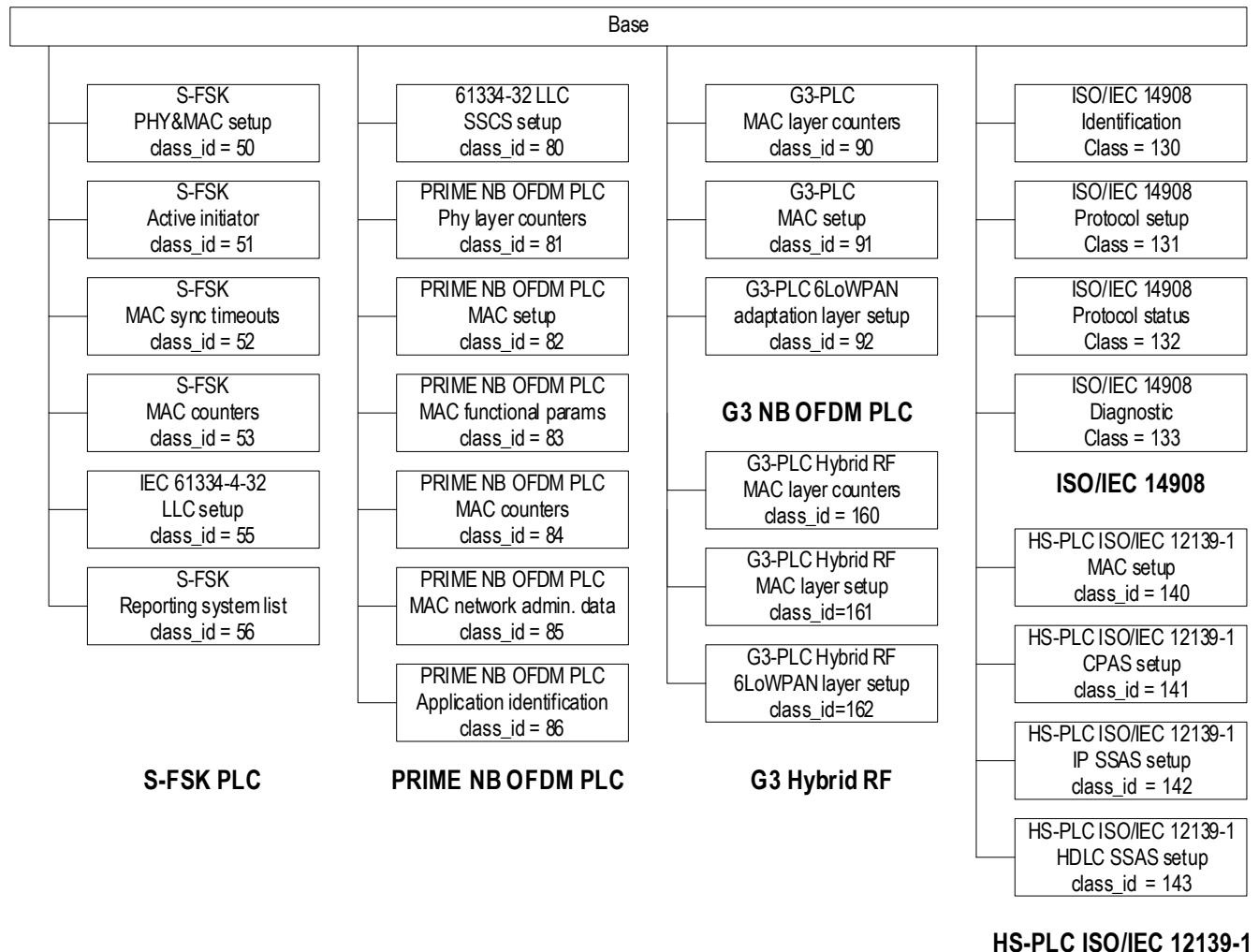
**Figure 6 – Overview of the interface classes – Part 1**

## COSEM Interface Classes



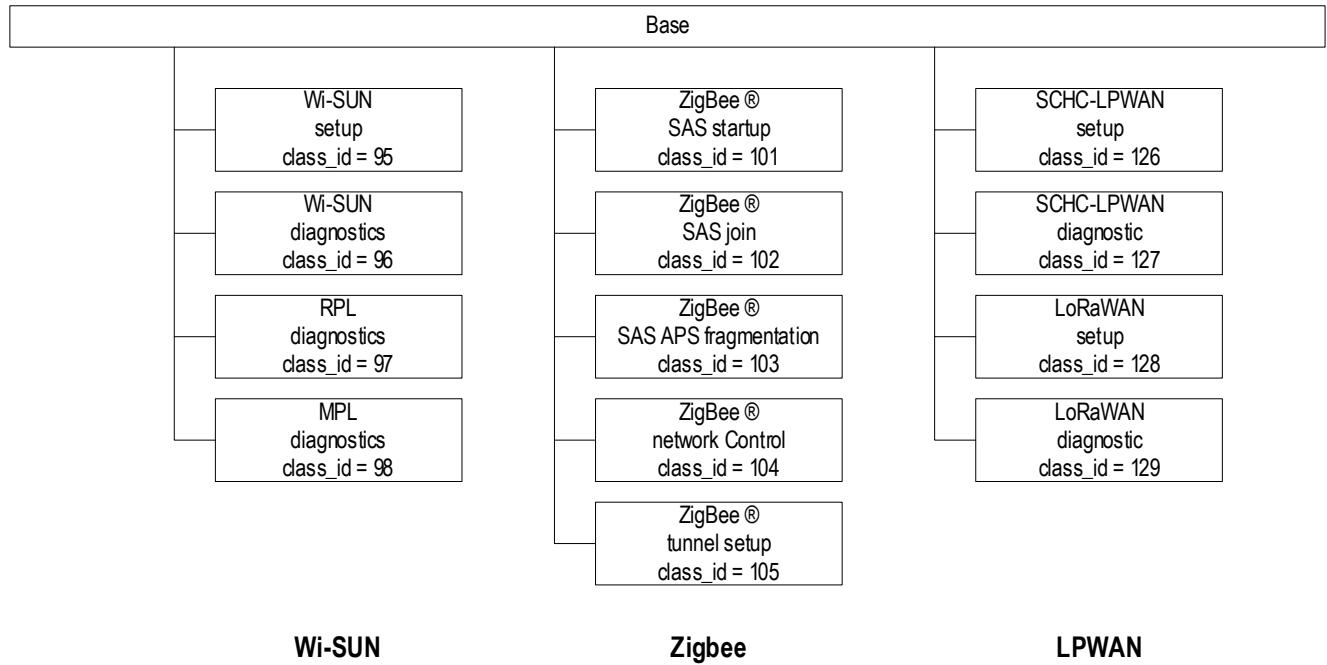
**Figure 7 – Overview of the interface classes – Part 2**

## COSEM Interface Classes



**Figure 8 – Overview of the interface classes – Part 3**

## COSEM Interface Classes



**Figure 9 – Overview of the interface classes - Part 4**

Table 4 lists the interface classes by class\_id.

**Table 4 – List of interface classes by class\_id**

| class_id | Interface class name | version(s)            | Clause                                    |
|----------|----------------------|-----------------------|---|
| 1        | Data                 | 0                     | 4.3.1                                     |
| 2        | Reserved             |                       |   |
| 3        | Register             | 0                     | 4.3.2                                     |
| 4        | Extended register    | 0                     | 4.3.3                                     |
| 5        | Demand register      | 0                     | 4.3.4                                     |
| 6        | Register activation  | 0                     | 4.3.5                                     |
| 7        | Profile generic      | 1<br>0                | 4.3.6<br>5.3.1                            |
| 8        | Clock                | 0                     | 4.5.1                                     |
| 9        | Script table         | 0                     | 4.5.2                                     |
| 10       | Schedule             | 0                     | 4.5.3                                     |
| 11       | Special days table   | 0                     | 4.5.4                                     |
| 12       | Association SN       | 4<br>3<br>2<br>1<br>0 | 4.4.3<br>5.4.4<br>5.4.3<br>5.4.2<br>5.4.1 |
| 13       | Reserved             |                       |   |
| 14       | Reserved             |                       |   |

## COSEM Interface Classes

| class_id                               | Interface class name                            | version(s)       | Clause                               |
|--|---|------------------|--------------------------------------|
| 15                                     | Association LN                                  | 3<br>2<br>1<br>0 | 4.4.4<br>5.4.7<br>5.4.6<br>5.4.5     |
| 16                                     | Reserved  |                  |                                      |
| 17                                     | SAP Assignment                                  | 0                | 4.4.5                                |
| 18                                     | Image transfer                                  | 0                | 4.4.6                                |
| 19                                     | IEC local port setup                            | 1<br>0           | 4.7.1<br>5.7.1                       |
| 20                                     | Activity calendar                               | 0                | 4.5.5                                |
| 21                                     | Register monitor                                | 0                | 4.5.6                                |
| 22                                     | Single action schedule                          | 0                | 4.5.7                                |
| 23                                     | IEC HDLC setup                                  | 1<br>0           | 4.7.2<br>5.7.2                       |
| 24                                     | IEC twisted pair (1) setup                      | 1<br>0           | 4.7.3<br>5.7.3                       |
| 25                                     | M-BUS slave port setup                          | 0                | 4.8.2                                |
| 26                                     | Utility tables                                  | 0                | 4.3.7                                |
| 27                                     | Modem configuration<br>PSTN modem configuration | 1<br>0           | 4.7.4<br>5.7.4                       |
| 28                                     | Auto answer                                     | 2<br>1<br>0      | 4.7.5<br>—<br>5.7.5                  |
| 29                                     | Auto connect<br>PSTN Auto dial                  | 2<br>1<br>0      | 4.7.6<br>5.7.7<br>5.7.6              |
| 30                                     | Data protection                                 | 0                | 4.4.9                                |
| 31 ... 39                              | Reserved  |                  |                                      |
| Interface classes for IP channel setup |   |                  |                                      |
| 40                                     | Push setup                                      | 3<br>2<br>1<br>0 | 4.4.8.2<br>5.4.11<br>5.4.10<br>5.4.9 |
| 41                                     | TCP-UDP setup                                   | 0                | 4.9.1                                |
| 42                                     | IPv4 setup                                      | 0                | 4.9.2                                |
| 43                                     | MAC address setup (Ethernet setup)              | 0                | 4.9.4<br>4.12.1<br>0                 |
| 44                                     | PPP setup                                       | 0                | 4.9.5                                |
| 45                                     | GPRS modem setup                                | 0                | 4.7.7                                |
| 46                                     | SMTP setup                                      | 0                | 4.9.6                                |

## COSEM Interface Classes

| class_id  | Interface class name                      | version(s) | Clause |
|---|---|------------|--------|
| 47  | GSM diagnostic                            | 2          | 4.7.8  |
|   |   | 1          | 5.7.9  |
|   |   | 0          | 5.7.8  |
| 48  | IPv6 setup                                | 0          | 4.9.3  |
| 49  | Reserved (proposed for FTP setup in 2012) |            |        |
| Interface classes for S_FSK PLC channel setup                 |   |            |        |
| 50  | S-FSK Phy&MAC setup                       | 1          | 4.10.3 |
|   |   | 0          | 5.10.1 |
| 51  | S-FSK Active initiator                    | 0          | 4.10.4 |
| 52  | S-FSK MAC synchronization timeouts        | 0          | 4.10.5 |
| 53  | S-FSK MAC counters                        | 0          | 4.10.6 |
| 55  | IEC 61334-4-32 LLC setup                  | 1          | 4.10.7 |
|   | S-FSK IEC 61334-4-32 LLC setup            | 0          | 5.10.2 |
| 56  | S-FSK Reporting system list               | 0          | 4.10.8 |
| Interface classes for setting up the ISO/IEC 8802-2 LLC layer |   |            |        |
| 57  | ISO/IEC 8802-2 LLC Type 1 setup           | 0          | 4.11.2 |
| 58  | ISO/IEC 8802-2 LLC Type 2 setup           | 0          | 4.11.3 |
| 59  | ISO/IEC 8802-2 LLC Type 3 setup           | 0          | 4.11.4 |
| 60  | Reserved                                  |            |        |
| 61  | Register table                            | 0          | 4.3.8  |
| 62  | Compact data                              | 1          | 4.3.10 |
|   |   | 0          | 5.3.2  |
| 63  | Status mapping                            | 0          | 4.3.9  |
| 64  | Security setup                            | 1          | 4.4.7  |
|   |   | 0          | 0      |
| 65  | Parameter monitor                         | 1          | 4.5.10 |
|   |   | 0          | 5.5.1  |
| 66  | Reserved                                  |            |        |
| 67  | Sensor manager                            | 0          | 4.5.11 |
| 68  | Arbitrator                                | 0          | 4.5.12 |
| 69  | Reserved                                  |            |        |
| 70  | Disconnect control                        | 0          | 4.5.8  |
| 71  | Limiter                                   | 0          | 4.5.9  |
| Interface classes for M-Bus channel setup                     |   |            |        |
| 72  | M-Bus client                              | 2          | 4.8.3  |
|   |   | 1          | 5.8.2  |
|   |   | 0          | 5.8.1  |
| 73  | Wireless Mode Q channel                   | 1          | 4.8.4  |
| 74  | M-Bus master port setup                   | 0          | 4.8.5  |
| 75  | Reserved                                  |            |        |
| 76  | DLMS server M-Bus port setup              | 0          | 4.8.6  |
| 77  | M-Bus diagnostic                          | 0          | 4.8.7  |

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 61/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

| class_id  | Interface class name                              | version(s) | Clause      |
|---|---|------------|-------------|
| 78  | Reserved  |            |             |
| 79  | Reserved  |            |             |
| Interface classes for OFDM Type 1 channel setup |   |            |             |
| 80  | 61334-4-32 LLC SSCS setup                         | 0          | 4.12.3      |
| 81  | PRIME NB OFDM PLC Physical layer counters         | 0          | 4.12.5      |
| 82  | PRIME NB OFDM PLC MAC setup                       | 0          | 4.12.6      |
| 83  | PRIME NB OFDM PLC MAC functional parameters       | 0          | 4.12.7      |
| 84  | PRIME NB OFDM PLC MAC counters                    | 0          | 4.12.8      |
| 85  | PRIME NB OFDM PLC MAC network administration data | 0          | 4.12.9      |
| 86  | PRIME NB OFDM PLC Application identification      | 0          | 4.12.1<br>1 |
| 87 ... 89                                       | Reserved  |            |             |
| Interface classes for OFDM Type 2 channel setup |   |            |             |
| 90  | G3-PLC MAC layer counters                         | 1          | 4.13.3      |
|   | G3 NB OFDM PLC MAC layer counters                 | 0          | 5.13.5      |
| 91  | G3-PLC MAC setup                                  | 3          | 4.13.4      |
|   | G3-PLC MAC setup                                  | 2          | 5.13.7      |
|   | G3-PLC MAC setup                                  | 1          | 5.13.6      |
|   | G3 NB OFDM PLC MAC setup                          | 0          | 5.13.5      |
| 92  | G3-PLC 6LoWPAN adaptation layer setup             | 3          | 4.13.5      |
|   | G3-PLC 6LoWPAN adaptation layer setup             | 2          | 5.13.7      |
|   | G3-PLC 6LoWPAN adaptation layer setup             | 1          | 5.13.9      |
|   | G3 NB OFDM PLC 6LoWPAN adaptation layer setup     | 0          | 5.13.8      |
| 93  | Reserved  |            |             |
| 94  | Reserved  |            |             |
| Interface classes for WiSUN setup               |   |            |             |
| 95  | Wi-SUN setup                                      | 0          | 4.18.1      |
| 96  | Wi-SUN diagnostic                                 | 0          | 4.18.2      |
| 97  | RPL diagnostic                                    | 0          | 4.18.3      |
| 98  | MPL diagnostic                                    | 0          | 4.18.4      |
| 100   | NTP Setup   | 0          | 4.9.7       |
| Interface classes for ZigBee® setup             |   |            |             |
| 101   | ZigBee® SAS startup                               | 0          | 4.15.2      |
| 102   | ZigBee® SAS join                                  | 0          | 4.15.3      |
| 103   | ZigBee® SAS APS fragmentation                     | 0          | 4.15.4      |
| 104   | ZigBee® network control                           | 0          | 4.15.5      |
| 105   | ZigBee® tunnel setup                              | 0          | 4.15.6      |
| 106 ...<br>110                                  | Reserved  |            |             |
| Interface classes for payment metering          |   |            |             |
| 111   | Account   | 0          | 4.6.2       |
| 112   | Credit  | 0          | 4.6.3       |
| 113   | Charge  | 0          | 4.6.4       |

## COSEM Interface Classes

| class_id   | Interface class name                                  | version(s) | Clause          |
|--|---|------------|-----------------|
| 115  | Token gateway   | 0          | 4.6.5           |
| 116  | IEC 62055-41 Attributes                               | 0          | 4.6.6           |
| 117 ...<br>121   | Reserved  |            |                 |
| 122  | Function control                                      | 0          | 4.4.10          |
| 123  | Array manager   | 0          | 4.4.11          |
| 124  | Communication port protection                         | 0          | 4.4.12          |
| 125  | Reserved  |            |                 |
| Interface classes for LP-WAN setup                                   |   |            |                 |
| 126  | SCHC-LPWAN setup                                      | 0          | 4.16.2.<br>1    |
| 127  | SCHC-LPWAN diagnostic                                 | 0          | 4.16.2.<br>2    |
| 128  | LoRaWAN setup   | 0          | 4.17.2.<br>2    |
| 129  | LoRaWAN diagnostic                                    | 0          | 4.17.2.<br>3    |
| Interface classes for ISO/IEC 14908 PLC setup                        |   |            |                 |
| 130  | ISO/IEC14908 Identification                           | 0          | 4.19.2          |
| 131  | ISO/IEC 14908 Protocol setup                          | 0          | 4.19.3          |
| 132  | ISO/IEC 14908 protocol status                         | 1          | 4.19.4          |
| 133  | ISO/IEC 14908 diagnostic                              | 1          | 4.19.5          |
| 134 ...<br>139   | Reserved  |            |                 |
| Interface classes for High Speed PLC setup                           |   |            |                 |
| 140  | HS-PLC ISO/IEC 12139-1 MAC setup                      | 0          | 4.14.2          |
| 141  | HS-PLC ISO/IEC 12139-1 CPAS setup                     | 0          | 4.14.3          |
| 142  | HS-PLC ISO/IEC 12139-1 IP SSAS setup                  | 0          | 4.14.4          |
| 143  | HS-PLC ISO/IEC 12139-1 HDLC SSAS setup                | 0          | 4.14.5          |
| 144 ...<br>150   | Reserved  |            |                 |
| Interface classes for IP channel setup (second decade)               |   |            |                 |
| 151  | LTE monitoring  | 1<br>0     | 4.7.9<br>5.7.10 |
| 152  | CoAP Set up   | 0          | 4.9.8           |
| 153  | CoAP Diagnostic                                       | 0          | 4.9.9           |
| 154 ... 15<br>9  | Reserved  |            |                 |
| Interface classes for G3-PLC Hybrid                                  |   |            |                 |
| 160  | G3-PLC Hybrid PLC & RF MAC layer counters             | 0          | 4.13.6          |
| 161  | G3-PLC Hybrid PLC & RF MAC setup                      | 0          | 4.13.7          |
| 162  | G3-PLC Hybrid PLC & RF 6LoWPAN adaptation layer setup | 0          | 4.13.8          |
| 163 ...<br>255   | Reserved  |            |                 |
| NOTE All unused class_ids are reserved for allocation by the DLMS UA |   |            |                 |

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 63/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

### 4.3 Interface classes for parameters and measurement data

#### 4.3.1 Data (class\_id = 1, version = 0)

##### 4.3.1.1 Overview

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<br>(static) | octet-string |                           |      |      | x          |
| 2. value                    | CHOICE       |                           |      |      | x + 0x08   |
| Specific methods            | m/o          |                           |      |      |            |

##### 4.3.1.2 Attribute description

###### 4.3.1.2.1 logical\_name

Identifies the “Data” object instance. See Clause 6 and DLMS UA 1000-1 Ed 15 Part 1:2021.

###### 4.3.1.2.2 value

Contains the data.

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

```

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],

```

## COSEM Interface Classes

```

time          [27],
delta-integer [28],
delta-long    [29],
delta-double-long [30],
delta-unsigned [31],
delta-long-unsigned [32],
delta-double-long-unsigned [33],  

-- complex data types  

array          [1],  

structure      [2],  

compact-array   [19]  

}

```

### **4.3.2 Register (class\_id = 3, version = 0)**

#### **4.3.2.1 Overview**

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              |                           |      |      |            |
|                          |                |                           |      |      |            |

#### **4.3.2.2 Attribute description**

##### **4.3.2.2.1 logical\_name**

Identifies the “Register” object instance. See Clause 6 and DLMS UA 1000-1 Ed 15 Part 1:2021.

##### **4.3.2.2.2 value**

Contains the current process or status value.

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

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

```

CHOICE
{
    -- simple data types

    null-data          [0],
    bit-string         [4],
    double-long        [5],
    double-long-unsigned [6],
    octet-string       [9],
    visible-string     [10],
    utf8-string        [12],
    integer            [15],
    long               [16],
    unsigned           [17],
    long-unsigned      [18],
    long64             [20],
    long64-unsigned    [21],
    enum               [22],
    float32            [23],
    float64            [24],
}

```

#### 4.3.2.2.3 **scaler\_unit**

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

```

scal_unit_type ::= structure
{
    scaler,
    unit
}

scaler:    integer

```

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

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

See Table 6 for examples for **scaler\_unit**.

```
unit:    enum
```

Enumeration defining the physical unit; for details see Table 5.

#### 4.3.2.3 **Method description**

##### 4.3.2.3.1 **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)
```

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 67/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

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

| <b>unit ::= enum</b> | <b>Unit</b>                 | <b>Quantity</b>   | <b>Unit name</b>     | <b>SI definition (comment)</b>                        |
|----------------------|-----------------------------|---|----------------------|---|
| <b>SI units</b>      |                             |   |                      |   |
| (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  | minute               | $60\text{ s}$   |
| (7)                  | s                           | time ( <i>t</i> )   | second               | s   |
| (8)                  | $^\circ$                    | (phase) angle   | degree               | $\text{rad} * 180/\pi$                                |
| (9)                  | $^\circ\text{C}$            | 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     | $\text{ms}^{-1}$                                      |
| (13)                 | $\text{m}^3$                | volume ( <i>V</i> )<br><i>r<sub>V</sub></i> , meter constant or pulse value (volume)    | cubic metre          | $\text{m}^3$  |
| (14)                 | $\text{m}^3$                | corrected volume <sup>a</sup>   | cubic metre          | $\text{m}^3$  |
| (15)                 | $\text{m}^3/\text{h}$       | volume flux   | cubic metre per hour | $\text{m}^3\text{s}^{-1}/(60*60)$                     |
| (16)                 | $\text{m}^3/\text{h}$       | corrected volume flux <sup>a</sup>  | cubic metre per hour | $\text{m}^3\text{s}^{-1}/(60*60)$                     |
| (17)                 | $\text{m}^3/\text{d}$       | volume flux   | cubic metre per day  | $\text{m}^3\text{s}^{-1}/(24*60*60)$                  |
| (18)                 | $\text{m}^3/\text{d}$       | corrected volume flux <sup>a</sup>  | cubic metre per day  | $\text{m}^3\text{s}^{-1}/(60*60)$                     |
| (19)                 | l                           | volume  | litre                | $10^{-3}\text{ m}^3$                                  |
| (20)                 | kg                          | mass ( <i>m</i> )   | kilogram             |   |
| (21)                 | N                           | force ( <i>F</i> )  | newton               | $\text{N} = \text{kg}\cdot\text{m}\cdot\text{s}^{-2}$ |
| (22)                 | Nm                          | energy  | newton meter         | $\text{J} = \text{Nm} = \text{Ws}$                    |
| (23)                 | Pa                          | pressure ( <i>p</i> )   | pascal               | $\text{N/m}^2$  |
| (24)                 | bar                         | pressure ( <i>p</i> )   | bar                  | $10^5\text{ Nm}^{-2}$                                 |
| (25)                 | J                           | energy  | joule                | $\text{J} = \text{Nm} = \text{Ws}$                    |
| (26)                 | J/h                         | thermal power, rate of change   | joule per hour       | $\text{Js}^{-1}/(60*60)$                              |
| (27)                 | W                           | active power ( <i>P</i> )   | watt                 | $\text{W} = \text{Js}^{-1}$                           |
| (28)                 | VA                          | apparent power ( <i>S</i> )   | volt-ampere          |   |
| (29)                 | var                         | reactive power ( <i>Q</i> )   | var                  |   |
| (30)                 | Wh                          | active energy<br><i>r<sub>w</sub></i> , active energy meter constant or pulse value     | watt-hour            | $\text{Ws}*(60*60)$                                   |
| (31)                 | VAh                         | apparent energy<br><i>r<sub>s</sub></i> , apparent energy meter constant or pulse value | volt-ampere-hour     | $\text{VAs}*(60*60)$                                  |
| (32)                 | varh                        | reactive energy<br><i>r<sub>B</sub></i> , reactive energy meter constant or pulse value | var-hour             | $\text{var s}^*(60*60\text{s})$                       |
| (33)                 | A                           | current ( <i>I</i> )  | ampere               | A   |
| (34)                 | C                           | electrical charge ( <i>Q</i> )  | coulomb              | $\text{C} = \text{As}$                                |
| (35)                 | V                           | voltage ( <i>U</i> )  | volt                 | V   |
| (36)                 | V/m                         | electric field strength ( <i>E</i> )  | volt per metre       | $\text{Vm}^{-1}$                                      |
| (37)                 | F                           | capacitance ( <i>C</i> )  | farad                | $\text{CV}^{-1} = \text{AsV}^{-1}$                    |
| (38)                 | $\Omega$                    | resistance ( <i>R</i> )   | ohm                  | $\Omega = \text{VA}^{-1}$                             |
| (39)                 | $\Omega\text{m}^2/\text{m}$ | resistivity ( <i>ρ</i> )  |                      | $\Omega\text{m}$                                      |

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 68/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

| <b>unit ::= enum</b> | <b>Unit</b>          | <b>Quantity</b>   | <b>Unit name</b>           | <b>SI definition (comment)</b>   |
|----------------------|----------------------|---|----------------------------|--|
| (40)                 | Wb                   | magnetic flux ( $\Phi$ )  | weber                      | $\text{Wb} = \text{Vs}$  |
| (41)                 | T                    | magnetic flux density ( $B$ )   | tesla                      | $\text{Wbm}^{-2}$  |
| (42)                 | A/m                  | magnetic field strength ( $H$ )   | ampere per metre           | $\text{Am}^{-1}$   |
| (43)                 | H                    | inductance ( $L$ )  | henry                      | $\text{H} = \text{WbA}^{-1}$   |
| (44)                 | Hz                   | frequency ( $f, \omega$ )   | hertz                      | $\text{s}^{-1}$  |
| (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)                 | $V^2\text{h}$        | volt-squared hour, $r_{U2h}$ , volt-squared hour meter constant or pulse value      | volt-squared-hours         | $\text{V}^2\text{s}^{-1} / (60*60)$  |
| (49)                 | $A^2\text{h}$        | ampere-squared hour, $r_{I2h}$ , ampere-squared hour meter constant or pulse value  | ampere-squared-hours       | $\text{A}^2 \text{s}^{-1} / (60*60)$   |
| (50)                 | kg/s                 | mass flux   | kilogram per second        | $\text{kg s}^{-1}$   |
| (51)                 | S, mho               | conductance   | siemens                    | $\Omega^{-1}$  |
| (52)                 | K                    | temperature ( $T$ )   | kelvin                     |  |
| (53)                 | 1/(V <sup>2</sup> h) | $R_{U2h}$ , volt-squared hour meter constant or pulse value                         |                            |  |
| (54)                 | 1/(A <sup>2</sup> h) | $R_{I2h}$ , ampere-squared hour meter constant or pulse value                       |                            |  |
| (55)                 | 1/m <sup>3</sup>     | $R_V$ , meter constant or pulse value (volume)                                      |                            |  |
| (56)                 |                      | percentage  | %                          |  |
| (57)                 | Ah                   | ampere-hours  | Ampere-hour                |  |
| (58),(59)            |                      | reserved  |                            |  |
| (60)                 | Wh/m <sup>3</sup>    | energy per volume   | $3,6*10^3 \text{ J/m}^3$   |  |
| (61)                 | J/m <sup>3</sup>     | calorific value, wobbe  |                            |  |
| (62)                 | Mol %                | molar fraction of gas composition   | mole percent               | (Basic gas composition unit)   |
| (63)                 | g/m <sup>3</sup>     | 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<br>NOTE The amount of energy per unit of mass of a substance        | Joule / kilogram           | $\text{m}^2 \cdot \text{kg} \cdot \text{s}^{-2} / \text{kg}$<br>$= \text{m}^2 \cdot \text{s}^{-2}$ |
| (66)                 | g/cm <sup>2</sup>    | pressure  | gram per square centimetre | 98,066 5 Pa  |
| (67)                 | atm                  | pressure  | atmosphere                 | 101,325*10 <sup>3</sup> Pa   |
| (68),(69)            |                      | reserved  |                            |  |
| (70)                 | dBm                  | signal strength, dB milliwatt (e.g. of GSM radio systems)                           |                            |  |
| (71)                 | db $\mu$ V           | signal strength, dB microvolt   |                            |  |
| (72)                 | dB                   | logarithmic unit that expresses the ratio between two values of a physical quantity |                            |  |
| (73)..(127)          |                      | reserved  |                            |  |
|                      |                      | Non – SI Units  |                            |  |
| (128)                | in                   | length (l)  | inch                       |  |
| (129)                | ft                   | length (l)  | foot                       |  |
| (130)                | lb                   | mass (m)  | pound                      |  |

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 69/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

| <b>unit ::= enum</b> | <b>Unit</b>     | <b>Quantity</b>                                       | <b>Unit name</b>                    | <b>SI definition (comment)</b> |
|----------------------|-----------------|---|-------------------------------------|--------------------------------|
| (131)                | °F              | temperature   | degree Fahrenheit                   |                                |
| (132)                | °R              | temperature   | degree Rankine                      |                                |
| (133)                | sq. in          | area  | square inch                         |                                |
| (134)                | sq ft           | area  | square foot                         |                                |
| (135)                | ac              | area  | acre                                |                                |
| (136)                | cu in           | volume  | cubic inch                          |                                |
| (137)                | cu ft           | volume  | cubic foot                          |                                |
| (138)                | ac ft           | volume  | acre-foot                           |                                |
| (139)                | gal (imp)       | volume  | gallon (imperial)                   |                                |
| (140)                | gal (US)        | volume  | gallon (US)                         |                                |
| (141)                | lbf             | force   | pound force                         |                                |
| (142)                | psi             | pressure (p)  | Pound force per square inch         |                                |
| (143)                | lb/cu ft        | density   | pound per cubic foot                |                                |
| (144)                | lb/(ft . s)     | dynamic viscosity                                     | pound per (foot . second)           |                                |
| (145)                | sq ft/s         | kinematic viscosity                                   | square foot per second              |                                |
| (146)                | Btu             | energy  | British thermal unit                |                                |
| (147)                | thm(EC)         | energy  | Therm(EU)                           |                                |
| (148)                | thm(US)         | energy  | Therm(US)                           |                                |
| (149)                | Btu/lb          | calorific value of mass, enthalpy                     | British thermal unit per pound      |                                |
| (150)                | Btu/cu ft       | calorific value of volume, wobbe                      | British thermal unit per cubic foot |                                |
| (151)                | cu ft           | volume (V) rv, meter constant or pulse value (volume) | cubic feet                          |                                |
| (152)                | ft/s            | speed (v)   | foot per second                     |                                |
| (153)                | cu ft/s         | volume flux   | cubic foot per second               |                                |
| (154)                | cu ft/min       | volume flux   | cubic foot per min                  |                                |
| (155)                | cu ft/h         | volume flux   | cubic foot per hour                 |                                |
| (156)                | cu ft/d         | volume flux   | cubic foot per day                  |                                |
| (157)                | ac ft/s         | volume flux   | acre foot per second                |                                |
| (158)                | ac ft/min       | volume flux   | acre foot per min                   |                                |
| (159)                | ac ft/h         | volume flux   | acre foot per hour                  |                                |
| (160)                | ac ft/d         | volume flux   | acre foot per day                   |                                |
| (161)                | gal (imp)       | volume (V) rv, meter constant or pulse value (volume) | imperial gallon                     |                                |
| (162)                | gal (imp) / s   | volume flux   | imperial gallon per second          |                                |
| (163)                | gal (imp) / min | volume flux   | imperial gallon per min             |                                |
| (164)                | gal (imp) / h   | volume flux   | imperial gallon per hour            |                                |
| (165)                | gal (imp) / d   | volume flux   | imperial gallon per day             |                                |
| (166)                | gal (US)        | volume (V) rv, meter constant or pulse value (volume) | US gallon                           |                                |
| (167)                | gal (US) / s    | volume flux   | US gallon per second                |                                |
| (168)                | gal (US) / min  | volume flux   | US gallon per min                   |                                |

## COSEM Interface Classes

| <b>unit ::= enum</b> | <b>Unit</b>  | <b>Quantity</b>                       | <b>Unit name</b>                | <b>SI definition (comment)</b> |
|----------------------|--------------|---------------------------------------|---------------------------------|--------------------------------|
| (169)                | gal (US) / h | volume flux                           | US gallon per hour              |                                |
| (170)                | gal (US) / d | volume flux                           | US gallon per day               |                                |
| (171)                | Btu/s        | energy flow, heat, power, change rate | British thermal unit per second |                                |
| (172)                | Btu/min      | energy flow, heat, power, change rate | British thermal unit per minute |                                |
| (173)                | Btu/h        | energy flow, heat, power, change rate | British thermal unit per hour   |                                |
| (174)                | Btu/d        | energy flow, heat, power, change rate | British thermal unit per day    |                                |
| (175) .. (252)       |              | reserved                              |                                 |                                |
| (253)                |              | extended table of units               |                                 |                                |
| (254)                | other        | other unit                            |                                 |                                |
| (255)                | count        | no unit, unitless, count              |                                 |                                |

<sup>a</sup> Usage of these units (16 & 18) is deprecated as OBIS codes specify a corrected volume (flux). This is to avoid contradiction of units associated with selected OBIS codes.

NOTE Enumeration values are administered by the DLMS UA.

**Table 6 – Examples for scalar\_unit**

| <b>Value</b> | <b>Scaler</b> | <b>Unit</b>    | <b>Data</b>            |
|--------------|---------------|----------------|------------------------|
| 263788       | -3            | m <sup>3</sup> | 263,788 m <sup>3</sup> |
| 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)

#### 4.3.3.1 Overview

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

| <b>Extended register</b> | <b>0...n</b>     | <b>class_id = 4, version = 0</b> |             |             |                   |
|--------------------------|------------------|----------------------------------|-------------|-------------|-------------------|
| <b>Attributes</b>        | <b>Data type</b> | <b>Min.</b>                      | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 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          |
| <b>Specific methods</b>  | <b>m/o</b>       |                                  |             |             |                   |
| 1. reset (data)          | o                |                                  |             |             |                   |

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 71/668 |
|-----------------------|------------|-----------------------------|--------|

**4.3.3.2 Attribute description****4.3.3.2.1 logical\_name**

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

**4.3.3.2.2 value**

See the specification of the IC "Register" in 4.3.2.2.2

**4.3.3.2.3 scaler\_unit**

See the specification of the IC "Register" in 4.3.2.2.3.

**4.3.3.2.4 status**

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

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

```

CHOICE
{
    -- simple data types

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

```

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

**4.3.3.2.5 capture\_time**

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

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

**4.3.3.3 Method description****4.3.3.3.1 reset (data)**

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

The attribute *capture\_time* is set to the time of the reset execution.

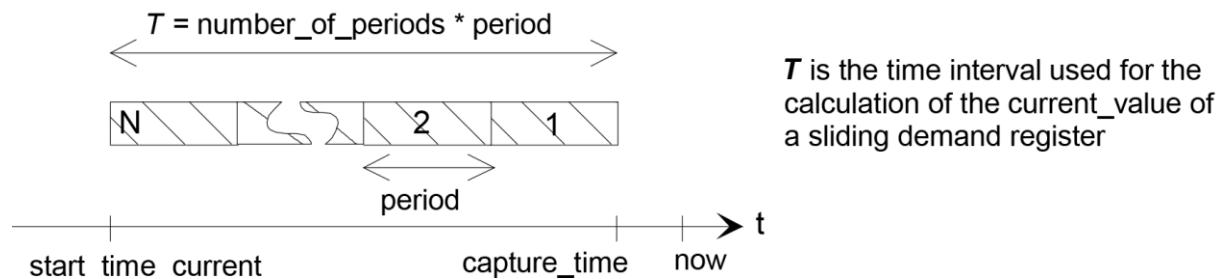
|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 72/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

```
data ::= integer (0)
```

#### 4.3.4 Demand register (class\_id = 5, version = 0)

##### 4.3.4.1 Overview

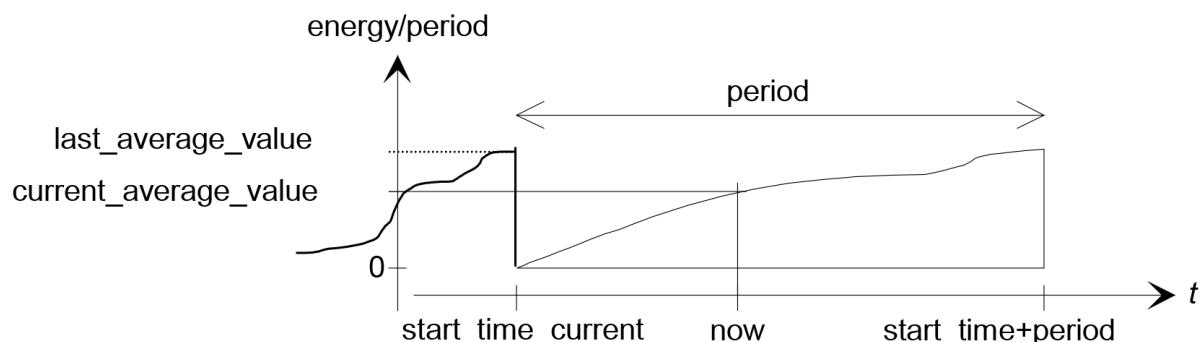
This IC allows modelling a demand value with its associated scaler, unit, status and time information. A “Demand register” object measures and computes a *current\_average\_value* periodically and it stores a *last\_average\_value*. The time interval  $T$  over which the demand is calculated is defined by specifying *number\_of\_periods* and *period*. See Figure 10.



**Figure 10 – The time attributes when measuring sliding demand**

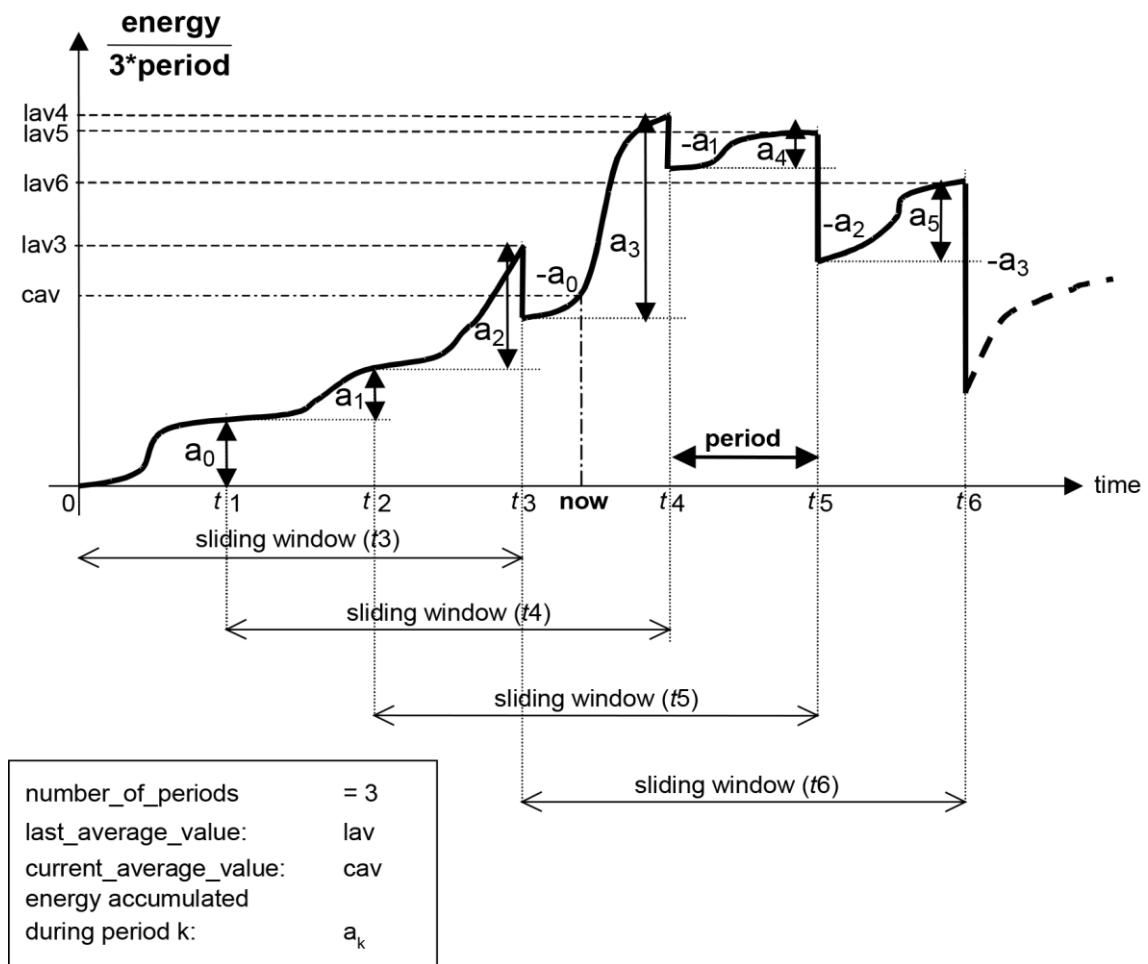
The demand register delivers two types of demand: *current\_average\_value* and *last\_average\_value* (see Figure 11 and Figure 12).

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



**Figure 11 – The attributes in the case of block demand**

## COSEM Interface Classes



**Figure 12 – 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   |

**4.3.4.2 Attribute description****4.3.4.2.1 logical\_name**

Identifies the “Demand register” object instance. See 6.3.1 and DLMS UA 1000-1 Ed 15 Part 1:2021.

**4.3.4.2.2 current\_average\_value**

Provides the current value (running demand) of the energy accumulated since *start\_time*, divided by *number\_of\_periods\*period*.

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

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

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

**4.3.4.2.3 last\_average\_value**

Provides the value of the energy accumulated (over the last *number\_of\_periods\*period*) divided by *number\_of\_periods\*period*. The energy of the current (not terminated) period is not considered by the calculation.

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

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

**4.3.4.2.4 scaler\_unit**

See the specification of IC “Register” in 4.3.2.2.3.

**4.3.4.2.5 status**

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

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

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

**4.3.4.2.6 capture\_time**

Provides the date and time when the *last\_average\_value* has been calculated.

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

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 75/668 |
|-----------------------|------------|-----------------------------|--------|

**4.3.4.2.7 start\_time\_current\_**

Provides the date and time when the measurement of the *current\_average\_value* has been started.

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

**4.3.4.2.8 period**

Period is the interval between two successive updates of the *last\_average\_value*. (*number\_of\_periods*\**period*) is the denominator for the calculation of the demand).

double-long-unsigned Measuring period in seconds

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

**4.3.4.2.9 number\_of\_periods**

The number of periods used to calculate the *last\_average\_value*. *number\_of\_periods* >= 1

- *number\_of\_periods* > 1 indicates that the *last\_average\_value* represents "sliding demand",
- *number\_of\_periods* = 1 indicates that the *last\_average\_value* represents "block demand".

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

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

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

- the current period is terminated;
- the *current\_average\_value* and the *last\_average\_value* are set to their default values;
- the *capture\_time* and the *start\_time\_current* are set to the time of the execution of *reset (data)*.

data ::= integer (0)

**4.3.4.3.2 next\_period (data)**

This method is used to trigger the regular termination (and restart) of a period. Closes (terminates) the current measuring period. Updates *capture\_time* and *start\_time* and copies *current\_average\_value* to *last\_average\_value*, sets *current\_average\_value* to its default value. Starts the next measuring period.

**REMARK** The old *last\_average\_value* (and *capture\_time*) can be read during the time "period". The old *current\_average\_value* is not available any more at the interface.

data ::= integer (0)

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 76/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

### 4.3.5 Register activation (class\_id = 6, version = 0)

#### 4.3.5.1 Overview

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            |                           |      |      |            |

#### 4.3.5.2 Attribute description

##### 4.3.5.2.1 logical\_name

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

##### 4.3.5.2.2 register\_assignment

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

```

array      object_definition

object_definition ::= structure

{
    class_id:      long-unsigned,
    logical_name: octet-string
}

```

##### 4.3.5.2.3 mask\_list

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

```

array      register_act_mask

register_act_mask ::= structure

{
    mask_name:          octet-string,
    index_list:         index_array
}

mask_name has to be uniquely defined within the object
index_array ::= array  unsigned

```

#### 4.3.5.2.4 active\_mask

Defines the currently active mask. The mask is defined by its *mask\_name* (see 4.3.5.2.3).

The *active\_mask* defines the registers currently enabled; all other registers listed in the *register\_assignment* are disabled.

#### 4.3.5.3 Method description

##### 4.3.5.3.1 add\_register (data)

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

```

data ::= structure
{
    class_id:        long-unsigned,
    logical_name:    octet-string
}

```

##### 4.3.5.3.2 add\_mask (data)

Adds another mask to the attribute *mask\_list*. If there exists already a mask with the same name, the existing mask will be overwritten by the new mask.

data ::= register\_act\_mask (see 4.3.5.2.3)

##### 4.3.5.3.3 delete\_mask (data)

Deletes a mask from the attribute *mask\_list*. The mask is defined by its *mask\_name*.

data ::= octet-string (*mask\_name*)

#### 4.3.6 Profile generic (class\_id = 7, version = 1)

##### 4.3.6.1 Overview

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.

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 78/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

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:

- a) the number of entries filled (*entries\_in\_use*). This will be zero after clearing the profile;
- b) 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;
- c) 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.

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

#### 4.3.6.2 Attribute description

##### 4.3.6.2.1 logical\_name

Identifies the “Profile generic” object instance. For examples, see 6.2.17, 6.2.19, 6.2.21, 6.2.22, 6.2.43, 6.2.46, 6.2.49, 6.2.50, 6.2.61, 6.2.62, 6.2.63, 6.2.64, 6.2.65, 6.2.66, 6.3.2, 6.3.6, etc.

##### 4.3.6.2.2 buffer

Contains a sequence of entries. Each entry contains values of the captured objects.  
 compact-array or array entry

```

entry ::= structure

{
  CHOICE
  {
    -- simple data types
    null-data          [0],
    boolean            [3],
    bit-string         [4],
    double-long        [5],
    double-long-unsigned [6],
    octet-string       [9],
    visible-string     [10],
    utf8-string        [12],
    bcd                [13],
    integer             [15],
    long                [16],
    unsigned            [17],
    long-unsigned       [18],
    long64              [20],
    long64-unsigned     [21],
    enum                [22],
    float32            [23],
    float64            [24],
    date-time           [25],
    date                [26],
    time                [27],
    delta-integer       [28],
    delta-long          [29],
    delta-double-long   [30],
    delta-unsigned      [31],
    delta-long-unsigned [32],
    delta-double-long-unsigned [33],
    -- complex data types
    array               [1],
    structure           [2],
    compact-array       [19]
  }
}
  
```

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

Default: The *buffer* is empty after reset.

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

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

*selective access* (see 4.1.4.17) to the attribute buffer may be available (optional). The selective access parameters are as defined in 4.3.6.2.9.

#### 4.3.6.2.3 capture\_objects

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

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

```
array      capture_object_definition

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

Where:

- attribute\_index is a pointer to the attribute within the object identified by its class\_id and *logical\_name*. Attribute\_index 1 refers to the 1st attribute (i.e. the *logical\_name*), attribute\_index 2 to the 2nd, etc.); attribute\_index 0 refers to all public attributes;
- data\_index is a pointer selecting a specific element of the attribute. The first element in the attribute structure is identified by data\_index 1. If the attribute is not a structure, then the data\_index has no meaning. If the capture object is the buffer of a profile, then the data\_index identifies the captured object of the buffer (i.e. the column) of the inner profile;

data\_index 0: references the whole attribute.

#### 4.3.6.2.4 capture\_period

>= 1: Automatic capturing assumed. Specifies the capturing period in seconds.

0: No automatic capturing; capturing is triggered externally or capture events occur asynchronously.

#### 4.3.6.2.5 sort\_method

If the profile is unsorted, it works as a “first in first out” buffer (it is hence sorted by capturing, and not necessarily by the time maintained in the “Clock” object). If the *buffer* is full, the next call to *capture* () will push out the first (oldest) entry of the *buffer* to make space for the new entry.

If the profile is sorted, a call to *capture* () will store the new entry at the appropriate position in the buffer, moving all following entries and probably losing the least interesting entry. If the new entry would enter the *buffer* after the last entry and if the *buffer* is already full, the new entry will not be retained at all.

enum: (1) fifo (first in first out),

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 81/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

- (2) lifo (last in first out),
- (3) largest,
- (4) smallest,
- (5) nearest\_to\_zero,
- (6) farest\_from\_zero

**Def.** fifo

### 4.3.6.2.6 sort\_object

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

capture\_object\_definition See 4.3.6.2.3

**Def.** no object to sort by (only possible with sort\_method fifo or lifo)

**NOTE** If the sort\_method is FIFO or LIFO, then all elements of the capture\_object\_definition specifying the sort object can be zero.

### 4.3.6.2.7 entries\_in\_use

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

double-long-unsigned 0...profile\_entries

**Def.** 0

### 4.3.6.2.8 profile\_entries

Specifies how many entries shall be retained in the buffer.

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

**Def.** 1

### 4.3.6.2.9 Parameters for selective access to the buffer attribute

Table 7 defines the parameters required for selective access to the buffer.

**Table 7 – Parameters for selective access to the buffer attribute**

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

range\_descriptor ::= structure  
{  
restricting\_object: capture\_object\_definition /\* (Defines the capture\_object restricting

## COSEM Interface Classes

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

### CHOICE

{as for 'from\_value' 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`

{

|                          |                      |   |
|--------------------------|----------------------|---|
| <code>from_entry:</code> | double-long-unsigned | -- first entry to retrieve,   |
| <code>to_entry:</code>   | double-long-unsigned | -- last entry to retrieve<br>/*(if <code>to_entry == 0</code> : highest possible entry)*/ |

|                                   |               |  |
|-----------------------------------|---------------|--|
| <code>from_selected_value:</code> | long-unsigned | -- index of first value to retrieve,   |
| <code>to_selected_value:</code>   | long-unsigned | -- index of last value to retrieve<br>/*(if <code>to_selected_value == 0</code> : highest possible selected_value)*/ |

}

NOTE 1 `from_entry` and `to_entry` identify the lines, `from_selected_value` to `selected_value` identify the columns of the buffer to be retrieved.

NOTE 2 Numbering of entries and selected values starts from 1.

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 83/668 |
|-----------------------|------------|-----------------------------|--------|

#### 4.3.6.3 Method description

##### 4.3.6.3.1 reset (data)

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

```
data ::= integer (0)
```

##### 4.3.6.3.2 capture(data)

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

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

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

```
data ::= integer (0)
```

#### 4.3.6.4 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.

#### 4.3.6.5 Restrictions

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

#### 4.3.6.6 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 6.2.19. Setting *capture\_period* to 1 ensures that the values are updated every second.

#### 4.3.7 Utility tables (class\_id = 26, version = 0)

##### 4.3.7.1 Overview

This IC allows encapsulating ANSI C12.19:2012 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 |
| <b>Specific methods</b> | <i>m/o</i>   |  |  |  |          |

**4.3.7.2 Attribute description****4.3.7.3 logical\_name**

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

**4.3.7.4 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.

**4.3.7.5 length**

Number of octets in table buffer.

**4.3.7.6 buffer**

Contents of the table.

Selective access (see 4.1.4.17) to the attribute *buffer* may be available (optional). The selective access parameters are defined in Table 8.

**Table 8 – Parameters for selective access to the buffer attribute**

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

```

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)

#### 4.3.8.1 Overview

This IC allows to group homogenous entries, identical attributes of multiple objects, which are all instances of the same IC, and in their *logical\_name* (OBIS code) the value in value groups A to D and F is identical. The possible values in value group E are defined in DLMS UA 1000-1 Ed 15 Part 1:2021 in a tabular form: the table header defines the common part of the OBIS code and each table cell defines one possible value of value group E. A “Register table” object may capture attributes of some or all of those objects.

NOTE 1 Some examples are the “Extended phase angle measurement” table, see Table 18 or the “UNIPEDE voltage dip quantities” table, see Table 20.

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                      |                            |      |      |            |

#### 4.3.8.2 Attribute description

##### 4.3.8.2.1 logical\_name

Identifies the “Register table” object instance.

When the format of the *logical\_name* is A.B.C.D.255.F; the values A to D and F define the common part of the logical name of the objects, the attributes of which are captured. Only one attribute of the objects concerned can be captured (for example the *value* attribute).

When the format of the *logical\_name* is A.B.98.10.x.255, several instances of the “Register table” IC can be used to capture different attributes of the objects concerned. The value group E numbers the instances. See DLMS UA 1000-1 Ed 15 Part 1:2021, 6.4.

##### 4.3.8.2.2 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*).

#### 4.3.8.2.3 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 DLMS UA 1000-1 Ed 15 Part 1:2021;
- 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 DLMS UA 1000-1 Ed 15 Part 1:2021 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 DLMS UA 1000-1 Ed 15 Part 1:2021;
- the attribute index shall be 2 (value).

#### 4.3.8.2.4 scaler\_unit

See the description of IC “Register”, 4.3.2.2.3.

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 87/668 |
|-----------------------|------------|-----------------------------|--------|

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.

#### 4.3.8.3 Method description

##### 4.3.8.3.1 reset (data)

Clears the *table\_cell\_values*. It has no effect on the attributes captured.

```
data ::= integer (0)
```

##### 4.3.8.3.2 capture (data)

Copies the values of the attributes into the *table\_cell\_values*. If the *attribute\_index* = 0, all attributes are captured.

#### 4.3.8.4 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)

##### 4.3.9.1 Overview

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          |                            |      |      |            |

##### 4.3.9.2 Attribute description

###### 4.3.9.2.1 logical\_name

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

###### 4.3.9.2.2 status\_word

Contains the current value of the status word.

```
CHOICE
{
    bit-string [4],
```

```

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

```

The size of the *status\_word* is n\*8 bits, the maximum size is 65 536 bits.

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

#### 4.3.9.2.3 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
        array long-unsigned      [18],
        [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 (class\_id: 62, version = 1)

#### 4.3.10.1 Overview

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

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.

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 89/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

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

### 4.3.10.2 Attribute description

#### 4.3.10.2.1 logical\_name

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

#### 4.3.10.2.2 compact\_buffer

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

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

#### 4.3.10.2.3 capture\_objects

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

When defining the *capture\_objects* attribute, circular references shall be avoided.

The *template\_id* attribute shall be the first element in the *capture\_objects* array.

Upon an explicit or implicit invocation of the *capture* (data) method the values of the selected attributes are captured into the *compact\_buffer*.

Two, mutually exclusive selective access mechanisms are available:

- relative selective access, i.e. entries defined relative to current date or entry are returned: this mechanism is controlled by the *data\_index* element; or
- absolute selective access, i.e. entries in an explicitly defined date range or entry range are returned: this mechanism is controlled by the *restriction\_element* and the columns are controlled by the lower nibble of the MS byte in the *data\_index*.

|        |            |                              |                       |
|--------|------------|------------------------------|-----------------------|
| 90/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|--------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

```

array capture_object_definition
capture_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 1<sup>st</sup> attribute (i.e. the *logical\_name*), attribute\_index 2 to the 2<sup>nd</sup> attribute etc.; attribute\_index 0 refers to all public attributes noting DLMS UA 1000-2 Ed.11:2021, 9.1.4.3.7;
- 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 – other than the *buffer* of a Profile generic object – 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 |

- 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 9.

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
    }
}

```

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
|                       |            | 91/668                      |

```

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:
  - MS Byte upper nibble set to 0x0;
  - MS Byte lower nibble 0x0 to 0xF in accordance with Table 9;
  - Lower byte set to 0x00
- 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

**Table 9 – Encoding of selective access parameters with data\_index**

|     |  |
|-----|--|
|     | <b>MS-Byte Upper nibble :</b> Selects the time intervals or entries  |
| 0xF | <b>Last number of complete months:</b> This is equal to a selective access to the profile <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 | <b>Last number of complete days:</b> This is equal to a selective access to the profile <i>buffer</i> resulting in all entries of the last complete number of days and the first entry at midnight of today.                 |
| 0xD | <b>Last number of complete hours:</b> This is equal to a selective access to the profile <i>buffer</i> resulting in all entries of the last complete number of hours and the first entry of the current hour.                |
| 0xC | <b>Last number of complete minutes:</b> This is equal to a selective access to the profile <i>buffer</i> resulting in all entries of the last complete number of minutes and the first entry of the current minute.          |
| 0xB | <b>Last number of seconds:</b> This is equal to a parameterised read access to the profile resulting in all entries of the last number of seconds.   |
| 0xA | <b>Last number of complete months including the current month:</b> This is the same as 0xF above but all entries up to now are retrieved.  |
| 0x9 | <b>Last number of complete days including the current day:</b> This is the same as 0xE above but all entries up to now are retrieved.  |
| 0x8 | <b>Last number of complete hours including the current hour:</b> This is the same as 0xD above but all entries up to now are retrieved.  |
| 0x7 | <b>Last number of complete minutes including the current minute:</b> This is the same as 0xC above but all entries up to now are retrieved.  |
|     | Values 0x6 to 0x2 apply only for the Push interface class (class_id = 40, Version = 3 (4.4.8.2) and Version = 2 (5.4.11)).   |
| 0x6 | <b>Complete number of months after last confirmation:</b> This is equal to a selective access to the profile <i>buffer</i> resulting in all entries of the complete number of months after the last confirmed entry.         |
| 0x5 | <b>Complete number of days after last confirmation:</b> This is equal to a selective access to the profile <i>buffer</i> resulting in all entries of the complete number of days after the last confirmed entry.             |

## COSEM Interface Classes

|             |   |
|-------------|---|
|             | <b>MS-Byte Upper nibble :</b> Selects the time intervals or entries   |
| 0x4         | <b>Complete number of hours after last confirmation:</b> This is equal to a selective access to the profile <i>buffer</i> resulting in all entries of the complete number of hours after the last confirmed entry.  |
| 0x3         | <b>Complete number of minutes after last confirmation:</b> This is equal to a selective access to the profile <i>buffer</i> resulting in all entries of the complete number of minutes after the last confirmed entry.  |
| 0x2*        | <b>Number of entries after last confirmation:</b> This is equal to a selective access to the profile <i>buffer</i> resulting in number of entries after the last confirmed entry.   |
| 0x1         | Last number of entries  |
| 0x0         | Whole attribute or a single element in an attribute with complex data type is selected (see above)  |
|             | <b>MS-Byte Lower nibble != 0:</b> Defines the number of columns selected  |
| 0x0         | All columns   |
| 0x1 ... 0xF | Number of columns, starting from column 1   |
| 0x00...0xFF | LS-Byte: <ul style="list-style-type: none"> <li>– when entries in time intervals are selected, specifies the number of time intervals, value 0x00 defines all time <b>intervals</b> ;</li> <li>– when entries are selected, specifies the number of entries, value 0x00 defines all entries.</li> </ul> |
| Example 1)  | 0xE401: The entries for the last complete day are selected. The first 4 columns are included.   |
| Example 2)  | 0xA300: The entries for the current month are selected. The first 3 columns are included.   |
| Example 3)  | 0x800C: The entries for the last complete 12 hours are selected. All columns are included.  |
| Example 4)  | 0x1080: The last 128 entries are selected. All columns are included.  |
| Example 5)  | 0x4608: The entries for the last complete 8 hours after the last confirmed entry are selected. The first 6 columns are included.  |

### 4.3.10.2.4 template\_id

Contains the identifier of the template. It shall uniquely identify the instance of the “Compact data” IC and the *template\_description*.

### 4.3.10.2.5 template\_description

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

- 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 *capture\_objects* array – encoded as a variable length integer;
- this is followed by the data type of each attribute, in the same order as in the *capture\_object* array:
  - in the case of attributes with simple data type, the data type is represented by a single octet, carrying the tag of the data type. In the case of *bit-string* [4], *octet-string* [9], *visible-string* [10], *utf8-string* [12] the length of the string is part of the data held in the *compact\_buffer*;
  - in the case of an *array* [1], the data type is represented by a single octet 0x01. This is followed by the type description of the elements in the array. The number of elements in the array is part of the data held in the *compact\_buffer*;
  - in the case of a *structure* [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.

#### 4.3.10.2.6 capture\_method

Defines the way the *compact\_buffer* is updated.

enum

- (0) Capture upon invoking the *capture* (data) method. This may occur remotely or locally (explicit capturing),
- (1) Capture upon reading the *compact\_buffer* attribute (implicit capturing).

#### 4.3.10.3 Method description

##### 4.3.10.3.1 reset (data)

Clears the *compact\_buffer*. After invoking this method the buffer holds an octet-string of 0 length until a new capture takes place.

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

data ::= integer(0)

##### 4.3.10.3.2 capture (data)

Copies the values of the attributes into the *compact\_buffer* by reading each capture object.

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

data ::= integer(0)

#### 4.3.10.4 Behaviour of the object after modification of certain attributes

Any modification of the *capture\_objects* shall reset the *compact\_buffer* and automatically update the *template\_description*.

#### 4.3.10.5 Examples for using compact data

##### 4.3.10.5.1 Example Daily billing data

Table 10 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 10 – Example daily billing data captured to *compact\_buffer***

| 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     |

## COSEM Interface Classes

|                        |    |                 |   |   |   |                      |          |
|------------------------|----|-----------------|---|---|---|----------------------|----------|
| 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 11 shows the attributes of the “Compact data” object.

**Table 11 – “Compact data“ object attributes – Daily billing data example**

|  |   |
|--|---|
| capture_objects<br>(array)                   | For the elements of the array, see columns 2, 3, 4 and 5 of Table 10.   |
| template_id<br>(unsigned)                    | 0   |
| template_description<br>(octet-string)       | -- For the data types, see column 7 of Table 10.<br>02 0A 11 06 11 11 06 06 06 06 09 12                                     |
| compact_buffer<br>(octet-string)<br>32 bytes | -- For the values see column 8 of Table 10.<br>00 51EE5305 29 18 0061FFD3 000D33D2 0012D687 0023CACE<br>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 12. Only the encoding of the result (SEQUENCE OF Get-Data-Result) is shown.

**Table 12 – Example daily billing data read using GET-WITH LIST**

| Encoding  | Explanation                   | Length   |
|---|-------------------------------|----------|
| 09  | <b>SEQUENCE of 9 elements</b> | 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        |
|   | <b>Total</b>                  | 50 bytes |
| NOTE The leading 00-s in each element are there to indicate the CHOICE “Data” in Get-Data-Result. |                               |          |

### 4.3.10.5.2 Example Diagnostic and Alarm data

Table 13 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.

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 95/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

**Table 13 – Example diagnostic and alarm data captured to *compact\_buffer***

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

**Table 14 – “Compact data” object attributes – Diagnostic and Alarm data example**

|   |  |
|---|--|
| capture_objects (array)                   | For the elements of the array, see columns 2, 3, 4 and 5 of Table 13.  |
| template_id (unsigned)                    | 1  |
| template_description (octet-string)       | -- For the data types, see column 7 of Table 13.<br>02 08 11 12 12 12 12 09 12 09                                    |
| compact_buffer (octet-string)<br>29 bytes | -- For the values, see column 8 of Table 13.<br>01 4200 4108 4308 02FB 084142434445464748 05FC<br>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 15 (only the Data is shown).

**Table 15 – Example diagnostic and alarm data read from “Profile generic” buffer**

| 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       |
|                        | <b>Total</b>             | 39 bytes |

#### 4.3.10.5.3 Example 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 16.

**Table 16 – Example logbook data entries in “Profile generic” *buffer***

| 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 17 shows the data to be captured by the “Compact data” object.

**Table 17 – Example logbook data captured to *compact\_buffer***

| 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 <sup>1</sup> | 7        | 0-0:99.98.0.255 | 2            | 0          | dyn. <sup>2</sup> | array of structure | 2     |

<sup>1</sup> See Table 16.

<sup>2</sup> The size is dynamic and depends on the number of entries captured.

Table 18 shows the attributes of the “Compact data” object.

## COSEM Interface Classes

**Table 18 – “Compact data“ object attributes – Logbook data example**

|  |   |
|--|---|
| capture_objects<br>(array)             | For the elements of the array, see columns 2, 3, 4 and 5 of Table 17. The capture object has only 2 elements: the Template_id and the buffer attribute of the Logbook.  |
| template_id<br>(unsigned)              | 2   |
| template-description<br>(octet-string) | <p>-- For the data types, see column 7 of Table 17.</p> <p>02 02 11 01 02 02 06 11</p> <p>Meaning:</p> <p>02 02 - a structure of 2 elements</p> <p>11 - first element is an unsigned</p> <p>01 02 02 - second element is an array of structure with two elements in the structure</p> <p>06 first one is a double-long-unsigned</p> <p>11 second one is an unsigned</p> |
| compact_buffer<br>(octet-string)       | <p>-- For the values, see column 8 of Table 17.</p> <p>02 -- value of the template-id</p> <p>32 -- number of the elements in the array</p> <p>and <math>50 \times 5 = 250</math> bytes (for 50 elements in the log book)</p> <p>252 bytes in total</p>  |

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 19.

**Table 19 – Example logbook data read from “Profile generic” *buffer***

| 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                |
| ....        | ...                     | ...              |
|             | <b>Total</b>            | <b>452 bytes</b> |

## 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 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;
- the “Function control” class – see 4.4.10 – allows enabling and disabling functions in the server;
- the “Array manager” class – see 4.4.11 – allows managing attributes of type array of other interface objects.
- the “Communication port protection” class – see 4.4.12 – allows protection of ports against unauthorised attempts at communication.

### 4.4.2 Client user identification

This feature enables the server to distinguish between different users from the client side and to log their activities accessing the device but it is not applicable to pre-established AAs.

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.

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

The list of users – identified by their *user\_id* and *user\_name* – is known both by the client and the server. In the server it is held by the *user\_list* attribute of the “Association” objects.

**NOTE 1** 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.

**NOTE 2** For this reason, this feature is not available with pre-established AAs.

|                       |            |                             |        |
|-----------------------|------------|-----------------------------|--------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 99/668 |
|-----------------------|------------|-----------------------------|--------|

## COSEM Interface Classes

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\_id = 12, version = 4)

#### 4.4.3.1 Overview

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 = 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                  |                            |      |      |            |

#### 4.4.3.2 Attribute description

##### 4.4.3.2.1 logical\_name

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

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 100/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

#### 4.4.3.2.2 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 4.1.4.17) to the attribute *object\_list* may be available. The access selector values and their parameters are as defined in Table 20.

#### 4.4.3.2.3 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,   |

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 101/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

```
        (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 4.1.4.17) to the attribute access\_rights\_list may be available (optional). The access selector values and their parameters are as defined in Table 20.

**Table 20 – 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.<br>For the response: data ::= objlist_type  |
| 2                     | structure<br>{<br>class_id: long-unsigned,<br>logical_name: octet-string<br>} | 2                        | Delivers the entry of the object_list for a specific class_id and logical_name.<br>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.<br>For the response: data ::= objlist_element<br>In the case of attribute 3, delivers the entry of the access_rights_list for a specific base_name.<br>For the response: data ::= access_rights_element |

#### 4.4.3.2.4 security\_setup\_reference

References the “Security setup” object by its *logical\_name*. The referenced object manages security for a given “Association SN” object instance.

#### 4.4.3.2.5 user\_list

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

```

array      user_list_entry

user_list_entry ::=   structure
{
    user_id:     unsigned,
    user_name:   CHOICE
    {
        visible-string [10],
        UTF-8-string [12]
    }
}

```

Where:

- user\_id is the identifier of the user (this value is carried by the calling-AE-invocation-id field of the AARQ);
- user\_name is the name of the user.

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

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

#### 4.4.3.2.6 current\_user

Holds the identifier of the current user.

`current_user ::= user_list_entry (see 4.4.3.2.5)`

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

#### 4.4.3.3 Method description

##### 4.4.3.3.1 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 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.11:2021, 9.5.

##### 4.4.3.3.2 change\_secret (data)

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

`data ::= octet-string new secret`

NOTE 1 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 2 In the case of HLS with GMAC, the (HLS\_) secret is held by the "Security setup" object referenced in attribute.

##### 4.4.3.3.3 reply\_to\_HLS\_authentication (data)

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

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

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 104/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

#### 4.4.3.3.4 reply\_to\_HLS\_authentication (data)

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.

#### 4.4.3.3.5 add\_user (data)

Adds a user to the user\_list.

data ::= user\_list\_entry (see 4.4.3.2.5)

#### 4.4.3.3.6 remove\_user (data)

Removes a user from the user\_list.

data ::= user\_list\_entry (see 4.4.3.2.5)

### 4.4.4 Association LN (class\_id = 15, version = 3)

#### 4.4.4.1 Overview

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 = 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   |

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 105/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

|                         |   |  |          |
|-------------------------|---|--|----------|
| 4. remove_object (data) | o |  | x + 0x78 |
| 5. add_user (data)      | o |  | x + 0x80 |
| 6. remove_user (data)   | o |  | x + 0x88 |

### 4.4.4.2 Attribute description

#### 4.4.4.2.1 logical\_name

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

#### 4.4.4.2.2 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 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:

```

{
    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
}

```

## COSEM Interface Classes

```

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 4.1.4.17) to the attribute *object\_list* may be available (optional). The selective access parameters are as defined in 4.4.4.2.2.1

### 4.4.4.2.2.1 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 shown in Table 21:

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 107/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

**Table 21 – Parameters for selective access to the object\_list attribute**

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

### 4.4.4.2.3 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.

### 4.4.4.2.4 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.11:2021, 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
{
```

## COSEM Interface Classes

```

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.11:2021, 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).

### **4.4.4.2.5 xDLMS\_context\_info**

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

```

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

```

Where:

- the conformance element contains the xDLMS conformance block supported by the server. The length of the bit-string is 24 bits;
- the max\_receive\_pdu\_size element contains the maximum length for an xDLMS APDU, expressed in bytes that the client may send negotiated during the application association process and limited by 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. It is limited by 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.11:2021, 9.5.

### **4.4.4.2.6 authentication\_mechanism\_name**

Contains the name of the authentication mechanism for the AA.

```

mechanism_name_type ::= CHOICE
{
    mechanism_name_structure [2],
}

```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 109/668 |
|-----------------------|------------|-----------------------------|---------|

```
octet-string [9]
}
```

The authentication mechanism name is specified as an OBJECT IDENTIFIER in DLMS UA 1000-2 Ed.11:2021, 9.5.

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.11:2021, 9.4.2.2.3.

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

#### **4.4.4.2.7 secret**

Contains the secret for the LLS or HLS authentication process.

NOTE In the case of HLS with GMAC, the (HLS) secret is held by the “Security setup” object referenced in attribute 9, *security\_setup\_reference*, 4.4.4.2.9.

#### **4.4.4.2.8 association\_status**

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

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

#### **4.4.4.2.9 security\_setup\_reference**

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

#### **4.4.4.2.10 user\_list**

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

```
array user_list_entry
```

```
user_list_entry ::= structure
```

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 110/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

```

{
    user_id:         unsigned,
    user_name: CHOICE
    {
        visible-string [10],
        UTF-8-string [12]
    }
}

```

Where:

- `user_id` is the identifier of the user (this value is carried by the calling-AE-invocation-id field of the AARQ);
- `user_name` is the name of the user.

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

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

### **4.4.4.2.11 current\_user**

Holds the identifier of the current user.

`current_user ::= user_list_entry (see 4.4.4.2.10)`

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

### **4.4.4.3 Method description**

#### **4.4.4.3.1 reply\_to\_HLS\_authentication (data)**

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

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

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

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

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

#### **4.4.4.3.2 change\_HLS\_secret (data)**

Changes the HLS secret (for example encryption key).

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

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 111/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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.

### **4.4.4.3.3 add\_object (data)**

Adds the referenced object to the *object\_list*.

data ::= object\_list\_element (see 4.4.4.2.2)

### **4.4.4.3.4 remove\_object (data)**

Removes the referenced object from the *object\_list*.

data ::= object\_list\_element (see 4.4.4.2.2)

### **4.4.4.3.5 add\_user (data)**

Adds a user to the *user\_list*.

data ::= user\_list\_entry (see 4.4.4.2.10)

### **4.4.4.3.6 remove\_user (data)**

Removes a user from the *user\_list*.

data ::= user\_list\_entry (see 4.4.4.2.10)

## **4.4.5 SAP assignment (class\_id = 17, version = 0)**

### **4.4.5.1 Overview**

This IC allows modelling the logical structure of physical devices, by providing information on the assignment of the logical devices to their SAPs. See DLMS UA 1000-2 Ed.11:2021, 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            |                            |      |      |            |

### **4.4.5.2 Attribute description**

#### **4.4.5.2.1 logical\_name**

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

#### **4.4.5.2.2 SAP\_assignment\_list**

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

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 112/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

```
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

### **4.4.5.3 Method description**

#### **4.4.5.3.1 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 (class\_id = 18, version = 0)**

### **4.4.6.1 General**

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

### **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).

For an example with more detailed explanations, see 4.4.6.6.

### **4.4.6.3 Overview**

This IC allows provides the attributes and methods with which to model the image transfer process.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 113/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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

### 4.4.6.4 Attribute description

#### 4.4.6.4.1 logical\_name

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

#### 4.4.6.4.2 image\_block\_size

Holds the ImageBlockSize, expressed in octets, which can be handled by the server. ImageBlockSize shall not exceed the ServerMaxReceivePduSize negotiated.

NOTE *image\_block\_size* is a property of the server.

#### 4.4.6.4.3 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 The size of the attribute may be dynamic, i.e. it may grow upon reception of new ImageBlocks.

#### 4.4.6.4.4 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.

#### 4.4.6.4.5 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).

#### 4.4.6.4.6 **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
```

#### 4.4.6.4.7 **image\_to\_activate\_info**

Provides information on the Image(s) ready for activation. It is generated as the result of the Image verification process. The client may check this information before activating the Images.

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

#### 4.4.6.5 **Method description**

##### 4.4.6.5.1 **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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 115/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

NOTE The *image\_identifier* 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 *image\_to\_activate* attribute after verification of the Image transferred.

After a successful invocation of the method the *image\_transfer\_status* attribute is set to (1) and the *image\_first\_not\_transferred\_block\_number* is set to 0. Any subsequent invocation of the method resets the whole Image transfer process and all ImageBlocks need to be transferred again.

### 4.4.6.5.2 **image\_block\_transfer (data)**

Transfers one block of the Image to the server.

```
data ::= structure
{
    image_block_number:      double-long-unsigned,
    image_block_value:       octet-string
}
```

After a successful invocation of the method the corresponding bit in the *image\_transferred\_blocks\_status* attribute is set to 1 and the *image\_first\_not\_transferred\_block\_number* attribute is updated.

### 4.4.6.5.3 **image\_verify (data)**

Verifies the integrity of the Image before activation.

```
data ::= integer (0)
```

The result of the invocation of this method may be success, temporary\_failure or other\_reason. If it is not success, then the result of the verification can be found by retrieving the value of the *image\_transfer\_status* attribute.

In the case of success, the *image\_to\_activate\_info* attribute holds the information about the images to activate.

NOTE xDLMS services Action-Result / Data-Access-Result codes are specified in DLMS UA 1000-2 Ed.11:2021, 9.5.

### 4.4.6.5.4 **image\_activate (data)**

Activates the Image.

```
data ::= integer (0)
```

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

The result of the invocation of this method may be success, temporary-failure or other-reason. If it is not success, then the result of the activation can be learned by retrieving the value of the *image\_transfer\_status* attribute.

NOTE xDLMS services Action-Result / Data-Access-Result codes are specified in DLMS UA 1000-2 Ed.11:2021, 9.5.

#### 4.4.6.6 Example for a standard image transfer process

##### 4.4.6.6.1 General

This subclause 4.4.6.6 provides an example of an Image transfer process from a client perspective, with a flow chart provided in Figure 13. Note that it is not mandatory to follow the process, some steps may be omitted or others may be necessary depending on the use case.

##### 4.4.6.6.2 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.

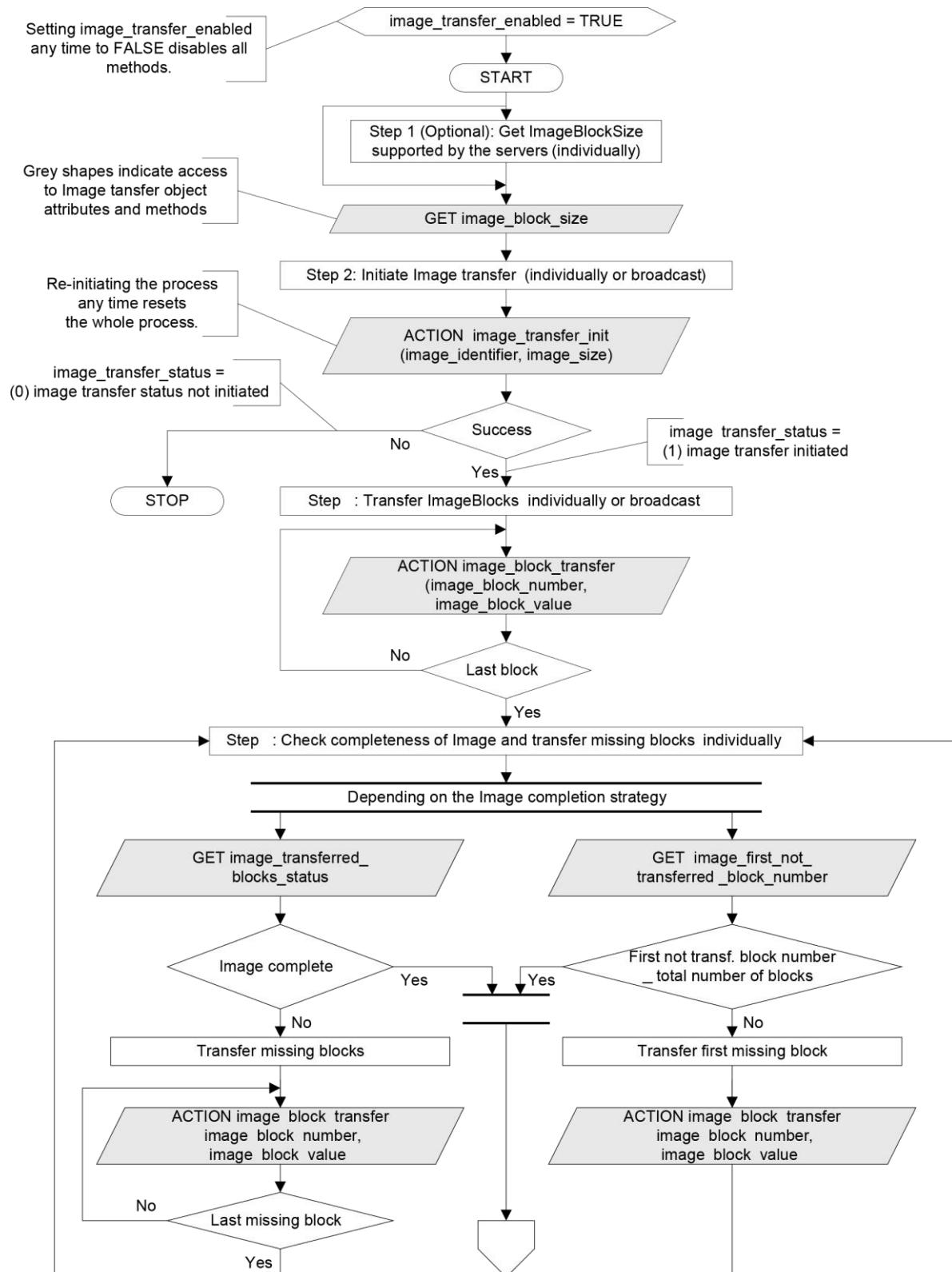
##### 4.4.6.6.3 Step 1 (Optional): Get ImageBlockSize

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.

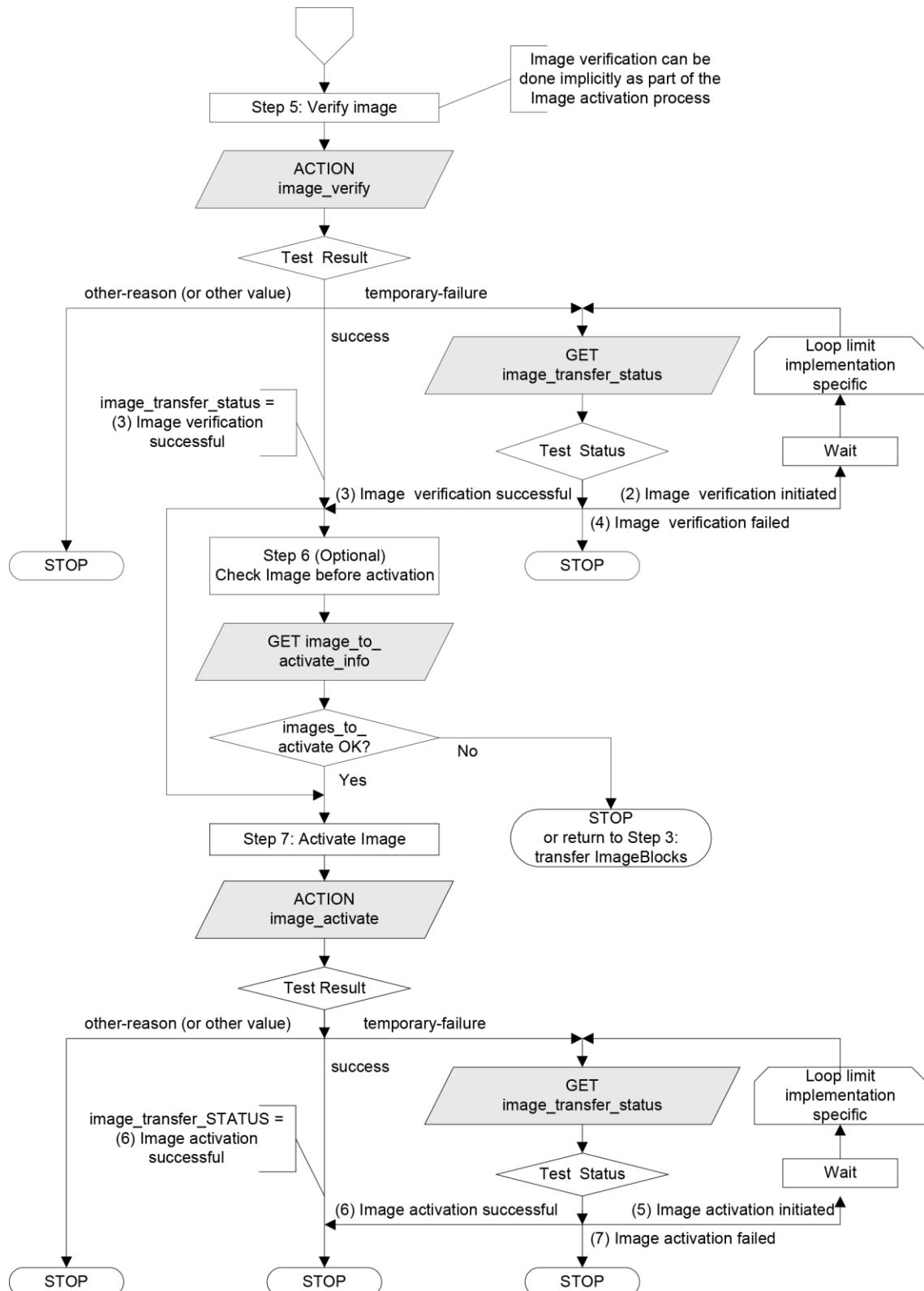
If ImageBlocks are sent using broadcast to a group of COSEM servers the ImageBlockSize shall be the same in each member of the group.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 117/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes



## COSEM Interface Classes



**Figure 13 – Image transfer process flow chart**

#### 4.4.6.6.4 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 sufficient memory space available to accommodate the Image.

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.

#### 4.4.6.6.5 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.

#### 4.4.6.6.6 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.

- 1) 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 **then** transfers the ImageBlocks not (yet) transferred;
- 2) 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 **then** transfers this ImageBlock not (yet) transferred;

After this, the client checks the completeness of the Image **again**.

**NOTE** The two mechanisms can be freely combined.

#### 4.4.6.6.7 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** The conditions of verifying the Image are out of the scope of this document.

#### 4.4.6.6.8 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.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 120/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

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

If the image transferred contains images with attributes that match those expected, the Server goes to step 7 and activates the image(s). However if the attributes do not match then the client can restart transferring the image.

#### 4.4.6.6.9 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.

#### 4.4.6.7 Image transfer for M-Bus devices

##### 4.4.6.7.1 Overview

The image transfer process for M-Bus devices is described in EN 13757-3:2018 Annex I. Using DLMS/COSEM, it is performed by following the steps described in 4.4.6.2.

- The M-Bus device image is first transferred from the DLMS client to the DLMS server.
- When the transfer of the image blocks to the DLMS server is complete, the DLMS client invokes the method for image verification. This prompts the DLMS server to initiate the process to transfer the image to the M-Bus device using M-Bus messages SND-UD2 and RSP-UD using CI-field as specified in EN 13757-3:2018, I.2.1.
- When the transfer is complete, the M-Bus device validates the image by setting its status to 'transfer successful'. This is communicated to the DLMS server.
- The DLMS server then informs the DLMS client that the status of the M-Bus device is now 'transfer successful'. The DLMS client then invokes the method for image activation which prompts the DLMS server to initiate the image activation process in the M-Bus device.
- When the activation has completed, the M-Bus device responds with its status 'activation successful'. The status is sent to the DLMS server which in turn sends it to the DLMS client and the process is image transfer is complete.

Figure 13 provides a sequence diagram showing this process in more detail.

##### 4.4.6.7.2 Image transfer steps for M-Bus devices

The image transfer steps described in 4.4.6.2 proceed as follows for M-Bus:

- Step 1: Optional, GET ImageBlockSize;
- Step 2: the image transfer is initiated by invoking the method "image\_transfer\_initiate" where the following structure is used for the parameter "image\_identifier":

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 121/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

|     |           |               |
|-----|-----------|---------------|
| MAN | M-Bus DEV | M-Bus PREPARE |
|-----|-----------|---------------|

MAN: Manufacturer (three letter) code according to FLAG, ASCII encoded

M-Bus DEV M-Bus DEV code (letters "MBUS"), ASCII encoded

M-Bus PREPARE M-Bus Prepare Command Structure (as specified in EN13757-3:2018, I.2.4 )

- Step 3: The DLMS client transfers the ImageBlocks to the DLMS server;
- Step 4: The DLMS client checks completeness of the Image. At this point, the DLMS client is aware that all the blocks in the image transfer have been successfully delivered to the DLMS server and the image is complete. This triggers Step 5;
- Step 5: The Server verifies the Image. This is when the image is transferred to the M-Bus device and hence is mandatory. The processes within this step are as follows:
  - i) Step 5 is initiated when the Client activates the *method image\_verify()*;
  - ii) The Server transfers the image to the M-Bus device using the procedure described in EN 13757-3:2018, Annex I;
  - iii) In the Transfer Preparation phase, the COSEM Server uses the M-Bus PREPARE information from the *image\_identifier* from Step 2 to issue the "Prepare Command" to the M-Bus device as specified in EN 13757-3:2018 I.2.4;
  - iv) In the Transfer Synchronization phase (optional) the COSEM Server issues the "Synchronize Command" to the M-Bus device as specified in EN 13757-3:2018 I.2.6;
  - v) In the Transfer of Image phase, the COSEM Server issues the "Transfer Command" to the M-Bus device for each Block of the Image as specified in EN 13757-3:2018 I.2.8.
  - vi) The image transfer is checked for completeness by issuing the "Completion Command" to the M-Bus device as specified in EN 13757-3:2018, I.2.10.
  - vii) In the Image Validation phase, the COSEM Server issues the "Validate Command" to the M-Bus device as specified in EN 13757-3:2018 I.2.14. In the case that the "State" received is not "Validation successful", then the "Validate Command" is followed (repeatedly) by a "State Command" to retrieve the results of the validation as described in EN13757-3:2018, I.2.12.).
  - viii) The attribute *image\_transfer\_status* is updated based on the value of the "State" received from the M-Bus device (see EN 13757-3:2018, Table I29). The M-Bus "state" maps to *image\_transfer\_status* according to Table 22:

**Table 22 – Mapping of M-Bus State to image\_transfer\_status**

| M-Bus "State"                | Attribute "image_transfer_status" |
|------------------------------|-----------------------------------|
| 0 Synchronization activated  | Not affected                      |
| 1 Synchronization terminated | Not affected                      |
| 2 Transfer initiated         | Not affected                      |
| 3 Transfer active            | Not affected                      |

## COSEM Interface Classes

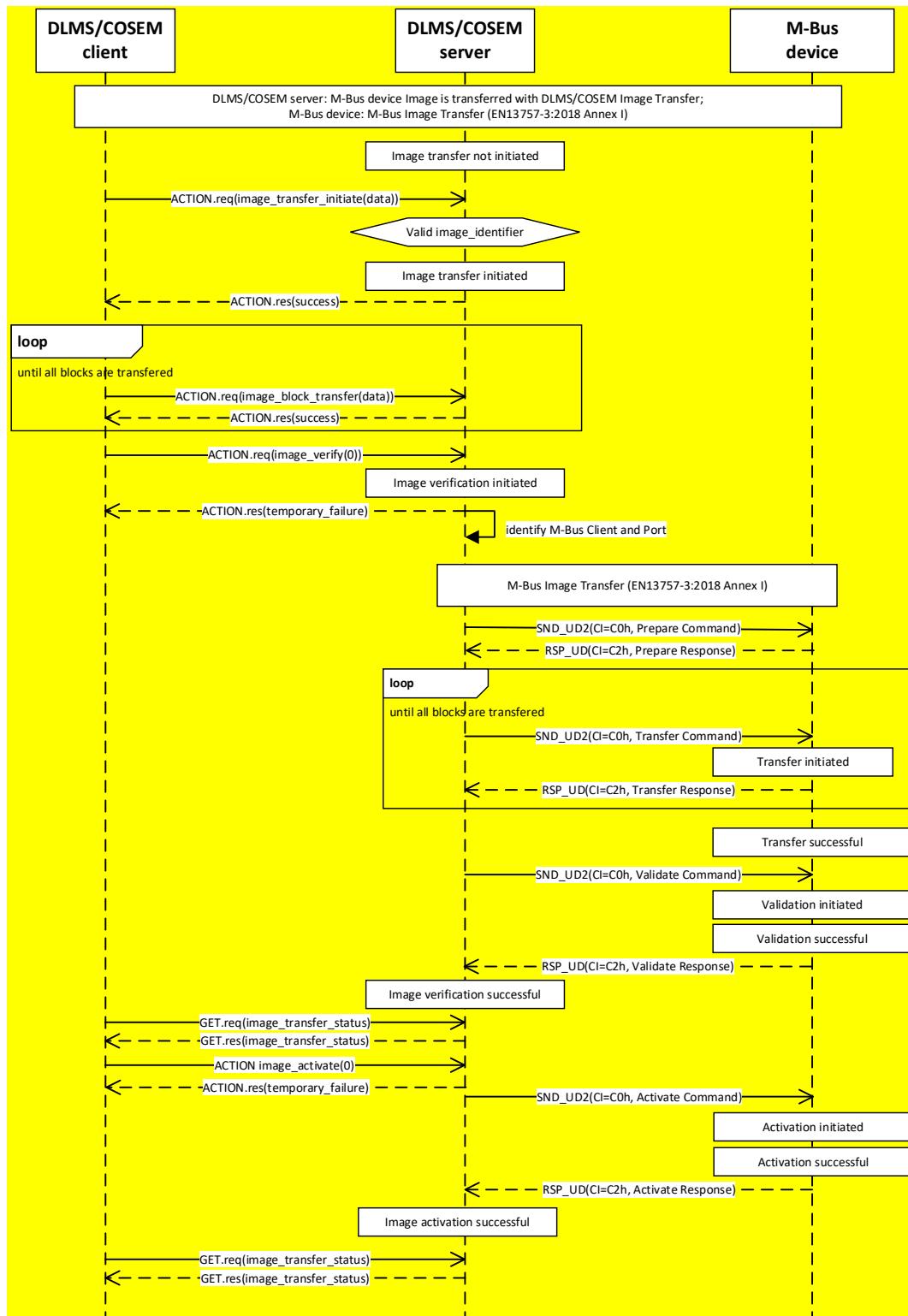
|                              |                                   |
|------------------------------|-----------------------------------|
| 4 Transfer terminated        | Not affected                      |
| 5 Transfer successful        | Not affected                      |
| 6 Transfer failed            | Not affected                      |
| 7 Validation initiated       | (2) Image verification initiated  |
| 8 Validation active          | Not affected                      |
| 9 Validation successful      | (3) Image verification successful |
| 10 Validation failed         | (4) Image verification failed     |
| 11 Activation initiated      | (5) Image activation initiated    |
| 12 Activation successful     | (6) Image activation successful   |
| 13 Activation failed         | (7) Image activation failed       |
| 14 Image Transfer Terminated | Not affected                      |
| 15 Idle                      | Not affected                      |

- 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). The image is activated by invoking the method `image_activate()`. This may be done directly by the COSEM client or via the object “Image transfer activation scheduler M-Bus channel x”. Upon invoking this method, the COSEM Server activates the image in the M-Bus device by issuing the “Activate command” with the parameters set for immediate activation. In the case that “State” received is not “Activation successful”, The “Activate Command” is followed (repeatedly) by a “State Command” to retrieve the results of the activation (see EN 13757-3:2018, I.2.12.)

The attribute `image_transfer_status` is updated based as defined in Table 22.

## COSEM Interface Classes

Image transfer with M-Bus transfer (EN 13757-3:2018 Annex A) is shown in Figure 14.



**Figure 14 – Image transfer with M-Bus transfer diagram**

#### 4.4.7 Security setup (class\_id = 64, version = 1)

##### 4.4.7.1 Overview

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   |
| <b>Specific methods</b>                | 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   |

##### 4.4.7.2 Attribute description

###### 4.4.7.2.1 logical\_name

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

###### 4.4.7.2.2 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, |

## COSEM Interface Classes

- 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 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.

### 4.4.7.2.3 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

### 4.4.7.2.4 client\_system\_title

Carries the (current) client system title:

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

NOTE 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, **such as** in the case of the S-FSK PLC profile, the client system title carried **by** 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.

### 4.4.7.2.5 server\_system\_title

Carries the server system title.

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

This attribute shall be read only.

### 4.4.7.2.6 certificates

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

array certificate\_info

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 126/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

```

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.11:2021, 9.2.6.4.

The octet-strings that hold the serial\_number, issuer, subject, and subject\_alt\_name shall include the tag, length and value (TLV) of the corresponding field of the Certificate.

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.

#### **4.4.7.3 Method description**

##### **4.4.7.3.1 security\_activate (data)**

Activates and strengthens the security policy:

data ::= enum, as specified at the *security\_policy* 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 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 127/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.4.7.3.2 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 1 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 2 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 3 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.

#### 4.4.7.3.3 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
{

```

## COSEM Interface Classes

```
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 1 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:  $Q_{e, U} \parallel \text{ECDSA}(\text{key\_id}) \parallel Q_{e, U}$ .

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 (array of) *key\_agreement\_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 2 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.

### 4.4.7.3.4 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 1 There is maximum one key pair for digital signature, one key pair for key agreement and one key pair for TLS.

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

### 4.4.7.3.5 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 129/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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

NOTE 1 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 2 It is the responsibility of the client to forward it to the Certification Authority.

```
response data ::= octet-string,
```

The octet-string contains the DER encoding of the CSR 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.

### **4.4.7.3.6 import\_certificate (data)**

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

```
data ::= octet-string
```

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

### **4.4.7.3.7 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  
    }  
}  
  
certificate_identification_by_entity ::= structure  
{  
    certificate_entity: enum:  
        (0) server,  
        (1) client,  
        (2) certification authority,  
        (3) other  
    certificate_type: enum:
```

## COSEM Interface Classes

```
(0) digital signature,  
(1) key agreement,  
(2) TL  
(3) other  
    system_title:          octet-string  
}  
  
certificate_identification_by_serial ::= structure  
{  
    serial_number:          octet-string,  
    issuer:                 octet-string  
}  
  
response data ::= octet-string  
response data octet-string is formatted as X.509 v3 DER format.
```

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

### 4.4.7.3.8 remove\_certificate (data)

Removes X.509 v3 certificate in the server.

```
data ::= certificate_identification  
  
certificate_identification see 4.4.7.3.7.
```

**NOTE** 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 signing request is generated and new certificate is imported

## 4.4.8 Push interface class

### 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).

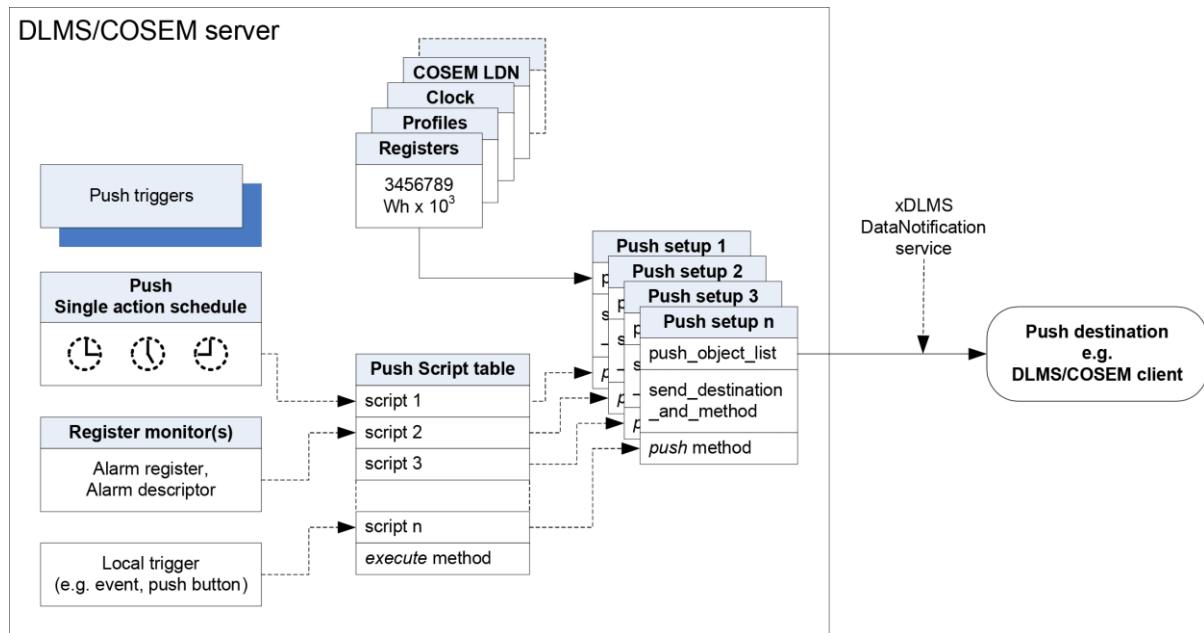
The DLMS/COSEM push mechanism follows the publish/subscribe pattern.

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 15.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 131/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes



**Figure 15 – 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.

When push uses a gateway, then the version of the gateway protocol for end devices without WAN/NN knowledge is to be applied. This is described in the DLMS UA 1000-2 Ed.11:2021, 10.9.4.3.

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 6.2.64.

**NOTE** 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* method of the “Push setup” object can be triggered (remotely or locally).. The security context is determined by the “Security setup” object referenced from the “Association” object.

NOTE Details are subject to project specific companion specifications.

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

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

#### 4.4.8.2 Push setup (class\_id = 40, version = 3)

##### 4.4.8.2.1 Overview

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.

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

In version 2 six new possibilities are specified:

- a new entry selection mechanism allows selecting entries related to the last confirmed entry;
- a new column selection mechanism allows explicitly selecting columns of Profile generic object buffer attributes;
- the **repetition\_delay** attribute now provides an exponential formula that allows increasing the repetition delay with each retry attempt;
- a new **push\_operation\_method** attribute allows specifying if the DataNotification.request service primitive is invoked with Service\_Class == Unconfirmed or confirmed and how the push retry operation works;
- a new **confirmation\_parameters** attribute allows limiting the time interval in which entries related to last confirmed entry can be selected;
- a new **last\_confirmation\_date\_time** attribute holds the date\_time when the DataNotification service was last confirmed.

The facility to use relative and absolute data selection was introduced from version 2 of the interface class. It would be common practice to use an instance of the Push setup IC with relative data selection for routine data push, and an instance of the Push setup IC with absolute data for special cases.

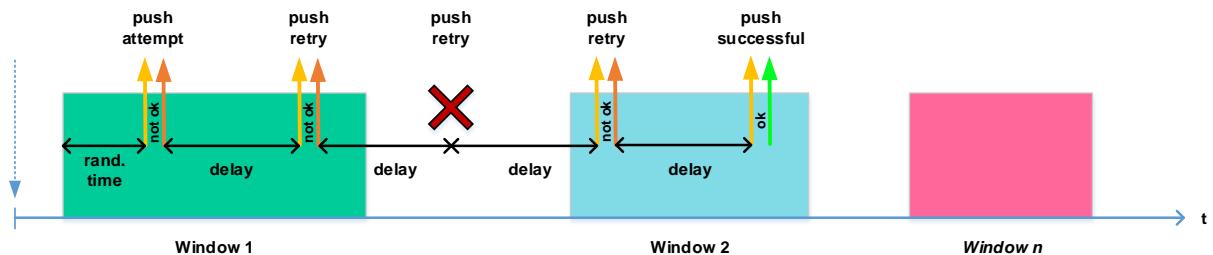
In this version 3, the *send\_destination\_and\_method* attribute and the *port\_reference* attribute of the Push IC is extended with a CoAP transport service type.

The push is started when the *push* method is invoked, triggered by a Push “Single action schedule” object, by an alarm “Register monitor” object, by a dedicated internal event or

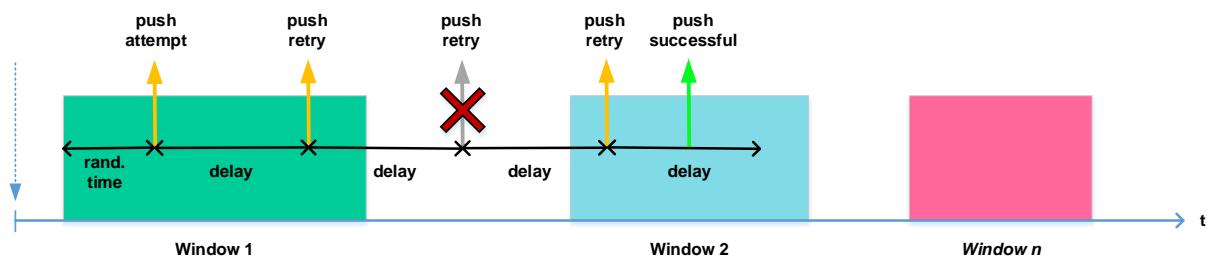
|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 133/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

externally. After the push operation has been triggered, it is executed according to the settings made in the given “Push setup” object. Depending on the communication window settings, the push is executed immediately or as soon as a communication window becomes active, after a random delay. If the push was not successful, retries are made. Push retries may be made when supporting protocol layer failure is indicated or when confirmation is missing. Push operation with windows, delays and retries is shown in Figure 16.



Case a) Retry on supporting layer failure



Case b) Retry on missing confirmation

**Figure 16 – Push windows, delays and retries**

## COSEM Interface Classes

| <b>Push setup</b>                        | <b>0...n</b>     | <b>class_id = 40, version = 2</b> |             |             |                   |
|--|------------------|-----------------------------------|-------------|-------------|-------------------|
| <b>Attribute (s)</b>                     | <b>Data type</b> | <b>Min.</b>                       | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 1. logical_name (static)                 | octet-string     |                                   |             |             | x                 |
| 2. push_object_list (static)             | array            |                                   |             |             | x + 0x08          |
| 3. send_destination_and_method (static)  | structure        |                                   |             |             | x + 0x10          |
| 4. communication_window (static)         | array            |                                   |             |             | x + 0x18          |
| 5. randomisation_start_interval (static) | long-unsigned    |                                   |             |             | x + 0x20          |
| 6. number_of_retries (static)            | unsigned         |                                   |             |             | x + 0x28          |
| 7. repetition_delay (static)             | structure        |                                   |             |             | x + 0x30          |
| 8. port_reference (static)               | octet-string     |                                   |             |             | x + 0x38          |
| 9. push_client_SAP (static)              | integer          |                                   |             |             | x + 0x40          |
| 10. push_protection_parameters (static)  | array            |                                   |             |             | x + 0x48          |
| 11. push_operation_method (static)       | enum             |                                   |             |             | x + 0x50          |
| 12. confirmation_parameters (static)     | structure        |                                   |             |             | x + 0x58          |
| 13. last_confirmation_date_time (dyn.)   | date-time        |                                   |             |             | x + 0x60          |
| <b>Specific methods</b>                  | <b>m/o</b>       |                                   |             |             |                   |
| 1. push (data)                           | m                |                                   |             |             | x + 0x38          |
| 2. reset (data)                          | o                |                                   |             |             | x + 0x70          |

### 4.4.8.2.2 Attribute description

#### 4.4.8.2.2.1 logical\_name

Identifies the “Push setup” object instance. See 6.2.24.

#### 4.4.8.2.2.2 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      push_object_definition
push_object_definition ::=   structure
{
    class_id:      long-unsigned,
    logical_name: octet-string,
    attribute_index: integer,
    data_index:      long-unsigned,
    restriction:    restriction_element,

```

## COSEM Interface Classes

```
        columns:          column_element
    }

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
}
column_element ::= array capture_object_definition
```

NOTE 1 For capture\_object\_definition please refer to the capture\_objects attribute of the Profile generic IC, see 4.3.6.2.3.

Where:

- class\_id, logical\_name and attribute\_index reference a COSEM object attribute. For referencing all attributes of an object, see DLMS UA 1000-2 Ed.11:2021, 9.1.4.3.7, attribute\_0 referencing.

If the attribute is simple or if attribute\_0 is referenced, then:

- data\_index has no meaning and shall be set to 0x0000;
- restriction\_element restriction\_type shall be 0 (none) and restriction\_element restriction\_value shall be null-data;
- column\_element shall be an array of 0 elements.

If the attribute is structure or an array – other than Profile generic object buffer then:

- data\_index set to 0x0000 selects the whole attribute;
- data\_index points to one element in the structure or array and can take any value between 0x0001 and 0xFFFF;
- restriction\_element restriction\_type shall be 0 (none) and restriction\_element restriction\_value shall be null-data;
- column\_element shall be an array of 0 elements.

If the attribute is a Profile generic object buffer, then data\_index, restriction and column define selective access parameters to select entries and columns.

There are three, mutually exclusive entry selection mechanisms available:

- 1) Relative selective access related to current date and time. Entries in last time intervals relative to current date and time or last entries as identified by data\_index are selected;
- 2) Relative selective access related to last confirmed entry. Entries in next time intervals relative to last confirmed entry or next entries as identified by data\_index are selected;
- 3) Absolute selective access. Entries in an explicitly defined date range or entry range as identified by restriction are selected.

There are two, mutually exclusive column selection mechanisms available:

- a) a defined number of columns starting from column 1 are selected;
- b) an explicitly defined list of columns are selected.

In the case of entry selection mechanism (1) and (2) the selective access parameters are defined by data\_index interpreted as three different fields as shown below. The encoding is shown in Table 9.

| <b>data_index</b> | <b>MS-Byte</b> |              | <b>LS-Byte</b> |
|-------------------|----------------|--------------|----------------|
|                   | Upper nibble   | Lower nibble |                |

Where:

- data\_index MS-Byte Upper nibble defines how the entries are selected;
- data\_index MS-Byte Lower nibble is used to specify the number of columns when applicable;
- data\_index LS-Byte defines the number of entries or time intervals.

In the case of entry selection mechanism (1) and (2), restriction\_element restriction\_type shall be 0 and restriction\_element restriction\_value shall be null\_data.

In the case of entry selection mechanism (3) the selective access parameters are defined by restriction\_element. In this case Data\_index MS\_byte Upper\_nibble shall be set to 0x0, and Data\_index LS\_byte shall be set to 0x00.

- for restriction by date range restriction\_type holds (1) range by date and restriction\_value holds restriction\_by\_date structure;
  - for restriction by entries restriction\_type holds (2) range by entry and restriction\_value holds restriction\_by\_entry structure;
- if selective access is not required, restriction\_type holds (0) none and restriction\_value holds null-data. This choice shall be taken also if relative selective access is used.

In the case of column selection mechanism (a), Data\_index MS\_byte Lower\_nibble defines the number of columns to be selected, starting from column 1. In this case, column\_element shall be an empty array.

In the case of column selection mechanism (b), the columns to be selected are identified by column\_element. In this case, Data\_index MS\_byte Lower\_nibble shall be set to 0.

If relative selective access related to last confirmed entry is used, the last confirmed entry is updated when the DataNotification.confirm service primitive is invoked with Result == CONFIRMED.

The DataNotification APDU carrying the push data is protected as stipulated by the security suite, security policy and the security material held by the “Security setup” object referenced

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 137/668 |
|-----------------------|------------|-----------------------------|---------|

from the Association SN / Association LN object within the application context of the AA which is linked to the push\_client\_SAP attribute; as well as by the push\_protection\_parameters attribute. For attributes to which no read access right is granted within this AA, or which cannot be accessed for any other reason, null-data should be returned.

NOTE 2 If the push\_object\_list array is empty, the push operation is disabled.

NOTE 3 The push\_object\_list attribute itself can be also pushed to identify the data pushed.

NOTE 4 Last confirmed entry is an internal variable maintained by the server AP

#### 4.4.8.2.2.3 send\_destination\_and\_method

Contains the destination address **and transport service** (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®  |
| (8)         | DLMS Gateway          |
| (9)         | Reliable CoAP,        |
| (10)        | Unreliable CoAP,      |
| (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.

**Format of a CoAP destination: The format of a CoAP destination is a CoAP URI.**

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 138/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

```
coap-URI = "coap:" "/" host [ ":" port ] path-abempty
```

Where the host part may be provided as an IP address. The host part shall not be empty. If a port is not specified, the default port 5683 is assumed for the CoAP-URI scheme. If a path is not specified, the empty path is assumed.

#### 4.4.8.2.2.4 communication\_window

Defines the time points when the communication window(s) for the push become(s) active (start\_time) and inactive (end\_time). See Figure 16.

```
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* including wildcards.

If the end of a communication window is reached when a push has already started the operation will be completed.

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

#### 4.4.8.2.2.5 randomisation\_start\_interval

Defines a maximum delay, in seconds, of sending the first DataNotification APDU after invoking the push method. This is to avoid a situation where many servers may push simultaneously. It is used as follows:

- the delay is random from 0 up to the maximum value. A value of 0 deactivates the mechanism;
- if the push is invoked within an active communication window, the random delay is applied. It is not applied for retries;
- if the push is invoked outside a communication window, then the random delay is applied when the communication windows opens;
- if the random delay extends beyond the current communication window, then the DataNotification APDU will be sent in the next window if available;
- if no communication windows are defined, then the random delay is applied whenever the push method is invoked including all retries.

#### 4.4.8.2.2.6 number\_of\_retries

Defines the maximum number of retries in the case of unsuccessful or skipped push attempts. After a successful push operation no further push attempts are made until the push operation is triggered again. For further details on the retransmission operation see the *push\_operation\_method* attribute (4.4.8.2.2.11).

#### 4.4.8.2.2.7 repetition\_delay

Defines the elements for calculating the time delay until the next attempt after an unsuccessful push. See Figure 16.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 139/668 |
|-----------------------|------------|-----------------------------|---------|

```

repetition_delay ::= structure
{
    repetition_delay_min: long-unsigned,
    repetition_delay_exponent: long-unsigned,
    repetition_delay_max: long-unsigned
}

```

Where:

- repetition\_delay\_min is the minimum delay, in seconds, until a next push attempt is started;
- repetition\_delay\_exponent is used for calculating the next delay;
- repetition\_delay\_max is the maximum delay, in seconds. Any calculated time delay will be capped by this value.

The repetition delay is calculated using the following formula:

$$\text{repetition\_delay} = \text{repetition\_delay\_min} \times (\text{repetition\_delay\_exponent} \times 0.01)^{n-1}$$

where n refers to the ordinal number of the push retries with n = 1 for the first retry etc.

**NOTE 1** The repetition delay itself is not influenced by the communication window. But a push retry only can be made if a communication window is active at that time. Otherwise it is handled like an unsuccessful push attempt.

**NOTE 2** The push data is not stored in an intermediate buffer. In the case of push retries, the current values of the attributes may change with every push retry attempt.

#### 4.4.8.2.2.8 port\_reference

Contains the logical name of a communication port setup object allowing the selection of a specific communication channel for the push based on the transport\_service\_type. This mainly applies in cases where several channels of the same type are supported.

If this information is not available or not needed, the attribute may be left empty (octet-string [0]).

- In the case where it is necessary to specify a specific CoAP transport layer instance to use, the port reference includes the logical name of the CoAP setup objects and optionally a UDP port:

`<CoAP setup object>[:<CoAP client UDP port>]`

The optional UDP port is specified when the CoAP client endpoint to use does not share port with the CoAP server endpoint specified in the CoAP setup object reference.

#### 4.4.8.2.2.9 push\_client\_SAP

Defines the client SAP where the push is directed to in the supporting layer of the DataNotification service. The push process takes place within the application context of the AA which is linked to the push\_client\_SAP attribute. The security context is determined by the Security setup object referenced in the related Association SN /LN object.

#### 4.4.8.2.2.10 push\_protection\_parameters

Specifies all protection parameters to be applied to the DataNotification APDU when the push data is sent. It offers the same options as the Data Protection IC but it is bound to the sending of the data defined in the push\_object\_list attribute.

## COSEM Interface Classes

```

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.

The fields transaction-id....other-information are A-XDR encoded OCTET STRINGS. The length and the value of each field are included in the AAD when applicable.

```

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

    key_info_options: CHOICE
    {
        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

```

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
| 141/668               |            |                             |

```
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.

The use of the various elements is the same as specified in Table 23 and Table 24 of the Data protection IC (class\_id = 30, version = 0).

#### **4.4.8.2.2.11 push\_operation\_method**

Defines if the DataNotification.request service primitive is invoked with Service\_Class == Unconfirmed or Confirmed and how the push retry operates.

```
enum:
(0) unconfirmed, retry on supporting protocol layer failure,
(1) unconfirmed, retry on missing supporting protocol layer confirmation,
(2) confirmed, retry on missing confirmation.
```

In case (0), the repetition delay for the next retry attempt is started upon supporting layer failure reported through the DataNotification.confirm service primitive.

In cases (1) and (2), the repetition delay for the next retry attempt is started when the DataNotification.request service primitive is invoked.

In cases (0) and (1), the push operation is deemed as successful when supporting layer confirmation is reported by invoking DataNotification.confim service primitive with Result == CONFIRMED.

In case (2), the push operation is deemed as successful when the server AL receives the data-notification-confirm APDU and invokes DataNotification.confirm service primitive with Service\_Class == Confirmed and Result == CONFIRMED.

#### **4.4.8.2.2.12 confirmation\_parameters**

Defines the selection of entries defined by data\_index to avoid pushing data from too far in the past. If entries have not yet been confirmed then all entries are selected. This attribute is applicable only for relative selective access related to last confirmation.

```
confirmation_parameters ::= structure
{
    confirmation_start_date:      date-time,
    confirmation_interval:       double-long-unsigned
}
```

Where:

- confirmation\_start\_date is the starting date and time for selecting entries. Fields of date-time not specified are not used. If all fields are not specified the use of confirmation\_start\_date is disabled;
- confirmation\_interval is a time interval in seconds backwards from the current date and time. If confirmation\_interval is set to zero, the use of confirmation\_interval is disabled.

#### **4.4.8.2.2.13 last\_confirmation\_date\_time**

Holds the date and time when the AL most recently invoked the DataNotification.confirm service primitive with Result == CONFIRMED.

*date-time* is formatted as specified in 4.1.6.1.

#### **4.4.8.2.3 Method description**

##### **4.4.8.2.3.1 push (data)**

Activates the push process leading to an attempt to send a DataNotification APDU carrying the push data.

```
data ::= integer(0)
```

##### **4.4.8.2.3.2 reset (data)**

Resets the push process to initial state.

```
data ::= integer(0)
```

#### **4.4.9 COSEM data protection (class\_id = 30, version = 0)**

##### **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.11:2021, 3.

Protection on COSEM data is aligned with and complements protection on xDLMS APDUs as defined in DLMS UA 1000-2 Ed.11:2021, 5.7.

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

- reading or writing 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;

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 143/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

- 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 server and another identified party, which may be a DLMS client or a third party.

Applying data protection between a DLMS 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.11:2021, 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 17 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:

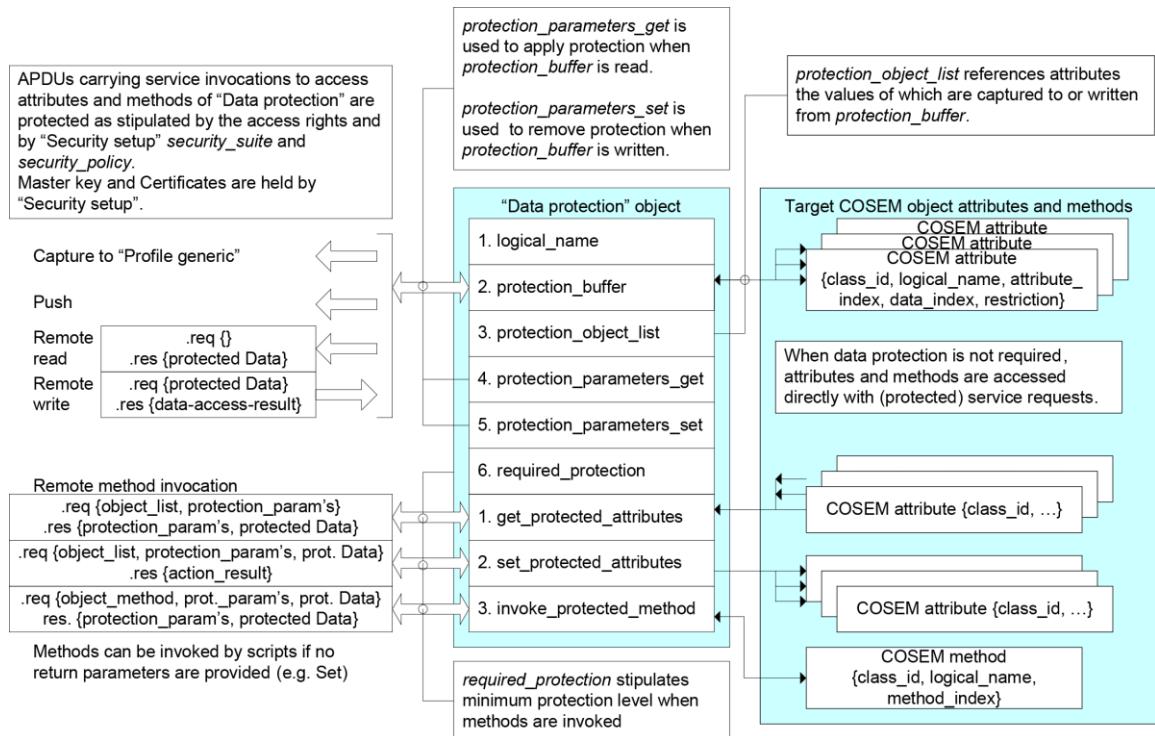
- reading or writing the *protection\_buffer* attribute. The *protection\_buffer* can be also captured in “Profile generic” objects or pushed using “Push setup” objects;
- invoking the *get\_protected\_attributes* / *set\_protected\_attributes* method.

For accessing a method of another COSEM object with protected data, the *invoke\_protected\_method* method is available.

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”.

## COSEM Interface Classes



**Figure 17 – COSEM model of data protection**

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

- 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.
- 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*.
- 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. If *protection\_parameters* do not satisfy *required\_protection* then the method invocation fails;
  - the protected attribute values are returned.
- 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*;

## COSEM Interface Classes

- the resulting attribute values are put in the attributes specified by the object\_list element;
- e) 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. If protection\_parameters do not satisfy required\_protection then the method invocation fails;
  - the protected method return parameters are returned.

Figure 18 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.11:2021, Figure 106.

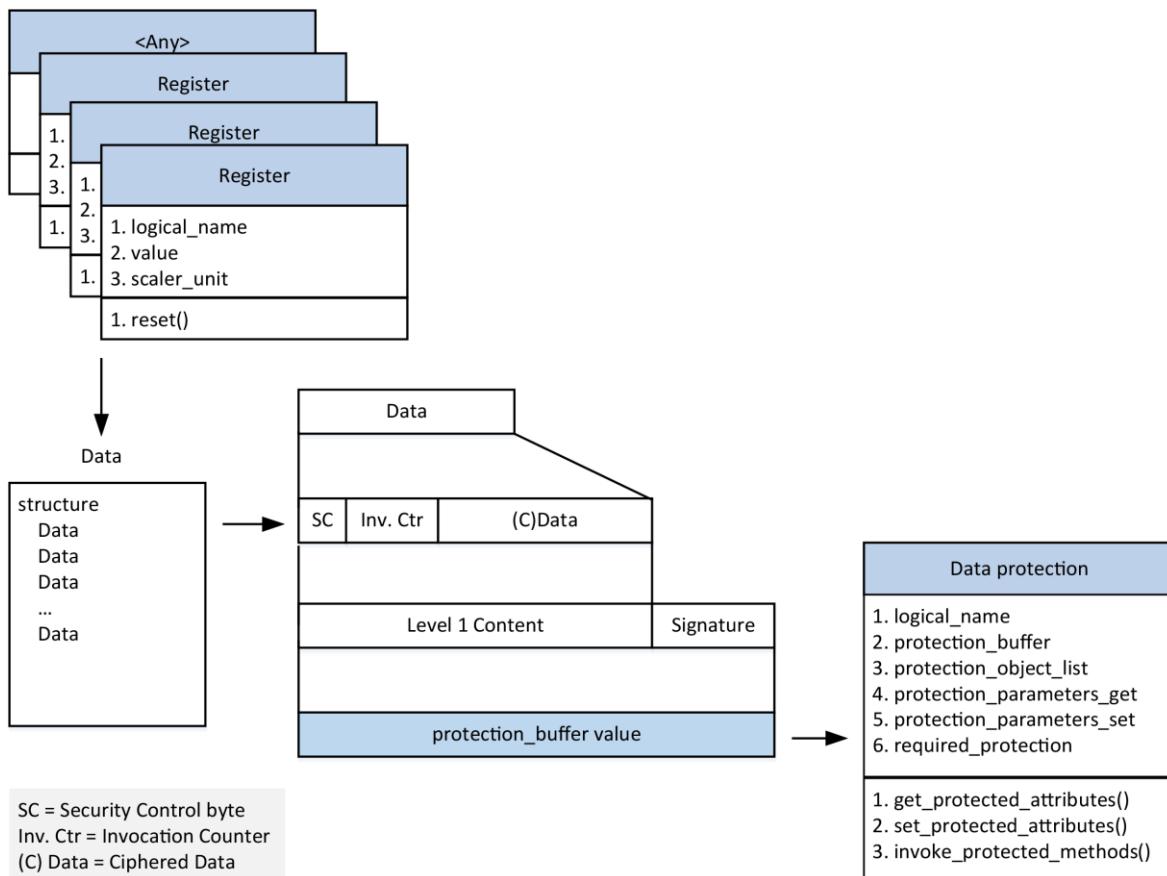
When the *protection\_buffer* attribute is read the following steps are performed:

- f) prerequisites: *protection\_object\_list*, *protection\_parameters\_get*, master key, key agreement and digital signature certificates as needed;
  - g) capture COSEM object attributes determined by *protection\_object\_list* and create Data, a structure containing the individual Data of the attributes captured;
  - h) protect Data according to *protection\_parameters\_get*.
- NOTE 4 In the example shown in Figure 18 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.
- i) put the protected data, of data type octet-string, into *protection\_buffer*;
  - j) 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.

## COSEM Interface Classes



**Figure 18 – Example: Read `protection_buffer` attribute**

| Data protection                    |          | 0...n        | class_id = 30, version = 0 |      |      |            |
|------------------------------------|----------|--------------|----------------------------|------|------|------------|
| Attribute (s)                      |          | Data type    | Min.                       | Max. | Def. | Short name |
| 1. logical_name                    | (static) | octet-string |                            |      |      | x          |
| 2. protection_buffer               | (dyn.)   | octet-string |                            |      |      | x + 0x08   |
| 3. protection_object_list          | (static) | array        |                            |      |      | x + 0x10   |
| 4. protection_parameters_get       | (static) | array        |                            |      |      | x + 0x18   |
| 5. protection_parameters_set       | (static) | array        |                            |      |      | x + 0x20   |
| 6. required_protection             | (static) | enum         |                            |      |      | x + 0x28   |
| Specific methods                   |          | m/o          |                            |      |      |            |
| 1. get_protected_attributes (data) |          | m            |                            |      |      |            |
| 2. set_protected_attributes (data) |          | m            |                            |      |      |            |
| 3. invoke_protected_method (data)  |          | m            |                            |      |      |            |

#### 4.4.9.2 Attribute description

##### 4.4.9.2.1 logical\_name

Identifies the “Data protection” object instance. See 6.2.36.

##### 4.4.9.2.2 protection\_buffer

Contains the protected Data.

When read, the attributes determined by *protection\_object\_list* are captured then protection is applied according to *protection\_parameters\_get*.

When written, the protected Data is put into the *protection\_buffer*, then the protection is verified / removed according to *protection\_parameters\_set* and the attributes determined by *protection\_object\_list* are set.

##### 4.4.9.2.3 protection\_object\_list

Defines the list of attributes to be captured to *protection\_buffer* when it is read or the list of attributes to be set when *protection\_buffer* is written.

Two, mutually exclusive selective access mechanisms are available:

- 1) relative selective access, i.e. entries defined relative to current date or entry are returned: this mechanism is controlled by the *data\_index* element; or
- 2) absolute selective access, i.e. entries in an explicitly defined date range or entry range are returned: this mechanism is controlled by the *restriction* 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 1<sup>st</sup> attribute (i.e. the *logical\_name*), *attribute\_index* 2 to the 2<sup>nd</sup> 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.

| <b>data_index:</b> | <b>MS-Byte</b> |              | <b>LS-Byte</b> |
|--------------------|----------------|--------------|----------------|
|                    | 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 9.

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:
  - MS Byte upper nibble set to 0x0;
  - MS Byte lower nibble set to 0x0 to 0xF in accordance with Table 9;
  - Lower Byte upper nibble set to 0x00;
- 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 149/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.4.9.2.4 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.

The fields *transaction-id* ....*other-information* are A-XDR encoded OCTET STRINGS. The length and the value of each field are included in the AAD when applicable.

```
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
    key_info_options: CHOICE
    {
        identified_key_info_options,
```

## COSEM Interface Classes

```
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 23 and Table 24

### **4.4.9.2.5 protection\_parameters\_set**

Contains all necessary parameters to verify / remove the protection applied when the *protection\_buffer* is written.

array protection\_parameters\_element

protection\_parameters\_element: see the *protection\_parameters\_get* attribute (4.4.9.2.4).

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 23 and Table 25

### **4.4.9.2.6 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, |

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 151/668 |
|-----------------------|------------|-----------------------------|---------|

- 2 authenticated request,
- 3 encrypted request,
- 4 digitally signed request,
- 5 authenticated response,
- 6 encrypted response,
- 7 digitally signed response

#### 4.4.9.3 Method description

##### 4.4.9.3.1 **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 (4.4.9.2.3);
- protection\_parameters ::= array protection\_parameters\_element.

protection\_parameters\_element: see the *protection\_parameters\_get* attribute (4.4.9.2.4).

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 23 and Table 26.

##### 4.4.9.3.2 **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.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 152/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

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 (4.4.9.2.3);
- protection\_parameters ::= array protection\_parameters\_element.

protection\_parameters\_element: see the *protection\_parameters\_get* attribute (4.4.9.2.4).

For the use of the various elements see Table 23 and Table 27.

#### **4.4.9.3.3 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 required\_protection – is applied on the return parameters, and invoke\_protected\_method\_response composed of protection\_parameters and protected\_method\_return\_parameters is returned.

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
}

```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 153/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

protection\_parameters\_element: see the specification of the protection\_parameters\_get attribute (4.4.9.2.4).

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.

To verify and remove protection from protected\_method\_invocation\_parameters, the server uses the protection\_parameters in invoke\_protected\_method\_request that must meet *required\_protection*.

The protection\_parameters in the 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 then applied to protect data in the protected\_method\_return\_parameters.

### 4.4.9.4 Key\_information and protection parameters

For the use of the various elements see Table 23 to Table 28.

**Table 23 – Key information required to establish data protection keys**

| <b>Key information choices</b>   |   | <b>Comment</b>   |
|--|---|--|
| key_info_options   |   |  |
| (0) <b>identified_key_info_options</b>                                 | S | The EK is identified   |
| (0) global_unicast_encryption_key                                      | S | GUEK   |
| (1) global_broadcast_encryption_key                                    | S | GBEK   |
| (1) <b>wrapped_key_info_options</b>                                    | 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.<br>0 = Master Key (KEK)  |
| key_ciphered_data  | M | Randomly generated key wrapped with KEK.   |
| (2) <b>agreed_key_info_options</b>                                     | S | The key is agreed by the parties using either: <ul style="list-style-type: none"><li>- the One-Pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme or</li><li>- the Static Unified Model C(0e, 2s, ECC CDH) scheme</li></ul>   |
| key_parameters   | M | Identifier of the key agreement scheme:<br>0x01: C(1e, 1s, ECC CDH)<br>0x02: C(0e, 2s, ECC CDH)<br>All other reserved.   |
| key_ciphered_data  | M | <ul style="list-style-type: none"><li>- In the case of the C(1e, 1s, ECC CDH) scheme: the public key <math>Q_u</math> of the ephemeral key agreement key pair of party U, signed with the private digital signature key of party U.</li><li>- In the case of the C(0e, 2s, ECC CDH) scheme: an octet-string of length zero.</li></ul> <p>NOTE In the second case party U has to provide a nonce, NonceU. See DLMS UA 1000-2 Ed.11:2021, 9.2.3.4.6.4 and 9.2.3.4.6.5.</p> |
| M: Mandatory (part of a structure)<br>S: Selectable (part of a CHOICE) |   |  |

## COSEM Interface Classes

**Table 24 – Protection parameters of *protection\_parameters\_get* attribute**

| <b>protection_parameters_get</b>   |   | <b>Written by the client</b> | <b>Filled in by the server</b> |
|--|---|------------------------------|--------------------------------|
| 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<br>(Shall carry the server system title)   | M | x                            |                                |
| recipient_system_title<br>(Shall carry the remote party system title)  | M | x                            |                                |
| other_information  | M | x                            |                                |
| key_info   | M | x                            |                                |
| key_info_type  | M | x                            |                                |
| (0) identified_key   | S | x                            |                                |
| (1) wrapped_key  | S | x                            |                                |
| (2) agreed_key   | S | x                            |                                |
| key_info_options   | M | x                            |                                |
| identified_key_options   | S | x                            |                                |
| global_unicast_encryption_key  | S | x                            |                                |
| global_broadcast_encryption_key  | S | x                            |                                |
| wrapped_key_options  | S | x                            |                                |
| kek_id   | M | x                            |                                |
| (0) master_key   | M | x                            |                                |
| key_ciphered_data  | M | x                            |                                |
| agreed_key_options   | S | x                            |                                |
| key_parameters   | M | x                            |                                |
| key_ciphered_data  | M |                              | x                              |
| The <i>protection_parameters_get</i> attribute is written by the client, but when <i>key_info_type</i> is (2) agreed key, the <i>key_ciphered_data</i> of <i>agreed_key_options</i> is empty.  |   |                              |                                |
| When the One-pass Diffie-Hellman c(1e, 1s, ECC CDH) scheme is used, this element is filled by the server and the remote party has to read back <i>protection_parameters_get</i> in order to be able to verify and remove protection. |   |                              |                                |
| M: Mandatory (part of a structure)<br>S: Selectable (part of a CHOICE)   |   |                              |                                |

## COSEM Interface Classes

**Table 25 – Protection parameters of *protection\_parameters\_set* attribute**

| <b>protection_parameters_set</b>                                       |   | <b>Written by<br/>the client</b> | <b>Filled in by<br/>the server</b> |
|--|---|----------------------------------|------------------------------------|
| 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<br>(Shall carry the remote party system title) | M | x                                |                                    |
| recipient_system_title<br>(Shall carry the server systems title)       | M | x                                |                                    |
| other_information  | M | x                                |                                    |
| key_info   | M | x                                |                                    |
| key_info_type  | M | x                                |                                    |
| (0) identified_key   | S | x                                |                                    |
| (1) wrapped_key  | S | x                                |                                    |
| (2) agreed_key   | S | x                                |                                    |
| key_info_options   | M | x                                |                                    |
| identified_key_options   | S | x                                |                                    |
| (0) global_unicast_encryption_key                                      | S | x                                |                                    |
| (1) global_broadcast_encryption_key                                    | S | x                                |                                    |
| wrapped_key_options  | S | x                                |                                    |
| kek_id   | M | x                                |                                    |
| (0) master_key   | M | x                                |                                    |
| key_ciphered_data  | M | x                                |                                    |
| agreed_key_options   | S | x                                |                                    |
| key_parameters   | M | x                                |                                    |
| key_ciphered_data  | M | x                                |                                    |

M: Mandatory (part of a structure)

S: Selectable (part of a CHOICE)

## COSEM Interface Classes

**Table 26 – 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 <sup>1</sup> |
| recipient_system_title   | M       | M <sup>2</sup> |
| 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 <sup>3</sup> |
| agreed_key_options   | S       | S (=)          |
| key_parameters   | M       | M (=)          |
| key_ciphered_data  | –       | M <sup>4</sup> |
| Notice that protection parameters are taken from request and put into response. Protection parameters are only applied for protection of response. |         |                |
| <sup>1</sup> originator_system_title from request is put into recipient_system_title of response.  |         |                |
| <sup>2</sup> recipient_system_title from request is put into originator_system_title of response.  |         |                |
| <sup>3</sup> key_ciphered_data is filled by the server.  |         |                |
| <sup>4</sup> When the One-pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme is used, this element is filled by the server.                             |         |                |
| M: Mandatory (part of a structure)   |         |                |
| S: Selectable (part of a CHOICE)   |         |                |

## COSEM Interface Classes

**Table 27 – Protection parameters of *set\_protected\_attributes* method**

| Protection parameters   | request | response |
|---|---------|----------|
| array protection_parameters_element   | M       |          |
| protection_type   | M       |          |
| (0) authentication  | S       |          |
| (1) encryption  | S       |          |
| (2) authentication and encryption   | S       |          |
| (3) digital signature   | S       |          |
| protection_options  | M       |          |
| transaction_id  | M       |          |
| originator_system_title   | M       |          |
| recipient_system_title  | M       |          |
| other_information   | M       |          |
| key_info  | M       |          |
| key_info_type   | M       |          |
| (0) identified_key  | S       |          |
| (1) wrapped_key   | S       |          |
| (2) agreed_key  | S       |          |
| key_info_options  | M       |          |
| identified_key_options  | S       |          |
| (0) global_unicast_encryption_key   | S       |          |
| (1) global_broadcast_encryption_key   | S       |          |
| wrapped_key_options   | S       |          |
| kek_id  | M       |          |
| (0) master_key  | M       |          |
| key_ciphered_data   | M       |          |
| agreed_key_options  | S       |          |
| key_parameters  | M       |          |
| key_ciphered_data   | M       |          |
| Notice that protection parameters are taken from request and are not present in response. Protection parameters are only used for removing protection on the request. |         |          |
| M: Mandatory (part of a structure)  |         |          |
| S: Selectable (part of a CHOICE)  |         |          |

## COSEM Interface Classes

**Table 28 – 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 <sup>1</sup> |
| recipient_system_title              | M       | M <sup>2</sup> |
| 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 <sup>3</sup> |
| agreed_key_options                  | S       | S(=)           |
| key_parameters                      | M       | M(=)           |
| key_ciphered_data                   | M       | M <sup>4</sup> |

Notice that protection parameters are taken from request and put into response. Protection parameters in the request are used to remove protection from request and protection parameters in the response are used to apply protection on response.

1 originator\_system\_title from request is put into recipient\_system\_title of response.

2 recipient\_system\_title from request is put into originator\_system\_title of response.

3 key\_ciphered\_data is sent by the originator in the request and filled by the server for the response.

4 When the One-pass Diffie-Hellman C(1e, 1s, ECC CDH) scheme is used, this element is filled by the server.

M: Mandatory (part of a structure)

S: Selectable (part of a CHOICE)

#### 4.4.10 Function control (class\_id: 122, version: 0)

##### 4.4.10.1 Overview

Instances of the IC “Function control” allow enabling and disabling functions in the server. Each function that can be enabled / disabled is identified by a name and is defined by a particular set of object identifiers referenced.

To allow enabling and disabling of functions controlled by time, “Single action schedule” and “Script table” objects are also specified.

| Function control              | 0...n        | class_id = 122, version = 0 |      |      |            |
|-------------------------------|--------------|-----------------------------|------|------|------------|
| Attributes                    | Data type    | Min.                        | Max. | Def. | Short name |
| 1. logical_name (static)      | octet-string |                             |      |      | x          |
| 2. activation_status (dyn.)   | array        |                             |      |      | x + 0x08   |
| 3. function_list (static)     | array        |                             |      |      | x + 0x10   |
| Specific methods              | m/o          |                             |      |      |            |
| 1. set_function_status (data) | m            |                             |      |      | x + 0x20   |
| 2. add_function (data)        | o            |                             |      |      | x + 0x28   |
| 3. remove_function (data)     | o            |                             |      |      | x + 0x30   |

##### 4.4.10.2 Attribute description

###### 4.4.10.2.1 logical\_name

Identifies the “Function control” object instance. See 6.2.38.

###### 4.4.10.2.2 activation\_status

Shows the current status of each functional block defined in the *function\_list* attribute.

```
activation_status ::= array function_status_type
function_status_type ::= structure
{
    function_name octet-string,
    function_status boolean
}
```

TRUE:      function enabled,  
 FALSE:     function disabled.

###### 4.4.10.3 function\_list

Defines a list of functions which can be enabled or disabled by invoking the *set\_function\_status* method.

```
function_list ::= array functional_block
functional_block ::= structure
{
    function_name: octet-string,
    function_specification: array function_definition
```

```

    }

function_definition ::= structure
{
    class_id:          long-unsigned,
    logical_name:      octet-string
}

```

The function\_specification element contains a list of objects involved in this function. The objects are referenced by their class ids and logical names.

In case of functions that cannot be linked to specific objects, class\_id = 0 is used and the second element holds a descriptive name instead of a logical\_name.

The names of the functions and the behaviour of the server in case of enabling or disabling a function have to be defined in project specific companion specifications.

Please also note that this attribute controls only the functions that are available in the server and that can be enabled or disabled. It is not possible to download new functions to the server by modifying this attribute or by invoking the *add\_function* method.

#### **4.4.10.4 Method description**

##### **4.4.10.4.1 set\_function\_status (data)**

Enables or disables one or more functions defined in attribute *function\_list*.

data ::= array function\_status\_type

as defined in attribute *activation\_status* (4.4.10.2.2).

The status of functions that are not listed is not modified.

##### **4.4.10.4.2 add\_function (data)**

Adds a new function to the attribute *function\_list*.

If a function with the same name already exists, the existing function will be replaced by the new function.

data ::= functional\_block

functional\_block see 4.4.10.3.

##### **4.4.10.4.3 remove\_function (data)**

Removes a function from the attribute *function\_list*.

data ::= octet-string, holding the function\_name.

#### 4.4.11 Array manager (class\_id = 123, version = 0)

##### 4.4.11.1 Overview

Instances of the “Array manager” IC allow managing attributes of type *array* of other interface objects, i.e.:

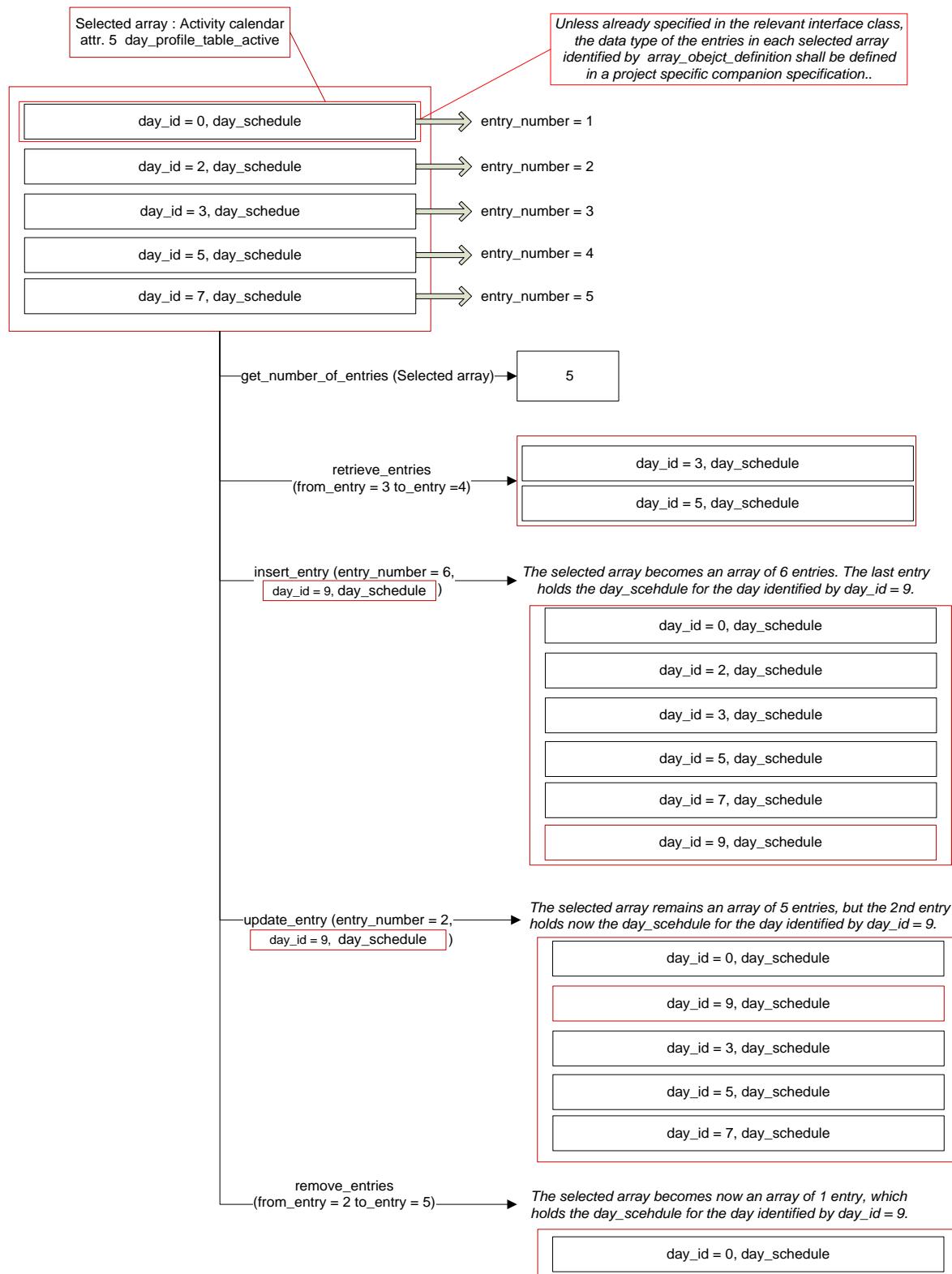
- retrieving the number of entries;
- selectively reading a range of entries;
- inserting a new entry or updating an existing entry;
- removing a range of entries.

Each instance allows managing several attributes of type *array* assigned to it.

An example of the application is shown in Figure 19.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 163/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes



**Figure 19 – Example of managing an array**

## COSEM Interface Classes

| Array manager                        | 0...n        | class_id = 123, version = 0 |      |      |            |
|--------------------------------------|--------------|-----------------------------|------|------|------------|
| Attributes                           | Data type    | Min.                        | Max. | Def. | Short name |
| 1. logical_name (static)             | octet-string |                             |      |      | x          |
| 2. array_object_list (static)        | array        |                             |      |      | x + 0x08   |
| Specific methods                     | m/o          |                             |      |      |            |
| 1. retrieve_number_of_entries( data) | m            |                             |      |      |            |
| 2. retrieve_entries (data)           | m            |                             |      |      |            |
| 3. insert_entry (data)               | o            |                             |      |      |            |
| 4. update_entry (data)               | o            |                             |      |      |            |
| 5. remove_entries (data)             | o            |                             |      |      |            |

### 4.4.11.2 Attribute description

#### 4.4.11.2.1 logical\_name

Identifies the “Array manager” object instance. See 6.2.16.

#### 4.4.11.2.2 array\_object\_list

Defines the list of attributes of type array that can be managed by an instance of this IC. Each “Array manager” object can manage 1 to n arrays.

```

array      array_object_list_definition

array_object_list_definition ::=   structure
{
    array_object_id:           unsigned,
    array_object_list_element: array_object_definition
}
array_object_definition ::=   structure
{
    class_id:     long-unsigned,
    logical_name: octet-string,
    attribute_index: integer
}

```

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.

Each attribute identified by the array\_object\_definition shall be of type array.

An attribute of type array holds a number of entries that can be referenced by their entry number. The number 1 identifies the first entry and the number m identifies the last entry, where m is the number of entries of a selected array.

Unless already specified in the relevant interface class, the data type of the entries in each array identified by array\_object\_definition shall be specified in a project specific companion specification.

When the target array is accessed through an “Array manager” object, its access rights are observed.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 165/668 |
|-----------------------|------------|-----------------------------|---------|

**4.4.11.3 Method description****4.4.11.3.1 retrieve\_number\_of\_entries (data)**

Returns the number of entries in the array identified.

The method invocation parameter identifies an array from the array\_object\_list by its array\_object\_id:

data ::= unsigned

The return parameter holds the number of entries in the array identified.

**4.4.11.3.2 retrieve\_entries (data)**

Returns a range of entries in one of the arrays of the array\_object\_list.

The method invocation parameters identify an array from the array\_object\_list by its array\_object\_id and the entries to be retrieved:

```
data ::= entries_to_retrieve

entries_to_retrieve ::= structure
{
    array_object_id: unsigned,
    list_of_entries: structure
    {
        from_entry:      long-unsigned,
        to_entry:        long-unsigned
    }
}
```

Where:

- array\_object\_id identifies one of the arrays managed by an “Array manager” object;
- list\_of\_entries identifies the entries to be retrieved.

If the number of entries in the selected array is m:

- if from\_entry < last entry < m, then the entries in this range are retrieved;
- if from\_entry < last entry but last\_entry > m, then the entries identified by from\_entry up to the last entry are retrieved;
- if from\_entry < last\_entry, but from\_entry > m, then the method returns an array of 0 elements;
- if from\_entry > last\_entry the method invocation fails.

The return parameters hold an array of the entries selected:

```
data ::=     retrieved_entries:

retrieved_entries      array entry

entry ::= attribute specific
```

The data type depends on the attribute of type *array* as specified in the IC specification or in the project specific companion specification.

#### 4.4.11.3.3 insert\_entry (data)

Inserts a new entry in a selected array.

```
data ::=      structure
{
    array_object_id:      unsigned,
    entry_to_insert:      structure
    {
        entry_number:    long-unsigned,
        entry:           attribute specific
    }
}
```

The data type of the entry depends on the attribute of type *array* as specified in the IC specification or in the project specific companion specification.

Where:

- *array\_object\_id* identifies one of the arrays managed by an “Array manager” object;
- *entry\_to\_insert* identifies the *entry\_number* and holds the entry itself to be inserted.

If *entry\_number* = 0:

- if the selected array is already full, then the method invocation fails;
- else, the new entry is inserted at the beginning of the array: it becomes the first entry and all the other entries of the selected array are shifted up by one;

If *entry\_number* > m (where m is the number of entries in the array):

- if the selected array is already full, then the method invocation fails;
- else, the new entry is inserted at the end of the selected array: it becomes the last entry. The other entries are not shifted.

If an *entry\_number* identifies an existing entry, it is inserted after the existing entry.

If the entries are shifted due to inserting a new entry then all the references to them have to be updated.

#### 4.4.11.3.4 update\_entry (data)

Updates an existing entry in a selected array.

```
data ::=      structure
{
    array_object_id:      unsigned,
    entry_to_update:      structure
    {
        entry_number:    long-unsigned,
        entry:           attribute specific
    }
}
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 167/668 |
|-----------------------|------------|-----------------------------|---------|

```

        The data type of the entry depends on the attribute of type
        array as specified in the IC specification or in the project
        specific companion specification.
    }
}

```

Where:

- array\_object\_id identifies one of the arrays managed by an “Array manager” object;
- entry\_to\_update identifies the entry\_number and holds the entry itself to be updated.

When an entry\_number identifies an existing entry, it is overwritten.

In any other case, the method invocation fails.

#### **4.4.11.3.5 remove\_entries (data)**

Removes a range of entries in one of the arrays of the array\_object\_list.

The method invocation parameters identify an array from the array\_object\_list by its array\_object\_id and the entries to be removed:

```

data ::= entries_to_remove

entries_to_remove ::= structure
{
    array_object_id: unsigned;
    list_of_entries: structure
    {
        from_entry:      long-unsigned,
        to_entry:        long-unsigned
    }
}

```

Where:

- array\_object\_id identifies one of the arrays managed by an “Array manager” object;
- list\_of\_entries identifies the entries to be removed.

If the number of entries in the selected array is m:

- if from\_entry < last\_entry < m, then the entries in this range are removed;
- if from\_entry < last\_entry but last\_entry > m, then the entries identified by from\_entry up to the last entry are removed;
- if from\_entry < last\_entry, but from\_entry > m, then no entries are removed;
- if from\_entry > last\_entry the method invocation fails.

If the entries are shifted due to removing entries, then all the references to them have to be updated.

#### 4.4.12 Communication port protection (class\_id = 124, version = 0)

##### 4.4.12.1 Overview

Instances of the “Communication port protection” IC can be used to protect communication ports of DLMS servers against possibly malicious communication attempts, in particular to prevent brute force attacks by reducing the possible number of attempts.

Each instance references a single communication port by its logical name (OBIS code). If an acceptable number of failed attempts is exceeded then the communication port is temporarily locked. The lockout time may increase with each failed attempt, until a maximum lockout time is reached.

A failed attempt is one that leads to discarding the APDU carrying a service request. The criteria for detecting a failed attempt are out of the Scope of this document.

The objects count both the number of failed attempts between two resets and the cumulative number of failed attempts.

It is possible to configure the communication ports such that they are locked to all attempts or unlocked to all attempts

| Communication port protection            | 0...n                | class_id = 124, version = 0 |      |      |            |
|--|----------------------|-----------------------------|------|------|------------|
| Attributes                               | Data type            | Min.                        | Max. | Def. | Short name |
| 1. logical_name<br>(static)              | octet-string         |                             |      |      | x          |
| 2. protection_mode<br>(static)           | enum                 |                             |      | 1    | x + 0x08   |
| 3. allowed_failed_attempts<br>(static)   | long-unsigned        |                             |      |      | x + 0x10   |
| 4. initial_lockout_time<br>(static)      | double-long-unsigned |                             |      |      | x + 0x18   |
| 5. steepness_factor<br>(static)          | unsigned             |                             |      | 1    | x + 0x20   |
| 6. max_lockout_time<br>(static)          | double-long-unsigned |                             |      |      | x + 0x28   |
| 7. port_reference<br>(static)            | octet-string         |                             |      |      | x + 0x30   |
| 8. protection_status<br>(dyn.)           | enum                 |                             |      |      | x + 0x38   |
| 9. failed_attempts<br>(dyn.)             | double-long-unsigned |                             |      |      | x + 0x40   |
| 10. cumulative_failed_attempts<br>(dyn.) | double-long-unsigned |                             |      |      | x + 0x48   |
| Specific methods                         | m/o                  |                             |      |      |            |
| 1. reset (data)                          | o                    |                             |      |      | x + 0x50   |

##### 4.4.12.2 Attribute description

###### 4.4.12.2.1 logical\_name

Identifies the “Communication port protection” object instance. See 6.2.39.

###### 4.4.12.2.2 protection\_mode

Controls the protection mode.

```
enum
  (0) locked,
  (1) locked_on_failed_attempts,
  (2) unlocked
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 169/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

When set to locked, no communication can take place via the port referenced by attribute 7.

When set to “locked\_on\_failed\_attempts”, the port becomes temporarily locked when the number of failed communication attempts exceeds an allowed number. The length of the lockout period may increase with each failed attempt, as determined by attributes 5 and 6.

When set to “unlocked” the lockout mechanism is disabled; communication through the port referenced by attribute 7 is always possible.

When the port is locked no communication can take place, even valid APDUs are discarded.

### 4.4.12.2.3 allowed\_failed\_attempts

Holds the number of allowed failed communication attempts before the lockout mechanism is triggered.

### 4.4.12.2.4 initial\_lockout\_time

Holds the initial value of the lockout time, in seconds, after the first failed communication attempt when the value of allowed\_failed\_attempts is reached.

### 4.4.12.2.5 steepness\_factor

Holds a factor that controls how the lockout time is increased with each failed attempt when allowed\_failed\_attempts is reached, until the max\_lockout\_time is reached.

The current lockout time is calculated using the following formulae:

$$NCA = \text{failed\_attempts} - \text{allowed\_failed\_attempts}$$

$$CLT = \text{initial\_lockout\_time} \times (\text{steepness\_factor}^{(NCA-1)})$$

Where:

- NCA is number of counted failed attempts: that is: the number of attempts that count towards extending the lockout time,
- CLT is the current lockout time.

Table 31 shows example values with an initial lockout time of 60s.

**Table 29 – Example values for NCA and CLT**

| NCA | steepness_factor |     |      |
|-----|------------------|-----|------|
|     | 1                | 2   | 3    |
|     | CLT (in seconds) |     |      |
| 1   | 60               | 60  | 60   |
| 2   | 60               | 120 | 180  |
| 3   | 60               | 240 | 540  |
| 4   | 60               | 480 | 1620 |
| 5   | 60               | 960 | 4860 |

Once lockout occurs, even valid attempts are discarded until the current lockout time has expired.

#### **4.4.12.2.6 max\_lockout\_time**

Holds the maximum time, in seconds, for which the communication port can be locked, even if the number of failed attempts keeps increasing. The purpose of this attribute is to avoid a denial of service attack.

#### **4.4.12.2.7 port\_reference**

Contains the logical name of an object that identifies the communication port being protected. That object may be a communication port setup object or any other suitably chosen object.

If this information is not available or not needed, the attribute may be left empty (octet-string [0]).

#### **4.4.12.2.8 protection\_status**

Holds the current protection status of the communication port.

enum

- (0) unlocked,
- (1) temporarily\_locked,
- (2) locked

The status is unlocked if the protection\_mode is unlocked or the number of failed attempts does not exceed the number of allowed\_failed\_attempts.

NOTE 1 The status becomes unlocked again when the current\_lockout\_time has elapsed.

The status is temporarily\_locked if the number of failed communication attempts exceeds allowed\_failed\_attempts and the current\_lockout\_time has not yet expired.

The status is locked if the protection\_mode is locked.

NOTE 2 When a port is locked, the server still may be accessed on another port.

#### **4.4.12.2.9 failed\_attempts**

Holds the total number of failed attempts since the last reset, independent of the protection\_mode and the triggering of the protection mechanism.

#### **4.4.12.2.10 cumulative\_failed\_attempts**

Holds the cumulative number of failed attempts, independent of the protection\_mode and the triggering of the protection mechanism.

This attribute is never reset.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 171/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.4.12.3 Method description

##### 4.4.12.3.1 reset (data)

Resets the variables of the lockout mechanism:

- the value of failed\_attempts is reset to 0;
- the current lockout time is reset to 0;
- the protection\_status is set to "(0) unlocked" if the protection\_mode is (1) locked\_on\_failed\_attempts, and is not affected otherwise.

```
data ::= integer (0)
```

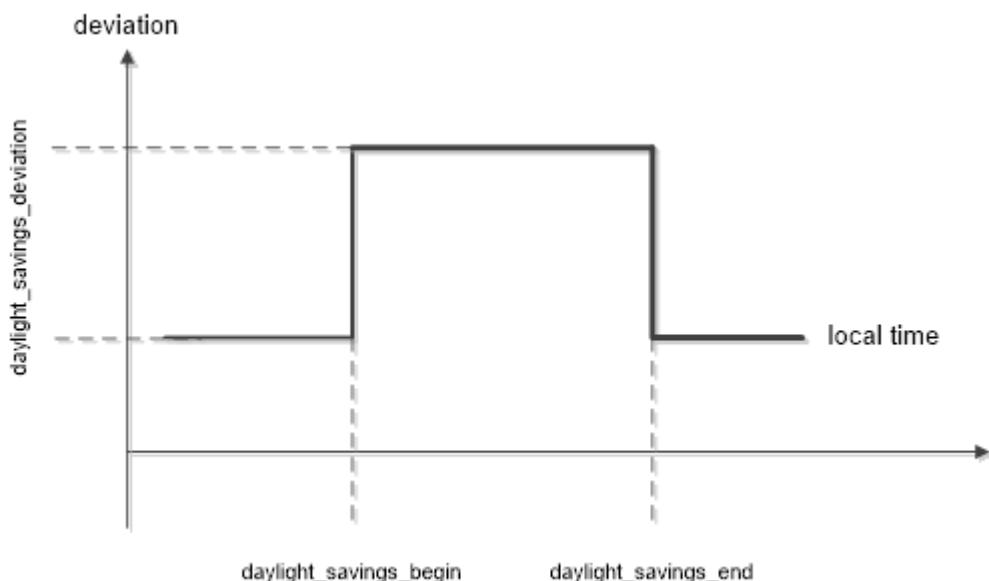
## 4.5 Interface classes for time- and event bound control

### 4.5.1 Clock (class\_id = 8, version = 0)

#### 4.5.1.1 Overview

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

The *date* information includes the elements year, month, day of month and day of week. The *time* information includes the elements hour, minutes, seconds, hundredths of seconds, and the deviation of the local time from UTC. The daylight saving time function modifies the deviation of local time to UTC depending on the attributes; see Figure 20. 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 20 – The generalized time concept**

## COSEM Interface Classes

| 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         | -840                      | +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                      | +120 |      | x + 0x30   |
| 8. daylight_savings_enabled (static)   | boolean      |                           |      |      | x + 0x38   |
| 9. clock_base (static)                 | enum         |                           |      |      | x + 0x40   |
| <b>Specific methods</b>                | <b>m/o</b>   |                           |      |      |            |
| 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   |

### 4.5.1.2 Attribute description

#### 4.5.1.2.1 logical\_name

Identifies the “Clock” object instance. See 6.2.5.

#### 4.5.1.2.2 time

Contains the meter’s local date and time, its deviation to UTC and the status.

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

When this attribute is set, all the fields in the octet-string representing *date-time* 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.

Only specified fields of the *date-time* are changed.

EXAMPLE For setting the *date* without changing the *time*, all time-relevant octets in the octet string representing *date-time* shall be set to “not specified”.

#### 4.5.1.2.3 time\_zone

The deviation of local, normal time to UTC in minutes. The value depends on the geographical location of the meter.

#### 4.5.1.2.4 status

The clock\_status maintained by the meter. The clock\_status element indicated in the *time* attribute is equal to the value of this attribute.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 173/668 |
|-----------------------|------------|-----------------------------|---------|

unsigned, formatted as specified in 4.1.6.1 for *clock\_status*

#### **4.5.1.2.5    *daylight\_savings\_begin***

Defines the local switch date and time when the local time starts to deviate from the normal time.

For generic definitions, wildcards are allowed.

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

#### **4.5.1.2.6    *daylight\_savings\_end***

Defines the local switch date and time when the local time ends to deviate from the local normal time.

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

#### **4.5.1.2.7    *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  $\pm 120$  min

#### **4.5.1.2.8    *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.

#### **4.5.1.2.9    *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.

#### **4.5.1.3    Method description**

##### **4.5.1.3.1    *adjust\_to\_quarter (data)***

Sets the meter's time to the nearest (+/-) quarter of an hour value (\*:00, \*:15, \*:30, \*:45).

data ::=    integer (0)

##### **4.5.1.3.2    *adjust\_to\_measuring\_period (data)***

Sets the meter's time to the nearest (+/-) starting point of a measuring period.

data ::=    integer (0)

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 174/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

**4.5.1.3.3    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)

**4.5.1.3.4    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)

**4.5.1.3.5    preset\_adjusting\_time (data)**

Presets the time to a new value (preset\_time) and defines a validity\_interval within which the new time can be activated.

```
data ::= structure
{
    preset_time:          octet-string,
    validity_interval_start: octet-string,
    validity_interval_end:  octet-string
}
```

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

**4.5.1.3.6    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)****4.5.2.1    Overview**

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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 175/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| 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            |                           |      |      |            |

### 4.5.2.2 Attribute description

#### 4.5.2.2.1 logical\_name

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

#### 4.5.2.2.2 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.

#### 4.5.2.3 Method description

##### 4.5.2.3.1 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)

##### 4.5.3.1 Overview

This IC, together with the IC “Special days”, allows modelling time- and date-driven activities within a device. Table 30 and Table 31 provide an overview and show the interactions between the two ICs.

**Table 30 – Schedule**

| Index | enable | action<br>(script) | Switch<br>_time | validity<br>_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  | 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  | x  |    |               |    |     |    |            | xx-04-01   | xx-09-30 |
| 202   | No     | xxxx:yy            | 11:00           |                     |               |    |    |    |    |    | x  |               |    |     |    |            | xx-04-01   | xx-09-30 |

**Table 31 – 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   |

#### 4.5.3.2 Attribute description

##### 4.5.3.2.1 logical\_name

Identifies the “Schedule” object instance. See 6.2.9

##### 4.5.3.2.2 entries

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

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

Where:

- script\_logical\_name: defines the logical name of the “Script table” object;
- script\_selector: defines the script\_identifier of the script to be executed;
- switch\_time accepts wildcards to define repetitive entries. The format of the octet-string follows the rules set in 4.1.6.1 for time;
- validity\_window defines a period in minutes, in which an entry shall be processed after power fail. (time between defined switch\_time and actual power\_up) 0xFFFF: the script is processed any time;
- exec\_weekdays defines the days of the week on which the entry is valid;
- exec\_specdays 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.

**4.5.3.3 Method description****4.5.3.3.1 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
}
```

Where:

- 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

**4.5.3.3.2 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
```

**4.5.3.3.3 delete (data)**

Deletes a range of entries in the table.

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

Where:

- 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:

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 179/668 |
|-----------------------|------------|-----------------------------|---------|

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.

#### 4.5.3.4 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).

#### 4.5.3.5 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.

#### 4.5.3.6 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.

#### 4.5.3.7 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.

#### 4.5.3.8 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.

#### 4.5.3.9 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)

#### 4.5.4.1 Overview

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*.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 180/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

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

#### 4.5.4.2 Attribute description

##### 4.5.4.2.1 logical\_name

Identifies the “Special days table” object instance. See 6.2.8.

##### 4.5.4.2.2 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 4.6.1 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”.

#### 4.5.4.3 Method description

##### 4.5.4.3.1 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.

##### 4.5.4.3.2 delete (data)

Deletes an entry in the table.

```
index      Index of the entry that shall be deleted.
```

```
data ::= long-unsigned
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 181/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.5.5 Activity calendar (class\_id = 20, version = 0)

##### 4.5.5.1 Overview

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

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

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

The “Activity calendar” object defines the activation of certain scripts, which can perform different activities inside the logical device. The interface to the IC “Script table” is the same as for the IC “Schedule” (see 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            |                            |      |      | x + 0x50   |

##### 4.5.5.2 Attribute description

NOTE: Attributes called ...\_active are currently active. Attributes called ...\_passive will be activated by the specific method activate\_passive\_calendar.

###### 4.5.5.2.1 logical\_name

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

**4.5.5.2.2 calendar\_name**

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

**4.5.5.2.3 season\_profile**

Contains a list of seasons defined by their starting date and a specific week\_profile to be executed. The list is sorted according to season\_start (in increasing order);

```
array      season

season ::= structure
{
    season_profile_name:      octet-string,
    season_start:            octet-string,
    week_name:               octet-string
}
```

Where:

- season\_profile\_name is a user defined name identifying the current season\_profile;
- season\_start defines the starting time of the season, formatted as specified in 4.6.1 for date-time. In season\_start, wildcards are allowed. If all fields are wildcards, the season will never start. When using wildcards, special care has to be taken to avoid conflicting parametrization, i.e. that the season\_start of two different seasons is the same;

NOTE The current season is terminated by the season\_start of the next season.

- week\_name defines the week\_profile active in this season.

**4.5.5.2.4 week\_profile\_table**

Contains an array of week\_profiles to be used in the different seasons. For each week\_profile, the day\_profile for every day of a week is identified.

```
array      week_profile

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
```

Where:

- week\_profile\_name is a user defined name identifying the current week\_profile;
- Monday defines the day\_profile valid on Monday;
- ...

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 183/668 |
|-----------------------|------------|-----------------------------|---------|

- Sunday defines the day\_profile valid on Sunday.

#### 4.5.5.2.5 **day\_profile\_table**

Contains an array of day\_profiles, identified by their day\_id. For each day\_profile, a list of scheduled actions is defined by a script to be executed and the corresponding activation time (start\_time). The list is sorted according to start\_time.

```

array      day_profile

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
}

```

Where:

- day\_id is a user defined identifier, identifying the current day\_profile;
- start\_time: defines the time when the script is to be executed (no wildcards); the format follows the rules set in 4.1.6.1 for *time*;
- script\_logical\_name: defines the *logical\_name* of the “Script table” object;
- script\_selector: defines the script\_identifier of the script to be executed.

#### 4.5.5.2.6 **activate\_passive\_calendar\_time**

Defines the time when the object itself calls the specific method *activate\_passive\_calendar*. A definition with "not specified" notation in all fields of the attribute will deactivate this automatism. Partial "not specified" notation in just some fields of date and time are not allowed.

octet-string, formatted as specified in 4.6.1 for *date-time*.

#### 4.5.5.3 **Method description**

##### 4.5.5.3.1 **activate\_passive\_calendar (data)**

This method copies all attributes called ...\_passive to the corresponding attributes called ...\_active.

data ::= integer (0)

#### 4.5.6 **Register monitor (class\_id = 21, version = 0)**

##### 4.5.6.1 **Overview**

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

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 184/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

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.

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

### 4.5.6.2 Attribute description

#### 4.5.6.2.1 logical\_name

Identifies the “Register monitor” object instance. See 6.2.13 and 6.3.10.

#### 4.5.6.2.2 thresholds

Provides the threshold values to which the attribute of the referenced register is compared.

array threshold

threshold: The threshold is of the same type as the monitored attribute of the referenced object.

#### 4.5.6.2.3 monitored\_value

Defines which attribute of an object is to be monitored. Only values with simple data types are allowed.

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

#### 4.5.6.2.4 actions

Defines the scripts to be executed when the monitored attribute of the referenced object crosses the corresponding threshold. The attribute actions has exactly the same number of elements as the attribute thresholds. The ordering of the action\_items correspond to the ordering of the thresholds (see 4.5.6.2.2).

```
array action_set
action_set ::= structure
{
    action_up:      action_item,
    action_down:    action_item
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 185/668 |
|-----------------------|------------|-----------------------------|---------|

```
}
```

Where:

- action\_up defines the action when the attribute value of the monitored register crosses the threshold in the upwards direction;
- action\_down defines the action when the attribute value of the monitored register crosses the threshold in the downwards direction.

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

#### 4.5.7 Single action schedule (class\_id = 22, version = 0)

##### 4.5.7.1 Overview

This IC allows modelling the execution of periodic actions within a meter. Such actions are not necessarily linked to tariffication (see “Activity calendar” (4.5.5) or “Schedule” (4.5.3)).

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

##### 4.5.7.2 Attribute description

###### 4.5.7.2.1 logical\_name

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

###### 4.5.7.2.2 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.

###### 4.5.7.2.3 type

enum:

- (1) size of execution\_time = 1; wildcard in date allowed,
- (2) size of execution\_time = n; all time values are the same, wildcards in date not allowed,

- (3) size of execution\_time = n; all time values are the same, wildcards in date are allowed,
- (4) size of execution\_time = n; time values may be different, wildcards in date not allowed,
- (5) size of execution\_time = n; time values may be different, wildcards in date are allowed

#### 4.5.7.2.4 execution\_time

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

```
array      execution_time_date

execution_time_date ::= structure
{
    time: octet-string,
    date: octet-string
}
```

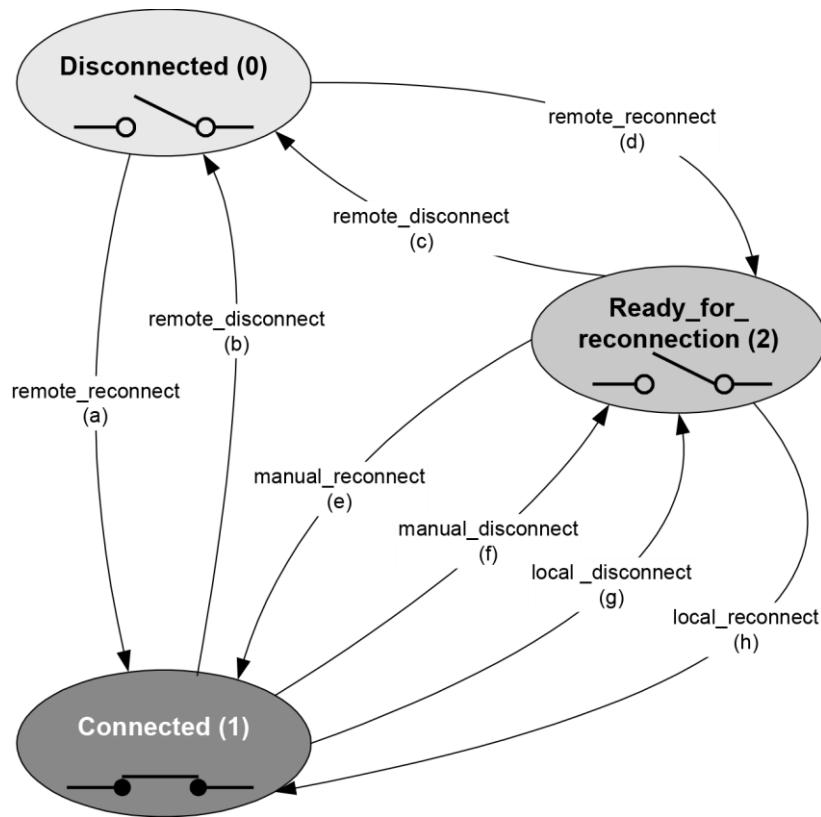
The two octet-strings contain *time* and *date*, in this order; *time* and *date* are formatted as specified in 4.1.6.1. Hundredths of seconds shall be zero.

### 4.5.8 Disconnect control (class\_id = 70, version = 0)

#### 4.5.8.1 Overview

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 21.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 187/668 |
|-----------------------|------------|-----------------------------|---------|

**Figure 21 – 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 32. 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.

## COSEM Interface Classes

**Table 32 – 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.<br>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.<br>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.<br>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).<br><br>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<br><br>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 (data) | m            |                            |      |      | x + 0x20   |
| 2.                 | remote_reconnect (data)  | m            |                            |      |      | x + 0x28   |

#### 4.5.8.2 Attribute description

##### 4.5.8.2.1 logical\_name

Identifies the “Disconnect control” object instance. See 6.2.22 and 6.2.46.

##### 4.5.8.2.2 output\_state

Shows the actual physical state of the device connection the supply.

boolean:

|        |              |
|--------|--------------|
| TRUE:  | Connected,   |
| FALSE: | 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.

##### 4.5.8.2.3 control\_state

Shows the internal state of the disconnect control object.

|       |     |                        |
|-------|-----|------------------------|
| enum: | (0) | Disconnected,          |
|       | (1) | Connected,             |
|       | (2) | Ready_for_reconnection |

##### 4.5.8.2.4 control\_mode

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

| control_mode | Disconnection |        |       |        | Reconnection |       |     |
|--------------|---------------|--------|-------|--------|--------------|-------|-----|
|              | Remote        | Manual | Local | Remote | Manual       | Local |     |
| enum:        | (b)           | (c)    | (f)   | (g)    | (a)          | (d)   | (e) |
| (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   |
| (6)          | x             | x      | –     | X      | –            | x     | x   |

NOTE 1 In Mode (0) the disconnect control object is always in 'connected' state.

NOTE 2 Local disconnection is always possible unless the corresponding trigger is inhibited.

#### 4.5.8.3 Method description

##### 4.5.8.3.1 remote\_disconnect (data)

Forces the “Disconnect control” object into ‘disconnected’ state if remote disconnection is enabled (control mode > 0).

data ::= integer (0)

#### 4.5.8.3.2 **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)

#### 4.5.9.1 Overview

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                        |      |      |            |

#### 4.5.9.2 Attribute description

##### 4.5.9.2.1 logical\_name

Identifies the “Limiter” object instance. See 6.2.15.

##### 4.5.9.2.2 monitored\_value

Defines an attribute of an object to be monitored. Only attributes with simple data types are allowed.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 191/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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

### **4.5.9.2.3 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

### **4.5.9.2.4 threshold\_normal**

Provides the threshold value to which the attribute monitored is compared when in normal operation.

threshold: see 4.5.9.2.3

### **4.5.9.2.5 threshold\_emergency**

Provides the threshold value to which the attribute monitored is compared when an emergency profile is active.

threshold: see 4.5.9.2.3

### **4.5.9.2.6 min\_over\_threshold\_duration**

Defines minimal over threshold duration in seconds required to execute the over threshold action.

### **4.5.9.2.7 min\_under\_threshold\_duration**

Defines minimal under threshold duration in seconds required to execute the under threshold action.

### **4.5.9.2.8 emergency\_profile**

An *emergency\_profile* is defined by three elements: *emergency\_profile\_id*, *emergency\_activation\_time*, *emergency\_duration*.

An emergency profile is activated if the *emergency\_profile\_id* element matches one of the elements on the *emergency\_profile\_group\_id\_list*, and time matches the *emergency\_activation\_time* and *emergency\_duration* element:

```
emergency_profile ::= structure
{
    emergency_profile_id:      long-unsigned,
    emergency_activation_time: octet-string,
    emergency_duration:        double-long-unsigned
}
```

Where:

- the `emergency_activation_time` element defines the date and time when the `emergency_profile` activated. The octet-string is encoded as specified in 4.1.6.1 for `date-time`,
- the `emergency_duration` element defines the duration in seconds, for which the `emergency_profile` is activated.

When an emergency profile is active, the `emergency_profile_active` attribute is set to TRUE.

#### 4.5.9.2.9 `emergency_profile_group_id_list`

Defines a list of group id-s of the emergency profile.

The emergency profile can be activated only if `emergency_profile_id` element of the `emergency_profile` attribute matches one of the elements on the `emergency_profile_group_id_list`:

```
array      emergency_profile_group_id
emergency_profile_group_id ::= long-unsigned
```

#### 4.5.9.2.10 `emergency_profile_active`

Indicates that the `emergency_profile` is active.

#### 4.5.9.2.11 `actions`

Defines the scripts to be executed when the monitored value crosses the threshold for minimal duration time.

```
action ::= structure
{
    action_over_threshold:      action_item,
    action_under_threshold:     action_item
}
```

Where:

- `action_over_threshold` 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;
- `action_under_threshold` 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.

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

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 193/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.5.10 Parameter monitor (class\_id = 65, version = 1)

Instances of the “Parameter monitor” IC are used to monitor a list of COSEM object attributes that hold parameters.

If the value of an attribute in the *parameter\_list* changes, 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 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 DLMS UA 1000-1 Ed 15 Part 1:2021, 6.5.

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

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

**NOTE 2** As the various parameters may be of different type and length, the entries in the profile column holding the parameters will be also of different type and length. This can be managed 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, such as the *time* attribute of the Clock object and any other relevant values.

#### Use of the Parameter monitor IC for detecting configuration parameter changes

Each client that has access to the server has to be aware of the current parametrisation in order to be able to correctly exchange data with servers. The parameters may be grouped and each group may have a name.

Although the parametrisation may be known initially, it may change during the lifetime of the meter, for example it may be changed by another client such as a field service device.

The current configuration may always be retrieved by reading the configuration parameters at the beginning of the exchange. This is not efficient and in the case of push operation it is not practical.

A solution is needed that allows clients to verify if the configuration of the server is as expected or to detect if any change has occurred.

To achieve this, version 1 of the Parameter monitor IC specifies new attributes to hold:

- the *parameter\_list\_name*;
- the *hash\_algorithm\_id*;
- the *parameter\_value\_digest*; and
- the *parameter values*.

When a client configures the server, it can retrieve these attributes and share them with other clients.

When a client starts a data exchange with the server, it can read the *parameter\_value\_digest* attribute and compare it to the value it has stored. If they match, the configuration is as

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 194/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

expected, if not, the client can read the parameter\_values attribute to identify which one(s) have changed.

The attributes can also be captured into Profile generic object buffer attributes to assure the validity of the data captured. The resulting overhead can be minimized by the use of null-data encoding.

| Parameter monitor          | 0...n        | class_id = 65, version = 1 |      |      |            |
|----------------------------|--------------|----------------------------|------|------|------------|
| Attributes                 | Data type    | Min.                       | Max. | Def. | Short name |
| 1. logical_name (static)   | octet-string |                            |      |      | x          |
| 2. changed_parameter       | structure    |                            |      |      | x + 0x08   |
| 3. capture_time            | date-time    |                            |      |      | x + 0x10   |
| 4. parameter_list          | array        |                            |      |      | x + 0x18   |
| 5. parameter_list_name     | octet-string |                            |      |      | x + 0x20   |
| 6. hash_algorithm_id       | enum         |                            |      |      | x + 0x28   |
| 7. parameter_value_digest  | octet-string |                            |      |      | x + 0x30   |
| 8. parameter_values        | structure    |                            |      |      | x + 0x38   |
| Specific methods           | m/o          |                            |      |      | x + 0x40   |
| 1. add_parameter (data)    | o            |                            |      |      | x + 0x48   |
| 2. delete_parameter (data) | o            |                            |      |      | x + 0x50   |

### 4.5.10.1 Attribute description

#### 4.5.10.1.1 logical\_name

Identifies the “Parameter monitor” object instance. See 6.2.14.

#### 4.5.10.1.2 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
}
```

#### 4.5.10.1.3 capture\_time

Provides data and time information showing when the value of the *changed\_parameter* attribute has been captured.

*date-time* is formatted as specified in 4.1.6.1.

#### 4.5.10.1.4 parameter\_list

Holds the list of parameters managed by a given instance of the “Parameter monitor” IC.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 195/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

```
parameter_list ::= array      parameter_list_element

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

NOTE 1 The list of parameters monitored may be changed by using the *add\_parameter* or *delete\_parameter* methods or writing this attribute.

NOTE 2 The list of parameters monitored may be changed by using the *add\_parameter* or *delete\_parameter* methods or writing this attribute.

### 4.5.10.1.5 parameter\_list\_name

Holds the name given to the list of parameter as defined by the *parameter\_list* attribute.

### 4.5.10.1.6 hash\_algorithm\_id

enum

- (0) SHA-256,
- (1) SHA-384,
- (2) Last 16 bytes of SHA-256,
- (3) Last 8 bytes of SHA-256,
- (4) Last 4 bytes of SHA-256

### 4.5.10.1.7 parameter\_value\_digest

Result of the digest calculation according to the algorithm specified by the *hash\_algorithm\_id* attribute. The digest is calculated on all the parameter list values as defined by attribute *parameter\_list* in the order they are placed in the array. To calculate the digest, the whole set of *parameter\_list* attribute values is first transformed into an octet-string. Rules for this transformation are defined DLMS UA 1000-2 Ed.11:2021, 9.2.3.4.3. Data conversion, for all the data types which are not an octet-string.

### 4.5.10.1.8 parameter\_values

Holds a copy of the values of the attributes referenced in *parameter\_list* attribute.

It is a structure that holds the A-XDR encoded value of each attribute in the order of the *parameter\_list* array.

## 4.5.10.2 Method description

### 4.5.10.2.1 add\_parameter (data)

Adds one parameter to the *parameter\_list*.

```
data ::= parameter_list_element
```

NOTE A parameter can be logged as soon as it is added to the list. Adding an element to the list does not affect the *buffer* of the “Profile generic” object capturing the *changed\_parameter* attribute.

### 4.5.10.2.2 delete\_parameter (data)

Deletes one parameter from the *parameter\_list*.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 196/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

```
data ::= parameter_list_element
```

**NOTE** 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 *buffer* of the “Profile generic” object capturing the *changed\_parameter* attribute.

#### 4.5.11 Sensor manager (class\_id = 67, version = 0)

##### 4.5.11.1 General

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

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

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

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

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

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

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

##### 4.5.11.2 Overview

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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 197/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| <b>Sensor manager</b>          |          | <b>0...n</b>               | <b>class_id = 67, version = 0</b> |             |             |                   |
|--------------------------------|----------|----------------------------|-----------------------------------|-------------|-------------|-------------------|
| <b>Attribute (s)</b>           |          | <b>Data type</b>           | <b>Min.</b>                       | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 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          |
| <b>Specific methods</b>        |          | <b>m/o</b>                 |                                   |             |             |                   |
| 1. reset (data)                |          | o                          |                                   |             |             |                   |

### 4.5.11.3 Attribute description

#### 4.5.11.3.1 logical\_name

Identifies the “Sensor manager” object instance.

NOTE Sensor manager objects are used in relation to non-electricity metering. Therefore no OBIS codes are defined in this document.

#### 4.5.11.3.2 serial\_number

Identifies the sensor (->site information).

NOTE For simple sensors, the serial number may be held by “Data” objects with appropriate OBIS codes if this is the only information required.

#### 4.5.11.3.3 metrological\_identification

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

#### 4.5.11.3.4 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.

#### **4.5.11.3.5 adjustment\_method**

Describes the sensor adjustment method, e.g. by 3-Measuring-point-equation.

#### **4.5.11.3.6 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)

#### **4.5.11.3.7 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 4.3.2.2.2.

#### **4.5.11.3.8 scaler\_unit**

The scaler and the unit of the *raw\_value*.

For the definition of *scaler\_unit*, see the “Register” IC in 4.3.2.2.3.

#### **4.5.11.3.9 status**

Status of last *raw\_value* captured (for the definition of status, see the “Extended register” IC, 4.3.3.2.4). The status keeps information such as:

- sensor failure,
- sensor activated / inactivated.

The meaning of the elements of the *status* shall be provided for each “Sensor manager” object instance.

#### **4.5.11.3.10 capture\_time**

Provides a “Sensor manager” object specific date and time information showing when the value of the attribute *raw\_value* has been captured.

*date\_time* is formatted as specified in 4.1.6.1.

For the possible data type choices, see the “Extended register” interface class in 4.3.3.2.5.

#### **4.5.11.3.11 raw\_value\_thresholds**

Provides the threshold values to which the *raw\_value* attribute is compared.

array threshold

threshold: The threshold is of the same type as the raw-value (4.5.11.3.7).

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 199/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.5.11.3.12 raw\_value\_actions

Defines the scripts to be executed when the *raw\_value* crosses the corresponding threshold. The attribute *raw\_value\_actions* has exactly the same number of elements as the attribute *raw\_value\_thresholds*. The ordering of the *action\_items* corresponds to the ordering of the thresholds. For the specification of actions, see the *Register monitor* IC in 4.5.6.2.4.

#### 4.5.11.3.13 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.

#### 4.5.11.3.14 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 4.5.11.3.13).

#### 4.5.11.3.15 processed\_value\_actions

Defines the scripts to be executed when the processed value crosses the corresponding threshold.

The attribute *processed\_value\_actions* has exactly the same number of elements as the attribute *processed\_value\_thresholds*. The ordering of the *action\_items* corresponds to the ordering of the thresholds.

For the specification of actions, see the *Register monitor* IC in 4.5.6.2.4.

### 4.5.11.4 Method description

#### 4.5.11.4.1 reset (data)

This method resets the *raw\_value* to the default value. The default value is an instance specific constant.

The attribute *capture\_time* is set to the time of the reset execution.

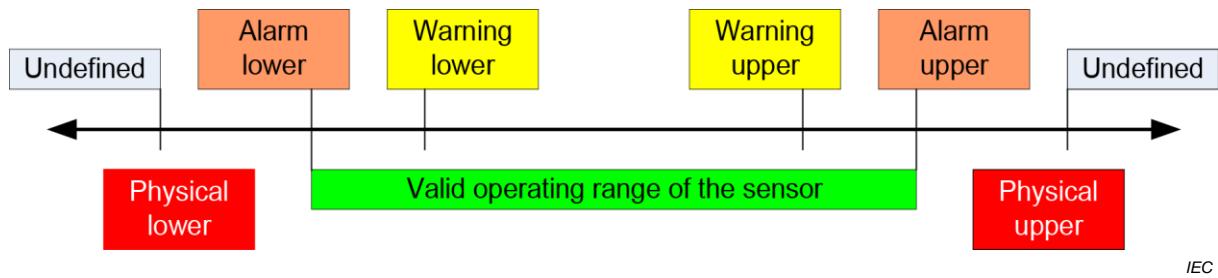
```
data ::= integer (0)
```

### 4.5.11.5 Example for absolute pressure sensor

Figure 22 illustrates the definition of relevant upper and lower thresholds.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 200/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

### COSEM Interface Classes



**Figure 22 – Definition of upper and lower thresholds**

Table 33 and Table 34 show examples of the various thresholds and the actions performed when the thresholds are crossed.

**Table 33 – 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 34 – 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 (class\_id = 68, version = 0)

##### 4.5.12.1 Overview

Instances of the “Arbitrator” IC allow determining, based on pre-configured rules comprising permissions and weightings, which action is carried out when multiple actors may request potentially conflicting actions to control the same resource. Generally, there is one “Arbitrator” object instantiated for each resource for which competing action requests need to be handled.

The “Arbitrator” IC allows:

- configuring the possible, potentially conflicting actions that can be requested;
- configuring the permissions for each actor to request the possible actions;
- configuring the weighting for each actor for each possible request.

NOTE 1 Examples for a resource are the supply control switch or a gas valve of the meter. Examples for possible actions are disconnect supply, enable reconnection, reconnect supply, prevent disconnection, prevent reconnection.

The actions that can be requested are held in the *actions* attribute as an array of script identifiers. The scripts – held by separate “Script table” objects – allow performing a wide range of functions. There may be actions designed to inhibit the execution of other actions: these are modelled as null-scripts, i.e. pointing to a script\_identifier (0) of a “Script table” object.

NOTE 2 The “Arbitrator” objects do not contain the names of the actions, but their purpose and effect can be deduced by looking at the relevant “Script table” objects.

The permissions are held in the *permissions\_table* attribute as an array of bit-strings: each element in the array represents one actor and each bit in the bit-string represents one action from the *actions* array.

The weightings are held in the *weightings\_table* attribute as a two-dimensional array: each line represents one actor and there is one weight allocated to each possible action for that actor. For actions designed to inhibit other actions a very high weight may be allocated compared to the weight of the action that is to be inhibited.

Actions are requested by invoking the *request\_action* method. The method invocation parameters contain:

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 202/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

- 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 ascertain 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.

| Arbitrator                    |          | 0...n        | class_id = 68, version = 0 |      |                  |            |
|-------------------------------|----------|--------------|----------------------------|------|------------------|------------|
| Attribute description         |          | Data type    | Min.                       | Max. | Def.             | Short name |
| 1. logical_name               | (static) | octet-string |                            |      |                  | x          |
| 2. actions                    | (static) | array        |                            |      | all empty        | x + 0x08   |
| 3. permissions_table          | (static) | array        |                            |      | all bits cleared | x + 0x10   |
| 4. weightings_table           | (static) | array        |                            |      | all zero         | x + 0x18   |
| 5. most_recent_requests_table | (dyn.)   | array        |                            |      | all bits cleared | x + 0x20   |
| 6. last_outcome               | (dyn.)   | unsigned     | 0                          | n    | 0                | x + 0x28   |
| Specific methods              |          | m/o          |                            |      |                  |            |
| 1. request_action (data)      |          | m            |                            |      |                  |            |
| 2. reset (data)               |          | o            |                            |      |                  |            |

#### 4.5.12.2 Attribute description

##### 4.5.12.2.1 logical\_name

Identifies the “Arbitrator” object instance. See 6.2.47.

##### 4.5.12.2.2 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.

##### 4.5.12.2.3 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 *actions* array. 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 in the *actions* array.

The bits shall be set for actions allowed and cleared for actions not allowed.

NOTE These *actor\_permissions* model business rules rather than acting as a form of access control.

##### 4.5.12.2.4 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 *weightings\_table* array shall be the same as in the *permissions\_table* array.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 204/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

The number of elements in the `actor_weighting_list` array shall be the same as in the `actions` array. The first element shows the weight for the first action and the last element shows the weight of the last action.

**NOTE** 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).

#### 4.5.12.2.5 `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 `most_recent_requests_table` array shall be the same as in the `permissions_table` array.

The length of the `most_recent_request` 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 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.

#### 4.5.12.2.6 `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 `request_action_list` bit-string: the number 1 corresponds to the leading bit and consequently to the first element in the `actions` array.

### 4.5.12.3 Method description

#### 4.5.12.3.1 `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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 205/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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.

### 4.5.12.3.2 reset (data)

Clears the configurable fields of the object instance, that is:

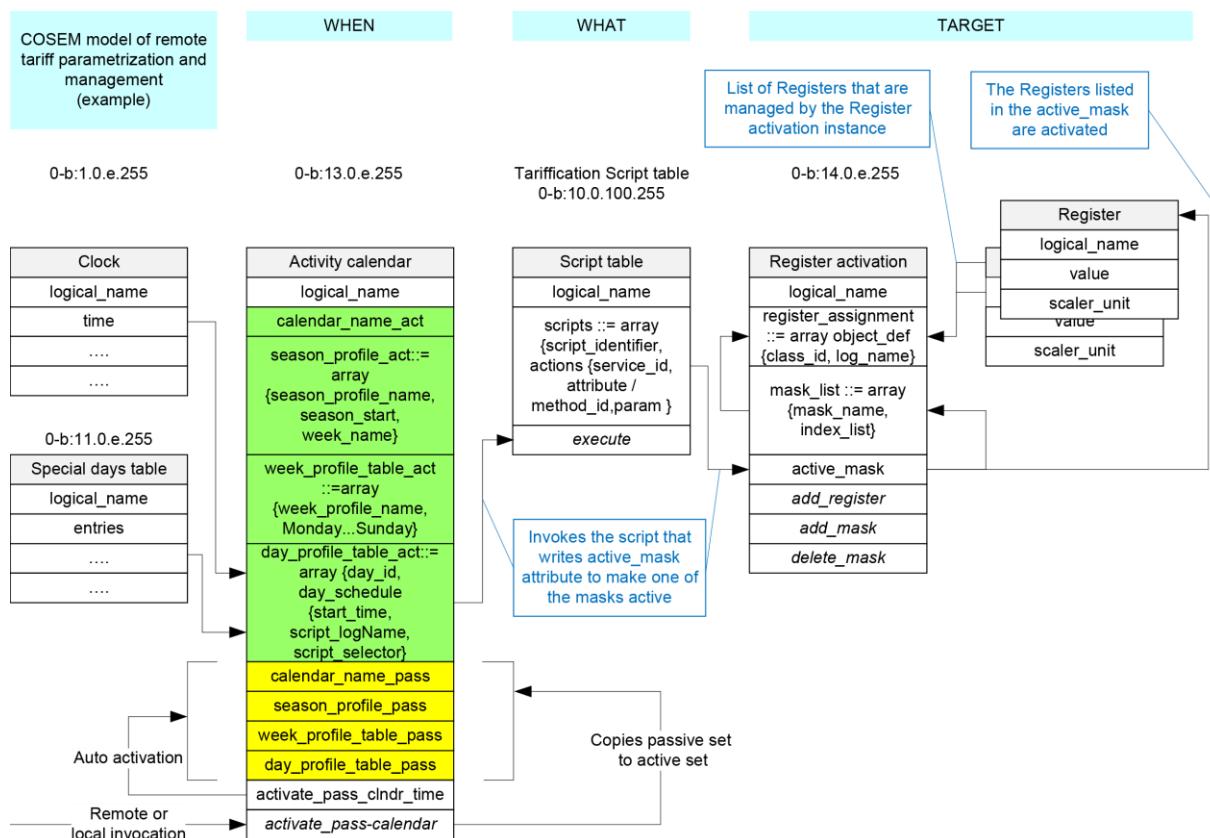
- clears all bits in the *permissions\_table* attribute;
- sets all values in the *weightings\_table* attribute to zero;
- clears all bits in the *most\_recent\_requests\_table* attribute;
- sets the *last\_outcome* attribute to zero.

**NOTE** 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

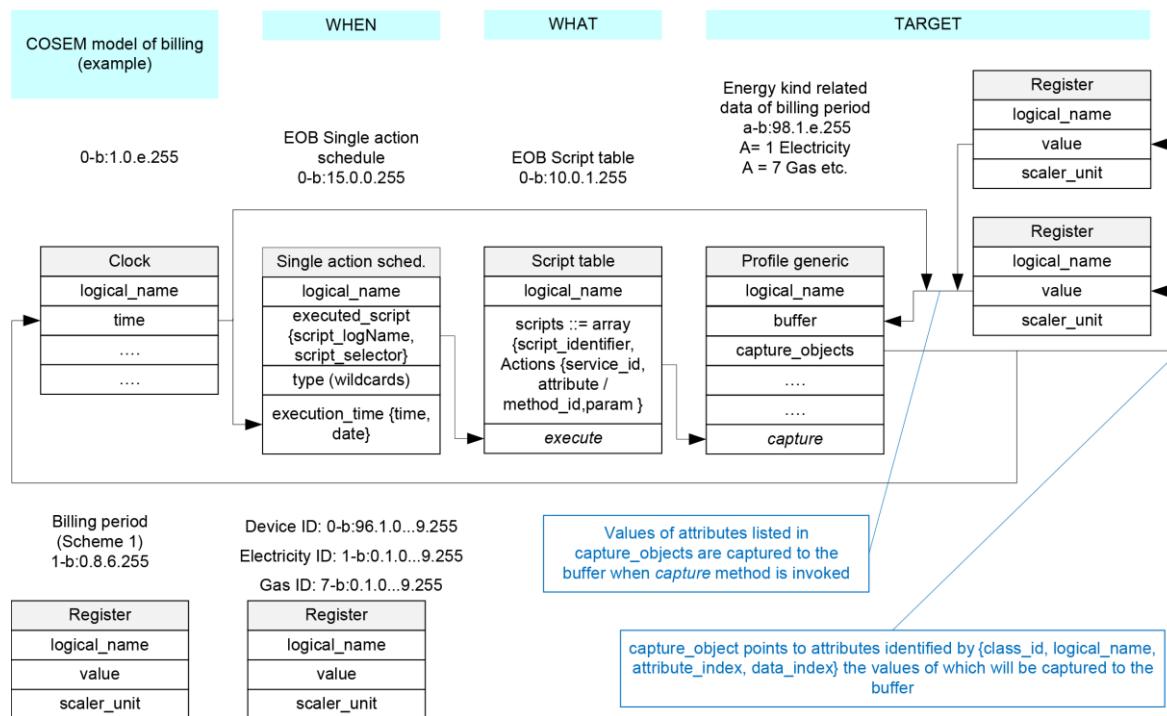
Figure 23 shows an example of modelling tariff parametrization and management using COSEM objects.



**Figure 23 – COSEM tariffication model (example)**

## COSEM Interface Classes

Figure 24 shows an example of modelling parametrization and management of billing using COSEM objects.



**Figure 24 – 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.

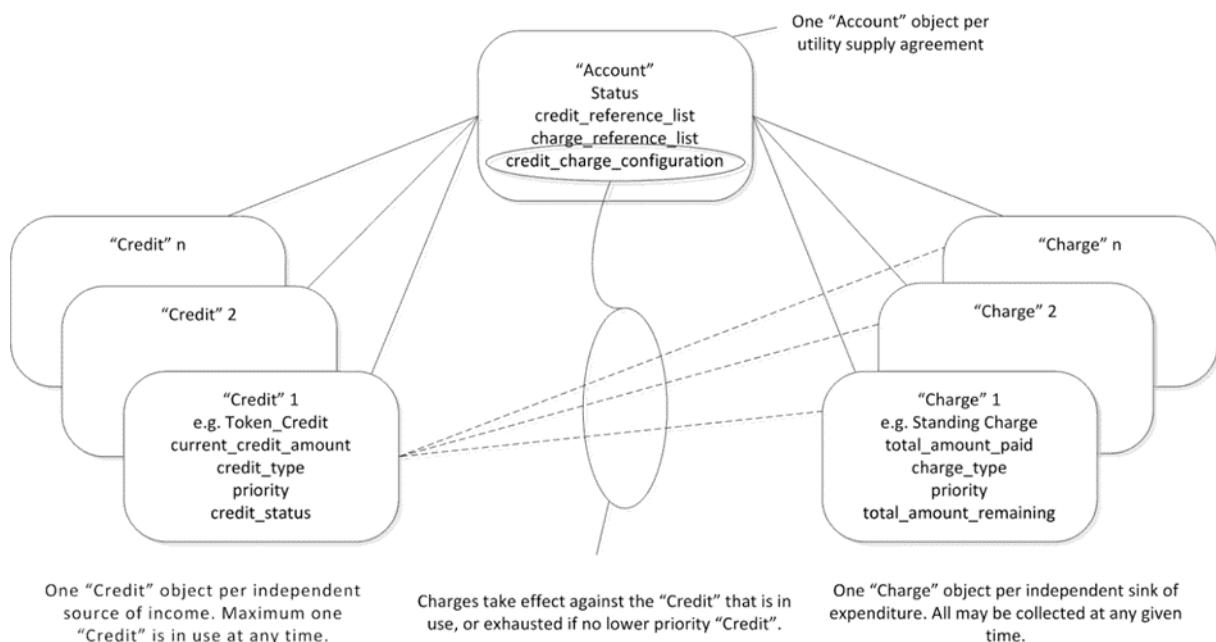
- **token\_credit:** Credit that is transferred to a meter operating in prepayment mode, normally in the form of Credit Tokens;
  - NOTE 1 In a meter operating in credit mode or managed payment mode, a "Credit" object configured with type token\_credit is used for recording the amount of credit used 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.

## COSEM Interface Classes

- **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 25 gives a general view of the account model.



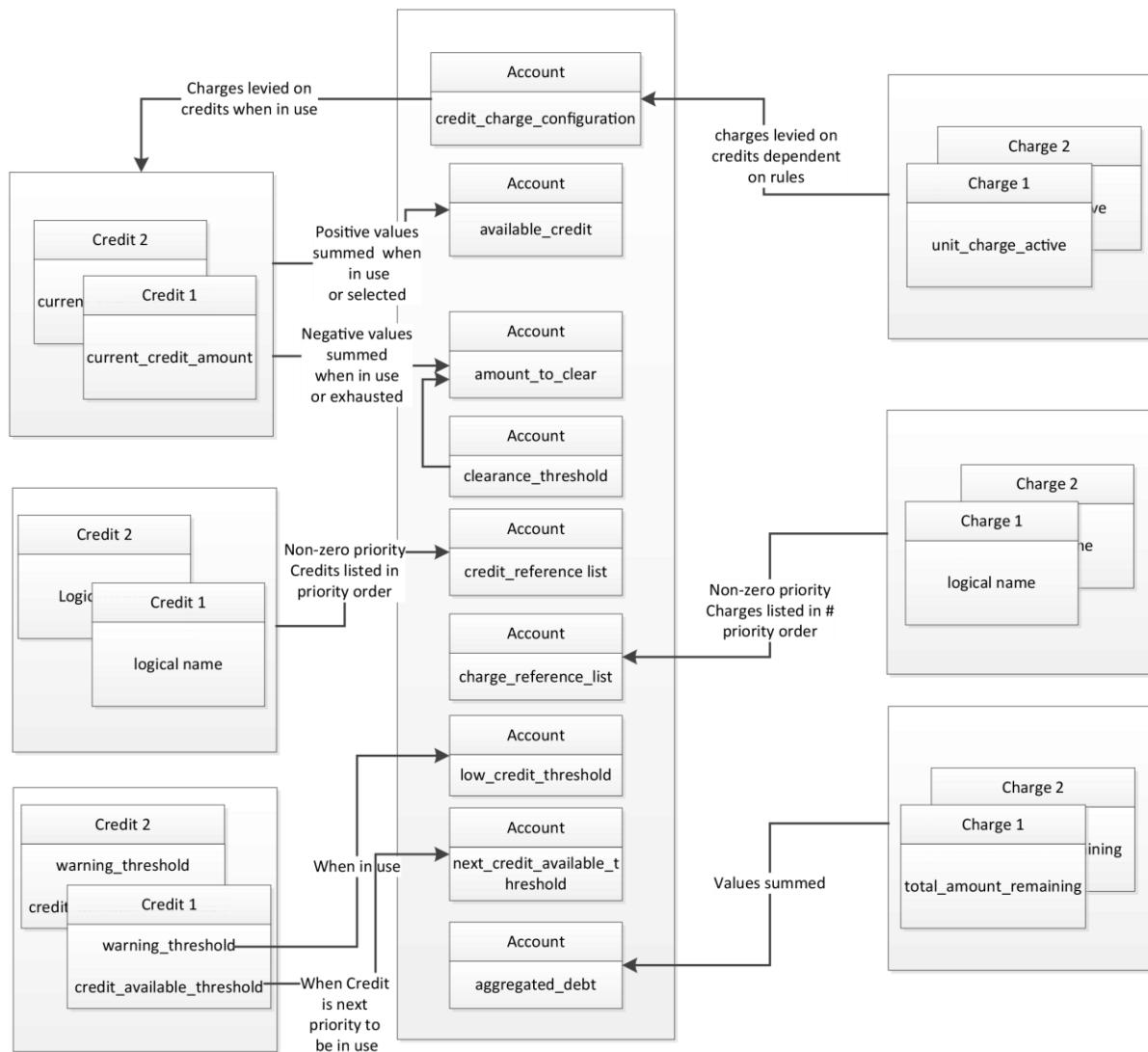
**Figure 25 – Outline Account model**

Figure 26 shows instances of “Account”, “Credit” and “Charge” interface classes with some of their attributes and the relationships between those attributes. In this example:

- a) There is one “Account”, two “Credit” and two “Charge” objects configured;
- b) “Credit” 1 is of type *token\_credit* and the *low\_credit\_threshold* and *limit* attributes are configured to be 0;
- c) interaction between multiple classes is covered in this diagram. Detailed configuration of individual “Credit” and “Charge” objects is not shown.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 209/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes



**Figure 26 – Diagram of attribute relationships**

### 4.6.2 Account (class\_id = 111, version = 0)

#### 4.6.2.1 Overview

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** 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.

## COSEM Interface Classes

| 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   |
| <b>Specific methods</b>             |          | m/o           |                             |      |                |            |
| 1. activate_account (data)          |          | o             |                             |      |                | x + 0x98   |
| 2. close_account (data)             |          | o             |                             |      |                | x + 0xA0   |
| 3. reset_account (data)             |          | o             |                             |      |                | x + 0xA8   |

### 4.6.2.2 Attribute description

#### 4.6.2.2.1 logical\_name

Identifies the “Account” object instance. See 6.2.17.

#### 4.6.2.2.2 account\_mode\_and\_status

Defines the payment\_mode, enumerated below, and the status of the “Account”, also enumerated.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 211/668 |
|-----------------------|------------|-----------------------------|---------|

```

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
}

```

The payment\_mode is an indication of a notional prepayment or credit/managed payment mode.

NOTE Payment\_mode does not force some other actions in the meter, or cause a change of behaviour.

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 *account\_activation\_time* or invocation of the *activate\_account* method.

#### 4.6.2.2.3 current\_credit\_in\_use

This attribute is an index into the *credit\_reference\_list* indicating which “Credit” object is *In use*.

#### 4.6.2.2.4 current\_credit\_status

This attribute provides the status of the current “Credit” object *In use* and some information about the next priority “Credit” object.

- Bit 0 = in\_credit, set when the *available\_credit* is above zero,
- Bit 1 = low\_credit, set when the “Account” object’s *available\_credit* level has passed below the “Account” object *low\_credit\_threshold*,
 

NOTE 1 The *low\_credit\_threshold* attribute of the associated “Credit” object *In use* is echoed in that attribute.
- Bit 2 = next\_credit\_enabled, set when the “Account” object *credit\_reference\_list* contains at least one other “Credit” object with non-zero priority, regardless of credit level,
- Bit 3 = next\_credit\_selectable, set when the next item in the “Account” object *credit\_reference\_list* contains a lower priority “Credit” object with non-zero priority with a *credit\_configuration* bit 1 (Requires confirmation) set such that it requires confirmation (for example “Emergency Credit” that is selectable but has not yet been selected),
 

NOTE 2 The next “Credit” object becomes selectable when the immediately higher priority “Credit” object has reached the “Account” object’s *next\_credit\_available\_threshold*.
- Bit 4 = next\_credit\_selected, set when the “Account” object *credit\_reference\_list* contains a “Credit” object of lower priority than the current “Credit” object *In use*, and that lower priority credit object is in the *Selected/Invoked* state,
 

NOTE 3 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*

## COSEM Interface Classes

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

When the next priority “Credit” object becomes the current “Credit” object then all bits have to update.

NOTE 4 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*.

### 4.6.2.2.5 available\_credit

The *available\_credit* attribute is the sum of the positive *current\_credit\_amount* values in the instances of the “Credit” class that:

- are listed in the “Account” object *credit\_reference\_list*;
- and have a “Credit” object *credit\_status* (2) *Selected/Invoked* or (3) *In use*.

This attribute holds the aggregate amount of credit available associated with the “Account” object. This value is positive or zero.

NOTE Negative values of “Credit” object *current\_credit\_amount* attributes are summed in *amount\_to\_clear*. See also the vertical axis in Figure 28.

double-long, scaled according to the *currency* attribute.

### 4.6.2.2.6 amount\_to\_clear

This value is the sum of:

- all negative values of *current\_credit\_amount* in “Credit” objects that:
  - are listed in the “Account” object *credit\_reference\_list* and
  - have a “Credit” object *credit\_status* (4) *Exhausted*,
  - have the *credit\_configuration* 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 *credit\_configuration* bit 2 (Requires the credit amount to be paid back) is set; and
- the negative (Value \* -1) of the value of the “Account” object *clearance\_threshold* attribute;

See also Figure 28.

NOTE 1 “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.

NOTE 2 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 213/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

This attribute is significant when the meter has accumulated a temporary debt, for instance, while an emergency credit is *In use*, and/or during a friendly credit period.

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) *In use*, (2) *Selected/Invoked* or (4) *Exhausted* state.

NOTE 3 For an explanation of the process of distributing top ups between “Credit” objects please refer to attribute 12 *token\_gateway\_configuration*.

NOTE 4 Where a meter is being used in prepayment mode, *amount\_to\_clear* is usually the amount needed to reverse a disconnection. Amount to clear is shown on the vertical axis in Figure 28 as a negative value.

double-long, scaled according to the *currency* attribute (4.6.2.2.15).

### 4.6.2.2.7 clearance\_threshold

This attribute is used in conjunction with the *amount\_to\_clear*, and is included in the description of that attribute.

This attribute becomes relevant when a meter has consumed all credit sources and has “Credit” objects with *current\_credit\_amount* attributes that have values less than zero.

This represents the value of credit that the *available\_credit* attribute must reach in order to make repayable “Credits” enabled again.

To achieve this, it is clear that enough credit has to be added to the meter to ensure that the *current\_credit\_available* attribute on all listed “Credit” objects is zero or greater.

double-long, scaled according to the *currency* attribute.

### 4.6.2.2.8 aggregated\_debt

This attribute is a simple sum of *total\_amount\_remaining* of all the “Charge” objects which are listed in the “Account” object *charge\_reference\_list*, where bit 1 (Continuous collection) of the *charge\_configuration* is cleared.

NOTE This value is not interchangeable with *amount\_to\_clear*; it is provided to assist external accounting.

double-long, scaled according to the *currency* attribute.

### 4.6.2.2.9 credit\_reference\_list

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).

Priority 0 credits shall NOT appear in this list, as by definition they are not enabled.

```
credit_reference_list ::= array      credit_reference
credit_reference ::= octet-string
```

#### 4.6.2.2.10 charge\_reference\_list

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.

```
charge_reference_list ::= array charge_reference
```

```
charge_reference ::= octet-string
```

#### 4.6.2.2.11 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 1** 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 215/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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 2 The meter application knows internally whether it is in a supply disconnected, load limiting period, or friendly credit period, and takes appropriate action based on the value of this attribute.

NOTE 3 It is implicit that charges are also collected at times when these specific conditions are not present.

### 4.6.2.2.12 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 1 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 2 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 3 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 is the subject of project specific companion specifications. However in the normal case an “Account” object has one “Token gateway” object and all tokens received 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” 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 4.6.2.2.6, Note 3.

#### **4.6.2.2.13 account\_activation\_time**

Defines the time when the object itself invokes the specific method *activate\_account*. 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 *date\_time*.

#### **4.6.2.2.14 account\_closure\_time**

Defines the time when the object itself invokes the specific method *close\_account*. 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 *date\_time*.

#### **4.6.2.2.15 currency**

Defines the “currency” unit used by all functions of an “Account” object. The *charge\_per\_unit* in the *unit\_charge* 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,

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 217/668 |
|-----------------------|------------|-----------------------------|---------|

sec = seconds,  
 kWh = kilowatt hours,  
 Whr = Watt hours,  
 mt3 = m<sup>3</sup>,  
 ft3 = ft<sup>3</sup>,  
 Jle = Joule

OR

Defined by project specific companion specification such that all characters used are lower case.

The *currency\_scale* 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 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\_unit: enum:  
 (0) time,  
 (1) consumption,  
 (2) monetary

The *currency\_unit* is an enumerated value that indicates the generic type of currency:

- time (in minutes, seconds, hours etc.);
- consumption (in kWhrs, Whrs, m<sup>3</sup>, Whrs etc.); or
- monetary (GBP, USD, EUR, etc.).

#### **4.6.2.2.16 low\_credit\_threshold**

This attribute reflects the *warning\_threshold* attribute of the “Credit” object currently *In use*. This threshold can be used to generate a warning to the consumer, for example. It has no other function.

double-long, scaled according to the *currency* attribute.

NOTE The value of this attribute will change depending on which “Credit” object is *In use* at the time of the query.

#### **4.6.2.2.17 next\_credit\_available\_threshold**

This threshold reflects the *credit\_available\_threshold* 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)).

When the *available\_credit* attribute of the “Account” object becomes less than or equal to this value, the *credit\_status* attribute of the next priority “Credit” object changes to:

- (1) *Selectable* in the case when the *credit\_configuration* attribute of the “Credit” object concerned has the bit 1 (Requires confirmation) set;
- (2) *Selected/Invoked* in the case where the bit 1 (Requires confirmation) is cleared and the *next\_credit\_available\_threshold* is greater than zero;
- (3) *In use* in the case where the bit 1 (Requires confirmation) is cleared and the *next\_credit\_available\_threshold* is equal to zero.

## COSEM Interface Classes

NOTE 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.

The *next\_credit\_available\_threshold* will change depending on which “Credit” is *In use*.

double-long, scaled according to the *currency* attribute.

### 4.6.2.2.18 max\_provision

This attribute affects the operation of “Charge” objects that are configured with *charge\_type* equal to (2) *payment\_event\_based\_collection*.

This attribute holds the limit amount scaled as defined in the *currency* attribute across all “Charge” objects with *charge\_type* (2) *payment\_event\_based\_collection*. The limit is related to the time period defined in *max\_provision\_period*.

NOTE 1 The combination of this attribute and *max\_provision\_period* is intended to provide a limit to the total value of collections from top-ups within, for example, one week.

NOTE 2 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).

long-unsigned, scaled according to the *currency* attribute.

### 4.6.2.2.19 max\_provision\_period

This attribute holds the time period applicable for the *max\_provision* attribute. This is specified in seconds.

double-long

## 4.6.2.3 Method description

### 4.6.2.3.1 activate\_account (data)

This method allows the activation of the “Account”. The *account\_status* element of *account\_mode\_and\_status* will be set to (2) *Account active*.

NOTE The *account\_activation\_time* will be set to the time of invoking this method.

data ::= integer (0)

### 4.6.2.3.2 close\_account (data)

This method forces the closure of the Account.

NOTE 1 This may be done pursuant to a change of supplier or change of tenancy or any other reason.

The *account\_status* element of *account\_mode\_and\_status* will be set to (3) *Account closed*. The *account\_closure\_time* is set to the time of invoking this method.

NOTE 2 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 *credit\_reference\_list* and *charge\_reference\_list* 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)

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 219/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.6.2.3.3 reset\_account (data)

If the *account\_status* of the attribute *account\_mode\_and\_status* is not equal to (2) *Active account*, then this method sets the attributes of the “Account” to default values except where an attribute does not have a default value, in which case the behaviour shall be project specific.

If the *account\_status* element of the attribute *account\_mode\_and\_status* is equal to 1 (New, inactive account), then this method shall have no effect.

```
data ::= integer (0)
```

### 4.6.3 Credit (class\_id = 112, version = 0)

#### 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.

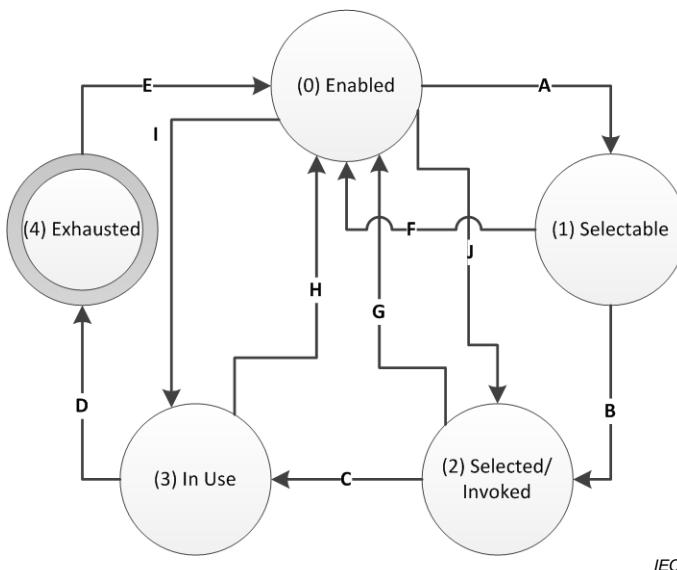
#### 4.6.3.2 Credit states

The credit states only have meaning when *priority* is non-zero. They are shown in Table 35 and Figure 27. The state transitions are shown in Table 36.

**Table 35 – Credit states**

| Priority | Credit state         | Meaning  |
|----------|----------------------|--|
| 0        | Any                  | The instance of the “Credit” object is inactive.<br><br>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.  |

## COSEM Interface Classes



**Figure 27 – Credit States when priority >0**

**Table 36 – 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>  |

## COSEM Interface Classes

| Credit state | From                   | To                    | Conditions   |
|--------------|------------------------|-----------------------|--|
| F            | (1) Selectable         | (0) Enabled           | <p>This transition occurs when the "Account" object <i>available_credit</i> becomes larger than the "Account" object <i>next_credit_available_threshold</i>.</p> <p>NOTE 7 This only applies when the <i>credit_configuration</i> attribute of the "Credit" object concerned has the bit 1 (Requires confirmation) set.</p> <p>NOTE 8 This is usually because the <i>current_credit_amount</i> attribute of a "Credit" object that is <i>In use</i> has been increased.</p>                                  |
| G            | (2) Selected / Invoked | (0) Enabled           | <p>This transition occurs when the "Account" object <i>available_credit</i> becomes larger than the "Account" object <i>next_credit_available_threshold</i>.</p> <p>NOTE 9 This is usually because the <i>current_credit_amount</i> attribute of a "Credit" object that is <i>In use</i> has been increased.</p>   |
| H            | (3) In use             | (0) Enabled           | <p>This transition occurs when the <i>credit_status</i> of a higher priority "Credit" object becomes <i>In use</i>.</p> <p>NOTE 10 This is usually because the higher priority "Credit" object <i>current_credit_amount</i> has been increased. At this point the <i>next_credit_available_threshold</i> of the "Account" object will also change.</p>   |
| I            | (0) Enabled            | (3) In use            | <p>This transition occurs when the immediately higher priority "Credit" object becomes exhausted.</p> <p>NOTE 11 This only applies when the <i>credit_configuration</i> attribute of the "Credit" object concerned has the bit 1 (Requires confirmation) cleared and the <i>credit_available_threshold</i> of the "Credit" object is set to zero and its <i>current_credit_amount</i> is greater than the <i>limit</i> (a credit cannot be <i>In use</i> until the higher priority Credit is Exhausted).</p> |
| J            | (1) Enabled            | (2) Selected/ Invoked | <p>The transition occurs when the <i>available_credit</i> in the "Account" object becomes less than the <i>next_credit_available_threshold</i> on the "Account" object.</p> <p>NOTE 12 This only applies when the <i>credit_configuration</i> attribute of the "Credit" object concerned has the bit 1 (Requires confirmation before it can be selected or <i>In use</i>) cleared, and the <i>credit_available_threshold</i> of this "Credit" object is greater than zero.</p>                               |

### 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 28 below.

In Figure 28 an example is given for possible cases involving a payment metering system with two "Credit" objects: Token Credit and Emergency Credit.

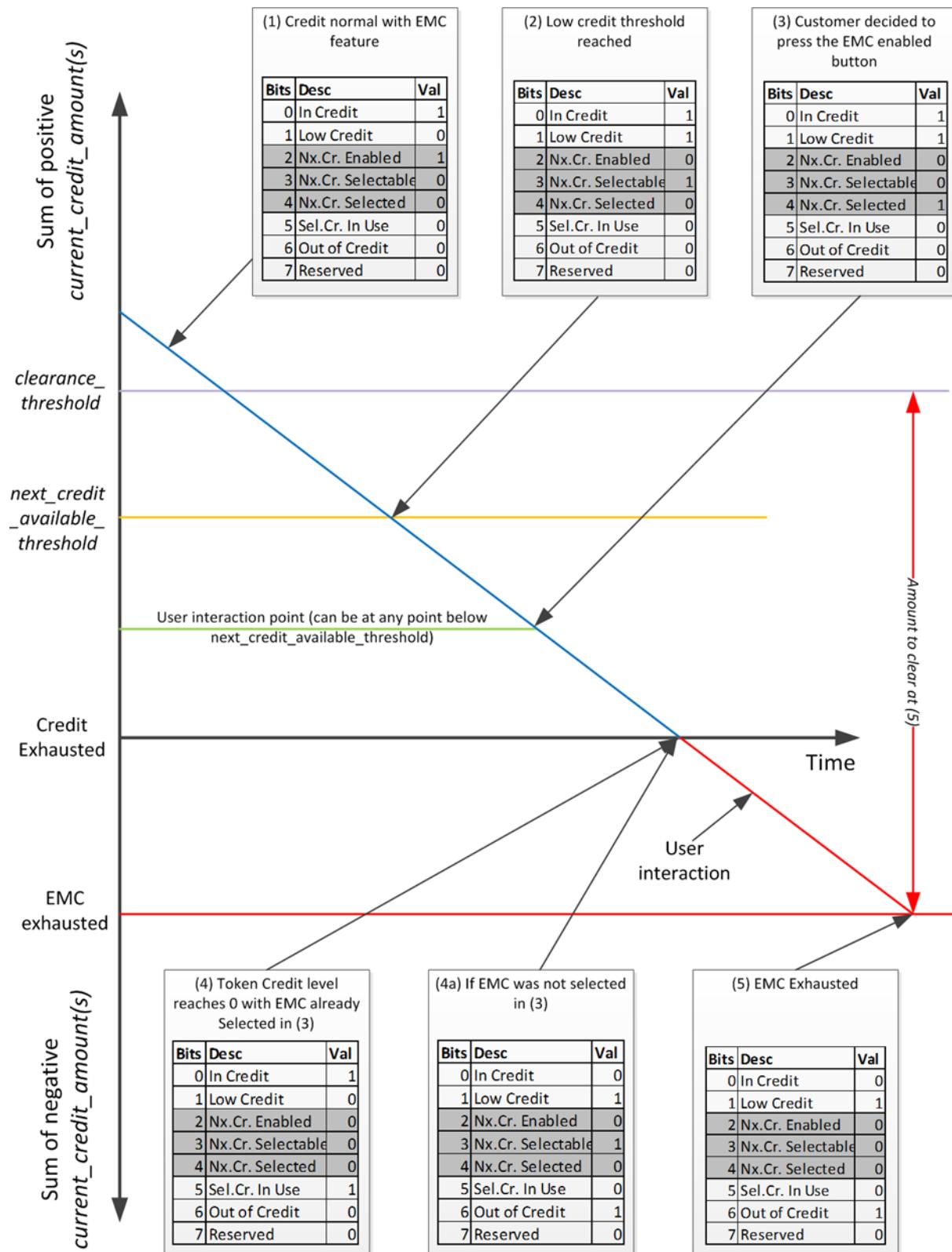
NOTE This is one possible implementation. The type and number of credits used in an actual project are subject to project specific companion specifications.

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

## COSEM Interface Classes

(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 28 – Operation of current\_credit\_status flags**

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 223/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

### 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   |
| <b>Specific methods</b>  |              | <b>m/o</b>                  |      |      |            |
| 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   |

### 4.6.3.5 Attribute description

#### 4.6.3.5.1 logical\_name

Identifies the “Credit” object instance. See 6.2.17.

#### 4.6.3.5.2 current\_credit\_amount

Provides the credit value of this particular “Credit” object. (See Figure 29).

**NOTE** 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 *available\_credit* attribute in the “Account” object from which the “Credit” object is referenced.

double-long, scaled according to the *currency* attribute of the “Account” object.

#### 4.6.3.5.3 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

#### 4.6.3.5.4 priority

Describes the activation priority of this “Credit” object.

Value 1 is the highest priority and value 255 the lowest.

Every “Credit” object shall have a different priority, except in the case of priority 0.

**NOTE** Behaviour will be undefined if there are multiple “Credit” objects configured with the same non-zero priority.

A value of 0 indicates that the “Credit” object is never activated, although it can still be configured and its *current\_credit\_amount* can be adjusted by invoking its methods, but it cannot receive top ups.

When a “Credit” object of priority zero is configured it shall not appear in the “Account” *credit\_reference\_list*, it will be not considered as part of the *available\_credit* calculation and it shall not be decremented by charges.

See the *credit\_reference\_list* attribute of the “Account” object (4.6.2.2.9) for more information.

#### 4.6.3.5.5 warning\_threshold

Holds a threshold value for *current\_credit\_amount*. When *current\_credit\_amount* is decremented to the value of *warning\_threshold*, a warning is triggered for the consumer that credit is low.

**NOTE** The *low\_credit\_threshold* attribute of the associated “Account” object echoes this attribute of the currently *In use* “Credit” object.

double-long, scaled according to the *currency* attribute of the “Account” object (4.6.2.2.15).

#### 4.6.3.5.6 limit

This attribute holds a threshold value for *current\_credit\_amount*. When *current\_credit\_amount* is decremented to the value of *limit*, then *credit\_status* becomes (4) *Exhausted*.

**NOTE** 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.

double-long, scaled according to the *currency* attribute of the “Account” object (4.6.2.2.15)

#### 4.6.3.5.7 credit\_configuration

Allows configuring the behaviour of the “Credit” object.

If **bit 0** is set then the credit item requires a visual indication on the meter display when it becomes selected.

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)

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 225/668 |
|-----------------------|------------|-----------------------------|---------|

If **bit 2** is set then this indicates that this “Credit” amount requires repayment and consequently the credit used will contribute to the *amount\_to\_clear* attribute in the “Account” object that references the particular “Credit” object.

If **bit 3** is set then the *current\_credit\_amount* can be cleared by an end of billing period action (using a script acting upon the *set\_amount\_to\_value* method. It is not recommended to configure “Credits” to permit this unless the meter is operating in credit mode.

If **bit 4** is set then this “Credit” object will be permitted to receive credit from tokens.

*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

#### 4.6.3.5.8 **credit\_status**

Driven by the prepayment application to indicate the state that the “Credit” object is in.

The states appear in Figure 27 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 *current\_credit\_amount*, *limit*, *preset\_credit\_amount* and *credit\_available\_threshold* attributes of the “Credit” objects and the *amount\_to\_clear* attribute of the “Account”.

When the *amount\_to\_clear* attribute of the ‘Account’ object referencing this “Credit” object transitions from a negative value to zero the EMC status shall be (0) *Enabled* again.

**NOTE** When a “Credit” object has a *credit\_status* (4) *Exhausted* then this “Credit” cannot be invoked again until the “Credit” transitions to (0) *Enabled*. 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 35.

#### 4.6.3.5.9 **preset\_credit\_amount**

This attribute is a value that is set as an initial amount of credit that is available for the “Credit” object.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 226/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

The value is added to the *current\_credit\_amount* attribute:

- a) when the *credit\_status* becomes (2), Selected/Invoked if *credit\_configuration* bit 1 (Requires confirmation) is set, and the application confirmation occurs, or
- b) when the *credit\_status* becomes (3) In use if *credit\_configuration* bit 1 (Requires confirmation) is cleared unless the *credit\_status* is (4) Exhausted, or
- c) when the method *invoke\_credit* is invoked, if the *credit\_configuration* bit 2 (Requires the credit amount to be paid back) is set, or
- d) when the *date\_time* in *period* occurs which is implicit.

When a “Credit” does not require a preset amount, the *preset\_credit\_amount* shall be 0 and the “Credit” can receive credit amounts from credit tokens or by invocation of the *update\_amount* and *set\_amount\_to\_value* methods. The value of *preset credit\_amount* does not change unless a new value is written by the client.

In the case of “Credits” of type *emergency\_credit*:

- the *preset\_credit\_amount* attribute shall be used,
- the “Credit” shall only be able to receive credit amounts from tokens when:
  - the status is (3) *In use* or (4) *Exhausted*;
  - some (or all) credit has been used; and
  - it is required to be paid back (*credit\_configuration* has bit 2 (Requires the credit amount to be paid back) set).

**NOTE** When the *current\_credit\_amount* has reached the *limit* then the *credit\_status* attribute will become (4) *Exhausted*. If the *credit\_configuration* bit 2 (Requires the credit amount to be paid back) is set, then the credit used is the difference between *preset\_credit\_amount* and *current\_credit\_amount* (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 *current\_credit\_amount* will be zero but the *credit\_status* will be (0) Enabled in the case of *amount\_to\_clear* = 0 or (4) *Exhausted* in the case where *amount\_to\_clear* is less than zero.

If this attribute is set to zero there is no behaviour associated with it.

double-long, scaled according to the *currency* attribute of the “Account” object.

#### 4.6.3.5.10 credit\_available\_threshold

A threshold value related to the “Account” object *available\_credit*.

When the *available\_credit* on the “Account” object decrements to the *credit\_available\_threshold* on the next-priority “Credit” object, the *credit\_status* of that object changes to:

- a) Selectable (1) if the *credit\_configuration* bit 1 (Requires confirmation) is set, or
- b) Selected/Invoked (2) if the *credit\_configuration* bit 1 (Requires confirmation) is cleared.

**NOTE 1** This reflects whether the “Credit” requires confirmation before it is put into use.

**NOTE 2** A “Credit” will not be *In use* until exhaustion of a higher-priority “Credit”, but if selected before a higher priority “Credit” is exhausted the value of the *current\_credit\_amount* attribute will contribute to *available\_credit* in the associated “Account”.

double-long, scaled according to the *currency* attribute of the “Account” object.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 227/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.6.3.5.11 period

Where the *credit\_type* = 3 (time\_based\_credit) or *credit\_type* = 4 (consumption\_based\_credit), this attribute holds the time at which the *current\_credit\_amount* is set automatically to the *preset\_credit\_amount*.

octet-string, formatted as specified in 4.1.6.1 for *date\_time*. Wildcards are allowed. If all fields are wildcards, the *preset\_credit\_amount* will never be added.

#### 4.6.3.6 Method description

##### 4.6.3.6.1 update\_amount (data)

This method adjusts the value of the *current\_credit\_amount* attribute. Positive values adjust the *current\_credit\_amount* positively. Negative values are also permitted.

*data* ::= double-long, scaled according to the *currency* attribute of the “Account” object.

##### 4.6.3.6.2 set\_amount\_to\_value (data)

This method sets the value of the *current\_credit\_amount* attribute. The amount previously in the updated attribute shall be given as the response parameter.

*data* ::= double-long, scaled according to the *currency* attribute of the “Account” object.

Returns double-long, scaled according to the *currency* attribute of the “Account” object.

##### 4.6.3.6.3 invoke\_credit (data)

This method is invoked to bring this “Credit” object to *credit\_status* (2) Selected/Invoked if it has a *credit\_configuration* bit 1 is set (Requires confirmation), and the *credit\_status* 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)

#### 4.6.3.7 Additional remarks

- a) 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.
- b) 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).

- c) 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*.

## COSEM Interface Classes

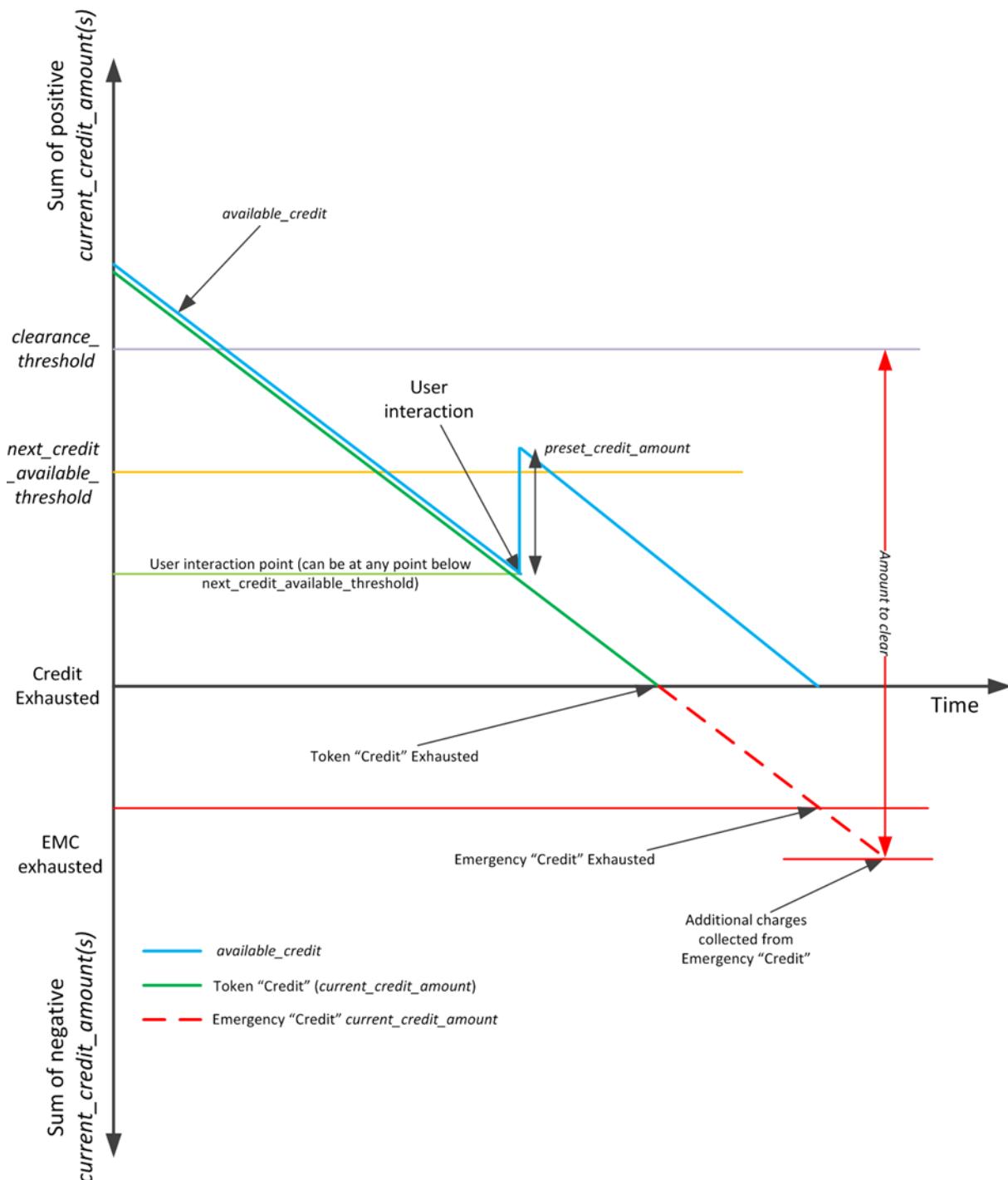
This will necessarily involve providing sufficient funds to repay the negative value of *current\_credit\_amount* of the “Credit” objects providing EMC function. To allow for the joint management of multiple “Credit” instances the credit used values for each “Credit” object are aggregated into the *amount\_to\_clear* 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.

- d) The “Account” object *token\_gateway\_configuration* attribute determines the distribution of credit to “Credit” objects.

Figure 29 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” *curren\_credit\_amount* is zero or below.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 229/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes



**Figure 29 – 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** The details of the collection (charge-taking) cycle may be project dependent, and thus they are subject to project specific companion specifications since they 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 (4.6.2.2.10).

| Charge  |          | 0...n                | class_id = 113, version = 0 |             |             |                   |
|---|----------|----------------------|-----------------------------|-------------|-------------|-------------------|
| <b>Attributes</b>   |          | <b>Data type</b>     | <b>Min.</b>                 | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 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          |
| <b>Specific methods</b>   |          | <b>m/o</b>           |                             |             |             |                   |
| 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          |

##### 4.6.4.1 Attribute description

###### 4.6.4.1.1 logical\_name

Identifies the “Charge” object instance. See 6.2.17.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 231/668 |
|-----------------------|------------|-----------------------------|---------|

**4.6.4.1.2 total\_amount\_paid**

Holds the total amount collected for this "Charge" object. It is not normally reset while the "Account" remains active.

**NOTE** This value is incremented by the application at the same time and at the same rate as the charge that is collected.

**4.6.4.1.3 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 *charge\_type* indicates which attributes are expected to be processed for this "Charge" object. The processing is also influenced by the *charge\_configuration* attribute.

**NOTE** In the case of *payment\_event\_based\_collection* charges this attribute dictates the mechanism of charge collection. The application collects *payment\_event\_based\_collection* charges from appropriate "Credits" as soon as credit is distributed from a new token.

**4.6.4.1.4 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 *total\_amount\_remaining* can be adjusted by invoking the appropriate methods of the "Charge" object.

"Charge" objects with priority value 0 shall not appear in the *charge\_reference\_list* of the "Account" object, but "Charge" objects with *priority <> 0* do necessarily appear in the *charge\_reference\_list* of an "Account".

**4.6.4.1.5 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 *scaler\_unit* attribute of the object identified by the *commodity\_reference* 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

## COSEM Interface Classes

```
{  
    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 1 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 2 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.

### Explanations to charge\_table elements

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 233/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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 1 For example, if the end consumer generates energy then the “Charge” should be negative, meaning that the utility pays the end consumer for the energy he produces.

NOTE 2 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.

### 4.6.4.1.6 *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 *unit\_charge\_active*. On activation, the whole structure of *unit\_charge\_passive* is copied into *unit\_charge\_active*.

### 4.6.4.1.7 *unit\_charge\_activation\_time*

Defines the time when the object itself invokes the specific method *activate\_unit\_charge\_passive*.

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 *date\_time*

### 4.6.4.1.8 *period*

Where *charge\_type* = (0) *consumption\_based\_collection* or (1) *time\_based\_collection* 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 *collect* method.

NOTE Implementations may vary: it is possible to collect the whole charge in a single collection, or to divide the collection into parts.

Where *charge\_type* = (2) *payment\_event\_based\_collection* this attribute is not used.

double-long-unsigned, expressed in seconds.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 234/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

#### 4.6.4.1.9 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 *charge\_type* = (2) *payment\_event\_based\_collection* then this “Charge” object will collect a proportion of each top-up in accordance with the proportion attribute (4.6.4.1.13). This applies only to token top ups and not to method invocations.

**NOTE** 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.

#### 4.6.4.1.10 last\_collection\_time

Holds the *date\_time* when the last collection took place.

This includes incremental collection made against consumption.

octet-string, formatted as specified in 4.1.6.1 for *date\_time*

#### 4.6.4.1.11 last\_collection\_amount

Holds the last collection amount relating to the *last\_collection\_time* attribute.

double-long, scaled according to the *price\_scale* element of *unit\_charge\_active*.

#### 4.6.4.1.12 total\_amount\_remaining

Where *charge\_configuration* bit 1 (Continuous collection) is cleared, this attribute holds the total amount remaining for this “Charge” object.

**NOTE 1** The value of this attribute is initially configured by invoking the *set\_total\_amount\_remaining* method.

**NOTE 2** 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 *charge\_configuration* bit 1 (Continuous collection) is set, this attribute is zero.

**NOTE 3** See also the Additional Notes at the end of the “Charge” IC specification.

double-long, scaled according to the *price\_scale* element of *unit\_charge\_active*.

#### 4.6.4.1.13 proportion

Where the *charge\_configuration* bit 0 (Percentage based collection) is set and *charge\_type* = (2) *payment\_event\_based\_collection*, 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 235/668 |
|-----------------------|------------|-----------------------------|---------|

The range 0x0000 to 0x2710 ( 0 to 10,000 ) represents 0 to 100 %.

**NOTE** The amount of collection that occurs may be limited by the *max\_provision* and *max\_provision\_period* attributes of the "Account" object.

If a fixed amount is required to be collected at every top-up (*charge\_type* = 2), then the amount to be taken should be set in the first element of the *charge\_table* array in the *unit\_charge\_active* / *unit\_charge\_passive* attribute.

Where either *charge\_type* <> (2) *payment\_event\_based\_collection* or *charge\_configuration* bit 0 (Percentage based collection) is cleared then this attribute has no effect.

In the case when *charge\_type* = (2) *payment\_event\_based\_collection* and the collection of a fixed amount is required see *unit\_charge\_active* (4.6.4.1.5) and *unit\_charge\_passive* (4.6.4.1.6).

#### 4.6.4.2 Method description

##### 4.6.4.2.1 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.

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** 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.

##### 4.6.4.2.2 activate\_passive\_unit\_charge (data)

Copies the whole structure of the *unit\_charge\_passive* attribute into the *unit\_charge\_active* attribute.

**NOTE** This method has no effect on the collection activity or the priority of the "Charge" object.

```
data ::= integer (0)
```

##### 4.6.4.2.3 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)
```

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 236/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

#### 4.6.4.2.4 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 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*.

#### 4.6.4.2.5 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 *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 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.)            | octet-string |                             |      |      | x + 0x10   |
| 4. token_description (dyn.)     | array        |                             |      |      | x + 0x18   |
| 5. token_delivery_method (dyn.) | enum         |                             |      |      | x + 0x20   |
| 6. token_status (dyn.)          | structure    |                             |      |      | x + 0x28   |
| Specific methods                | m/o          |                             |      |      |            |
| 1. enter (data)                 | m            |                             |      |      | x + 0x30   |

#### 4.6.5.1 Attribute description

##### 4.6.5.1.1 logical\_name

Identifies the “Token gateway” object instance. See 6.2.17.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 237/668 |
|-----------------------|------------|-----------------------------|---------|

**4.6.5.1.2 token**

Contains the unprocessed octet string of the most recently received or token for the purpose of capture in a history profile.

**4.6.5.1.3 token\_time**

Contains, for the most recently received and processed token, the time and date at which a received token arrived at the DLMS server.

octet-string, formatted as specified in 4.1.6.1 for *date\_time*

**4.6.5.1.4 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** 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
```

**4.6.5.1.5 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
```

**4.6.5.1.6 token\_status**

*token\_status* 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 *status\_code*.

**NOTE 1** It is expected that the optional bit string content will be documented in a project specific companion specification.

**NOTE 2** 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 62055-41:2018 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

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.

#### 4.6.5.2 Method description

##### 4.6.5.2.1 enter (data)

This method is invoked to transfer a token to the DLMS server in the form of octet-string. If successful it may return the `token_status` attribute.

`data ::= octet-string, and returns token_status`

#### 4.6.5.3 Additional remarks

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 programed 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 6.2.17.

#### 4.6.6 IEC 62055-41 Attributes (class\_id = 116, version =0)

##### 4.6.6.1 Overview

An instance of the “IEC 62055-41 Attributes” IC presents a selection of IEC 62055-41 data elements necessary to manage STS functions implemented within a DLMS server.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 239/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| IEC 62055-41 Attributes |           | 0..n                 | class_id = 116 , version = 0 |        |      |            |
|-------------------------|-----------|----------------------|------------------------------|--------|------|------------|
| Attributes              |           | Data type            | Min.                         | Max.   | Def. | Short name |
| 1. logical_name         | (static)  | octet-string         |                              |        |      | x          |
| 2. meter_pan            | (static)  | structure            |                              |        |      | x + 0x08   |
| 3. commodity            | (static)  | visible-string       |                              |        |      | x + 0x10   |
| 4. token_carrier_types  | (static)  | array                |                              |        |      | x + 0x18   |
| 5. encryption_algorithm | (static)  | unsigned             | 0                            | 99     |      | x + 0x20   |
| 6. supply_group_code    | (dynamic) | double-long-unsigned | 0                            | 999999 |      | x + 0x28   |
| 7. tariff_index         | (dynamic) | unsigned             | 0                            | 99     |      | x + 0x30   |
| 8. key_revision_number  | (dynamic) | unsigned             | 1                            | 9      |      | x + 0x38   |
| 9. key_type             | (dynamic) | unsigned             | 0                            | 3      |      | x + 0x40   |
| 10. key_expiry_number   | (dynamic) | unsigned             | 0                            | 255    |      | x + 0x48   |
| 11. no_of_kct_supported | (static)  | unsigned             | 2                            | 4      |      | x + 0x50   |
| 12. sts_certificate_no  | (static)  | visible-string       |                              |        |      | x + 0x58   |
| <b>Specific methods</b> |           | m/o                  |                              |        |      |            |
|                         |           |                      |                              |        |      |            |

### 4.6.6.2 Attribute description

#### 4.6.6.2.1 logical\_name

Identifies the “STS Attributes” object instance, see 6.2.17.

#### 4.6.6.2.2 meter\_pan

MeterPrimaryAccountNumber defined in IEC 62055-41:2018, 6.1.2.

```

meter_pan ::=structure
{
    issuer_identification_number: double-long-unsigned,
    decoder_reference_number: long64-unsigned,
    pan_check_digit: unsigned.
}

```

This attribute shall be set at the time of manufacture and shall be “read only” through the DLMS server’s communication interfaces.

#### 4.6.6.2.3 commodity

Holds the name of the metered commodity, this can be “ELECTRICITY”, “WATER”, “GAS”, or “TIME”.

This *commodity* attribute and the Account IC’s *currency* attribute shall be used to determine the applicable TokenSubclass and its unit as shown in Table 37 below.

This attribute shall be set at the time of manufacture and shall be “read only” through the DLMS server’s communication interfaces.

**Table 37 – Mapping of commodity attribute to Account IC's currency attribute**

| Clause 6.2.2 IEC62055-41 |                          |                                | Account IC (class_id =111) |               | IEC62055-41<br>attributes IC<br>(class_id = 116) |
|--------------------------|--------------------------|--------------------------------|----------------------------|---------------|--|
| Token Class              | Token SubClass           | Units of measurement           | currency_name              | currency_unit | commodity  |
| 0                        | 0 – electricity          | 0.1 kWh                        | kWh                        | Consumption   | ELECTRICITY                                      |
|                          | 1 – water                | 0.1 m <sup>3</sup>             | mt3                        | Consumption   | WATER  |
|                          | 2 – gas                  | 0.1 m <sup>3</sup>             | mt3                        | Consumption   | GAS  |
|                          | 3 – time                 | 0.1 minutes                    | Min                        | Consumption   | TIME   |
|                          | 4 – electricity currency | 10 <sup>-5</sup> base currency | As per ISO 4217            | Monetary      | ELECTRICITY                                      |
|                          | 5 – water currency       | 10 <sup>-5</sup> base currency | As per ISO 4217            | Monetary      | WATER  |
|                          | 6 – gas currency         | 10 <sup>-5</sup> base currency | As per ISO 4217            | Monetary      | GAS  |
|                          | 7 – time currency        | 10 <sup>-5</sup> base currency | As per ISO 4217            | Monetary      | TIME   |
|                          | 8 to 15 - reserved       | -                              | -                          | -             | -  |

**4.6.6.2.4 token\_carrier\_types**

All available TokenCarrierType instances as defined in 6.1.3 of IEC 62055-41:2018.

```
array token_carrier_type
token_carrier_type: unsigned
```

This attribute shall be set at the time of manufacture and shall be “read only” through the DLMS server’s communication interfaces.

**4.6.6.2.5 encryption\_algorithm**

EncryptionAlgorithm as defined in 6.1.5 of IEC 62055-41:2018.

This attribute shall be set at the time of manufacture and shall be “read only” through the DLMS server’s communication interfaces.

**4.6.6.2.6 supply\_group\_code**

SupplyGroupCode as defined in 6.1.6 of IEC 62055-41:2018.

This attribute shall be set at the time of manufacture and updated only through the insertion of a key change token (3 and 4 token sets).

**4.6.6.2.7 tariff\_index**

TariffIndex as defined in 6.1.7 of IEC 62055-41:2018.

## COSEM Interface Classes

This attribute shall be updated only through the insertion of a key change token and shall be “read only” through the DLMS server’s communication interfaces.

### **4.6.6.2.8 key\_revision\_number**

KeyRevisionNumber defined in 6.1.8 of IEC 62055-41:2018.

This attribute shall be updated only through the insertion of a key change token and shall be “read only” through the DLMS server’s communication interfaces.

### **4.6.6.2.9 key\_type**

KeyType as defined in 6.1.9 of IEC 62055-41:2018.

This attribute shall be updated only through the insertion of a key change token and shall be “read only” through the DLMS server’s communication interfaces.

### **4.6.6.2.10 key\_expiry\_number**

KeyExpiryNumber as defined in 6.1.10 of IEC 62055-41:2018.

This attribute shall be updated only through the insertion of a key change token and shall be “read only” through the DLMS server’s communication interfaces.

### **4.6.6.2.11 no\_of\_kct\_supported**

Defines the number of key change tokens supported by the server as defined in 6.2.7 and 6.2.8 of IEC 62055-41:2018.

This attribute shall be set at the time of manufacture and shall be “read only” through the DLMS server’s communication interfaces.

### **4.6.6.2.12 sts\_certificate\_no**

Conformance certificate number as issued by the STS Association.

This attribute shall be set at the time of manufacture and shall be “read only” through the DLMS server’s communication interfaces.

A firmware upgrade shall reset the value of this attribute to the new certificate number as part of the upgrade, but shall remain “read only” thereafter.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 242/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## 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)

#### 4.7.1.1 Overview

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          |                            |      |      |            |

#### 4.7.1.2 Attribute description

##### 4.7.1.2.1 logical\_name

Identifies the “IEC local port setup” object instance. 6.2.18.

##### 4.7.1.2.2 default\_mode

Defines the protocol used by the meter on the port.

enum:

- (0) protocol according to IEC 62056-21:2002 (modes A...E),
- (1) protocol according to IEC 62056-46:2002/AMD1:2006. Using this enumeration value all other attributes of this IC are not applicable,
- (2) protocol not specified. Using this enumeration value, attribute 4), prop\_baud is used for setting the communication speed on the port. All other attributes are not applicable.

##### 4.7.1.2.3 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,

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 243/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

- (7) 38 400 baud,
- (8) 57 600 baud,
- (9) 115 200 baud

### **4.7.1.2.4 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

### **4.7.1.2.5 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

### **4.7.1.2.6 device\_addr**

Device address according to IEC 62056-21:2002.

### **4.7.1.2.7 pass\_p1**

Password 1 according to IEC 62056-21:2002.

### **4.7.1.2.8 pass\_p2**

Password 2 according to IEC 62056-21:2002.

### **4.7.1.2.9 pass\_w5**

Password W5 reserved for national applications.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 244/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

#### 4.7.2 IEC HDLC setup (class\_id = 23, version = 1)

##### 4.7.2.1 Overview

This IC allows modelling and configuring communication channels according to DLMS UA 1000-2 Ed.11:2021, Clause 8IEC 62056-46:2002/AMD1:2006 . Several communication channels can be configured.

| IEC HDLC setup                    |          | 0...n         | class_id = 23, version = 1 |        |      |            |
|-----------------------------------|----------|---------------|----------------------------|--------|------|------------|
| Attributes                        |          | Data type     | Min.                       | Max.   | Def. | Short name |
| 1. logical_name                   | (static) | octet-string  |                            |        |      | x          |
| 2. comm_speed                     | (static) | enum          | 0                          | 9      | 5    | x + 0x08   |
| 3. window_size_transmit           | (static) | unsigned      | 1                          | 7      | 1    | x + 0x10   |
| 4. window_size_receive            | (static) | unsigned      | 1                          | 7      | 1    | x + 0x18   |
| 5. max_info_field_length_transmit | (static) | long-unsigned | 32                         | 2030   | 128  | x + 0x20   |
| 6. max_info_field_length_receive  | (static) | long-unsigned | 32                         | 2030   | 128  | x + 0x28   |
| 7. inter_octet_time_out           | (static) | long-unsigned | 20                         | 6000   | 25   | x + 0x30   |
| 8. inactivity_time_out            | (static) | long-unsigned | 0                          |        | 120  | x + 0x38   |
| 9. device_address                 | (static) | long-unsigned | 0x0010                     | 0x3FFD |      | x + 0x40   |
| <b>Specific methods</b>           |          | <b>m/o</b>    |                            |        |      |            |

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 2 030 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 1 000 ms to 6 000 ms in order to allow using communication media, where long delays may occur. The default value has been changed to 25 ms to align with 6.4.4.3.4 of IEC 62056-46:2002/AMD1:2006 .

##### 4.7.2.2 Attribute description

###### 4.7.2.2.1 logical\_name

Identifies the “IEC HDLC setup” object instance. See 6.2.20.

###### 4.7.2.2.2 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 245/668 |
|-----------------------|------------|-----------------------------|---------|

**4.7.2.2.3 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.

**4.7.2.2.4 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.

**4.7.2.2.5 max\_info\_length\_transmit**

The maximum information field length that a device can transmit. During logon, a smaller value can be negotiated.

**4.7.2.2.6 max\_info\_length\_receive**

The maximum information field length that a device can receive. During logon, a smaller value can be negotiated.

**4.7.2.2.7 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.

**4.7.2.2.8 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.

**4.7.2.2.9 device\_address**

Contains the physical device address of a device.

In the case of one byte addressing:

|             |                           |
|-------------|---------------------------|
| 0x00        | NO_STATION Address,       |
| 0x01...0x0F | Reserved for future use,  |
| 0x10...0x7D | Usable address space,     |
| 0x7E        | 'CALLING' device address, |
| 0x7F        | Broadcast address         |

In the case of two byte addressing:

|                 |                                    |
|-----------------|------------------------------------|
| 0x0000          | NO_STATION address,                |
| 0x0001...0x000F | Reserved for future use,           |
| 0x0010...0x3FFD | Usable address space,              |
| 0x3FFE          | 'CALLING' physical device address, |
| 0x3FFF          | Broadcast address                  |

#### 4.7.3 IEC twisted pair (1) setup (class\_id = 24, version = 1)

##### 4.7.3.1 General

Instances of this IC allow setting up data exchange over the medium *twisted pair with carrier signalling* as specified in IEC 62056-3-1:2021. Several communication channels can be configured.

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](http://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 6.2.21.

##### 4.7.3.2 Overview

Instances of this IC allow setting up data exchange over the medium *twisted pair with carrier signalling* as specified in IEC 62056-3-1:2021. Several communication channels can be configured.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 247/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| 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   |
| <b>Specific methods</b>    |          | m/o                       |                            |      |      |            |

### 4.7.3.3 Attribute description

#### 4.7.3.3.1 logical\_name

Identifies the “IEC twisted pair setup” object instance. See 6.2.21.

#### 4.7.3.3.2 mode

This attribute specifies the working mode of this interface

enum:

- (0) inactive. The interface ignores all frames received,
- (1) always active,
- (2)... (127) reserved,
- (128)...(250) manufacturer specific.

#### 4.7.3.3.3 comm\_speed

Holds the communication speed supported by the port.

enum:

- (2) 1200 baud,
- (3) 2400 baud,
- (4) 4800 baud,
- (5) 9600 baud,
- (6) 19200 baud,
- (7) 38400 baud

NOTE IEC 62056-3-1:2021 supports baud rates from 1 200 to 9 600.

#### 4.7.3.3.4 primary\_address\_list

Holds the list of Primary Station Addresses (ADP) for which each logical device of the real equipment (the secondary station) has been programmed. See IEC 62056-3-1:2021, 5.2.4.

primary\_address\_list\_type ::= array unsigned

#### 4.7.3.3.5 tabi\_list

Represents the list of the TAB(i) for which the real equipment (the secondary station) has been programmed in the case of forgotten station call (see IEC 62056-3-1:2021).

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 248/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

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

```
tabi_list_type ::= array tabi_element
tabi_element: integer
```

### 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 38.

**Table 38 – 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)

#### 4.7.4.1 Overview

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

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

#### 4.7.4.2 Attribute description

##### 4.7.4.2.1 logical\_name

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

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 249/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.7.4.2.2 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
```

#### 4.7.4.2.3 initialization\_string

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

```
array      initialization_string_element

initialization_string_element ::= structure
{
    request:          octet-string,
    response:         octet-string,
    delay_after_response: long-unsigned
}
```

If the array contains more than one initialization\_string\_element, the requests are sent in a sequence. The next request is sent after the expected response matching the previous request and waiting a delay\_after\_response time [ms], to allow the modem to execute the request.

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

#### 4.7.4.2.4 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)

##### 4.7.5.1 Overview

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 |             |             |                   |
|----------------------------|------------|------------------|----------------------------|-------------|-------------|-------------------|
| <b>Attributes</b>          |            | <b>Data type</b> | <b>Min.</b>                | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 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          |
| <b>Specific methods</b>    | <b>m/o</b> |                  |                            |             |             |                   |

##### 4.7.5.2 Attribute description

###### 4.7.5.2.1 logical\_name

Identifies the “Auto answer” object instance. See 6.2.6.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 251/668 |
|-----------------------|------------|-----------------------------|---------|

**4.7.5.2.2 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

**4.7.5.2.3 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*.

**4.7.5.2.4 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.

**4.7.5.2.5 number\_of\_calls**

This number is the reference used in modes 1 and 2.

When set to 0, this means there is no limit.

#### 4.7.5.2.6 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 *listening\_window* and the number of rings outside the *listening\_window*.

```
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 *listening\_window*.

#### 4.7.5.2.7 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 "+99419350000" to "+99419359999" are accepted.

Example 3: "7777\*" = calls from all numbers starting with "7777" are accepted.

Example 4: "\*9000" = calls from all numbers ending with "9000" are accepted.

- the *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. *number\_of\_rings*, *listening\_windows*, etc.

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

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 253/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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

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

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

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

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

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

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

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

### 4.7.6 Auto connect (class\_id = 29, version = 2)

#### 4.7.6.1 Overview

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<br>(static)     | octet-string  |                            |      |      | x          |
| 2. mode<br>(static)             | enum          |                            |      |      | x + 0x08   |
| 3. repetitions<br>(static)      | unsigned      |                            |      |      | x + 0x10   |
| 4. repetition_delay<br>(static) | long-unsigned |                            |      |      | x + 0x18   |
| 5. calling_window<br>(static)   | array         |                            |      |      | x + 0x20   |
| 6. destination_list<br>(static) | array         |                            |      |      | x + 0x28   |
| Specific methods                | m/o           |                            |      |      |            |
| 1. connect (data)               | o             |                            |      |      |            |

**4.7.6.2 Attribute description****4.7.6.2.1 logical\_name**

Identifies the “Auto connect” object instance. See 6.2.6.

**4.7.6.2.2 mode**

Controls the auto connect functionality in terms of the timing, the message type and the infrastructure to be used.

Modes (1) to (3) are dedicated to CSD services.

Modes (4) to (6) are dedicated to sending specific messages using a specific infrastructure.

Modes (101) to (104) apply to packet switched network connections only (e.g. GPRS).

enum:

- (0) no auto connect; the device never connects,
- (1) auto dialling allowed anytime, the values defined in the *calling\_window* are ignored,
- (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,
- (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.

**4.7.6.2.3 repetitions**

The maximum number of retries in case of unsuccessful connection attempts.

**4.7.6.2.4 repetition\_delay**

The time delay, expressed in seconds until an unsuccessful connection attempt can be repeated.

repetition\_delay = 0 means delay is not specified

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 255/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.7.6.2.5 calling\_window

Contains the time points when the window becomes active (start\_time), and inactive (end\_time). The start\_time implicitly defines the period.

EXAMPLE When the day of month is not specified (equal to 0xFF) this means that the calling window is managed on a daily basis. Daily, monthly ...window management can be defined.

```
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*.

#### 4.7.6.2.6 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
```

#### 4.7.6.3 Method description

##### 4.7.6.3.1 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)

##### 4.7.7.1 Overview

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<br>(static)       | octet-string  |       |                            |      | x          |  |
| 2. APN<br>(static)                | octet-string  |       |                            |      | x + 0x08   |  |
| 3. PIN_code<br>(static)           | long-unsigned |       |                            |      | x + 0x10   |  |
| 4. quality_of_service<br>(static) | structure     |       |                            |      | x + 0x18   |  |
| Specific methods                  | m/o           |       |                            |      |            |  |

**4.7.7.2 Attribute description****4.7.7.2.1 logical\_name**

Identifies the “GPRS modem setup” object instance. See 6.2.23.

**4.7.7.2.2 APN**

Defines the access point name of the network.

**4.7.7.2.3 PIN\_code**

Holds the personal identification number.

**4.7.7.2.4 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: 2)****4.7.8.1 Overview**

The cellular 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, UMTS, CDMA or LTE 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 DLMS UA 1000-1 Ed 15 Part 1:2021, 6.5.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 257/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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

### 4.7.8.2 Attribute description

#### 4.7.8.2.1 logical\_name

Identifies the “GSM Diagnostic” object instance. For logical names, see 6.2.23.

#### 4.7.8.2.2 operator

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

#### 4.7.8.2.3 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

#### 4.7.8.2.4 cs\_attachment

Indicates the current circuit switched status.

- enum:
- (0) inactive,
  - (1) incoming call,
  - (2) active,
  - (3)...(255) reserved

#### 4.7.8.2.5 ps\_status

The *ps\_status* value field indicates the packet switched status of the modem.

enum:

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 258/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

(0) inactive,  
 (1) GPRS,  
 (2) EDGE,  
 (3) UMTS,  
 (4) HSDPA,  
 (5) LTE,  
 (6) CDMA,  
 (7) LTE Cat M1,  
 (8) LTE Cat NB1,  
 (9) LTE Cat NB2,  
 (10)...(255) reserved

#### 4.7.8.2.6 cell\_info

Represents the cell information:

```
cell_info_type ::= structure
{
    cell_ID:           double-long-unsigned,
    location_ID:      long-unsigned,
    signal_quality:   unsigned,
    ber:               unsigned,
    mcc:               long-unsigned,
    mnc:               long-unsigned,
    channel_number:   double-long-unsigned
}
```

Where:

- cell\_ID: Four-byte cell ID in hexadecimal format;
- location\_ID: Two-byte location area code (LAC) in the case of GSM networks or Tracking Area Code (TAC) in the case of UMTS, CDMA or LTE networks in hexadecimal format (e.g. "00C3" equals 195 in decimal);
- 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 Rate (BER) measurement in percent:
  - (0...7) as RXQUAL\_n values specified in ETSI GSM 05.08:1996, 8.2.4
  - (99) not known or not detectable.
- mcc: Mobile Country Code of the serving network, as defined in ITU-T E.212 (05.2008) SERIES E;
- mnc: Mobile Network Code of the serving network, as defined in ITU-T E.212 (05.2008) SERIES E;
- channel\_number: Represents the absolute radio-frequency channel number (ARFCN or EARFCN for LTE network).

#### 4.7.8.2.7 adjacent\_cells

```
array        adjacent_cell_info

adjacent_cell_info ::= structure
{
    cell_ID:           double-long-unsigned,
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 259/668 |
|-----------------------|------------|-----------------------------|---------|

```

        signal_quality: unsigned
    }

```

Where:

- cell\_ID: Four-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. |

#### 4.7.8.2.8 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.

### 4.7.9 LTE monitoring (class\_id: 151, version = 1)

#### 4.7.9.1 Overview

Instances of the ‘LTE monitoring’ IC allow monitoring LTE modems by handling all data necessary data for this purpose.

Version 1 of the LTE monitoring IC also covers LTE Cat M1 and NB-IoT networks.

| LTE monitoring                      | 0...n                       | class_id = 151, version = 1 |      |      |            |
|-------------------------------------|-----------------------------|-----------------------------|------|------|------------|
| Attributes                          | Data type                   | Min.                        | Max. | Def. | Short name |
| 1. logical_name<br>(static)         | octet-string                |                             |      |      | x          |
| 2. LTE_network_parameters<br>(dyn.) | LTE_network_parameters_type |                             |      |      | x + 0x08   |
| 3. LTE_quality_of_service<br>(dyn)  | LTE_QoS_type                |                             |      |      | x + 0x10   |
| Specific methods                    | m/o                         |                             |      |      |            |

#### 4.7.9.2 Attribute description

##### 4.7.9.2.1 logical\_name

Identifies the “LTE monitoring” object instance. See 6.2.23.

##### 4.7.9.2.2 LTE\_network\_parameters

Represents the network parameters for the LTE network

```
LTE_network_parameters_type ::= structure
```

```
{
    T3402: long-unsigned,
```

## COSEM Interface Classes

```

T3412:           long-unsigned,
T3412ext2:       double-long-unsigned,
T3324:           long-unsigned,
TeDRX:            double-long unsigned,
TPTW:             long-unsigned,
qRxlevMin:        integer,
qRxlevMinCE-r13: integer,
qRxlevMinCE1-r13: integer
}

```

Where:

- T3402 timer in seconds, used on PLMN selection procedure and sent by the network to the modem. Refer to 3GPP TS 24.301 V13.4.0 (2016-01) for details.
- T3412 timer in seconds, used to manage the periodic tracking area updating procedure and sent by the network to the modem. Refer to 3GPP TS 24.301 V13.4.0 (2016-01) for details.
- T3412ext2 timer in seconds (extended periodic tracking area update timer). Refer to 3GPP TS 24.301 V13.4.0 (2016-01) and 3GPP TS 24.008 V13.7.0 (2016-10) for details.
- T3324 timer in seconds (Power saving mode active timer). Refer to 3GPP TS 24.301 V13.4.0 (2016-01) and 3GPP TS 24.008 V13.7.0 (2016-10) for details.
- TeDRX timer (Extended Idle mode DRX cycle timer). Refer to 3GPP TS 24.301 V13.4.0 (2016-01) and 3GPP TS 24.008 V13.7.0 (2016-10) for details. The double-long unsigned value shall be multiplied by 0,01 to get the real value in seconds. E.g. 512 represents 5,12 seconds
- TPTW timer (Extended Idle mode DRX paging time window). Refer to 3GPP TS 24.301 V13.4.0 (2016-01) and 3GPP TS 24.008 V13.7.0 (2016-10) for details. The long unsigned value shall be multiplied by 0,01 to get the real value in seconds. E.g. 512 represents 5,12 seconds
- qRxlevMin the minimum required Rx level in the cell in dBm as defined in 3GPP TS 36.304 V13.8.0 (2018-01).
- qRxlevMinCE-r13 the minimum required Rx level in enhanced coverage CE Mode A, LTE Cat M1. For this mode a value from -70...-22 in steps of 1 is required.  
NOTE 1 This field is not used in case of LTE Cat NB1 or NB2.
- qRxlevMinCE1-r13 the minimum required Rx level in enhanced coverage CE Mode B, LTE Cat M1. For this mode a value from -78...-22 in steps of 1.  
NOTE 2 This field is not used in case of LTE Cat NB1 or NB2.

### **4.7.9.2.3    LTE\_quality\_of\_service**

Represents the quality of service of the LTE network

```

LTE_QoS_type ::= structure
{
  (N)RSRQ:           integer,
  (N)RSRP:           integer,
  SNR:              integer,
  Coverage Enhancement: enum
}

```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 261/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

Where:

- (N)RSRQ represents the signal quality as defined in 3GPP TS 36.133. The use of this parameter can be determined from the GSM diagnostic IC, ps\_status attribute:
  - For LTE Cat M1, a value range from -30 up to 46 is necessary to represent RSRQ. Refer to 3GPP TS 36.133 V13.11.0 (2018-03) for details.
  - For LTE Cat NB1 and LTE Cat NB2 a value range from -30 up to 46 is necessary to represent NRSRQ. Refer to 3GPP TS 36.133 V14.4.0 (2017-06) for details.

|                 |              |                              |                      |
|-----------------|--------------|------------------------------|----------------------|
|                 | (-30)        | -34 dB,                      |                      |
|                 | (-29)        | -33,5 dB,                    |                      |
|                 | (-28)..(-1)  | -33 dB .. -19,5 dB,          |                      |
|                 | (0)          | -19,5 dB or less,            |                      |
|                 | (1)          | -19 dB,                      |                      |
| - (N)RSRP       | (2)..(31)    | -18,5 dB .. -4 dB,           | represents the       |
| signal level as | (32)         | -3,5 dB,                     | defined in 3GPP TS   |
| 36.133. The use | (33)         | -3 dB,                       | of this parameter is |
| determined from | (34)         | -3 dB or less,               | the GSM diagnostic   |
| IC, ps_status   | (35)         | -2,5 dB,                     | attribute:           |
| - For           | (36)         | -2 dB,                       | ps_status = 7 (LTE   |
| Cat M1) a       | (37)..(45)   | -1,5 dB .. +2,5 dB,          | value range from -   |
| 17 up to 97     | (46)         | 2,5 dB or better,            | is necessary to      |
| represent       | (99)         | not known or not detectable, | RSRP. Refer to       |
| 3GPP TS         |              | other values reserved        | 36.133 V13.11.0      |
| (2018-03) for   |              |                              | details              |
|                 | (-17)        | -156 dBm,                    |                      |
|                 | (-16)        | -155 dBm,                    |                      |
|                 | (-15)..(-1)  | -154 dBm .. -140 dBm,        |                      |
|                 | (0)          | -140 dBm or less,            |                      |
|                 | (1)          | -139 dB,                     |                      |
|                 | (2)          | -138 dBm,                    |                      |
|                 | (3)..(96)    | -137 dBm .. -44 dBm,         |                      |
|                 | (97)         | -44 dBm or better,           |                      |
|                 | (127)        | not known or not detectable, |                      |
|                 | other values | reserved                     |                      |

NOTE 1 depending on the use of Coverage Enhancement modes, a reported value of (0) will be used or not (for devices that do not use CE modes, values (-17..-1) will not be used. A Reported value of (0) means in that case a RSRP < -140 dBm).

- For ps\_status = 8 or 9 (LTE Cat NB1 or LTE Cat NB2) a value range from 0 to 113 is necessary to represent NRSRP value. Refer to 3GPP TS 36.133 V14.4.0 (2017-06) for details.

|            |                      |
|------------|----------------------|
| (0)        | -156 dBm,            |
| (1)        | -155 dBm,            |
| (2)        | -154 dBm             |
| (3)..(112) | -153 dBm .. -44 dBm, |
| (113)      | -44 dBm or better,   |

## COSEM Interface Classes

|              |                              |
|--------------|------------------------------|
| (127)        | not known or not detectable, |
| other values | reserved                     |

NOTE 2 For the NRSRP value (NB-IoT) the mapping is different from the mapping for the RSRP value (LTE Cat M1).

- SNR the Signal to Noise Ratio in a range from -20 dB to 50 dB. Refer to 3GPP TS 36.101 V15.4.0 (2019-01) for details.
- Coverage Enhancement Coverage Enhancement Mode A or B in case of LTE Cat M1, CE Level 0,1 or 2 in case of LTE Cat NB1/NB2. Refer to 3GPP TS 36.331 V15.5.1 (2019-05), 3GPP TS 36.321 V15.5.0 (2019-05) and 3GPP TS 36.213 V15.5.0 (2019-05) for details.

| Enum         | LTE Cat M1 | NB-IoT     |
|--------------|------------|------------|
| (0)          | CE Mode A  | CE Level 0 |
| (1)          | CE Mode B  | CE Level 1 |
| (2)          | (not used) | CE Level 2 |
| other values | reserved   |            |

## 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) a DLMS server hosted by an M-Bus master and exchanging dedicated M-Bus APDUs with M-Bus slaves;
- b) a DLMS client hosted by an M-Bus master and exchanging DLMS/COSEM APDUs with DLMS 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 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 server:

- DLMS 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)

#### 4.8.2.1 Overview

**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<br>(static) | octet-string |                            |      |      | x          |  |
| 2. default_baud<br>(static) | enum         | 0                          | 5    | 0    | x + 0x08   |  |
| 3. avail_baud<br>(static)   | enum         | 0                          | 7    |      | x + 0x10   |  |
| 4. addr_state<br>(static)   | enum         |                            |      |      | x + 0x18   |  |
| 5. bus_address<br>(static)  | unsigned     |                            |      |      | x + 0x20   |  |
| Specific methods            | m/o          |                            |      |      |            |  |

#### 4.8.2.2 Attribute description

##### 4.8.2.2.1 logical\_name

Identifies the “M-Bus slave port setup” object instance. See 6.2.22.

**4.8.2.2.2 default\_baud**

Defines the baud rate for the opening sequence.

enum:

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

**4.8.2.2.3 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

**4.8.2.2.4 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.

**4.8.2.2.5 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 = 2)****4.8.3.1 Overview**

Instances of the “M-Bus client” allow setting up M-Bus slave devices using wired and wireless 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 13757-3:2018; for details on wireless M-Bus communication see EN 13757-4 and for details on the M-Bus transport and security services see EN 13757-7:2018.

NOTE Version 2 of the “M-Bus client” IC is in line with EN 13757-3:2018, EN 13757-4:2019 and EN 13757-7:2018.

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 as defined in EN 13757-2:2018, 5.7.5, Identification Number, Manufacturer ID etc. as defined in EN 13757-7:2018, Clause 7,

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 265/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

**Transport Layer (TPL).** 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 “Extended Register” IC. M-Bus value objects may be captured in M-Bus “Profile generic” objects, eventually along with other, non M-Bus specific objects. If the data type used by M-Bus is not a data type specified in COSEM, then a data type conversion has to take place. The conversion process is specified in EN 13757-7:2018, Annex H.

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, transferring the security information etc.

Configuration field as defined in EN 13757-7:2018, 7.5.8 provides information about the security mode and number of encrypted blocks.

Configuration extension field as defined in EN 13757-7:2018, 7.6.2 provides information about security mode dependant additional parameters like Key ID and KDF function.

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 = 2 |      |       |            |
|------------------------------------|----------------------|----------------------------|------|-------|------------|
| 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   |
| 15. configuration_extension (dyn.) | unsigned             |                            |      | 0     | x + 0x70   |
| 16. invocation_status (dyn.)       | array                |                            |      |       | x + 0x78   |
| Specific methods                   | m/o                  |                            |      |       |            |

## COSEM Interface Classes

|   |  |   |  |  |          |
|---|--|---|--|--|----------|
| 1. slave_install (data)                 |  | o |  |  | x + 0x80 |
| 2. slave_deinstall (data)               |  | o |  |  | x + 0x88 |
| 3. capture (data)                       |  | o |  |  | x + 0x90 |
| 4. reset_alarm (data)                   |  | o |  |  | x + 0x98 |
| 5. synchronize_clock (data)             |  | o |  |  | x + 0xA0 |
| 6. data_send (data)                     |  | o |  |  | x + 0xA8 |
| 7. set_encryption_key (data)            |  | o |  |  | x + 0xB0 |
| 8. transfer_key (data)                  |  | o |  |  | x + 0xB8 |
| 9. transfer_security_information (data) |  | o |  |  | x + 0xC0 |
| 10. invocations_reset(data)             |  | o |  |  | X + 0xC8 |

### 4.8.3.2 Attribute description

#### 4.8.3.2.1 logical\_name

Identifies the “M-Bus client” object instance. See 6.2.22.

#### 4.8.3.2.2 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.

#### 4.8.3.2.3 capture\_definition

Provides the *capture\_definition* for M-Bus slave devices.

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

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

NOTE 2 The elements data\_information\_block and value\_information\_block correspond to Data Information Block (DIB) and Value Information Block (VIB) described in EN 13757-3:2018, 6.2 and Clause 7 respectively.

#### 4.8.3.2.4 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.

#### 4.8.3.2.5 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 267/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

NOTE 1 The specification of the B field of the OBIS codes limits the range to 1...64 within one logical device. See DLMS UA 1000-1 Ed 15 Part 1:2021, 5.2.

If the slave device is already configured and thus, its primary address is different from 0, then this value shall be written to the *primary\_address* attribute. From this moment, the data exchange with the M-Bus slave device is possible.

Otherwise, the *slave\_install* method shall be used; see 4.8.3.3.1.

NOTE 2 The *primary\_address* 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.

### 4.8.3.2.6 identification\_number

Carries the Identification Number element of the data header as specified in EN 13757-7:2018, 7.4.

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.

### 4.8.3.2.7 manufacturer\_id

Carries the Manufacturer Identification element of the data header as specified in EN 13757-7:2018, 7.4.

### 4.8.3.2.8 version

Carries the Version element of the data header as specified in EN 13757-7:2018, 7.4.

### 4.8.3.2.9 device\_type

Carries the Device type identification element of the data header as specified in EN 13757-7:2018, 7.5.4, Table 13.

### 4.8.3.2.10 access\_number

Carries the Access Number element of the data header as specified in EN 13757-7:2018, 7.4.

### 4.8.3.2.11 status

Carries the Status byte element of the data header as specified in EN 13757-7:2018, 7.5.6, Table 14 and 15.

It is updated with every readout of the M-Bus slave device.

### 4.8.3.2.12 alarm

Carries the Alarm state specified in EN 13757-7:2018, Annex D.

It is updated with every readout of the M-Bus slave device.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 268/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

**4.8.3.2.13 configuration**

Carries the Configuration field (previously: Signature field) as specified in EN 13757-7:2018, 7.5.8. 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.

**4.8.3.2.14 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.

**4.8.3.2.15 configuration\_extension**

Carries the Configuration Extension field as specified in EN 13757-7:2018, 7.6.2. It contains information about security mode dependant additional parameters like Key ID and KDF function.

It is updated with every readout of the M-Bus slave device.

**4.8.3.2.16 invocation\_status**

Carries the invocation status of pending M-Bus method invocations. When invocation\_status indicates (1) success then the result is carried in return-data of invocation\_status. If no return data is available, the return\_data octet-string is empty, i.e has length of 0. When there are no pending invocations, then the array is empty.

```
array invocation_status
invocation_status ::= structure
{
    method_id:      unsigned,
    completion:    enum
                    (0) pending,
                    (1) success,
                    (2) error
    return_data:    octet-string
}
```

Method\_id is taken from the request of the method invocation.

Multiple invocations of the same method will generate entries with the same method\_id.

**4.8.3.3 Method description****4.8.3.3.1 slave\_install (data)**

Installs a slave device, which is currently unconfigured (its primary address is 0).

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 269/668 |
|-----------------------|------------|-----------------------------|---------|

```
data ::= unsigned
```

This method can be successfully invoked only if the current value of the *primary\_address* attribute is 0. The following actions are performed:

- the M-Bus address 0 is checked for presence of a new device;
- if no uninstalled M-Bus slave is found, the method invocation fails;
- if the *slave\_install* method is invoked with **no** parameter, then the primary\_EN 13757-7:2018 address is assigned automatically. This is done by checking the *primary\_address* attribute of all M-Bus client objects in the DLMS/COSEM device and then selecting the first unused number. The *primary\_address* attribute is set to this address and it is then transferred to the M-Bus slave device;
- if the *slave\_install* method is invoked with a primary address as a parameter, then the *primary\_address* attribute is set to this value and it is then transferred to the M-Bus slave device.

```
data ::= CHOICE
{
    primary_address:     unsigned,
    meter_address:      meter_address_definition //secondary address
}
meter_address_definition ::= structure
{
    identification_number: double-long-unsigned,
    manufacturer_id:      long-unsigned,
    version:               unsigned,
    device_type:           unsigned
}
```

*meter\_address\_definition* contains a set of 4 values which is compared against the M-Bus telegram header of the SND\_IR telegram sent out periodically by a non bound M-Bus device.

The DLMS server confirms the SND\_IR with a CNF\_IR if the values in M-Bus telegram header match the fields configured in “M-Bus secondary address”.

NOTE Unconfigured slave devices are configured with primary address as specified in EN 13757-7:2018, 8.4 Selection and secondary addressing.

#### 4.8.3.3.2 slave\_deinstall (data)

De-installs the slave device. The main purpose of this service is to de-install the M-Bus slave device and to prepare the master for the installation of a new device. The following actions are performed:

- the M-Bus address is set to 0 in the M-Bus slave device;
- the encryption key transferred previously to the M-Bus slave device is destroyed; the default key is not affected;
- the *encryption\_key\_status* is set to (0): no encryption\_key;
- the attribute *primary\_address* is also set to 0.
- **all attributes are set to default values.**

NOTE A new M-Bus slave can be installed only once the value of the *primary\_address* attribute is 0.

```
data ::= unsigned (0)
```

**4.8.3.3.3 capture (data)**

Captures values – as specified by the *capture\_definition* attribute – from the M-Bus slave device.

```
data ::= integer (0)
```

**4.8.3.3.4 reset\_alarm (data)**

Resets alarm state of the M-Bus slave device.

```
data ::= integer (0)
```

**4.8.3.3.5 synchronize\_clock (data)**

Synchronize the clock of the M-Bus slave device with that of the M-Bus client device.

```
data ::= integer (0)
```

**4.8.3.3.6 data\_send(data)**

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]
    }
}
```

**4.8.3.3.7 set\_encryption\_key(data)**

Sets the encryption key in the M-Bus client and enables encrypted communication with the M-Bus slave device.

```
data ::= octet-string (encryption_key)
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 271/668 |
|-----------------------|------------|-----------------------------|---------|

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 `set_encryption_key` method with an octet-string of zero length.

**NOTE** Changing the encryption key requires two steps: First, the key is sent to the M-Bus slave, encrypted with the default key, using the `transfer_key` method; or transferred with SITP; using the `transfer_security_information` method. Second, the key is set in the M-Bus master using the `set_encryption_key` method.

#### 4.8.3.3.8 `transfer_key(data)`

Transfers an encryption key to the M-Bus slave device.

`data ::= octet-string (encrypted_key)`

Before encrypted M-Bus telegrams can be used, an operational encryption key has to be sent to the M-Bus slave, by invoking the `transfer_key` 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.

Each M-Bus slave device must be delivered with a default encryption key. The default key has the role of Key Encryption Key.

A new encryption key can be set in the M-Bus client by invoking the `set_encryption_key` method with the new encryption key as a parameter.

With further invocations of the `transfer_key` method, new encryption keys can be sent to the M-Bus slave. The method invocation parameter is the new encryption key encrypted with the default key. The M-Bus telegram is encrypted.

When an M-Bus slave is de-installed, the encryption key is destroyed but the default key is not affected. Encryption remains disabled until a new encryption key is transferred.

#### 4.8.3.3.9 `transfer_security_information(data)`

Transfer security information to and from the M-Bus slave device with the use of Security Information Transfer Protocol defined in EN 13757-7:2018, Annex A.

**Method invocation parameters:**

`data ::= octet-string`

**Return parameters:**

`data ::= octet-string`

Method invocation parameters contain SITP-command and return parameters contain SITP-response as defined in EN 13757-7:2018, A4.

#### 4.8.3.3.10 invocations\_reset

Reset the method invocations with M-Bus slave device to initial state.

Method invocation parameters:

`data ::= integer()`

0: resets all method invocations,  
 1: resets all successful method invocations,  
 2: resets all method invocations showing an error,  
 3: resets all pending method invocations.

### 4.8.4 Wireless Mode Q channel (`class_id = 73, version = 1`)

#### 4.8.4.1 Overview

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        | <code>class_id = 73, version = 1</code> |      |      |            |
|-------------------------------|--------------|---|------|------|------------|
| Attributes                    | Data type    | Min.                                    | Max. | Def. | Short name |
| 1. logical_name<br>(static)   | octet-string |   |      |      | x          |
| 2. addr_state<br>(static)     | enum         |   |      |      | x + 0x08   |
| 3. device_address<br>(static) | octet-string |   |      |      | x + 0x10   |
| 4. address_mask<br>(static)   | octet-string |   |      |      | x + 0x18   |
| <i>Specific methods</i>       | <i>m/o</i>   |   |      |      |            |

#### 4.8.4.2 Attribute description

##### 4.8.4.2.1 logical\_name

Identifies the “Wireless Mode Q channel” object instance. See 6.2.22.

##### 4.8.4.2.2 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.

##### 4.8.4.2.3 device\_address

The currently assigned address of the device on the network.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 273/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.8.4.2.4 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)

#### 4.8.5.1 Overview

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          |                            |      |      |            |

#### 4.8.5.2 Attribute description

##### 4.8.5.2.1 logical\_name

Identifies the “M-Bus master port setup” object instance. See 6.2.22.

##### 4.8.5.2.2 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 server M-Bus port setup (class\_id = 76, version = 0)

#### 4.8.6.1 Overview

Instances of the “DLMS server M-Bus port setup” are used in DLMS servers hosted by M-Bus slave devices, using the DLMS/COSEM wired or wireless M-Bus (wM-Bus) communication profile.

| DLMS 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   |

## COSEM Interface Classes

|                                   |          |                      |  |  |   |          |
|-----------------------------------|----------|----------------------|--|--|---|----------|
| 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 |
| <b>Specific methods</b>           |          | <i>m/o</i>           |  |  |   |          |

### 4.8.6.2 Attribute description

#### 4.8.6.2.1 logical\_name

Identifies the “DLMS server M-Bus port setup” object instance. See 6.2.22.

#### 4.8.6.2.2 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).

#### 4.8.6.2.3 M-Bus\_port\_communication\_state

Carries the communication status of the M-Bus node.

See [EN 13757-4:2019](#), 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)
- (4) This attribute is relevant only in wM-Bus.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 275/668 |
|-----------------------|------------|-----------------------------|---------|

#### **4.8.6.2.4 M-Bus\_Data\_Header\_Type**

Carries the type of the M-Bus Data Header, derived from the CITL 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 CITL 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 CITL 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 CITL field shall be 0x60 in an M-Bus frame sent by a master and 0x7C in an M-Bus frame sent by a slave.

#### **4.8.6.2.5 primary\_address**

Carries the primary address of the M-Bus slave device.

See DLMS UA 1000-2 Ed.11:2021, Table 113.

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.

#### **4.8.6.2.6 identification\_number**

Carries the Identification Number element of the Data Header as specified in EN 13757-7:2018, 7.5.1.

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 If in subsequent messages these values are not the same, the message is discarded.

#### **4.8.6.2.7 manufacturer\_id**

Carries the Manufacturer Identification element of the Data Header as specified EN 13757-7:2018, 7.5.2.

#### **4.8.6.2.8 version**

Carries the Version element of the Data Header as specified in EN 13757-7:2018, 7.5.3.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 276/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

**4.8.6.2.9 device\_type**

Carries the Device type identification element of the Data Header as specified in EN 13757-7:2018, 7.5.4, Table 13.

**4.8.6.2.10 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.

**4.8.6.2.11 listening\_window**

This attribute is relevant only in wM-Bus.

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, etc., window management can be defined.

```
array window_element

window_element ::= structure

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

start\_time and end\_time are formatted as specified in 4.6.1 for *date-time*.

**4.8.7 M-Bus diagnostic (class\_id = 77, version = 0)****4.8.7.1 Overview**

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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 277/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| 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   |
| <b>Specific methods</b>            | <i>m/o</i>           |                            |      |      |            |
| 1. reset (data)                    | o                    |                            |      |      |            |

### 4.8.7.2 Attribute description

#### 4.8.7.2.1 logical\_name

Identifies the “M-Bus diagnostic” object instance. See 6.2.22.

#### 4.8.7.2.2 received\_signal\_strength

This attribute is relevant only for wireless bidirectional M-Bus communication.

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.

#### 4.8.7.2.3 channel\_Id

This attribute is relevant only for wM-Bus communication.

When the wM-Bus profile is used, this attribute holds the identification of the channel currently used.

The default value is 0.

#### 4.8.7.2.4 link\_status

This attribute is relevant only for wM-Bus communication.

When the wM-Bus profile is used, this attribute holds the current status of link to the M-Bus network.

The four possible values for link\_status 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

**NOTE** 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.

Link\_status changes to normal operation when the link is re-established.

Link\_status changes to permanently interrupted , when the manufacturer specified number of attempts is exceeded.

**4.8.7.2.5 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.

**4.8.7.2.6 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.

**4.8.7.2.7 FCS\_OK\_frames\_counter**

Counts the number of frames received with a correct checksum. When the FCS\_OK\_frames\_counter field reaches the maximum value, it automatically returns to 0 on the next increment.

The default value is 0.

**4.8.7.2.8 FCS\_NOK\_frames\_counter**

Counts the number of frames received with an incorrect checksum. When the FCS\_NOK frames counter field reaches the maximum value, it automatically returns to 0 on the next increment. The default value is 0.

**4.8.7.2.9 capture\_time**

Holds the time stamp of the most recent change of the value of the attributes 2, 4, 6, 7 or 8.

```
capture_time ::= structure
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 279/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

```
{  
    attribute_id:      unsigned,  
    time_stamp:       date-time  
}
```

### **4.8.7.3    Method description**

#### **4.8.7.3.1    reset (data)**

Clears all counters, received\_signal\_strength, link\_status and capture\_time.

```
data ::= integer(0)
```

NOTE Channel\_id management is outside the scope of the specification of this IC.

See [EN 13757-4:2019](#).

## 4.9 Interface classes for setting up data exchange over the Internet

### 4.9.1 TCP-UDP setup (class\_id = 41, version = 0)

#### 4.9.1.1 Overview

Instances of the IC TCP-UDP setup allow 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 or the UDP sub-layer of the DLMS/COSEM CoAP transport layer of a DLMS/COSEM CoAP 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.

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.

A DLMS/COSEM CoAP transport layer may support multiple UDP connections used for communication between a physical device and one or more 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 | x + 0x18   |
| 5. nb_of_sim_conn      | (static)  | unsigned                   | 1    |          | x + 0x20   |
| 6. inactivity_time_out | (static)  | long-unsigned              |      | 180      | x + 0x28   |
| Specific methods       | m/o       |                            |      |          |            |

#### 4.9.1.2 Attribute description

##### 4.9.1.2.1 logical\_name

Identifies the “TCP-UDP setup” object instance. See 6.2.23.

##### 4.9.1.2.2 TCP-UDP\_port

Holds the TCP-UDP port number on which the physical device is listening for the DLMS/COSEM application.

Alternatively, holds the UDP port number of the CoAP endpoint of the DLMS/COSEM CoAP transport layer of a physical device.

## COSEM Interface Classes

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 |

### 4.9.1.2.3 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.

### 4.9.1.2.4 MSS

With the help of the Maximum Segment Size (MSS) option, a TCP entity 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.

### 4.9.1.2.5 nb\_of\_sim\_conn

The maximum number of simultaneous connections the COSEM TCP-UDP based transport layer is able to support.

### 4.9.1.2.6 inactivity\_time\_out

Defines the time, expressed in seconds over which, if no frame is received from the COSEM client, the inactive TCP connection shall be aborted.

When this value is set to 0, this means that the *inactivity\_time\_out* is not operational. In other words, a TCP connection, once established, in normal conditions – no power failure, etc. – will never be aborted by the COSEM server.

Note, that all actions related to the management of the inactivity time-out function such as measuring the inactivity time, aborting the TCP connection if the time-out is over, etc. are managed inside the TCP-UDP layer implementation.

## 4.9.2 IPv4 setup (class\_id = 42, version = 0)

### 4.9.2.1 Overview

This IC allows modelling the setup of the IPv4 layer, handling all information related to the IP Address settings associated to a given device and to a lower layer connection on which these settings are used.

There shall be an instance of this IC in a device for each different network interface implemented. For example, if a device has two interfaces (using the TCP-UDP/IPv4 profile on both of them), there shall be two instances of the IPv4 setup IC in that device: one for each of these interfaces.

| IPv4 setup                         | 0...n                | class_id = 42, version = 0 |      |      |            |
|------------------------------------|----------------------|----------------------------|------|------|------------|
| Attributes                         | Data type            | Min.                       | Max. | Def. | Short name |
| 1. logical_name (static)           | octet-string         |                            |      |      | x          |
| 2. DL_reference (static)           | octet-string         |                            |      |      | x + 0x08   |
| 3. IP_address                      | double-long-unsigned |                            |      |      | x + 0x10   |
| 4. multicast_IP_address            | array                |                            |      |      | x + 0x18   |
| 5. IP_options                      | array                |                            |      |      | x + 0x20   |
| 6. subnet_mask                     | double-long-unsigned |                            |      |      | x + 0x28   |
| 7. gateway_IP_address              | double-long-unsigned |                            |      |      | x + 0x30   |
| 8. use_DHCP_flag (static)          | boolean              |                            |      |      | x + 0x38   |
| 9. primary_DNS_address             | double-long-unsigned |                            |      |      | x + 0x40   |
| 10. secondary_DNS_address          | double-long-unsigned |                            |      |      | x + 0x48   |
| Specific methods                   | m/o                  |                            |      |      |            |
| 1. add_mc_IP_address (data)        | o                    |                            |      |      | x + 0x60   |
| 2. delete_mc_IP_address (data)     | o                    |                            |      |      | x + 0x68   |
| 3. get_nbof_mc_IP_addresses (data) | o                    |                            |      |      | x + 0x70   |

#### 4.9.2.2 Attribute description

##### 4.9.2.2.1 logical\_name

Identifies the “IPv4 setup” object instance. See 6.2.23.

##### 4.9.2.2.2 DL\_reference

References a Data link layer (e.g. Ethernet or PPP) setup object by its logical name.

The referenced object contains information about the specific settings of the data link layer supporting the IP layer.

##### 4.9.2.2.3 IP\_address

Carries the value of the IP address (IPv4) of this physical device on the network to which the device is connected via the referenced lower layer interface. It can be either (static) or (dynamic). In the latter case, dynamic IP address assignment (for example DHCP) is used.

If no IP address is assigned, the value is 0.

EXAMPLE The IPv4 address 192.168.0.1 (in dotted decimal notation) corresponds to C0A80001 (hexa) which gives 3232235521 (double-long-unsigned).

##### 4.9.2.2.4 multicast\_IP\_address

Contains an array of IP addresses. IP addresses in this array shall fall into the multicast group address range (“Class D” addresses, including IP addresses in the range of 224.0.0.0 to 239.255.255.255). When a device receives an IP datagram with one of these IP addresses in the destination IP address field, it shall consider that this datagram is addressed to it.

multicast\_IP\_address::= array double-long-unsigned

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 283/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.9.2.2.5 IP\_options

Contains the necessary parameters to support the selected IP options – for example Datagram time-stamping or security services (IPSec).

```

IP_options ::= array    IP_options_element

IP_options_element ::= structure

{
    IP_Option_Type:      unsigned,
    IP_Option_Length:    unsigned,
    IP_Option_Data:      octet-string
}

```

NOTE In all cases, as specified in RFC 791, the IP\_Option\_Length field includes the total length of all three fields: IP\_Option\_Type, IP\_Option\_Length and IP\_Option\_Data.

Allowed IP\_Option\_Types:

- Security: IP\_Option\_Type = 0x82, IP\_Option\_Length = 11

If this option is present, the device shall be allowed to send security, compartmentation, handling restrictions and TCC (closed user group) parameters within its IP Datagrams. The IP\_Option\_Data shall contain the value of the Security, Compartments, Handling Restrictions and Transmission Control Code values, as specified in RFC 791.

- Loose Source and Record Route: IP\_Option\_Type = 0x83

If this option is present, the device shall supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information. The IP\_Option\_Length and IP\_Option\_Data values are specified in RFC 791.

- Strict Source and Record Route: IP\_Option\_Type = 0x89

If this option is present, the device shall supply routing information to be used by the gateways in forwarding the datagram to the destination, and to record the route information. The IP\_Option\_Length and IP\_Option\_Data values are specified in RFC 791.

- Record Route: IP\_Option\_Type = 0x07

If this option is present, the device shall additionally:

- 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 additionally:

- 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.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 284/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

**4.9.2.2.6 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. (See more about sub-networking in RFC 940 and RFC 950.)

**4.9.2.2.7 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.

**4.9.2.2.8 use\_DHCP\_flag**

boolean:

TRUE: The device uses DHCP (Dynamic Host Configuration Protocol) to dynamically determine the *IP\_address*, *subnet\_mask* and *gateway\_IP\_address* parameters.

FALSE: The *IP\_address*, *subnet\_mask* and *gateway\_IP\_address* parameters shall be set locally.

**4.9.2.2.9 primary\_DNS\_address**

The IP Address of the primary Domain Name Server (DNS).

If no IP address is assigned, the value is 0.

**4.9.2.2.10 secondary\_DNS\_address**

The IP Address of the secondary Domain Name Server (DNS).

If no IP address is assigned, the value is 0.

**4.9.2.3 Method description****4.9.2.3.1 add\_mc\_IP\_address (IP\_Address)**

Adds one multicast IP address to the *multicast\_IP\_address* array.

*IP\_Address*::= double-long-unsigned

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 285/668 |
|-----------------------|------------|-----------------------------|---------|

**4.9.2.3.2 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

**4.9.2.3.3 get\_nb\_of\_mc\_IP\_addresses (data)**

Returns the number of IP Addresses contained in the *multicast\_IP\_address* array.

data ::= unsigned

**4.9.3 IPv6 setup (class\_id = 48, version = 0)****4.9.3.1 Overview**

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            |                            |      |      |            |

**4.9.3.2 Attribute description****4.9.3.2.1 logical\_name**

Identifies the “IPv6 setup” object instance. See 6.2.23.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 286/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

**4.9.3.2.2 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.

**4.9.3.2.3 address\_config\_mode**

Defines the IPv6 address configuration mode

enum:

- (0) Auto-configuration (default),
- (1) DHCPv6,
- (2) Manual,
- (3) ND (Neighbour Discovery)

NOTE The address\_config\_mode is common for all IPv6 addresses managed by an instance of the IPv6 setup class.

**4.9.3.2.4 unicast\_IPv6\_addresses**

Carries unicast IPv6 address(es) assigned to the related interface of the physical device on the network (unique local unicast, link local unicast and / or global unicast addresses). An IPv6 address can be either (static) or (dynamic) or both.

`unicast_IPv6_addresses ::= array octet-string`

The format of each unicast IPv6 address shall be as specified in RFC 3513.

If no unicast IPv6 address is assigned to the interface, an array of zero elements is present (default).

To reset all unicast IPv6 address(es) configured, an array of zero elements shall be written.

**4.9.3.2.5 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.

**4.9.3.2.6 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 287/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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.

### **4.9.3.2.7 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.

### **4.9.3.2.8 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.

### **4.9.3.2.9 traffic\_class**

Contains the traffic class element of the IPv6 header. The 6 most-significant bits are used for DSCP (Differentiated Services Codepoint), which is used to classify packets as specified in RFC 2474:1998, Clause 3.

### **4.9.3.2.10 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

#### 4.9.3.3 Method description

##### 4.9.3.3.1 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.

##### 4.9.3.3.2 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)

##### 4.9.4.1 Overview

NOTE 1 The name and the use of this interface class has been changed from "Ethernet setup" to "MAC address setup" to allow a more general use, without changing the version.

Instances of this IC hold the MAC address of the physical device (or, more generally, a device or software.) There shall be an instance of this IC for each network interface of a physical device.

NOTE 2 In the case of the three-layer HDLC based communication profile, the MAC address (lower HDLC address) is carried by an IEC HDLC setup object.

NOTE 3 In the case of the S-FSK PLC communication profile, the MAC address is carried by a S-FSK Phy&MAC setup object.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 289/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

|                          |                  |                                   |             |             |                   |
|--------------------------|------------------|-----------------------------------|-------------|-------------|-------------------|
| <b>MAC address setup</b> | 0...n            | <b>class_id = 43, version = 0</b> |             |             |                   |
| <b>Attributes</b>        | <b>Data type</b> | <b>Min.</b>                       | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 1. logical_name (static) | octet-string     |                                   |             |             | x                 |
| 2. MAC_address           | octet-string     |                                   |             |             | x + 0x08          |
| <b>Specific methods</b>  | <b>m/o</b>       |                                   |             |             |                   |

### 4.9.4.2 Attribute description

#### 4.9.4.2.1 logical\_name

Identifies the “MAC address setup” object instance. See 6.2.21, 6.2.23 and 6.2.27.

#### 4.9.4.2.2 mac\_address

Holds the MAC address.

## 4.9.5 PPP setup (class\_id = 44, version = 0)

### 4.9.5.1 Overview

NOTE Compared to earlier editions, 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.

|                                |                   |                                   |             |             |                   |
|--------------------------------|-------------------|-----------------------------------|-------------|-------------|-------------------|
| <b>PPP setup</b>               | 0...n             | <b>class_id = 44, version = 0</b> |             |             |                   |
| <b>Attributes</b>              | <b>Data type</b>  | <b>Min.</b>                       | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 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          |
| <b>Specific methods</b>        | <b>m/o</b>        |                                   |             |             |                   |

### 4.9.5.2 Attribute description

#### 4.9.5.2.1 logical\_name

Identifies the “PPP setup” object instance. See 6.2.23.

#### 4.9.5.2.2 PHY\_reference

References another object by its logical\_name. The object referenced contains information about the specific physical layer interface, supporting the PPP layer.

#### 4.9.5.2.3 LCP\_options

Contains the necessary parameters to support the selected LCP configuration options.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 290/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

```
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 1 In all cases, as specified in IETF STD 51 / RFC 1661, the LCP\_Option\_Length field includes the total length of all three fields: LCP\_Option\_Type, LCP\_Option\_Length and LCP\_Option\_Data.

NOTE 2 For assigned values see Point-to-Point (PPP) Protocol Field Assignments, 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 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 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 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 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 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 RFC 1661.

This configuration option provides a method to negotiate the compression of the data link layer address and control fields;

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 291/668 |
|-----------------------|------------|-----------------------------|---------|

- 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`

```
{
    callback_active:          boolean, // default: false,
    callback_data_length:     unsigned,
    callback_operation:       unsigned,
    callback_message:         octet-string
}
```

Where:

- the `callback-active` field indicates whether the callback option is active on this PPP link;
- the `callback_operation` field indicates the contents of the Message field;

`callback_operation: unsigned`

|     |   |
|-----|---|
| (0) | Location determined by user authentication,     |
| (1) | 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.

#### 4.9.5.2.4 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,
```

## COSEM Interface Classes

```

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 1 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:

```

0x0000 – No IP Compression is used (default),
0x002d – Van Jacobson Compressed TCP/IP (RFC 1332),
0x0003 – Robust Header Compression (ROHC) (RFC 3241),
0x0061 – IP Header Compression (RFC 2507, RFC 3544)

```

- Preferred-Local-IP-Address (Pref-Local-I), IPCP\_Option\_Type = 3. See RFC 1332.

This configuration option provides a way to negotiate the IP address to be used on the local end of the link. It allows the sender of the Configure-Request to state, which IP-address is desired, or to request that the peer provide the information. The peer can provide this information by NAK-ing the option, and returning a valid IP-Address;

NOTE 2 In RFC 1332 the name of option Type 3 is "IP-Address".

NOTE 3 Option types 20, 21 and 22 have been specified for the purposes of DLMS/COSEM; they are not specified in RFC 1332.

- Preferred-Peer-IP-Addresses (Pref-Peer-IP), IPCP\_Option\_Type = 20.

This configuration option provides a way to negotiate the IP Address to be used on the remote end of the link. When the Grant-Access-Only-to-Pref-Peer-on-List (GAO) parameter is set to be TRUE, the device shall accept PPP connection only with a remote device having one of the IP Addresses on this list. When the Use-Static-IP-Pool (USIP) parameter is set to TRUE, the COSEM Server device shall try to assign one of these IP Addresses to the remote device;

- Grant-Access-Only-to-Pref-Peer-on-List (GAO), IPCP\_Option\_Type = 21.

This configuration option indicates whether the device can accept PPP connection only with peer devices with IP Address on the above list or not. Its default value is FALSE;

- Use-Static-IP-Pool (USIP), IPCP\_Option\_Type = 22.

This configuration option indicates whether the device should try to assign one of the IP Addresses of the Preferred-Peer-IP-Addresses to the remote end device during the IP Address negotiation phase or not.

### 4.9.5.2.5 PPP\_authentication

Contains the parameters required by the PPP authentication procedure used.

```
PPP_auth_type ::= CHOICE
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 293/668 |
|-----------------------|------------|-----------------------------|---------|

```

{
    null-data [0] -- used when no authentication is required,
    structure [2] -- PAP_login or CHAP_algorithm or EAP_params
}

PAP_login ::= structure
{
    user-name: octet-string,
    PAP-password: octet-string
}

CHAP_algorithm ::= structure
{
    user-name: octet-string,
    algorithm_id: unsigned
}

```

Possible values for CHAP-algorithm-id parameter today are as follows:  
 0x05 – CHAP with MD5 (default), see RFC 1994.  
 0x06 – SHA-1,  
 0x80 – MS-CHAP, see RFC 2433,  
 0x81 – MS-CHAP-2, see RFC 2759.

NOTE 1 New values can be used as they 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 2 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 3 The Extensible Authentication Protocol (EAP) is specified in RFC 3748.

#### 4.9.6 SMTP setup (class\_id = 46, version = 0)

##### 4.9.6.1 Overview

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           |                            |      |      |            |

#### 4.9.6.2 Attribute description

##### 4.9.6.2.1 logical\_name

Identifies the “SMTP setup” object instance. See 6.2.23.

##### 4.9.6.2.2 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

##### 4.9.6.2.3 user\_name

Defines the user name to be used for the login to the SMTP server.

##### 4.9.6.2.4 login\_password

Password to be used for login. When the string is void, this means that there is no authentication.

##### 4.9.6.2.5 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.

##### 4.9.6.2.6 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.

#### 4.9.7 NTP setup (class\_id = 100, version = 0)

##### 4.9.7.1 Overview

Instances of the “NTP setup” IC allow setting up time synchronisation using the NTP protocol as specified in RFC 5905. One or several instances may be configured to support multiple time servers.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 295/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| NTP Setup                           |          | 0...n         | class_id = 100, version = 0 |      |       |            |
|-------------------------------------|----------|---------------|-----------------------------|------|-------|------------|
| Attributes                          |          | Data type     | Min.                        | Max. | Def.  | Short name |
| 1. logical_name                     | (static) | octet-string  |                             |      |       | x          |
| 2. activated                        | (static) | boolean       |                             |      | FALSE | x + 0x08   |
| 3. server_address                   | (static) | octet-string  |                             |      |       | x + 0x10   |
| 4. server_port                      | (static) | long-unsigned |                             |      | 123   | x + 0x18   |
| 5. authentication_method            | (static) | enum          |                             |      |       | x + 0x20   |
| 6. authentication_keys              | (static) | array         |                             |      |       | x + 0x28   |
| 7. client_key                       | (static) | octet-string  |                             |      |       | x + 0x30   |
| Specific methods                    |          | m/o           |                             |      |       |            |
| 1. synchronize (data)               |          | m             |                             |      |       |            |
| 2. add_authentication_key (data)    |          | o             |                             |      |       |            |
| 3. delete_authentication_key (data) |          | o             |                             |      |       |            |

### 4.9.7.2 Attribute description

#### 4.9.7.2.1 logical\_name

Identifies the “NTP setup” object instance. See 6.2.23.

#### 4.9.7.2.2 activated

Defines if the NTP time synchronisation is active or not.

boolean

|        |                            |
|--------|----------------------------|
| TRUE:  | Synchronisation active     |
| FALSE: | Synchronisation not active |

#### 4.9.7.2.3 server\_address

Defines the NTP server address as an octet string. This server address can be a name, which must be resolvable by the primary DNS or the secondary DNS. In the case when it is directly the IP address of the server the dot-decimal notation shall be used for IPv4 addresses and the text format for IPv6 addresses.

Examples: 163.187.45.87 (IPv4) or 2001:db8::1:0:0:1 (IPv6)

#### 4.9.7.2.4 server\_port

Defines the value of the UDP port related to this protocol. By default, this value is the NTP port number ID assigned by IANA:

ntp 123/udp Network Time Protocol

#### 4.9.7.2.5 authentication\_method

Defines the authentication mode used for NTP protocol

enum:

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 296/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

- (0) no\_security,
- (1) shared\_secrets,
- (2) auto\_key\_IFF

#### **4.9.7.2.6 authentication\_keys**

Contains the necessary symmetric keys if shared secrets mode of authentication is used.

```
authentication_keys ::= array authentication_key

authentication_key ::= structure

{
    key_id      double-long-unsigned,
    key         octet-string
}
```

#### **4.9.7.2.7 client\_key**

Specifies the client key (NTP server public key) for NTP auto key authentication mechanism using IFF authentication scheme (auto\_key\_IFF).

#### **4.9.7.3 Method description**

##### **4.9.7.3.1 synchronize (data)**

Synchronizes the time of the DLMS server with the NTP server.

```
data ::= integer(0)
```

##### **4.9.7.3.2 add\_authentication\_key (data)**

Adds a new symmetric authentication key to authentication key array.

```
data ::= structure

{
    key_id      double-long-unsigned,
    key         octet-string
}
```

##### **4.9.7.3.3 delete\_authentication\_key (data)**

Deletes a symmetric authentication key from the key array. The key to be deleted is identified by its key\_id.

```
data ::= double-long-unsigned
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 297/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.9.8 CoAP setup (class\_id = 152 version = 0)

##### 4.9.8.1 Overview

This IC allows modelling the setup of the CoAP sublayer – based on RFC 7252 of the DLMS/COSEM CoAP based transport layer of a DLMS/COSEM CoAP based communication profile.

| CoAP setup                       | 0...n         | class_id = 152, version = 0 |      |        |            |
|----------------------------------|---------------|-----------------------------|------|--------|------------|
| Attributes                       | Data type     | Min.                        | Max. | Def.   | Short name |
| 1. logical_name (static)         | octet-string  |                             |      |        | x          |
| 2. UDP_reference (static)        | octet-string  |                             |      |        | x + 0x08   |
| 3. ack_timeout (static)          | long-unsigned |                             |      |        | x + 0x10   |
| 4. ack_random_factor (static)    | long-unsigned |                             |      | 150    | x + 0x18   |
| 5. max_retransmit (static)       | long-unsigned |                             |      | 4      | x + 0x20   |
| 6. nstart (static)               | long-unsigned | 1                           |      | 1      | x + 0x28   |
| 7. delay_ack_timeout (static)    | long-unsigned |                             |      |        | x + 0x30   |
| 8. exponential_back_off (static) | long-unsigned |                             | 200  | 200    | x + 0x38   |
| 9. probing_rate (static)         | long-unsigned |                             |      | 1      | x + 0x40   |
| 10. CoAP_uri_path (static)       | octet-string  |                             |      | "dlms" | x + 0x48   |
| 11. transport_mode (static)      | enum          |                             |      |        | x + 0x50   |
| 12. version (dyn.)               | CHOICE        |                             | 15   |        | x + 0x58   |
| 13. token_length (static)        | unsigned      | 0                           | 8    | 1      | x + 0x60   |
| Specific methods                 | m/o           |                             |      |        |            |

##### 4.9.8.2 Attribute description

###### 4.9.8.2.1 Logical\_name

Identifies the “CoAP setup” object instance. See 6.2.23.

###### 4.9.8.2.2 UDP\_reference

References a TCP-UDP setup object by its logical name. The referenced object contains information about the UDP layer supporting the CoAP layer on which the CoAP server endpoint accepts UDP messages.

###### 4.9.8.2.3 ack\_timeout

The minimum initial ACK timeout, in ms, for confirmable messages.

###### 4.9.8.2.4 ack\_random\_factor

The random factor to apply for randomness of the initial ACK timeout (see RFC 7252).

For a new Confirmable CoAP message, the initial timeout is set to a random duration in between ack\_timeout and ack\_timeout x ack\_random\_factor x 0.01.

**4.9.8.2.5 max\_retransmit**

The maximum number of retransmissions for a confirmable message.

**4.9.8.2.6 nstart**

The number of simultaneous outstanding CoAP request messages of either of the following form: a CON CoAP message for which no CoAP acknowledgement has been received or a NON CoAP message for which no CoAP response message has been received. For further reference see RFC 7252.

**4.9.8.2.7 delay\_ack\_timeout**

The time, in ms, the CoAP messaging layer waits for the application layer to return a response before it will return an acknowledgement to prevent spurious retransmissions from peer.

**4.9.8.2.8 exponential\_back\_off**

The exponential back-off factor used to control the exponential back-off of the retransmission delay in between retransmissions of Confirmable messages by the CoAP messaging layer.

The retransmission\_delay in between the n-1 and the n'th retransmission of a Confirmable CoAP message will be:

$$\text{retransmission\_delay} = \text{initial\_ack\_timeout} \times (\text{exponential\_back\_off} \times 0.01) ^ {(n-1)}$$

Where the initial\_ack\_timeout is the random timeout in between ack\_timeout and ack\_timeout x ack\_random\_factor x 0.01 chosen for when the Confirmable message is first transmitted and where n refers to the ordinal number of retransmissions with n = 1 for the first retransmission.

**4.9.8.2.9 probing\_rate**

Defines the average data rate in number of bytes/seconds that an endpoint shall not exceed in sending to another endpoint that does not respond.

**4.9.8.2.10 CoAP\_uri\_path**

The Uri-path of the CoAP wrapper

**4.9.8.2.11 transport\_mode**

Defines the supported transport modes

enum:

- (1) Reliable operation supported only,
- (2) Unreliable operation supported only,
- (3) Reliable and Unreliable operation supported

**4.9.8.2.12 version**

The version of the DLMS/COSEM CoAP wrapper

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 299/668 |
|-----------------------|------------|-----------------------------|---------|

## CHOICE

```
{
    --simple data types
    octet-string [9],
    unsigned [17]
}
```

**4.9.8.2.13 token\_length**

The length of the Token, expressed in number of bytes, used by the CoAP clients within the CoAP layer

**4.9.9 CoAP diagnostic (class\_id = 153, version = 0)****4.9.9.1 Overview**

Instances of the IC “CoAP diagnostic” hold information related to the DLMS/COSEM CoAP transport layer operation of a DLMS server.

| CoAP diagnostic                    | 0...n        | class_id = 153, version = 0 |      |      |            |
|------------------------------------|--------------|-----------------------------|------|------|------------|
| Attributes                         | Data type    | Min.                        | Max. | Def. | Short name |
| 1. logical_name (static)           | octet-string |                             |      |      | x          |
| 2. messages_counter (dyn.)         | Structure    |                             |      |      | x + 0x10   |
| 3. request_response_counter (dyn.) | Structure    |                             |      |      | x + 0x18   |
| 4. coap_bt_counter (dyn.)          | Structure    |                             |      |      | x + 0x20   |
| 5. capture_time (dyn.)             | Structure    |                             |      |      | x + 0x28   |
| Specific methods                   | m/o          |                             |      |      |            |
| 1. reset (data)                    | O            |                             |      |      |            |
|                                    |              |                             |      |      |            |

**4.9.9.2 Attribute description****4.9.9.2.1 logical\_name**

Identifies the “CoAP diagnostic” object instance. See 6.2.23.

**4.9.9.2.2 messages\_counter**

Holds the CoAP messaging layer counters of an RFC 7252 CoAP messaging layer within a DLMS/COSEM CoAP transport layer.

```
structure
{
    CoAP_tx: double-long-unsigned,
```

## COSEM Interface Classes

```

CoAP_rx:           double-long-unsigned,
CoAP_rtx:          double-long-unsigned,
CoAP_tx_reset:     double-long-unsigned,
CoAP_rx_reset:     double-long-unsigned,
CoAP_tx_ack:        double-long-unsigned,
CoAP_rx_ack:        double-long-unsigned,
CoAP_rx_drop:       double-long-unsigned,
CoAP_tx_non-piggybacked: double-long-unsigned,
CoAP_max_rtx_exceeded: double-long-unsigned
}

```

Where:

- CoAP\_tx: represents the number of CoAP messages sent;
- CoAP\_rx: represents the number of CoAP messages received;
- CoAP\_rtx: represents the number of CoAP messages that have been re-sent;
- CoAP\_tx\_reset: represents the number of CoAP reset messages sent;
- CoAP\_rx\_reset: represents the number of CoAP reset messages received;
- CoAP\_tx\_ack: represents the number of CoAP acknowledgement messages sent;
- CoAP\_rx\_ack: represents the number of CoAP acknowledgement messages received;
- CoAP\_rx\_drop: represents the number of CoAP messages received but silently dropped due to message format error or other reason;
- CoAP\_tx\_non-piggybacked: represents the number of CoAP responses that were returned non-piggybacked (does not include retransmission of these by messaging layer);
- CoAP\_max\_rtx\_exceeded: represent the number of times transmission of a CoAP message was abandoned due to exceed of the max retransmission counter of the CoAP messaging layer

### **4.9.9.2.3 request\_response\_counter**

Holds counters for the CoAP request/response layer of an RFC 7252 CoAP request/response layer within a DLMS/COSEM CoAP transport layer.

```

structure
{
    CoAP_rx_requests:      double-long-unsigned,
    CoAP_tx_requests:      double-long-unsigned,
    CoAP_rx_response:      double-long-unsigned,
    CoAP_tx_response:      double-long-unsigned,
    CoAP_tx_client_error:   double-long-unsigned,
    CoAP_rx_client_error:   double-long-unsigned,
    CoAP_tx_server_error:   double-long-unsigned,
    CoAP_rx_server_error:   double-long-unsigned
}

```

Where:

- CoAP\_rx\_requests: the number of CoAP requests received (does not include duplicates received in messaging layer);
- CoAP\_tx\_requests: the number of CoAP requests sent (does not include retransmission of these by messaging layer);
- CoAP\_rx\_response: the number of CoAP responses received (does not include duplicates received in messaging layer, does not include CoAP client or server error responses);
- CoAP\_tx\_response: the number of CoAP responses sent (does not include retransmission of these by messaging layer, does not include client or server error responses);

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
|                       |            | 301/668                     |

## COSEM Interface Classes

- CoAP\_tx\_client\_error: represents the number of CoAP client errors sent;
- CoAP\_rx\_client\_error: represents the number of CoAP client errors received;
- CoAP\_tx\_server\_error: represents the number of CoAP server errors sent;
- CoAP\_rx\_server\_error: the number of CoAP server errors received.

### 4.9.9.2.4 coap\_BT\_counter

Holds the CoAP Block-Wise transfer layer counters of an RFC 7959 CoAP Block-Wise transfer layer within a DLMS/COSEM CoAP transport layer.

structure

```
{  
    CoAP_Block_Wise_Transfer_started:      double-long-unsigned,  
    CoAP_Block_Wise_Transfer_completed:    double-long-unsigned,  
    CoAP_Block_Wise_Transfer_timeout:      double-long-unsigned  
}
```

Where:

- CoAP\_Block\_Wise\_Transfer\_started: represents the number of CoAP Block-Wise Transfers started;
- CoAP\_Block\_Wise\_Transfer\_completed: represents the number of CoAP Block-Wise Transfers completed;
- CoAP\_Block\_Wise\_Transfer\_timeout: represents the number of CoAP Block-Wise Transfers timed-out due to inactivity.

### 4.9.9.2.5 capture\_time

Holds the time stamp of the most recent change of the value of the attributes 3, 4, 5.

```
capture_time ::= structure  
{  
    attribute_id:      unsigned,  
    time_stamp:       date-time  
}
```

### 4.9.9.3 Method description

#### 4.9.9.3.1 reset (data)

By invoking this method all counters are cleared.

The attribute capture\_time is set to the time of the reset execution.

data ::= integer (0)

## 4.10 Interface classes for setting up data exchange using S-FSK PLC

### 4.10.1 General

This subclause 4.10 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.

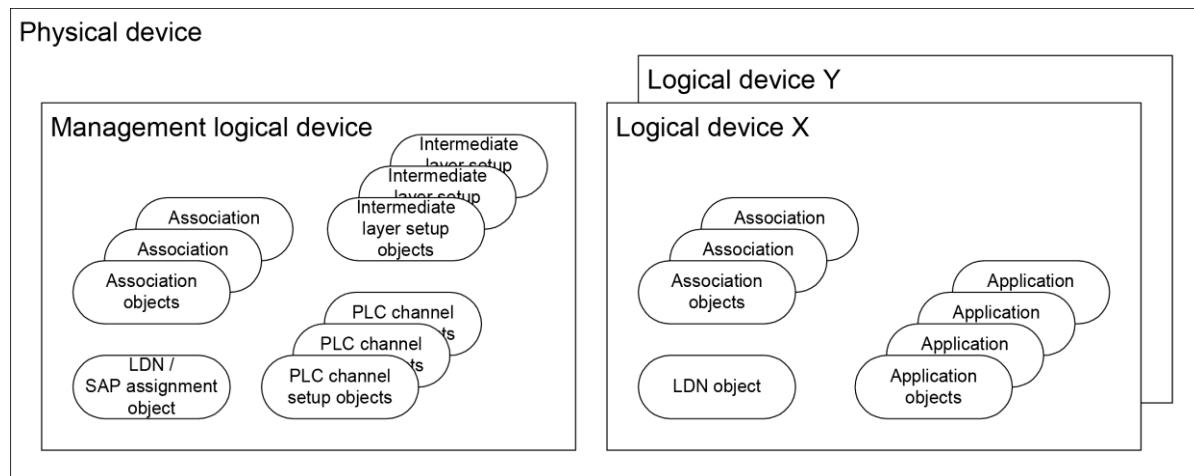
For definitions related to S-FSK PLC profile see 3.2.

### 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 30 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 30 – Object model of DLMS 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 303/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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 39 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 39 – Mapping IEC 61334-4-512:2001 MIB variables to COSEM IC attributes / methods**

| Name                                       | Reference<br>(unless otherwise indicated) | Interface class  | class_id / attribute / method |
|--|---|--|-------------------------------|
| <b>S-FSK Physical layer management</b>     |   |  |                               |
| delta-electrical-phase                     | variable 1                                | S-FSK Phy&MAC set-up<br>(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                 |
| <b>MAC layer management</b>                |   |  |                               |
| mac-address                                | variable 3                                | S-FSK Phy&MAC set-up<br>(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<br>(class_id = 52, version = 0) | 52 / Attr. 2                  |
| synchronization-confirmation-time-out      | variable 6                                |  | 52 / Attr. 3                  |
| time-out-not-addressed                     | variable 7                                |  | 52 / Attr. 4                  |
| time-out-frame-not-OK                      | variable 8                                |  | 52 / Attr. 5                  |
| min-delta-credit                           | variable 9                                |  | 50 / Attr. 12                 |
| initiator-mac-address                      | IEC 61334-5-1:2001,<br>4.3.7.6            | S-FSK Phy&MAC set-up<br>(class_id = 50, version = 1)               | 50 / Attr. 13                 |
| synchronization-locked                     | variable 10                               |  | 50 / Attr. 14                 |
| <b>IEC 61334-4-32 LLC layer management</b> |   |  |                               |
| max-frame-length                           | IEC 61334-4-32:1996<br>5.1.4              | IEC 61334-4-32 LLC setup<br>(class_id = 55, version = 1)           | 55 / Attr. 2                  |
| reply-status-list                          | variable 11                               |  | 55 / Attr. 3                  |
| broadcast-list                             | variable 12                               |  | –                             |

## COSEM Interface Classes

| Name                          | Reference<br>(unless otherwise indicated) | Interface class   | class_id / attribute / method |
|-------------------------------|---|---|-------------------------------|
| L-SAP-list                    | variable 13                               | NOTE In DLMS/COSEM, L-SAPs of logical devices are held by a SAP Assignment object |                               |
| <b>ACSE management</b>        |   |   |                               |
| application-context-list      | variable 14                               | NOTE In DLMS/COSEM the Association objects play a similar role.                   |                               |
| <b>Application management</b> |   |   |                               |
| active-initiator              | variable 15                               | S-FSK Active initiator (class_id = 51, version = 0)                               | 51 / Attr. 2                  |
| <b>MIB system objects</b>     |   |   |                               |
| reporting-system-list         | variable 16                               | S-FSK Reporting system list (class_id = 56, version = 0)                          | 56 / Attr. 2                  |
| <b>Other MIB objects</b>      |   |   |                               |
| 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)

#### 4.10.3.1 Overview

NOTE 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.

## COSEM Interface Classes

| 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   |
| <b>Specific methods</b>       | <b>m/o</b> |                  |                            |      |      |            |

**Table 40 – 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.  |                |

### 4.10.3.2 Attribute description

#### 4.10.3.2.1 logical\_name

Identifies the “S-FSK Phy&MAC setup” object instance. See 6.2.24.

#### 4.10.3.2.2 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 This enumeration is different from that of IEC 61334-4-512:2001.

#### 4.10.3.2.3 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

#### 4.10.3.2.4 max\_receiving\_gain

Holds the MIB variable *max-receiving-gain* (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 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.

#### 4.10.3.2.5 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 307/668 |
|-----------------------|------------|-----------------------------|---------|

#### **4.10.3.2.6 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 $\mu$ V.

#### **4.10.3.2.7 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.

#### **4.10.3.2.8 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 1 MAC addresses are expressed on 12 bits.

Contains the value of the address of the physical attachment (MAC address) associated to the local system. In the unconfigured state, the MAC address is "NEW-address".

This attribute is locally written by the CIASE when the system is registered (with a Register service). The value is used in each outgoing or incoming frame. The default value is "NEW-address".

This attribute is set to NEW:

- by the MAC sub-layer, once the time-out-not-addressed delay is exceeded;
- when a client system “resets” the server system. See 4.10.4.

When this attribute is set to NEW:

- the system loses its synchronization (function of the MAC-sublayer);
- the *mac\_group\_address* attribute is reset (array of 0 elements);
- the system automatically releases all AAs which can be released.

NOTE 2 The second item is not present in IEC 61334-4-512:2001.

The predefined MAC addresses are shown in Table 40.

#### **4.10.3.2.9 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).

#### **4.10.3.2.10 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 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 shall be specified in project specific companion specifications.

#### **4.10.3.2.11 repeater\_status**

Holds the current *repeater status* of the device.

boolean:

- |        |              |
|--------|--------------|
| FALSE: | no repeater, |
| TRUE:  | repeater     |

#### **4.10.3.2.12 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 Only the three least significant bits are used.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 309/668 |
|-----------------------|------------|-----------------------------|---------|

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.

#### **4.10.3.2.13 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 4.10.3.2.14 below). See also IEC 61334-5-1:2001 3.5.3, 4.1.6.3 and 4.1.7.2.

#### **4.10.3.2.14 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 (4.10.4.2.2).

If the value of this attribute is equal to FALSE, the system is in the synchronization-unlocked state. In this state, the *initiator\_mac\_address* 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 *initiator\_mac\_address* attribute which remains at the NO-BODY value. The default value of this variable shall be specified in the implementation specifications.

**NOTE** 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 *initiator\_mac\_address* attribute.

#### **4.10.3.2.15 transmission\_speed**

The transmission speed supported by the physical device. See also IEC 61334-5-1:2001, 3.2.2.

|                |            |             |
|----------------|------------|-------------|
| enum:          | 50 Hz      | 60 Hz       |
| (0)            | 300 baud   | 360 baud    |
| (1)            | 600 baud   | 720 baud    |
| (2)            | 1 200 baud | 1 440 baud  |
| (3) -- default | 2 400 baud | 2 880 baud  |
| (4)            | 4 800 baud | 5 760 baud  |
| (5)            | 7 200 baud | 8 640 baud  |
| (6)            | 9 600 baud | 11 520 baud |

#### 4.10.4 S-FSK Active initiator (class\_id = 51, version = 0)

##### 4.10.4.1 Overview

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<br>(static)          | octet-string         |                            |      |      | x          |
| 2. active_initiator<br>(dyn.)        | initiator_descriptor |                            |      |      | x + 0x08   |
| Specific methods                     | m/o                  |                            |      |      |            |
| 1. reset_NEW_not_synchronized (data) |                      |                            |      |      | x + 0x10   |

##### 4.10.4.2 Attribute description

###### 4.10.4.2.1 logical\_name

Identifies the “S-FSK Active initiator” object instance. See 6.2.25.

###### 4.10.4.2.2 active\_initiator

Holds the MIB variable *active-initiator* (variable 15) specified in IEC 61334-4-512:2001, 5.6.

Contains the identifiers of the active initiator, which has last registered the system with a Register request. See IEC 61334-4-511:2000, 7.2.

The Initiator system is identified with its System Title, MAC address and L-SAP selector:

```
initiator_descriptor ::= structure
{
    system_title:      octet-string,
    MAC_address:       long-unsigned,
    L_SAP_selector:    unsigned
}
```

The size and the structure of the system title may be specified in system specifications. When the system title is used as part of the initialisation vector of cryptographic algorithms, then the size shall meet the requirements applicable for the initialisation vector.

The MAC\_address element is used to update the *initiator-mac-address* MAC management variable when the system is configured in the synchronization-locked state. See the specification of the *initiator\_mac\_address* and the *synchronization\_locked* attributes of the S-FSK Phy&MAC setup IC in 5.9.3.

As long as the server is not registered by an active initiator, the L\_SAP\_selector field is set to 0 and the system\_title field is equal to an octet string of 0s.

The default value of the initiator-descriptor is: system\_title = octet-string of 0s, MAC\_address = NO-BODY and L\_SAP\_selector = 0.

The value of this attribute can be updated by the invocation of the *reset\_NEW\_not\_synchronized* method or by the CIASE Register service.

#### 4.10.4.3 Method description

##### 4.10.4.3.1 *reset\_NEW\_not\_synchronized* (data)

Holds the MIB variable *reset-NEW-not-synchronized* (variable 17) specified in IEC 61334-4-512:2001, 5.8.

Allows a client system to “reset” the server system. The submitted value corresponds to a client MAC address. The writing is refused if:

- the value does not correspond to a valid client MAC address or the predefined NO-BODY address;
- the submitted value is different from the NO-BODY address and the *synchronization\_locked* attribute is not equal to TRUE.

For the description of the Intelligent Search Initiator process, see IEC 62056-8-3:2013, 10.7.

When this method is invoked, the following actions are performed:

- the system returns to the unconfigured state (UNC: MAC-address equals NEW-address). This transition automatically causes the synchronization lost (function of the MAC sub layer);
- the system changes the value of the *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)

##### 4.10.5.1 Overview

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   |
| <i>Specific methods</i>                 |          | m/o           |                            |      |      |            |

##### 4.10.5.2 Attribute description

###### 4.10.5.2.1 logical\_name

Identifies the “S-FSK synchronization timeouts” object instance. See 6.2.25.

#### **4.10.5.2.2 search\_initiator\_timeout**

This timeout supports the intelligent search initiator function.

It defines the value of the time, expressed in seconds, during which the server system is searching for the initiator with the strongest signal.

During this timeout, all initiators, which may be heard by the servers, are expected to talk.

After the expiry of this timeout, the server will accept a Register request from the initiator having provided the strongest signal and it will be locked to that initiator.

If the value of the timeout is equal to 0, this means that the feature is not used.

The timeout is started at the beginning of the Search Initiator Phase, when the server receives the first frame with a valid initiator MAC address. The timeout is restarted when the Search Initiator Phase is over and the server locks on the initiator. During the Check Initiator Phase, it is restarted on the reception of each valid frame.

A Fast synchronization may be performed if the level of signal is good enough (Level of initiator signal  $\geq$  Search-Initiator-Threshold) and one of the MAC addresses (Source or Destination) is an Initiator MAC address. This means that the module (the meter) is next to a DC or next to a module that is already locked on that DC. The module locks in this case on that initiator.

#### **4.10.5.2.3 synchronization\_confirmation\_timeout**

Holds the MIB variable *synchronization-confirmation-timeout* (variable 6) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6.

Defines the value of the time, expressed in seconds, after which a server system which just gets frame synchronized (detection of a data path equal to AAAA54C7 hex) will automatically lose its frame synchronization if the MAC sublayer does not identify a valid MAC frame. The timeout starts after the reception of the first four bytes of a physical frame.

The value of this variable can be modified by a client system. This time-out ensures a fast desynchronization of a system, which has synchronized on a wrong physical frame. See IEC 61334-5-1:2001, 3.5.3 for more details.

The default value of this variable should be specified in the implementation specifications.

A value equal to 0 is equivalent to cancel the use of the related *synchronization\_confirmation\_timeout* counter.

#### **4.10.5.2.4 time\_out\_not\_addressed**

Holds the MIB variable *time-out-not-addressed* (variable 7) specified in IEC 61334-4-512:2001, 5.3 and in IEC 61334-5-1:2001, 4.3.7.6.

Defines the time, in minutes, after which a server system that has not been individually addressed:

- returns to the non configured state (UNC: MAC-address equals NEW-address): this transition automatically involves the loss of the synchronization (function of the MAC sub layer) and releasing all AAs that can be released;

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 313/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

- loses its active initiator: the MAC address of the active-initiator is set to NO-BODY, the LSAP selector is set to the value 00 and the System Title is set to an octet-string of 0s.

Because broadcast addresses are not individual system addresses, the timer associated with the *time-out-not-addressed* delay ensures that a forgotten system will sooner or later return to the unconfigured state. It will be then discovered again.

A forgotten system is a system, which has not been individually addressed for more than the "*time-out-not-addressed*" amount of time.

The default value of this variable should be specified in the implementation specifications.

A value equal to 0 is equivalent to cancel the use of the related *time-out-not-addressed* counter.

### **4.10.5.2.5 time\_out\_frame\_not\_OK**

Holds the MIB variable *time-out-frame-not-OK* (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 *time-out-frame-not-OK* counter.

## **4.10.6 S-FSK MAC counters (class\_id = 53, version = 0)**

### **4.10.6.1 Overview**

An instance of the "S-FSK MAC counters" IC stores counters related to the frame exchange, transmission and repetition phases.

| <b>S-FSK MAC counters</b>    |           | <b>0...n</b>         | <b>class_id = 53, version = 0</b> |             |             |                   |
|------------------------------|-----------|----------------------|-----------------------------------|-------------|-------------|-------------------|
| <b>Attributes</b>            |           | <b>Data type</b>     | <b>Min.</b>                       | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 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          |
| <b>Specific methods</b>      |           | <b>m/o</b>           |                                   |             |             |                   |
| 1. reset (data)              |           |                      |                                   |             |             |                   |

#### 4.10.6.2 Attribute description

##### 4.10.6.2.1 logical\_name

Identifies the "S-FSK MAC counters" object instance. See 6.2.25.

##### 4.10.6.2.2 synchronization\_register

Holds the MIB variable *synchronization-register* (variable 23), specified in IEC 61334-4-32:1996, 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 315/668 |
|-----------------------|------------|-----------------------------|---------|

For details on the MA\_Sync.indication service primitive, see IEC 61334-5-1:2001, 4.1.7.1.

If the synchronization process ends with one of the three primitives listed above, the synchronization-register variable is updated by taking into account the SA and DA fields of the primitive.

The updating of the *synchronization-register* variable is carried out as follows:

First, the Management Entity checks the SA and DA fields.

- If one of these fields corresponds to a client MAC address (CMA) the Entity:
  - checks if the client MAC address (CMA) appears in one of the couples contained in the synchronization-register variable;
  - if it appears, the related synchronizations-counter subfield is incremented;
  - if it does not appear, a new (mac-address, synchronizations-counter) couple is added. This couple is initialized to the (CMA, 1) value.
- If none of the SA and DA fields correspond to a client MAC address, it is supposed that the system found its synchronization reference on a DiscoverReport type frame. In that case, the mac-address which should be registered in the synchronization-register variable is the predefined NEW value (FFE). The updating of the synchronization-register variable is carried out in the same way as it is done for a normal client MAC address (CMA).

When a *synchronizations-counter* field reaches the maximum value, it automatically returns to 0 on the next increment.

The maximum number of synchronization couples {mac-address, synchronizations-counter} contained in this variable should be specified in the implementation specifications. When this maximum is reached, the updating of the variable follows a First-In-First-Out (FIFO) mechanism: only the newest source MAC addresses are memorized.

The default value of this variable is an empty array.

#### 4.10.6.2.3 desynchronization\_listing

Holds the MIB variable *desynchronization-listing* (variable 24), specified in IEC 61334-4-32:1996, 5.8.

structure

```
{
    nb_physical_layer_desynchronization:          double-long-unsigned,
    nb_time_out_not_addressed_desynchronization: double-long-unsigned,
    nb_timeout_frame_not_OK_desynchronization:   double-long-unsigned,
    nb_write_request_desynchronization:           double-long-unsigned,
    nb_wrong_initiator_desynchronization:         double-long-unsigned
}
```

This variable counts the number of desynchronizations that occurred depending on their cause. On reception of synchronization loss notification, the Management Entity updates this attribute by incrementing the counter related to the cause of the desynchronization.

When one of the counters reaches the maximum value, it automatically returns to 0 on the next increment.

The default value of this variable is a structure with all elements equal to 0.

#### 4.10.6.2.4 broadcast\_frames\_counter

Holds the MIB variable *broadcast-frames-counter* (variable 19) specified in IEC 61334-4-32:1996, 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.

#### 4.10.6.2.5 repetitions\_counter

Holds the MIB variable *repetitions-counter* (variable 20) specified in IEC 61334-4-32:1996, 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.

#### 4.10.6.2.6 transmissions\_counter

Holds the MIB variable *transmissions-counter* (variable 21) specified in IEC 61334-4-32:1996, 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.

#### 4.10.6.2.7 CRC\_OK\_frames\_counter

Holds the MIB variable *CRC-OK-frames-counter* (variable 22) specified in IEC 61334-4-32:1996, 5.8.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 317/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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.

### 4.10.6.2.8 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.

### 4.10.6.3 Method description

#### 4.10.6.3.1 reset (data)

Clears all counters.

data ::= integer (0)

## 4.10.7 IEC 61334-4-32 LLC setup (class\_id = 55, version = 1)

### 4.10.7.1 Overview

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 61334-4-32:1996.

| IEC 61334-4-32 LLC setup        | 0...n         | class_id = 55, version = 1 |      |      |            |
|---------------------------------|---------------|----------------------------|------|------|------------|
| Attributes                      | Data type     | Min.                       | Max. | Def. | Short name |
| 1. logical_name<br>(static)     | octet-string  |                            |      |      | x          |
| 2. max_frame_length<br>(static) | long-unsigned |                            |      |      | x + 0x08   |
| 3. reply_status_list<br>(dyn.)  | array         |                            |      |      | x + 0x10   |
| Specific methods                | m/o           |                            |      |      |            |

### 4.10.7.2 Attribute description

#### 4.10.7.2.1 logical\_name

Identifies the “IEC 61334-4-32 LLC setup” object instance. See 6.2.25.

#### 4.10.7.2.2 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.

#### 4.10.7.2.3 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.

```

array      reply_status

reply_status ::= structure

{
    L_SAP_selector:      unsigned,
    length_of_waiting_L_SDU:  unsigned
}

```

length\_of\_waiting\_L\_SDU in the case of the S-FSK profile is in number of sub-frames; valid values are 1 to 7.

#### **4.10.8 S-FSK Reporting system list (class\_id = 56, version = 0)**

##### **4.10.8.1 Overview**

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          |                            |      |      |            |

##### **4.10.8.2 Attribute description**

###### **4.10.8.2.1 logical\_name**

Identifies the “S-FSK Reporting system list” object instance. See 6.2.25.

###### **4.10.8.2.2 reporting\_system\_list**

Holds the MIB variable *reporting-system-list* (variable 16) specified in IEC 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:

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 319/668 |
|-----------------------|------------|-----------------------------|---------|

- 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 subclause 4.11 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)

#### 4.11.2.1 Overview

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<br>(static)      | octet-string  |                            |      |      | x          |
| 2. max_octets_ui_pdu<br>(static) | long unsigned |                            |      | 128  | x + 0x08   |
| Specific methods                 | m/o           |                            |      |      |            |

#### 4.11.2.2 Attribute description

##### 4.11.2.2.1 logical\_name

Identifies the “ISO/IEC 8802-2 LLC Type 1 setup” object instance. See 6.2.26.

##### 4.11.2.2.2 max\_octets\_ui\_pdu

Refer to the appropriate MAC protocol specification for any limitation on the maximum number of octets in a UI PDU. No restrictions are imposed by the LLC sublayer. However, in the interest of having a value that all users of Type 1 LLC may depend upon, all MACs shall at least be capable of accommodating UI PDUs with information fields up to and including 128 octets in length.

See ISO/IEC 8802-2:1998, 6.8.1 *Maximum number of octets* in a UI PDU.

### 4.11.3 ISO/IEC 8802-2 LLC Type 2 setup (class\_id = 58, version = 0)

#### 4.11.3.1 Overview

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           |                            |      |      |            |

#### 4.11.3.2 Attribute description

##### 4.11.3.2.1 logical\_name

Identifies the “ISO/IEC 8802-2 LLC Type 2 setup” object instance. See 6.2.26.

transmit\_window\_size\_k

The transmit window size (k) shall be a data link connection parameter that can never exceed 127. It shall denote the maximum number of sequentially numbered I PDUs that the sending LLC may have outstanding (i.e., unacknowledged). The value of k is the maximum number by which the sending LLC send state variable V(S) can exceed the N(R) of the last received I PDU.

See ISO/IEC 8802-2:1998, 7.8.4 *Transmit window size, k*.

##### 4.11.3.2.2 receive\_window\_size\_rw

The receive window size (RW) shall be a data link connection parameter that can never exceed 127. It shall denote the maximum number of unacknowledged sequentially numbered I PDUs that the local LLC allows the remote LLC to have outstanding. It is transmitted in the information field of XID (see ISO/IEC 8802-2:1998, 5.4.1.1.2) and applies to the XID sender. The XID receiver shall set its transmit window (k) to a value less than or equal to the receive window of the XID sender to avoid overrunning the XID sender.

See ISO/IEC 8802-2:1998, 7.8.6 *Receive window size, RW*.

##### 4.11.3.2.3 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

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 321/668 |
|-----------------------|------------|-----------------------------|---------|

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 *Maximum number of octets in an I PDU, N1*.

#### **4.11.3.2.4 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 *Maximum number of transmissions, N2*.

#### **4.11.3.2.5 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 *Acknowledgement timer*.

#### **4.11.3.2.6 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 *P-bit timer*.

#### **4.11.3.2.7 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 *Reject timer*.

#### **4.11.3.2.8 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 *Busy-state timer*.

### **4.11.4 ISO/IEC 8802-2 LLC Type 3 setup (class\_id = 59, version = 0)**

#### **4.11.4.1 Overview**

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.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 322/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

| 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   |
| <b>Specific methods</b>                 | <i>m/o</i>    |                            |      |      |            |

### 4.11.4.2 Attribute description

#### 4.11.4.2.1 logical\_name

Identifies the “ISO/IEC 8802-2 LLC Type 3 setup” object instance. See 6.2.26.

#### 4.11.4.2.2 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 *Maximum number of octets in an ACn command PDU, N3*.

#### 4.11.4.2.3 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 *Maximum number of transmissions, N4*.

#### 4.11.4.2.4 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, *Acknowledgement time, T1*.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 323/668 |
|-----------------------|------------|-----------------------------|---------|

#### **4.11.4.2.5 receive\_lifetime\_var\_t2**

This time value is a logical link parameter that determines the period of all of the receive variable lifetime timers. T2 shall be longer by a margin of safety than the longest possible period during which the first transmission and all retries of a single PDU may occur. The margin of safety shall take into account anything affecting LLCs perception of the arrival time of PDUs, such as LLC response time, timer resolution, and variations in the time required for the medium access control sublayer to pass received PDUs to LLC.

If the destruction of the received state variables is not desired, the value of time T2 may be set to infinity. In this case the receive variable lifetime timer need not be implemented.

The unit is seconds. Infinity is indicated by all bits set to 1.

See ISO/IEC 8802-2:1998, 8.6.5 *Receive lifetime variable, T2*.

#### **4.11.4.2.6 transmit\_lifetime\_var\_t3**

This time value is a logical link parameter that determines the minimum lifetime of the transmit sequence state variables. T3 shall be longer by a margin of safety than:

- a) the logical link variable T2 at stations to which ACn commands are sent; and
- b) the longest possible lifetime of an ACn command-response pair. The lifetime of an ACn command-response pair shall take into account the sum of processing time, queuing delays, and transmission time for the command and response PDUs at the local and remote stations.

If the destruction of the transmit state variables is not desired, the value of time T3 may be set to infinity. Note, if the receive variable lifetime parameter, T2 is set to infinity at remote stations to which ACn commands are sent, then the T3 parameter shall be set to infinity at the local station.

The unit is seconds. Infinity is indicated by all bits set to 1.

See ISO/IEC 8802-2:1998, 8.6.6 *Transmit lifetime variable, T3*.

### **4.12 Interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for PRIME networks**

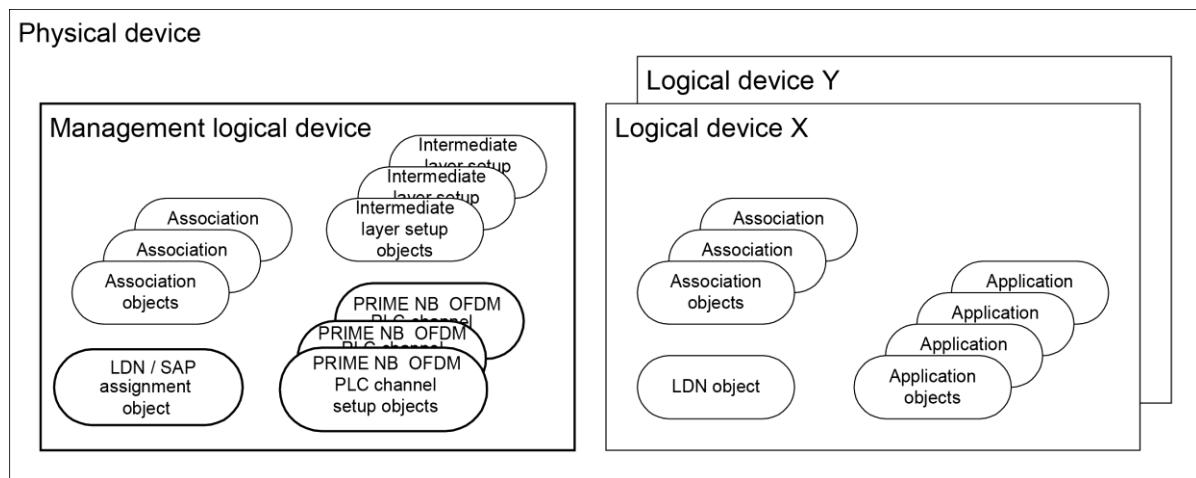
#### **4.12.1 Overview**

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 31 shows an example with a COSEM physical device comprising three logical devices.

## COSEM Interface Classes



**Figure 31 – Object model of DLMS 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.

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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 325/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

Table 41 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 41 – Mapping of PRIME NB OFDM PLC PIB attributes to COSEM IC attributes**

| Name   | Identifier | Interface class   | class_id / attribute |
|--|------------|---|----------------------|
| <b>PHY PIB attributes – PHY read-only variable that provide statistical information <sup>1</sup></b> |            |   |                      |
| phyStatsCRCIncorrectCount  | 0x00A0     | PRIME NB OFDM PLC<br>Physical layer counters<br>(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         |
| <b>PHY read-only parameters, providing information on specific implementation <sup>2</sup></b>       |            |   |                      |
| phyTxQueueLen  | 0x00B0     |   | Not modelled         |
| phyRxQueueLen  | 0x00B1     |   |                      |
| phyTxProcessingDelay   | 0x00B2     |   |                      |
| phyRxProcessingDelay   | 0x00B3     |   |                      |
| PhyAgcMinGain  | 0x00B4     |   |                      |
| PhyAgcStepValue  | 0x00B5     |   |                      |
| PhyAgcStepNumber   | 0x00B6     |   |                      |
| <b>MAC read-write variables, read-only variables <sup>3</sup></b>                                    |            |   |                      |
| macMinSwitchSearchTime   | 0x0010     | PRIME NB OFDM PLC MAC<br>setup (class_id = 82, version<br>= 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         |
| <b>MAC read-only variables that provide functional information <sup>4</sup></b>                      |            |   |                      |
| macLNID  | 0x0020     | PRIME NB OFDM PLC MAC<br>functional parameters<br>(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         |

## COSEM Interface Classes

| Name   | Identifier | Interface class  | class_id / attribute |  |
|--|------------|--|----------------------|--|
| 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        |  |
| <b>MAC read-only variable that provide statistical information <sup>5</sup></b>  |            |  |                      |  |
| 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         |  |
| <b>Read-only lists, made available by MAC layer through management interface <sup>6</sup></b>  |            |  |                      |  |
| macListMcastEntries  | 0x0052     | PRIME NB OFDM PLC 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         |  |
| <b>Application PIB attributes <sup>7</sup></b>   |            |  |                      |  |
| 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         |  |
| <sup>1</sup> See ITU-T G.9904:2012 Table 10-1.<br><sup>2</sup> See ITU-T G.9904:2012 Table 10-2.<br><sup>3</sup> See ITU-T G.9904:2012 Table 10-3, 10-4.<br><sup>4</sup> See ITU-T G.9904:2012 Table 10-5.<br><sup>5</sup> See ITU-T G.9904:2012 Table 10-6.<br><sup>6</sup> See ITU-T G.9904:2012 Table 10-7.<br><sup>7</sup> See ITU-T G.9904:2012 Table 10-9. |            |  |                      |  |
| NOTE Whereas in COSEM interface class specifications the underscore notation is used, in Recommendation ITU-T G.9904:2012 and in Table 1, the camel notation is used.  |            |  |                      |  |

### 4.12.3 61334-4-32 LLC SSCS setup (class\_id = 80, version = 0)

#### 4.12.3.1 Overview

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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 327/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| 61334-4-32 LLC SSCS setup |          | 0...n            | class_id = 80, version = 0 |             |             |                   |
|---------------------------|----------|------------------|----------------------------|-------------|-------------|-------------------|
| <b>Attributes</b>         |          | <b>Data type</b> | <b>Min.</b>                | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 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          |
| <b>Specific methods</b>   |          | <b>m/o</b>       |                            |             |             |                   |
| 1. reset (data)           |          | o                |                            |             |             |                   |

### 4.12.3.2 Attribute description

#### 4.12.3.2.1 logical\_name

Identifies the “61334-4-32 LLC SSCS setup object instance”. See 6.2.27.

#### 4.12.3.2.2 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.

#### 4.12.3.2.3 base\_node\_address

Holds the value of the base node address to which the service node is registered.

After deregistration this address is 0.

### 4.12.3.3 Method description

#### 4.12.3.3.1 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)

### 4.12.5.1 Overview

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.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 328/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

| PRIME NB OFDM PLC Physical layer counters |          | 0...n            | class_id = 81, version = 0 |             |             |                   |
|---|----------|------------------|----------------------------|-------------|-------------|-------------------|
| <b>Attributes</b>                         |          | <i>Data type</i> | <i>Min.</i>                | <i>Max.</i> | <i>Def.</i> | <i>Short name</i> |
| 1. logical_name                           | (static) | octet-string     |                            |             |             | x                 |
| 2. 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          |
| <b>Specific methods</b>                   |          | <i>m/o</i>       |                            |             |             |                   |
| 1. reset (data)                           |          | o                |                            |             |             |                   |

#### 4.12.5.2 Attribute description

##### 4.12.5.2.1 logical\_name

Identifies the “PRIME NB OFDM PLC Physical layer counters” object instance. See 6.2.27.

##### 4.12.5.2.2 phy\_stats\_crc\_incorrect\_count

PIB attribute 0x00A0: Number of bursts received on the physical layer for which the CRC was incorrect.

##### 4.12.5.2.3 phy\_stats\_crc\_failed\_count

PIB attribute 0x00A1: Number of bursts received on the physical layer for which the CRC was correct, but the Protocol field of PHY header had invalid value. This count would reflect number of times corrupt data was received and the CRC calculation failed to detect it.

##### 4.12.5.2.4 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.

##### 4.12.5.2.5 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.

#### 4.12.5.3 Method description

##### 4.12.5.3.1 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)

##### 4.12.6.1 Overview

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          |                            |      |      |            |

##### 4.12.6.2 Attribute description

###### 4.12.6.2.1 logical\_name

Identifies the “PRIME NB OFDM PLC MAC setup” object instance. See 6.2.27..

###### 4.12.6.2.2 mac\_min\_switch\_search\_time

PIB attribute 0x0010: Minimum time for which a service node in *Disconnected* status should scan the channel for beacons before it can broadcast PNPDUs. This attribute is not maintained in base nodes.

The unit of this attribute is seconds.

###### 4.12.6.2.3 mac\_max\_promotion\_pdu

PIB attribute 0x0011: Maximum number of PNPDUs that may be transmitted by a service node in a period of *mac\_promotion\_pdu\_tx\_period* seconds. This attribute is not maintained in base nodes.

###### 4.12.6.2.4 mac\_promotion\_pdu\_tx\_period

PIB attribute 0x0012: Time quantum for limiting the number of PNPDUs transmitted from a service node. No more than *mac\_max\_promotion\_pdu* may be transmitted in a period of *mac\_promotion\_pdu\_tx\_period*.

The unit of this attribute is seconds.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 330/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

**4.12.6.2.5 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.

**4.12.6.2.6 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.

**4.12.6.2.7 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.

**4.12.6.2.8 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.

**4.12.7 PRIME NB OFDM PLC MAC functional parameters (class\_id = 83 version = 0)****4.12.7.1 Overview**

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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 331/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| 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   |
| <b>Specific methods</b>                     | <b>m/o</b>    |                            |        |      |            |

### 4.12.7.2 Attribute description

#### 4.12.7.2.1 logical\_name

Identifies the “PRIME NB OFDM PLC MAC functional parameters” object instance. See 6.2.27.

#### 4.12.7.2.2 mac\_LNID

PIB attribute 0x0020: LNID allocated to this node at time of its registration.

#### 4.12.7.2.3 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 *Terminal* functional state.

#### 4.12.7.2.4 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.

#### 4.12.7.2.5 mac\_SNA

PIB attribute 0x0023: Subnetwork address to which this node is registered. The base node returns the SNA it is using.

**4.12.7.2.6 mac\_state**

PIB attribute 0x0024: Present functional state of the node.

```
enum:
    (0) Disconnected,
    (1) Terminal,
    (2) Switch,
    (3) Base
```

**4.12.7.2.7 mac\_scp\_length**

PIB attribute 0x0025: The SCP length, in symbols, in present frame.

**4.12.7.2.8 mac\_node\_hierarchy\_level**

PIB attribute 0x0026: Level of this node in subnetwork hierarchy.

**4.12.7.2.9 mac\_beacon\_slot\_count**

PIB attribute 0x0027: Number of beacon slots provisioned in present frame structure.

**4.12.7.2.10 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.

**4.12.7.2.11 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 *Terminal* functional state.

**4.12.7.2.12 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.

**4.12.7.2.13 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 *Terminal* functional state.

**4.12.7.2.14 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

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 333/668 |
|-----------------------|------------|-----------------------------|---------|

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)

##### 4.12.8.1 Overview

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                    |                            |      |      |            |

##### 4.12.8.2 Attribute description

###### 4.12.8.2.1 logical\_name

Identifies the “PRIME NB OFDM PLC MAC counters” object instance. See 6.2.27.

###### 4.12.8.2.2 mac\_tx\_data\_pkt\_count

PIB attribute 0x0040: Count of successfully transmitted MSDUs.

###### 4.12.8.2.3 mac\_rx\_data\_pkt\_count

PIB attribute 0x0041: Count of successfully received MSDUs whose destination address was this node.

###### 4.12.8.2.4 mac\_tx\_ctrl\_pkt\_count

PIB attribute 0x0042: Count of successfully transmitted MAC control packets.

**4.12.8.2.5 mac\_rx\_ctrl\_pkt\_count**

PIB attribute 0x0043: Count of successfully received MAC control packets whose destination was this node.

**4.12.8.2.6 mac\_csma\_fail\_count**

PIB attribute 0x0044: Count of failed CSMA transmit attempts.

**4.12.8.2.7 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.

**4.12.8.3 Method description****4.12.8.3.1 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)****4.12.9.1 Overview**

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            |                            |      |      |            |  |

#### 4.12.9.2 Attribute description

##### 4.12.9.2.1 logical\_name

Identifies the “PRIME NB OFDM PLC MAC network administration data” object instance. See 6.2.27.

##### 4.12.9.2.2 mac\_list\_multicast\_entries

PIB attribute 0x0052: List of entries in multicast switching table. This list is not maintained in service nodes in a *Terminal* functional state.

```
mac_list_multicast_entries_type ::= array mac_list_multicast_entries_element
mac_list_multicast_entries_element ::= structure
{
    mcast_entry_LCID:           integer, -- LCID of multicast group
    mcast_entry_members:        long     -- number of child nodes
}
```

The number of child nodes is the number of the members of this group, including the Node itself.

##### 4.12.9.2.3 mac\_list\_switch\_table

PIB attribute 0x0053: Switch table. This table is not maintained by service nodes in a *Terminal* state.

```
mac switch_table ::= array    stbl_entry_LSID
stbl_entry_LSID ::=          long   -- SID of attached Switch node
```

##### 4.12.9.2.4 mac\_list\_direct\_table

PIB attribute 0x0055: Direct table.

```
mac_direct_table ::= array    mac_direct_table_element
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)
}
```

Where:

- dconn\_entry\_src\_SID is the SID of switch through which the source service node is connected;
- dconn\_entry\_src\_LNID is the NID allocated to the source service node;
- dconn\_entry\_src\_LCID is the LCID allocated to this connection at the source;

- dconn\_entry\_dst\_SID is the SID of the switch through which the destination service node is connected;
- dconn\_entry\_dst\_LNID is the NID allocated to the destination service node;
- dconn\_entry\_dst\_LCID is the LCID allocated to this connection at the destination;
- dconn\_entry\_DID is the EUI-48 of the direct switch.

#### **4.12.9.2.5 mac\_list\_available\_switches**

PIB attribute 0x0056: List of switch nodes whose beacons are received.

```
mac_list_available_switches ::= array mac_list_available_switches_element
mac_list_available_switches_element ::= structure
{
    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
}
```

Where:

- slist\_entry\_SNA is EUI-48 of the subnetwork;
- slist\_entry\_LSID is SID of this switch;
- slist\_entry\_level is level of this switch in subnetwork hierarchy;
- slist\_entry\_rx\_level is the received signal level for this Switch;
- slist\_entry\_rx\_snr is the signal to noise ratio for this switch.

#### **4.12.9.2.6 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.

```
mac_list_phy_comm ::= array phy_comm_element
phy_comm_element ::= structure
{
    phy_Comm_EUI:            octet-string,
    phy_Comm_Tx_Pwr:         integer,
    phy_Comm_Tx_Cod:         integer,
    phy_Comm_Rx_Cod:         integer,
    phy_Comm_Rx_Lvl:         integer,
    phy_Comm_SNR:            integer,
    phy_Comm_Tx_Pwr_Mod:     integer,
    phy_Comm_Tx_Cod_Mod:     integer,
    phy_Comm_Rx_Cod_Mod:     integer
}
```

Where:

- phy\_Comm\_EUI is the EUI-48 of the other device;
- phy\_Comm\_Tx\_Pwr is the Tx power of GPDUs sent to the device;
- phy\_Comm\_Tx\_Cod is the Tx coding of GPDUs sent to the device;

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 337/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

- 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.

### **4.12.9.3 Method description**

#### **4.12.9.3.1 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 6.2.27.

### **4.12.11 PRIME NB OFDM PLC Application identification (class\_id = 86, version = 0)**

#### **4.12.11.1 Overview**

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<br>(static)                  | octet-string  |                            |      |      | x          |  |
| 2. firmware_version<br>(static)              | octet-string  |                            | 128  |      | x + 0x08   |  |
| 3. vendor_Id<br>(static)                     | long-unsigned |                            |      |      | x + 0x10   |  |
| 4. product_Id<br>(static)                    | long-unsigned |                            |      |      | x + 0x18   |  |
| Specific methods                             | m/o           |                            |      |      |            |  |

#### **4.12.11.2 Attribute description**

##### **4.12.11.2.1 logical\_name**

Identifies the device setup object instance. See 6.2.27.

##### **4.12.11.2.2 firmware\_version**

PIB attribute 0x0075: Textual description of the firmware version running on the device.

**4.12.11.2.3 vendor\_Id**

PIB attribute 0x0076: Unique vendor identifier assigned by PRIME Alliance.

**4.12.11.2.4 product\_Id**

PIB attribute 0x0077: Vendor assigned unique identifier for specific product.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 339/668 |
|-----------------------|------------|-----------------------------|---------|

## 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 Amd. 1:2021.

**NOTE 1** Previous versions of these interface classes based on previous versions of ITU-T G.9903:2014 – see 5.13 – are 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 PLC profile layers (including PHY, IEEE 802.15.4: 2006 MAC and 6LoWPAN) and Hybrid PLC & RF profile layers (IEEE 802.15.4:2020 RF PHY, IEEE 802.15.4:2015 RF MAC and additional 6LoWPAN features), six 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.
- “G3-PLC Hybrid RF MAC layer counters”, see 4.13.6
- “G3-PLC Hybrid RF MAC setup”, see 4.13.7
- “G3-PLC Hybrid 6LoWPAN adaptation layer setup”, see 4.13.8.

A G3-PLC device implementing the PLC profile layers shall implement: “G3-PLC MAC layer counters”, “G3-PLC MAC setup”, and “G3-PLC 6LoWPAN adaptation layer setup”.

A G3-PLC device implementing the Hybrid PLC & RF profile layers shall implement: “G3-PLC MAC layer counters”, “G3-PLC MAC setup”, “G3-PLC 6LoWPAN adaptation layer setup”, “G3-PLC Hybrid RF MAC layer counters”, “G3-PLC Hybrid RF MAC setup” and “G3-PLC Hybrid 6LoWPAN adaptation layer setup”.

An instance of the existing COSEM interface class “MAC address” (class\_id = 43, version = 0) is needed to indicate the EUI-48 MAC address of the G3-PLC modem (corresponding to aExtendedAddress constant in IEEE 802.15.4: 2006 ).

IPv6 configuration is provided by an instance of “IPv6 setup” class.

**NOTE 2** The PHY layer of ITU-T G.9903 Amd. 1:2021 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: 2006 and 802.15.4:2015 (for Hybrid PLC & RF devices), both a meter (PAN device) and a data concentrator (PAN coordinator) / Neighbourhood Network Access Point (NNAP) are Full Function Devices (FFD). In terms of DLMS/COSEM, the meter is the server and the data 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.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 340/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

Table 42 shows the mapping of G3-PLC PIB attributes to attributes of COSEM interface classes.

**Table 42 – Mapping of G3-PLC IB attributes to COSEM IC attributes**

| Name   | Identifier | Interface class  | class_id / attribute |
|--|------------|--|----------------------|
| <b>MAC counters – Read only PIB attributes that provide statistic information <sup>1</sup></b>     |            |  |                      |
| mac_Tx_data_packet_count   | 0x0101     | G3-PLC MAC layer counters (class_id = 90, version = 1<br>(4.13.3)) | 90 / 2               |
| mac_Rx_data_packet_count   | 0x0102     |  | 90 / 3               |
| mac_Tx_cmd_packet_count  | 0x0103     |  | 90 / 4               |
| mac_Rx_cmd_packet_count  | 0x0104     |  | 90 / 5               |
| mac_CSMA_fail_count  | 0x0105     |  | 90 / 6               |
| mac_CSMA_no_ACK_count  | 0x0106     |  | 90 / 7               |
| mac_bad_CRC_count  | 0x0109     |  | 90 / 8               |
| mac_Tx_data_broadcast_count  | 0x0108     |  | 90 / 9               |
| mac_Rx_data_broadcast_count  | 0x0107     |  | 90 / 10              |
| <b>MAC setup PIB attributes – Read only and read-write and write only variables <sup>1,2</sup></b> |            |  |                      |
| mac_short_address  | 0x0053     | G3-PLC MAC setup (class_id = 91, version = 3<br>(4.13.4))          | 91 / 2               |
| mac_RC_coord   | 0x010F     |  | 91 / 3               |
| mac_PAN_id   | 0x0050     |  | 91 / 4               |
| mac_key_table  | 0x0071     |  | 91 / 5               |
| mac_frame_counter  | 0x0077     |  | 91 / 6               |
| mac_tone_mask  | 0x0110     |  | 91 / 7               |
| mac_TMR_TTL  | 0x010D     |  | 91 / 8               |
| mac_max_frame_retries  | 0x0059     |  | 91 / 9               |
| mac_POS_table_entry_TTL  | 0x010E     |  | 91 / 10              |
| mac_neighbour_table  | 0x010A     |  | 91 / 11              |
| mac_high_priority_window_size  | 0x0100     |  | 91 / 12              |
| mac_CSMA_fairness_limit  | 0x010C     |  | 91 / 13              |
| mac_beacon_randomization_window_length   | 0x0111     |  | 91 / 14              |
| mac_A  | 0x0112     |  | 91 / 15              |
| mac_K  | 0x0113     |  | 91 / 16              |
| mac_min_CW_attempts  | 0x0114     |  | 91 / 17              |
| mac_cenelec_legacy_mode  | 0x0115     |  | 91 / 18              |
| mac_FCC_legacy_mode  | 0x0116     |  | 91 / 19              |
| mac_max_BE   | 0x0047     |  | 91 / 20              |
| mac_max_CSMA_backoffs  | 0x004E     |  | 91 / 21              |
| mac_min_BE   | 0x004F     |  | 91 / 22              |
| mac_broadcast_max_CW_enabled   | 0x011E     |  | 91 / 23              |
| mac_transmit_atten   | 0x011F     |  | 91 / 24              |
| mac_POS_table  | 0x0120     |  | 91 / 25              |
| mac_duplicate_detection_TTL  | 0x0078     |  | 91 / 26              |
| <b>6LoWPAN adaptation layer IB attributes – Read only and read-write variables <sup>3,4</sup></b>  |            |  |                      |

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 341/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| Name                              | Identifier  | Interface class   | class_id / attribute |
|-----------------------------------|---|---|----------------------|
| adp_max_hops                      | 0x0F  |   | 92 / 2               |
| adp_weak_LQI_value                | 0x1A  |   | 92 / 3               |
| adp_security_level                | 0x00  |   | 92 / 4               |
| adp_prefix_table                  | 0x01  |   | 92 / 5               |
| adp_routing_configuration         | 0x06,<br>0x09,<br>0x0A,<br>0x0D,<br>0x11-<br>0x19,<br>0x1B,<br>0x1F |   | 92 / 6               |
| adp_broadcast_log_table_entry_TTL | 0x02  |   | 92 / 7               |
| adp_routing_table <sup>5</sup>    | 0x0C  |   | 92 / 8               |
| adp_context_information_table     | 0x07  |   | 92 / 9               |
| adp_blacklist_table <sup>5</sup>  | 0x1E  | G3-PLC 6LoWPAN adaptation layer setup<br>(class_id = 92, version = 3) | 92 / 10              |
| adp_broadcast_log_table           | 0x0B  | (4.13.5)  | 92 / 11              |
| adp_group_table                   | 0x0E  |   | 92 / 12              |
| adp_max_join_wait_time            | 0x20  |   | 92 / 13              |
| adp_path_discovery_time           | 0x21  |   | 92 / 14              |
| adp_active_key_index              | 0x22  |   | 92 / 15              |
| adp_metric_type                   | 0x03  |   | 92 / 16              |
| adp_coord_short_address           | 0x08  |   | 92 / 17              |
| adp_disable_default_routing       | 0xF0  |   | 92 / 18              |
| adp_device_type                   | 0x10  |   | 92 / 19              |
| adp_default_coord_route_enabled   | 0x24  |   | 92 / 20              |
| adp_destination_address_set       | 0x23  |   | 92 / 21              |
| adp_low_LQI_value                 | 0x04  |   | 92 / 22              |
| adp_high_LQI_value                | 0x05  |   | 92 / 23              |

<sup>1</sup> See ITU-T G.9903 Amd. 1:2021 (05/2021), 9.3.6.2.2 and 9.3.6.2.3.  
<sup>2</sup> The following attributes of the G3-PLC MAC sublayer IB attributes have been excluded as there is no need to expose them: *macBSN*, *macDSN*, *macTimeStampSupported*, *macPromiscuousMode*, *macSecurityEnabled*.  
<sup>3</sup> See ITU-T G.9903 Amd. 1:2021 (05/2021), 9.4.1.1.  
<sup>4</sup> The following attributes of the G3-PLC Adaptation sublayer IB attributes have been excluded as there is no need to expose them; *adpSoftVersion*, *adpSnifferMode*.  
<sup>5</sup> adp\_routing\_table and adp\_blacklist\_table as defined in the 6LoWPAN adaptation layer setup (class\_id = 92, version = 3) shall exclusively be used with the G3-PLC PLC profile.

**NOTE** Whereas in ITU-T G.9903 Amd. 1:2021 (05/2021) the camel-case notation is used, in COSEM interface class specifications – and in this table – the underscore notation is used.

Table 43 shows the mapping of additional PIB attributes related to the Hybrid PLC & RF profile to attributes of COSEM interface classes. Indeed, attributes from both Table 42 and Table 43 shall be considered for the G3-PLC Hybrid PLC & RF profile.

## COSEM Interface Classes

**Table 43 – Mapping of G3-PLC Hybrid PLC & RF IB attributes to COSEM IC attributes**

| Name   | Identifier | Interface class  | class_id / attribute |
|--|------------|--|----------------------|
| <b>Hybrid RF MAC counters – Read only PIB attributes that provide statistic information <sup>1,2</sup></b>   |            |  |                      |
| mac_retry_count_RF   | 0x20A      | G3-PLC Hybrid RF MAC layer counters<br>(class_id = 160, version = 0)<br><br>(4.13.6)       | 160 / 2              |
| mac_multiple_retry_count_RF  | 0x20B      |  | 160 / 3              |
| mac_Tx_fail_count_RF   | 0x20C      |  | 160 / 4              |
| mac_Tx_success_count_RF  | 0x20D      |  | 160 / 5              |
| mac_fcs_error_count_RF   | 0x20E      |  | 160 / 6              |
| mac_security_failure_RF  | 0x20F      |  | 160 / 7              |
| mac_duplicate_frame_count_RF   | 0x210      |  | 160 / 8              |
| mac_Rx_success_count_RF  | 0x211      |  | 160 / 9              |
| mac_nack_count_RF  | 0x212      |  | 160 / 10             |
| <b>Hybrid RF MAC setup PIB attributes – Read only and read-write and write only variables <sup>2,3</sup></b> |            |  |                      |
| mac_max_BE_RF  | 0x201      | G3-PLC Hybrid RF MAC setup<br>(class_id = 161, version = 0)<br><br>(4.13.7)                | 161 / 2              |
| mac_max_CSMA_backoffs_RF   | 0x202      |  | 161 / 3              |
| mac_max_frame_retries_RF   | 0x203      |  | 161 / 4              |
| mac_min_BE_RF  | 0x204      |  | 161 / 5              |
| mac_frame_counter_RF   | 0x207      |  | 161 / 6              |
| mac_duplicate_detection_TTL_RF   | 0x208      |  | 161 / 7              |
| mac_POS_table_RF   | 0x21D      |  | 161 / 8              |
| mac_operating_mode_RF  | 0x21E      |  | 161 / 9              |
| mac_channel_number_RF  | 0x21F      |  | 161 / 10             |
| mac_duty_cycle_usage_RF  | 0x220      |  | 161 / 11             |
| mac_duty_cycle_period_RF   | 0x221      |  | 161 / 12             |
| mac_duty_cycle_limit_RF  | 0x222      |  | 161 / 13             |
| mac_duty_cycle_threshold_RF  | 0x223      |  | 161 / 14             |
| mac_disable_PHY_RF   | 0x224      |  | 161 / 15             |
| <b>Hybrid 6LoWPAN adaptation layer IB attributes – Read only and read-write variables <sup>4</sup></b>       |            |  |                      |
| adp_routing_table <sup>5</sup>   | 0x0C       | G3-PLC Hybrid 6LoWPAN adaptation layer setup (class_id = 162, version = 0)<br><br>(4.13.8) | 162 / 2              |
| adp_blacklist_table <sup>5</sup>   | 0x1E       |  | 162 / 3              |
| adp_low_LQI_value_RF   | 0xD0       |  | 162 / 4              |
| adp_high_LQI_value_RF  | 0xD1       |  | 162 / 5              |
| adp_routing_configuration_RF   | 0xD2-0xD5  |  | 162 / 6              |
| adp_use_backup_media   | 0xD6       |  | 162 / 7              |

<sup>1</sup> See ITU-T G.9903 Amd. 1:2021, H.6.3.

<sup>2</sup> The following attributes of the G3-PLC Hybrid RF MAC sublayer IB attributes have been excluded as there is no need to expose them: *macEbsn\_RF*, *macDsn\_RF*, *macTimeStampSupported\_RF*, *macDeviceTable\_RF*, *macCounterOctets\_RF*.

<sup>3</sup> See ITU-T G.9903 Amd. 1:2021, H.6.3 and H.6.5.2.

<sup>4</sup> See ITU-T G.9903 Amd. 1:2021, H.8.1.

<sup>5</sup> adp\_routing\_table and adp\_blacklist\_table as defined in the Hybrid 6LoWPAN adaptation layer setup (class\_id = 162, version = 0) shall be exclusively used with the G3-PLC Hybrid PLC & RF profile.

## COSEM Interface Classes

| Name   | Identifier | Interface class | class_id / attribute |
|--|------------|-----------------|----------------------|
| NOTE Whereas in ITU-T G.9903 Amd. 1:2021 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)

#### 4.13.3.1 Overview

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   |

#### 4.13.3.2 Attribute description

NOTE When a counter reaches the maximum value (0xFFFFFFFF), it's automatically rolled-over.

##### 4.13.3.2.1 logical\_name

Identifies the “G3-PLC MAC layer counters” object instance. See 6.2.28.

##### 4.13.3.2.2 mac\_Tx\_data\_packet\_count

PIB attribute 0x0101: Statistic counter of successfully transmitted data packets (MSDUs).

##### 4.13.3.2.3 mac\_Rx\_data\_packet\_count

PIB attribute 0x0102: Statistic counter of successfully received data packets (MSDUs).

**4.13.3.2.4 mac\_Tx\_cmd\_packet\_count**

PIB attribute 0x0103: Statistic counter of successfully transmitted command packets.

**4.13.3.2.5 mac\_Rx\_cmd\_packet\_count**

PIB attribute 0x0104: Statistic counter of successfully received command packets.

**4.13.3.2.6 mac\_CSMA\_fail\_count**

PIB attribute 0x0105: Counts the number of times when CSMA backoffs reach macMaxCSMABackoffs.

**4.13.3.2.7 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).

**4.13.3.2.8 mac\_bad\_CRC\_count**

PIB attribute 0x0109: Statistic counter of the number of frames received with bad CRC.

**4.13.3.2.9 mac\_Tx\_data\_broadcast\_count**

PIB attribute 0x0108: Statistic counter of the number of broadcast frames sent.

**4.13.3.2.10 mac\_Rx\_data\_broadcast\_count**

PIB attribute 0x0107: Statistic counter of successfully received broadcast packets.

**4.13.3 Method description****4.13.3.1 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 = 3)****4.13.4.1 Overview**

An instance of the “G3-PLC MAC setup” IC holds the necessary parameters to set up and manage the G3-PLC IEEE 802.15.4: 2006 MAC sub-layer.

These attributes influence the functional behaviour of an implementation. Implementations may allow changes to the attributes during normal running, i.e. even after the device start-up sequence has been executed.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 345/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| G3-PLC MAC setup                           |          | 0...n                | class_id = 91, version = 3 |                  |                                 |            |
|--|----------|----------------------|----------------------------|------------------|---------------------------------|------------|
| 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<br>967 295 | 0                               | x + 0x28   |
| 7. mac_tone_mask                           | (static) | bit-string           |                            |                  | 0x00000<br>0000FF<br>FFFFF<br>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_POS_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<br>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<br>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   |
| 23. mac_broadcast_max_CW_enabled           | (static) | boolean              |                            |                  | FALSE                           | x + 0xB0   |
| 24. mac_transmit_atten                     | (static) | unsigned             | 0                          | 25               | 0                               | x + 0xB8   |
| 25. mac_POS_table                          | (dyn.)   | array                |                            |                  |                                 | x + 0xC0   |
| 26. mac_duplicate_detection_TTL            | (static) | unsigned             | 0                          | 255              | 3                               | x + 0xC8   |
| <b>Specific methods</b>                    |          | m/o                  |                            |                  |                                 |            |
| 1. mac_get_neighbour_table_entry (data)    |          | o                    |                            |                  |                                 |            |
| 2. mac_get_POS_table_entry (data)          |          | o                    |                            |                  |                                 |            |

### 4.13.4.2 Attribute description

#### 4.13.4.2.1 logical\_name

Identifies the “G3-PLC MAC setup” object instance. See 6.2.28.

#### 4.13.4.2.2 mac\_short\_address

PIB attribute 0x0053: The 16-bit address the device is using to communicate through the PAN. Its value shall be equal to 0xFFFF when the device does not have a short address. An

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 346/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

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.

#### 4.13.4.2.3 mac\_RC\_coord

PIB attribute 0x010F: Route cost to coordinator, to be used in the beacon payload as RC\_COORD

#### 4.13.4.2.4 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.

#### 4.13.4.2.5 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
}
```

**Where:**

- 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.

#### 4.13.4.2.6 mac\_frame\_counter

PIB attribute 0x0077: The outgoing frame counter for this device, used when ciphering frames at MAC layer.

#### 4.13.4.2.7 mac\_tone\_mask

PIB attribute 0x0110: Defines the tone mask to use during symbol formation. **Each bit controls if a given carrier is used (value = 1) or notched (value = 0).**

The bit-string is always 72 bits long, the number of meaningful bits depending on the device band-plan. The first bit in the bit-string (bit 0) represents the lowest frequency of the CENELEC-A, CENELEC-B and FCC band-plans.

The mapping of the bandplans is given in

Table 44 – .

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 347/668 |
|-----------------------|------------|-----------------------------|---------|

COSEM Interface Classes

**Table 44 – Bandplans**

| Bit number        | Carrier number | Carrier Frequency (kHz) | Carrier Frequency (kHz) | Carrier Frequency (kHz) |
|-------------------|----------------|-------------------------|-------------------------|-------------------------|
|                   |                | FCC                     | CENELEC-A               | CENELEC-B               |
| bit 0 (first bit) | 1              | 154.6875                | 35.9375                 | 98.4375                 |
| bit 1             | 2              | 159.375                 | 37.5                    | 100                     |
| bit 2             | 3              | 164.0625                | 39.0625                 | 101.5625                |
| bit 3             | 4              | 168.75                  | 40.625                  | 103.125                 |
| bit 4             | 5              | 173.4375                | 42.1875                 | 104.6875                |
| bit 5             | 6              | 178.125                 | 43.75                   | 106.25                  |
| bit 6             | 7              | 182.8125                | 45.3125                 | 107.8125                |
| bit 7             | 8              | 187.5                   | 46.875                  | 109.375                 |
| bit 8             | 9              | 192.1875                | 48.4375                 | 110.9375                |
| bit 9             | 10             | 196.875                 | 50                      | 112.5                   |
| bit 10            | 11             | 201.5625                | 51.5625                 | 114.0625                |
| bit 11            | 12             | 206.25                  | 53.125                  | 115.625                 |
| bit 12            | 13             | 210.9375                | 54.6875                 | 117.1875                |
| bit 13            | 14             | 215.625                 | 56.25                   | 118.75                  |
| bit 14            | 15             | 220.3125                | 57.8125                 | 120.3125                |
| bit 15            | 16             | 225                     | 59.375                  | 121.875                 |
| bit 16            | 17             | 229.6875                | 60.9375                 | not used                |
| bit 17            | 18             | 234.375                 | 62.5                    | not used                |
| bit 18            | 19             | 239.0625                | 64.0625                 | not used                |
| bit 19            | 20             | 243.75                  | 65.625                  | not used                |
| bit 20            | 21             | 248.4375                | 67.1875                 | not used                |
| bit 21            | 22             | 253.125                 | 68.75                   | not used                |
| bit 22            | 23             | 257.8125                | 70.3125                 | not used                |
| bit 23            | 24             | 262.5                   | 71.875                  | not used                |
| bit 24            | 25             | 267.1875                | 73.4375                 | not used                |
| bit 25            | 26             | 271.875                 | 75                      | not used                |
| bit 26            | 27             | 276.5625                | 76.5625                 | not used                |
| bit 27            | 28             | 281.25                  | 78.125                  | not used                |
| bit 28            | 29             | 285.9375                | 79.6875                 | not used                |
| bit 29            | 30             | 290.625                 | 81.25                   | not used                |
| bit 30            | 31             | 295.3125                | 82.8125                 | not used                |
| bit 31            | 32             | 300                     | 84.375                  | not used                |
| bit 32            | 33             | 304.6875                | 85.9375                 | not used                |
| bit 33            | 34             | 309.375                 | 87.5                    | not used                |
| bit 34            | 35             | 314.0625                | 89.0625                 | not used                |
| bit 35            | 36             | 318.75                  | 90.625                  | not used                |
| bit 36            | 37             | 323.4375                | not used                | not used                |
| bit 37            | 38             | 328.125                 | not used                | not used                |
| bit 38            | 39             | 332.8125                | not used                | not used                |

## COSEM Interface Classes

| Bit number        | Carrier number | Carrier Frequency (kHz) | Carrier Frequency (kHz) | Carrier Frequency (kHz) |
|-------------------|----------------|-------------------------|-------------------------|-------------------------|
|                   |                | FCC                     | CENELEC-A               | CENELEC-B               |
| bit 39            | 40             | 337.5                   | not used                | not used                |
| bit 40            | 41             | 342.1875                | not used                | not used                |
| bit 41            | 42             | 346.875                 | not used                | not used                |
| bit 42            | 43             | 351.5625                | not used                | not used                |
| bit 43            | 44             | 356.25                  | not used                | not used                |
| bit 44            | 45             | 360.9375                | not used                | not used                |
| bit 45            | 46             | 365.625                 | not used                | not used                |
| bit 46            | 47             | 370.3125                | not used                | not used                |
| bit 47            | 48             | 375                     | not used                | not used                |
| bit 48            | 49             | 379.6875                | not used                | not used                |
| bit 49            | 50             | 384.375                 | not used                | not used                |
| bit 50            | 51             | 389.0625                | not used                | not used                |
| bit 51            | 52             | 393.75                  | not used                | not used                |
| bit 52            | 53             | 398.4375                | not used                | not used                |
| bit 53            | 54             | 403.125                 | not used                | not used                |
| bit 54            | 55             | 407.8125                | not used                | not used                |
| bit 55            | 56             | 412.5                   | not used                | not used                |
| bit 56            | 57             | 417.1875                | not used                | not used                |
| bit 57            | 58             | 421.875                 | not used                | not used                |
| bit 58            | 59             | 426.5625                | not used                | not used                |
| bit 59            | 60             | 431.25                  | not used                | not used                |
| bit 60            | 61             | 435.9375                | not used                | not used                |
| bit 61            | 62             | 440.625                 | not used                | not used                |
| bit 62            | 63             | 445.3125                | not used                | not used                |
| bit 63            | 64             | 450                     | not used                | not used                |
| bit 64            | 65             | 454.6875                | not used                | not used                |
| bit 65            | 66             | 459.375                 | not used                | not used                |
| bit 66            | 67             | 464.0625                | not used                | not used                |
| bit 67            | 68             | 468.75                  | not used                | not used                |
| bit 68            | 69             | 473.4375                | not used                | not used                |
| bit 69            | 70             | 478.125                 | not used                | not used                |
| bit 70            | 71             | 482.8125                | not used                | not used                |
| bit 71 (last bit) | 72             | 487.5                   | not used                | not used                |

### 4.13.4.2.8 mac\_TMR\_TTL

PIB attribute 0x010D: Maximum time to live of tone map parameters entry in the neighbour table in minutes.

**NOTE** This attribute was used with different definitions in earlier versions of the Interface Class. See 5.13.6.2.8 and 5.13.5.2.8.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 349/668 |
|-----------------------|------------|-----------------------------|---------|

**4.13.4.2.9 mac\_max\_frame\_retries**

PIB attribute 0x0059: Maximum number of retransmissions.

**4.13.4.2.10 mac\_POS\_table\_entry\_TTL**

PIB attribute 0x010E: Maximum time to live for an entry in the neighbour table in minutes.

**NOTE** This attribute was used with different name and different definitions in earlier versions of the Interface Class. See 5.13.6.2.10 and 5.13.5.2.10.

**4.13.4.2.11 mac\_neighbour\_table**

PIB attribute 0x010A: See [ITU-T G.9903 Amd. 1:2021, \(9.3.7.2.2\)](#) for CENELEC and FCC bands.

Every device shall maintain a "neighbour table" which contains information about all the devices with which a unicast message has been exchanged. Each time a tone map response command is received the table is updated. In case the table is full, the entry corresponding to the shortest valid time is removed. This table shall be accessible by the adaptation, MAC sublayers and physical layer.

```
array neighbour_table

neighbour_table ::= structure

{
    short_address: long-unsigned,
    payload_modulation_scheme: boolean,
    tone_map: bit-string,
    modulation_type: enum,
    tx_gain: integer,
    tx_res: enum,
    tx_coeff: bit-string,
    reverse_lqi: unsigned,
    phase_differential: integer,
    TMR_valid_time: unsigned,
    no_data: unsigned
}
```

**NOTE** This table is actualized each time any frame is received from a neighbour device, and each time a Tone Map Response is received.

Where:

- short\_address                          The MAC Short Address of the node which this entry refers to.
- payload\_modulation\_scheme            Payload Modulation scheme to be used when transmitting to this neighbour.  

0: Differential,  
1: Coherent
- tone\_map                                The tone map to be used when transmitting to this neighbour

**NOTE** 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.

## COSEM Interface Classes

The first bit in the bit-string (bit 0) corresponds to Tone Map bit TM[0].

In CENELEC-A bandplan, each bit of the tone map field is associated with one sub-band consisting of a group of 6 tones.

| <b>Bit number</b> | <b>TM field</b> | <b>Sub-band (kHz)</b> |
|-------------------|-----------------|-----------------------|
| 0                 | TM[0]           | 35.9375 to 43.75      |
| 1                 | TM[1]           | 45.3125 to 53.125     |
| 2                 | TM[2]           | 54.6875 to 62.5       |
| 3                 | TM[3]           | 64.0625 to 71.875     |
| 4                 | TM[4]           | 73.4375 to 81.25      |
| 5                 | TM[5]           | 82.8125 to 90.625     |

In CENELEC-B bandplan, each bit of the tone map field is associated with one sub-band consisting of a group of 4 tones.

| <b>Bit number</b> | <b>TM field</b> | <b>Sub-band (kHz)</b> |
|-------------------|-----------------|-----------------------|
| 0                 | TM[0]           | 98.4375 to 103.125    |
| 1                 | TM[1]           | 104.6875 to 109.375   |
| 2                 | TM[2]           | 110.9375 to 115.625   |
| 3                 | TM[3]           | 117.1875 to 121.875   |

In FCC bandplan, each bit of the tone map field is associated with one sub-band consisting of a group of 3 tones.

| <b>Bit number</b> | <b>TM field</b> | <b>Sub-band (kHz)</b> |
|-------------------|-----------------|-----------------------|
| 0                 | TM[0]           | 154.6875 to 164.0625  |
| 1                 | TM[1]           | 168.75 to 178.125     |
| 2                 | TM[2]           | 182.8125 to 192.1875  |
| 3                 | TM[3]           | 196.875 to 206.25     |
| 4                 | TM[4]           | 210.9375 to 220.3125  |
| 5                 | TM[5]           | 225 to 234.375        |
| 6                 | TM[6]           | 239.0625 to 248.4375  |
| 7                 | TM[7]           | 253.125 to 262.5      |
| 8                 | TM[8]           | 267.1875 to 276.5625  |
| 9                 | TM[9]           | 281.25 to 290.625     |
| 10                | TM[10]          | 295.3125 to 304.6875  |
| 11                | TM[11]          | 309.375 to 318.75     |
| 12                | TM[12]          | 323.4375 to 332.8125  |
| 13                | TM[13]          | 337.5 to 346.875      |
| 14                | TM[14]          | 351.5625 to 360.9375  |
| 15                | TM[15]          | 365.625 to 375        |
| 16                | TM[16]          | 379.6875 to 389.0625  |
| 17                | TM[17]          | 393.75 to 403.125     |
| 18                | TM[18]          | 407.8125 to 417.1875  |

## COSEM Interface Classes

| Bit number | TM field | Sub-band (kHz)       |
|------------|----------|----------------------|
| 19         | TM[19]   | 421.875 to 431.25    |
| 20         | TM[20]   | 435.9375 to 445.3125 |
| 21         | TM[21]   | 450 to 459.375       |
| 22         | TM[22]   | 464.0625 to 473.4375 |
| 23         | TM[23]   | 478.125 to 487.5     |

- Modulation\_type      The modulation type to use for communicating with the device.  
 enum:  
   (0) Robust Mode,  
   (1) DBPSK,  
   (2) DQPSK,  
   (3) D8PSK,  
   (4) 16-QAM  
 NOTE The 16-QAM modulation is optional and only applicable for FCC band.
- tx\_gain      Tx Gain to be used when transmitting to this neighbour.
- 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.

NOTE 1 One group of tones gathers 6 consecutive tones (or carriers) for CENELEC-A band, 4 for CENELEC-B band and 3 consecutive tones for FCC band.

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.

The first bit in the bit-string (bit 0) corresponds to TXCOEF[0] and the last bit in the bit-string (bit 23) corresponds to TXCOEF[23], noting that the bits are ordered in groups as defined in table 9-21.

The first bit in the bit-string (bit 0) corresponds to TXCOEF[0] and the last bit in the bit-string (bit 47) corresponds to TXCOEF[47], noting that the bits are ordered in groups as defined in table 9-22.

## COSEM Interface Classes

NOTE 2 A description of tx\_coeff is included in ITU-T G.9903 Amd. 1:2021 tables 9-21 and 9-22.

In CENELEC-A bandplan, each group of 4 bits of the TXCOEF field is associated with one sub-band consisting of a group of 6 tones.

| Bit number | TXCOEF field | Sub-band (kHz)    |
|------------|--------------|-------------------|
| 0          | TXCOEF[0]    | 35.9375 to 43.75  |
| 1          | TXCOEF[1]    |                   |
| 2          | TXCOEF [2]   |                   |
| 3          | TXCOEF [3]   |                   |
| 4          | TXCOEF [4]   | 45.3125 to 53.125 |
| 5          | TXCOEF [5]   |                   |
| 6          | TXCOEF[6]    |                   |
| 7          | TXCOEF[7]    |                   |
| 8          | TXCOEF[8]    | 54.6875 to 62.5   |
| 9          | TXCOEF[9]    |                   |
| 10         | TXCOEF[10]   |                   |
| 11         | TXCOEF[11]   |                   |
| 12         | TXCOEF[12]   | 64.0625 to 71.875 |
| 13         | TXCOEF[13]   |                   |
| 14         | TXCOEF[14]   |                   |
| 15         | TXCOEF[15]   |                   |
| 16         | TXCOEF[16]   | 73.4375 to 81.25  |
| 17         | TXCOEF[17]   |                   |
| 18         | TXCOEF[18]   |                   |
| 19         | TXCOEF[19]   |                   |
| 20         | TXCOEF[20]   | 82.8125 to 90.625 |
| 21         | TXCOEF[21]   |                   |
| 22         | TXCOEF[22]   |                   |
| 23         | TXCOEF[23]   |                   |

In CENELEC-B bandplan, each group of 4 bits of the TXCOEF field is associated with one sub-band consisting of a group of 4 tones.

| Bit number | TXCOEF field | Sub-band (kHz)      |
|------------|--------------|---------------------|
| 0          | TXCOEF[0]    | 98.4375 to 103.125  |
| 1          | TXCOEF[1]    |                     |
| 2          | TXCOEF [2]   |                     |
| 3          | TXCOEF [3]   |                     |
| 4          | TXCOEF [4]   | 104.6875 to 109.375 |
| 5          | TXCOEF [5]   |                     |
| 6          | TXCOEF[6]    |                     |
| 7          | TXCOEF[7]    |                     |
| 8          | TXCOEF[8]    | 110.9375 to 115.625 |
| 9          | TXCOEF[9]    |                     |

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 353/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| Bit number | TXCOEF field | Sub-band (kHz)      |
|------------|--------------|---------------------|
| 10         | TXCOEF[10]   |                     |
| 11         | TXCOEF[11]   |                     |
| 12         | TXCOEF[12]   |                     |
| 13         | TXCOEF[13]   |                     |
| 14         | TXCOEF[14]   | 117.1875 to 121.875 |
| 15         | TXCOEF[15]   |                     |

In FCC bandplan, each group of 2 bits of the TXCOEF field is associated with one sub-band consisting of a group of 3 tones.

| Bit number | TXCOEF field | Sub-band (kHz)       |
|------------|--------------|----------------------|
| 0          | TXCOEF[0]    |                      |
| 1          | TXCOEF[1]    | 154.6875 to 164.0625 |
| 2          | TXCOEF[2]    |                      |
| 3          | TXCOEF[3]    | 168.75 to 178.125    |
| 4          | TXCOEF[4]    |                      |
| 5          | TXCOEF[5]    | 182.8125 to 192.1875 |
| 6          | TXCOEF[6]    |                      |
| 7          | TXCOEF[7]    | 196.875 to 206.25    |
| 8          | TXCOEF[8]    |                      |
| 9          | TXCOEF[9]    | 210.9375 to 220.3125 |
| 10         | TXCOEF[10]   |                      |
| 11         | TXCOEF[11]   | 225 to 234.375       |
| 12         | TXCOEF[12]   |                      |
| 13         | TXCOEF[13]   | 239.0625 to 248.4375 |
| 14         | TXCOEF[14]   |                      |
| 15         | TXCOEF[15]   | 253.125 to 262.5     |
| 16         | TXCOEF[16]   |                      |
| 17         | TXCOEF[17]   | 267.1875 to 276.5625 |
| 18         | TXCOEF[18]   |                      |
| 19         | TXCOEF[19]   | 281.25 to 290.625    |
| 20         | TXCOEF[20]   |                      |
| 21         | TXCOEF[21]   | 295.3125 to 304.6875 |
| 22         | TXCOEF[22]   |                      |
| 23         | TXCOEF[23]   | 309.375 to 318.75    |
| 24         | TXCOEF[24]   |                      |
| 25         | TXCOEF[25]   | 323.4375 to 332.8125 |
| 26         | TXCOEF[26]   |                      |
| 27         | TXCOEF[27]   | 337.5 to 346.875     |
| 28         | TXCOEF[28]   |                      |
| 29         | TXCOEF[29]   | 351.5625 to 360.9375 |
| 30         | TXCOEF[30]   | 365.625 to 375       |

## COSEM Interface Classes

| Bit number | TXCOEF field | Sub-band (kHz)       |
|------------|--------------|----------------------|
| 31         | TXCOEF[31]   |                      |
| 32         | TXCOEF[32]   | 379.6875 to 389.0625 |
| 33         | TXCOEF[33]   |                      |
| 34         | TXCOEF[34]   | 393.75 to 403.125    |
| 35         | TXCOEF[35]   |                      |
| 36         | TXCOEF[36]   | 407.8125 to 417.1875 |
| 37         | TXCOEF[37]   |                      |
| 38         | TXCOEF[38]   | 421.875 to 431.25    |
| 39         | TXCOEF[39]   |                      |
| 40         | TXCOEF[40]   | 435.9375 to 445.3125 |
| 41         | TXCOEF[41]   |                      |
| 42         | TXCOEF[42]   | 450 to 459.375       |
| 43         | TXCOEF[43]   |                      |
| 44         | TXCOEF[44]   | 464.0625 to 473.4375 |
| 45         | TXCOEF[45]   |                      |
| 46         | TXCOEF[46]   | 478.125 to 487.5     |
| 47         | TXCOEF[47]   |                      |

- Reverse\_lqi
 

Link Quality Indicator of the link to the neighbour (reverse LQI)

NOTE 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 seven integer values between 0 and 6.
- TMR\_valid\_time
 

Remaining time in minutes until which the tone map response parameters in the neighbour table are considered valid.

  - 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.
  - When a MAC fail occurs (when mac\_max\_frame\_retries attempts have failed), this value shall be set to 0.
- no\_data
 

Value that may be used as needed for some implementations as determined by a companion specification.

### 4.13.4.2.12 mac\_high\_priority\_window\_size

PIB attribute 0x0100: The high priority contention window size in number of slots.

**4.13.4.2.13 mac\_CSMA\_fairness\_limit**

PIB attribute 0x010C: Channel access fairness limit. Specifies how many failed back-off attempts, back-off exponent is set to **macMinBE**. This attribute can take a value between  $2 \times (\text{macMaxBE} - \text{macMinBE})$  and 255.

**4.13.4.2.14 mac\_beacon\_randomization\_window\_length**

PIB attribute 0x0111: Duration time in seconds for the beacon randomization.

**4.13.4.2.15 mac\_A**

PIB attribute 0x0112: This parameter controls the adaptive CW linear decrease.

**4.13.4.2.16 mac\_K**

PIB attribute 0x0113: Rate adaptation factor for channel access fairness limit. This attribute can take a value between 1 and **macCSMAFairnessLimit**.

**4.13.4.2.17 mac\_min\_CW\_attempts**

PIB attribute 0x0114: Number of consecutive attempts while using minimum CW.

**4.13.4.2.18 mac\_cenelec\_legacy\_mode**

PIB attribute 0x0115: This read only attribute indicates the capability of the node.

0: The following configuration is used (legacy mode):

- Elementary interleaving;
- Interleaver parameters  $n_i$  and  $n_j$  are not swapped when  $I(i,j) = 0$ .

1: The following configuration is used (non legacy mode):

- Full Block interleaving;
- Interleaver parameters  $n_i$  and  $n_j$  are swapped when  $I(i,j) = 0$ .

**4.13.4.2.19 mac\_FCC\_legacy\_mode**

PIB attribute 0x0116: This read only attribute indicates the capability of the node.

0: The following configuration is used (legacy mode):

- Differential FCH modulation;
- Elementary interleaving;
- Interleaver parameters  $n_i$  and  $n_j$  are not swapped when  $I(i,j) = 0$ ;
- Single RS block.

1: The following configuration is used (non legacy mode):

- Coherent FCH modulation;
- Full Block interleaving;
- Interleaver parameters  $n_i$  and  $n_j$  are swapped when  $I(i,j) = 0$ ;
- Two RS blocks.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 356/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

**4.13.4.2.20 mac\_max\_BE**

PIB attribute 0x0047: Maximum value of backoff exponent. It should always be greater than macMinBE.

**4.13.4.2.21 mac\_max\_CSMA\_backoffs**

PIB attribute 0x004E: Maximum number of backoff attempts.

**4.13.4.2.22 mac\_min\_BE**

PIB attribute 0x004F: Minimum value of backoff exponent.

**4.13.4.2.23 mac\_broadcast\_max\_CW\_enabled**

PIB attribute 0x011E: If enabled, MAC uses maximum contention window.

**4.13.4.2.24 mac\_transmitt\_atten**

PIB attribute 0x011F: Attenuation of the output level in dB.

**4.13.4.2.25 mac\_POS\_table**

PIB attribute 0x0120: The POS table contains some information about all the devices within the POS of the device

```
array POS_table

neighbour_table ::= structure

{
    short_address:          long-unsigned,
    LQI:                   unsigned,
    POS_valid_time:         unsigned,
}
```

NOTE: Each time a message is received (filtered according to IEEE 802.15.4: 2006 , 7.5.6.2), an entry in the POS table is created or updated (if the entry is already present). In case the table is full, the entry corresponding to the shortest valid time is removed.

**4.13.4.2.25.1 short\_address**

The MAC Short Address of the node which this entry refers to.

**4.13.4.2.25.2 LQI**

Link quality indicator of the last received packet from this neighbour.

NOTE: LQI is referred to as forward\_LQI in some implementations.

**4.13.4.2.25.3 POS\_valid\_time**

Remaining time in minutes until when this entry is considered valid.

Every time an entry is created, it is set to mac\_POS\_table\_entry\_TTL. When it reaches zero, this entry is no longer valid in the table and may be removed.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 357/668 |
|-----------------------|------------|-----------------------------|---------|

**4.13.4.2.26 mac\_duplicate\_detection\_TTL**

PIB attribute 0x0078: Time a received tuple [source address + sequence-number] is retained for duplicate frame detection, in seconds.

**NOTE** This value should be greater than (1+macMaxFrameRetries)\*aEIFS.

**4.13.4.3 Method description****4.13.4.3.1 mac\_get\_neighbour\_table\_entry (data)**

This method is used to retrieve the mac neighbour table for one MAC short address. It may be used to perform topology monitoring by the client.

The method invocation parameter contains a mac\_short\_address.

data ::= long-unsigned

The response parameter includes the neighbour table for this mac\_short\_address.

data ::= array neighbour\_table

where mac\_neighbour\_table is as defined in the *mac\_neighbour\_table* attribute of the present IC.

**4.13.4.3.2 mac\_get\_POS\_table\_entry (data)**

This method is used to retrieve the mac POS 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 POS table for this mac\_short\_address.

data ::= array POS\_table

**Where:**

- mac\_POS\_table is as defined in the *mac\_POS\_table* attribute of the present IC.

**4.13.5 G3-PLC 6LoWPAN adaptation layer setup (class\_id = 92, version = 3)****4.13.5.1 Overview**

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.

## COSEM Interface Classes

| G3-PLC 6LoWPAN adaptation layer setup |            | 0...n          | class_id = 92, version = 3 |        |        |            |
|---------------------------------------|------------|----------------|----------------------------|--------|--------|------------|
| Attribute (s)                         |            | 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                          | 5      | 5      | x + 0x18   |
| 5. adp_prefix_table                   | (dyn)      | array          |                            |        |        | x + 0x20   |
| 6. adp_routing_configuration          | (static)   | array          |                            |        |        | x + 0x28   |
| 7. adp_broadcast_log_table _entry_TTL | (static)   | long- unsigned | 0                          | 65535  | 2      | x + 0x30   |
| 8. adp_routing_table                  | (dyn)      | array          |                            |        |        | x + 0x38   |
| 9. adp_context_information _table     | (dyn)      | array          |                            |        |        | x + 0x40   |
| 10. adp_blacklist_table               | (dyn)      | array          |                            |        |        | x + 0x48   |
| 11. adp_broadcast_log_table           | (dyn)      | array          |                            |        |        | x + 0x50   |
| 12. adp_group_table                   | (dyn)      | array          |                            |        |        | x + 0x58   |
| 13. adp_max_join_wait_time            | (static)   | long- unsigned | 0                          | 1023   | 20     | x + 0x60   |
| 14. adp_path_discovery_time           | (static)   | unsigned       | 0                          | 255    | 40     | x + 0x68   |
| 15. adp_active_key_index              | (static)   | unsigned       | 0                          | 1      | 0      | x + 0x70   |
| 16. adp_metric_type                   | (static)   | unsigned       | 0x00                       | 0x0F   | 0x0F   | x + 0x78   |
| 17. adp_coord_short_address           | (static)   | long- unsigned | 0x0000                     | 0x7FFF | 0x0000 | x + 0x80   |
| 18. adp_disable_default_routing       | (static)   | boolean        |                            |        | FALSE  | x + 0x88   |
| 19. adp_device_type                   | (static)   | enum           | 0                          | 2      | 2      | x + 0x90   |
| 20. adp_default_coord_route _enabled  | (static)   | boolean        |                            |        | FALSE  | x + 0x98   |
| 21. adp_destination_address _set      | (dyn)      | array          |                            |        |        | x + 0xA0   |
| 22. adp_low_LQI_value                 | (static)   | unsigned       | 0                          | 255    | 0      | x + 0xA8   |
| 23. adp_high_LQI_value                | (static)   | unsigned       | 0                          | 255    | 255    | x + 0xB0   |
| <b>Specific methods</b>               | <i>m/o</i> |                |                            |        |        |            |

### 4.13.5.2 Attribute description

#### 4.13.5.2.1 logical\_name

Identifies the “G3-PLC OFDM 6LoWPAN adaptation layer setup” object instance. See 6.2.28.

#### 4.13.5.2.2 adp\_max\_hops

PIB attribute 0x0F: Defines the maximum number of hops to be used by the routing algorithm.

#### 4.13.5.2.3 adp\_weak\_LQI\_value

PIB attribute 0x1A: The weak link value defines the LQI value below which a link to a neighbour is considered as a weak link. A value of 52 represents an SNR of 3 dB.

**4.13.5.2.4 adp\_security\_level**

PIB attribute 0x00: The minimum security level to be used for incoming and outgoing adaptation frames. Only values 0 (no ciphering) and 5 (ciphering with 32 bits integrity code) are supported.

**4.13.5.2.5 adp\_prefix\_table**

PIB attribute 0x01: Contains the list of prefixes defined on this PAN.

NOTE it is assumed that the link local IPv6 address exists independently and is not affected by the prefixes defined in the prefix table.

**4.13.5.2.6 adp\_routing\_configuration**

The routing configuration element specifies all parameters linked to the routing mechanism described in [ITU-T G.9903 Amd. 1:2021](#). 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.

```
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_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
}
```

Where:

- 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

## COSEM Interface Classes

- 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
- adp\_Kh PIB attribute 0x17: A weight factor for hop to calculate link cost.  
Range : 0-31  
Default value : 4
- adp\_Krt PIB attribute 0x1B: A weight factor for the number of active routes in the routing table to calculate link cost.  
Range : 0-31  
Default value : 0
- adp\_RREQ\_retries PIB attribute 0x18: The number of RREQ re-transmission in case of RREP reception time out.  
Range : 0-255  
Default value : 0
- adp\_RREQ\_wait PIB attribute 0x19: Time in seconds to wait between two consecutive RREQ or RLCREQ 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\_ink\_cost PIB attribute 0x0A: It represents an additional cost to take into account a possible asymmetry in the link.  
Range : 0-255  
Default value : 0

### 4.13.5.2.7 adp\_broadcast\_log\_table\_entry\_TTL

PIB attribute 0x02: Maximum time to live of an adpBroadcastLogTable entry (in minutes).

### 4.13.5.2.8 adp\_routing\_table

PIB attribute 0x0C: Contains the routing table.

```
array routing_table
    routing_table ::= structure
    {
        destination_address: long-unsigned,
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 361/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

```
next_hop_address:    long-unsigned,  
route_cost:          long-unsigned,  
hop_count:           unsigned,  
weak_link_count:    unsigned,  
valid_time:          long-unsigned  
}
```

NOTE 1 This table is actualized each time a route is built or updated (triggered by data traffic) and each time the TTL timer expires.

Where:

- 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 2 Practically the maximum allowed value is limited by adp\_max\_hops.
- weak\_link\_count Number of weak links to destination.  
Range: 0-14
  - NOTE 3 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.

### **4.13.5.2.9 adp\_context\_information\_table**

PIB attribute 0x07: Contains the context information associated to each CID extension field. See [ITU-T G.9903 Amd. 1:2021](#). The elements are specified in Table 9-30 of that Recommendation.

```
array context_information_table  
  
context_information_table ::= structure  
  
{  
    CID:                 bit-string,  
    context_length:      unsigned,  
    context:              octet-string,  
    C:                   boolean,  
    valid_lifetime:      long-unsigned  
}
```

Where:

- CID Corresponds to the 4-bit context information used for source and destination addresses (SCI, DCI). The first bit in the bit-string (bit 0) corresponds to the least significant bit of CID in [ITU-T G.9903 Amd. 1:2021](#), Table 9-30.  
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.

- 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

#### 4.13.5.2.10 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
}
```

Where:

- 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.

#### 4.13.5.2.11 adp\_broadcast\_log\_table

PIB attribute 0x0B: Contains the broadcast log table.

NOTE 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
}
```

Where:

- 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,
- valid\_time Remaining time in minutes until when this entry in the broadcast log table is considered valid.

#### 4.13.5.2.12 adp\_group\_table

PIB attribute 0x0E: Contains the group addresses to which the device belongs.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 363/668 |
|-----------------------|------------|-----------------------------|---------|

```

array group_address
group_address ::= long-unsigned

```

**Where:**

- group\_address      Group address to which this node has been subscribed.

**4.13.5.2.13 adp\_max\_join\_wait\_time**

PIB attribute 0x20: Network join timeout in seconds for LBD.

**4.13.5.2.14 adp\_path\_discovery\_time**

PIB attribute 0x21: Timeout for path discovery in seconds.

**4.13.5.2.15 adp\_active\_key\_index**

PIB attribute 0x22: Index of the active GMK to be used for data transmission.

**4.13.5.2.16 adp\_metric\_type**

PIB attribute 0x03: Metric Type to be used for routing purposes

**4.13.5.2.17 adp\_coord\_short\_address**

PIB attribute 0x08: Defines the short address of the coordinator.

**4.13.5.2.18 adp\_disable\_default\_routing**

PIB attribute 0xF0:

```

boolean:
TRUE:          the default routing (LOADng) is disabled,
FALSE:         the default routing (LOADng) is enabled.

```

**4.13.5.2.19 adp\_device\_type**

PIB attribute 0x10: Defines the type of the device connected to the modem:

```

enum:
(0)  PAN device,
(1)  PAN coordinator,
(2)  Not Defined

```

**4.13.5.2.20 adp\_default\_coord\_route\_enabled**

PIB attribute 0x24:

```

boolean:
TRUE:          the adaptation layer adds a default route to the
               coordinator after successful completion of the
               bootstrapping procedure.
FALSE:         no default route will be created.

```

#### 4.13.5.2.21 adp\_destination\_address\_set

PIB attribute 0x23: Contains the list of the addresses of the devices for which this LOADng router is providing connectivity.

array D\_address

D\_address ::= long-unsigned

Where:

- D\_Address The 16-bit address of a destination node attached to this LOADng router and for which this LOADng router provides connectivity.

#### 4.13.5.2.22 adp\_low\_LQI\_value

PIB attribute 0x04: The low LQI value defines the LQI value, used in metric computation, below which a link to a neighbour is considered as an unreliable link. This value shall be lower than adpHighLQIValue.

#### 4.13.5.2.23 adp\_high\_LQI\_value

PIB attribute 0x05: If. The high LQI value defines the LQI value, used in metric computation, above which a link to a neighbour is considered as a reliable link. This value is greater than adpLowLQIValue.

### 4.13.6 G3-PLC Hybrid RF MAC layer counters (class\_id = 160, version = 0)

#### 4.13.6.1 Overview

An instance of the “G3-PLC Hybrid RF 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 Hybrid RF MAC layer counters |          | 0...n                | class_id = 160, version = 0 |               |      |            |
|-------------------------------------|----------|----------------------|-----------------------------|---------------|------|------------|
| Attributes                          |          | Data type            | Min                         | Max.          | Def. | Short name |
| 1. logical_name                     | (static) | octet-string         | -                           | -             | -    | x          |
| 2. mac_retry_count_RF               | (dyn.)   | double-long-unsigned | 0                           | 4 294 967 295 | 0    | x + 0x08   |
| 3. mac_multiple_retry_count_RF      | (dyn.)   | double-long-unsigned | 0                           | 4 294 967 295 | 0    | x + 0x10   |
| 4. mac_Tx_fail_count_RF             | (dyn.)   | double-long-unsigned | 0                           | 4 294 967 295 | 0    | x + 0x18   |
| 5. mac_Tx_success_count_RF          | (dyn.)   | double-long-unsigned | 0                           | 4 294 967 295 | 0    | x + 0x20   |
| 6. mac_fcs_error_count_RF           | (dyn.)   | double-long-unsigned | 0                           | 4 294 967 295 | 0    | x + 0x28   |
| 7. mac_security_failure_count_RF    | (dyn.)   | double-long-unsigned | 0                           | 4 294 967 295 | 0    | x + 0x30   |
| 8. mac_duplicate_frame_count_RF     | (dyn.)   | double-long-unsigned | 0                           | 4 294 967 295 | 0    | x + 0x38   |
| 9. mac_Rx_success_count_RF          | (dyn.)   | double-long-unsigned | 0                           | 4 294 967 295 | 0    | x + 0x40   |
| <b>Specific methods</b>             |          | m/o                  |                             |               |      |            |
| 1. reset (data)                     |          | 0                    |                             |               |      | x + 0x50   |

**4.13.6.2 Attribute description**

NOTE When a counter reaches the maximum value (0xFFFFFFFF), it shall be automatically rolled-over.

**4.13.6.2.1 logical\_name**

Identifies the "G3-PLC Hybrid RF MAC layer counters" object instance. See 6.2.28.

**4.13.6.2.2 mac\_retry\_count\_RF**

PIB attribute 0x20A: Statistic counter of the number of transmitted data packets (MSDUs) that required exactly one retry before acknowledgement.

**4.13.6.2.3 mac\_multiple\_retry\_count\_RF**

PIB attribute 0x20B: Statistic counter of the number of transmitted data packets (MSDUs) that required more than one retry before acknowledgement.

**4.13.6.2.4 mac\_Tx\_fail\_count\_RF**

PIB attribute 0x20C: Statistic counter of the number of transmitted data packets (MSDUs) that did not result in an acknowledgement after macMaxFrameRetries\_RF.

**4.13.6.2.5 mac\_Tx\_success\_count\_RF**

PIB attribute 0x20D: Statistic counter of the number of transmitted data packets (MSDUs) that were acknowledged successfully after the initial data packet transmission.

**4.13.6.2.6 mac\_fcs\_error\_count\_RF**

PIB attribute 0x20E: Statistic counter of the number of transmitted data packets (MSDUs) that were discarded due to an incorrect FCS.

**4.13.6.2.7 mac\_security\_failure\_count\_RF**

PIB attribute 0x20F: Statistic counter of the number of transmitted data packets (MSDUs) that were returned from the procedure referred to in ITU-T G.9903 Amd. 1:2021, H.6.4 (selection of IEEE 802.15.4:2015, 9.2.3) with any Status other than SUCCESS.

**4.13.6.2.8 mac\_duplicate\_frame\_count\_RF**

PIB attribute 0x210: Statistic counter of the number of transmitted data packets (MSDUs) that contained the same sequence number as a frame previously received.

**4.13.6.2.9 mac\_Rx\_success\_count\_RF**

PIB attribute 0x211: Statistic counter of the number of data packets (MSDUs) that were received correctly.

**4.13.6.3 Method description****4.13.6.3.1 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)
```

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 366/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

**4.13.7 G3-PLC Hybrid RF MAC setup (class\_id = 161, version = 0)****4.13.7.1 Overview**

An instance of the “G3-PLC Hybrid RF MAC setup” IC holds the necessary additional parameters to set up and manage the G3-PLC Hybrid PLC & RF IEEE 802.15.4:2015 RF 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 Hybrid RF MAC setup                 | 0...n                | class_id = 161, version = 0 |               |       |            |
|--|----------------------|-----------------------------|---------------|-------|------------|
| Attributes                                 | Data type            | Min.                        | Max.          | Def.  | Short name |
| 1. logical_name (static)                   | octet-string         |                             |               |       | x          |
| 2. mac_max_BE_RF (static)                  | unsigned             | 3                           | 14            | 8     | x + 0x08   |
| 3. mac_max_CSMA_backoffs_RF (static)       | unsigned             | 0                           | 5             | 4     | x + 0x10   |
| 4. mac_max_frame_retries_RF (static)       | unsigned             | 0                           | 7             | 3     | x + 0x18   |
| 5. mac_min_BE_RF (static)                  | unsigned             | 0                           | 14            | 3     | x + 0x20   |
| 6. mac_frame_counter_RF (dyn.)             | double-long-unsigned | 0                           | 4 294 967 295 | 0     | x + 0x28   |
| 7. mac_duplicate_detection_TTL_RF (static) | bit-string F         | 0                           | 255           | 3     | x + 0x30   |
| 8. mac_POS_table_RF (dyn.)                 | array                |                             |               |       | x + 0x38   |
| 9. mac_operating_mode_RF (static)          | unsigned             | 1                           | 2             | 1     | x + 0x40   |
| 10. mac_channel_number_RF (static)         | unsigned             | 0                           | 7279          | 0     | x + 0x48   |
| 11. mac_duty_cycle_usage_RF (dyn.)         | unsigned             | 0                           | 100           | 0     | x + 0x50   |
| 12. mac_duty_cycle_period_RF (static)      | long-unsigned        | 1                           | 65535         | 3600  | x + 0x58   |
| 13. mac_duty_cycle_limit_RF (static)       | long-unsigned        | 1                           | 65535         | 90    | x + 0x60   |
| 14. mac_duty_cycle_threshold_RF (static)   | unsigned             | 1                           | 100           | 90    | x + 0x68   |
| 15. mac_disable_PHY_RF (static)            | boolean              |                             |               | FALSE | x + 0x70   |
| Specific methods                           | m/o                  |                             |               |       |            |
| 1. mac_get_POS_table_entry_RF (data)       | o                    |                             |               |       | x + 0xA0   |

**4.13.7.2 Attribute description****4.13.7.2.1 logical\_name**

Identifies the “G3-PLC Hybrid RF MAC setup” object instance. See 6.2.28.

**4.13.7.2.2 mac\_max\_BE\_RF**

PIB attribute 0x201: Maximum value of backoff exponent. It should always be greater than macMinBE\_RF.

**4.13.7.2.3 mac\_max\_CSMA\_backoffs\_RF**

PIB attribute 0x202: Maximum number of backoff attempts.

**4.13.7.2.4 mac\_max\_frame\_retries\_RF**

PIB attribute 0x203: Maximum number of retransmissions.

**4.13.7.2.5 mac\_min\_BE\_RF**

PIB attribute 0x204: Minimum value of backoff exponent.

**4.13.7.2.6 mac\_frame\_counter\_RF**

PIB attribute 0x207: The outgoing frame counter for this device, used when ciphering frames at MAC layer.

**4.13.7.2.7 mac\_duplicate\_detection\_TTL\_RF**

PIB attribute 0x208: Time a received tuple [source address + sequence-number] is retained for duplicate frame detection, in seconds.

**4.13.7.2.8 mac\_POS\_table\_RF**

PIB attribute 0x21D: The RF POS table contains some information about all the devices within the POS of the device

```
array POS_table_RF

neighbour_table ::= structure

{
    short_address: long-unsigned,
    forward_LQI: unsigned,
    reverse_LQI: unsigned,
    duty_cycle: unsigned,
    forward_Tx_power_offset: unsigned,
    reverse_Tx_power_offset: unsigned,
    POS_valid_time: unsigned,
}
```

NOTE Each time a message is received (filtered according to IEEE 802.15.4: 2006 , 7.5.6.2), an entry in the POS table is created or updated (if the entry is already present). In case the table is full, the entry corresponding to the shortest valid time is removed.

Where:

- short\_address      The MAC Short Address of the node to which this entry refers.
- forward\_LQI      Link quality indicator computed as the exponentially weighted moving average (EWMA) of the LQI derived from the received packets from this neighbour using a smoothing factor of 1/8.  
The forward LQI is updated only if the incoming frame carries an LI-IE.
- reverse\_LQI      Link quality indicator computed as the exponentially weighted moving average (EWMA) of the LQI derived from RLQ-IEs received from this neighbour using a smoothing factor of 1/8. If no reverse LQI measurements are available, this value shall be set to 0xFF indicating "not measured".  
The reverse LQI is updated only if the destination address of the Incoming frame matches macShortAddress.
- duty\_cycle      Duty cycle usage of the neighbour in percent as reported via the last received LI-IE.

- forward\_Tx\_power\_offset TX output power offset used for transmissions from this device to the neighbour node. Defined as the power reduction in dB compared to a reference TX output power of 30 dBm. The initial value is the difference in dB of the implementation's TX output power to 30 dBm. The value is propagated to the neighbour via the LI-IE.
- reverse\_Tx\_power\_offset TX output power offset used for transmissions from the neighbour node to this device. Defined as the power reduction in dB compared to a reference TX output power of 30 dBm. The value is received from the neighbour via the LI-IE.
- POS\_valid\_time Remaining time in minutes until when this entry is considered valid.  
Every time an entry is created, it is set to mac\_POS\_table\_entry\_TTL. When it reaches zero, this entry is no longer valid in the table and may be removed.

#### **4.13.7.2.9 mac\_operating\_mode\_RF**

PIB attribute 0x21E: The RF operating mode as defined in clause 19.3 of IEEE 802.15.4: 2006 .

#### **4.13.7.2.10 mac\_channel\_number\_RF**

PIB attribute 0x21F: The channel number as defined in clause 10.1.3.9 of IEEE 802.15.4: 2006 :2020 to be used for channel scanning and data transmission.

#### **4.13.7.2.11 mac\_duty\_cycle\_usage\_RF**

PIB attribute 0x220: Current usage of maximum allowed duty cycle in percent over the current sliding-window measurement period, i.e.  $(t_{on} / \text{macDutyCycleLimit\_RF}) * 100$ .

#### **4.13.7.2.12 mac\_duty\_cycle\_period\_RF**

PIB attribute 0x221: Duration of the measurement period in seconds, e.g. 3600 seconds as default for ETSI EN 303 204 V2.1.2 (2016-09) .

#### **4.13.7.2.13 mac\_duty\_cycle\_limit\_RF**

PIB attribute 0x222: Duration of the allowed transmission time in seconds, e.g. 2,5%/10% of 3600 seconds as default for ETSI EN 303 204. If this attribute is set equal to macDutyCyclePeriod\_RF the value of macDutyCycleUsage\_RF will be fixed to zero.

#### **4.13.7.2.14 mac\_duty\_cycle\_threshold\_RF**

PIB attribute 0x223: Duty cycle threshold for stopping RF transmissions.

#### **4.13.7.2.15 mac\_disable\_PHY\_RF**

PIB attribute 0x224: Disable RF PHY Tx and Rx.

### **4.13.7.3 Method description**

#### **4.13.7.3.1 mac\_get\_POS\_table\_entry\_RF (data)**

This method is used to retrieve the RF mac POS table for one MAC short address. It may be used to perform topology monitoring by the client.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 369/668 |
|-----------------------|------------|-----------------------------|---------|

The method invocation parameter contains a mac\_short\_address.

```
data ::= long-unsigned
```

The response parameter includes the RF POS table for this mac\_short\_address.

```
data ::= array POS_table
```

where mac\_POS\_table\_RF is as defined in the *mac\_POS\_table\_RF* attribute of the present IC.

#### 4.13.8 G3-PLC Hybrid 6LoWPAN adaptation layer setup (class\_id = 162, version = 0)

##### 4.13.8.1 Overview

An instance of the “G3-PLC Hybrid 6LoWPAN adaptation layer setup” IC holds the necessary additional parameters to set up and manage the G3-PLC Hybrid PLC & RF 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 Hybrid 6LoWPAN adaptation layer setup | 0...n        | class_id = 162, version = 0 |      |      |            |
|--|--------------|-----------------------------|------|------|------------|
| Attribute(s)                                 | Data type    | Min.                        | Max. | Def. | Short name |
| 1. logical_name (static)                     | octet-string |                             |      |      | x          |
| 2. adp_routing_table (dyn.)                  | array        |                             |      |      | x + 0x08   |
| 3. adp_blacklist_table (dyn.)                | array        |                             |      |      | x + 0x10   |
| 4. adp_low_LQI_value_RF (static)             | unsigned     | 0                           | 254  | 0    | x + 0x18   |
| 5. adp_high_LQI_value_RF (static)            | unsigned     | 0                           | 254  | 254  | x + 0x20   |
| 6. adp_routing_configuration_RF (static)     | array        |                             |      |      | x + 0x28   |
| 7. adp_use_backup_media (static)             | boolean      |                             |      | TRUE | x + 0x30   |
| Specific methods                             | m/o          |                             |      |      |            |

##### 4.13.8.2 Attribute description

###### 4.13.8.2.1 logical\_name

Identifies the “G3-PLC Hybrid 6LoWPAN adaptation layer setup” object instance. See 6.2.28.

###### 4.13.8.2.2 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,
```

## COSEM Interface Classes

```

route_cost:          long-unsigned,
hop_count:          unsigned,
weak_link_count:    unsigned,
valid_time:          long-unsigned,
media_type:          enum
}

```

**NOTE 1** This table is generated each time a route is built or updated (triggered by data traffic) and each time the TTL timer expires.

Where:

- 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 2** Practically the maximum allowed value is limited by adp\_max\_hops.

- weak\_link\_count Number of weak links to destination.  
Range: 0-14

**NOTE 3** 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.
- media\_type The medium to be used when transmitting to the next hop.  
0: PLC,  
1: RF

### 4.13.8.2.3 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,
    media_type:                      enum
}

```

Where:

- 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.
- media\_type The medium to be used when transmitting to the next hop, where:  
0: PLC,  
1: RF

### 4.13.8.2.4 adp\_low\_LQI\_value\_RF

PIB attribute 0xD0: The low LQI value defines the LQI value, used in metric computation, below which a link to a neighbour is considered as an unreliable RF link. This value shall be lower than adpHighLQIValue\_RF.

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
|                       |            | 371/668                     |

**4.13.8.2.5 adp\_high\_LQI\_value\_RF**

PIB attribute 0xD1: If. The high LQI value defines the LQI value, used in metric computation, above which a link to a neighbour is considered as a reliable RF link. This value is greater than adpLowLQIValue\_RF.

**4.13.8.2.6 adp\_routing\_configuration\_RF**

The routing configuration element specifies all additional parameters linked to the routing mechanism described in ITU-T G.9903 Amd. 1:2021 for the Hybrid PLC & RF profile. The elements are specified in H.8.1 of that Recommendation.

**NOTE** The Link cost calculation is provided in ITU-T G.9903 Amd. 1:2021, H.8.2.

```
array routing_configuration
routing_configuration ::= structure
{
    adp_Kq_RF:             unsigned,
    adp_Kh_RF:             unsigned,
    adp_Krt_RF:            unsigned,
    adp_Kdc_RF:            unsigned,
}
```

Where:

- adp\_Kq\_RF              PIB attribute 0xD2: A weight factor for LQI to calculate route cost.  
Range : 0-50  
Default value : 10.
- adp\_Kh\_RF              PIB attribute 0xD3: A weight factor for hop to calculate link cost.  
Range : 0-31  
Default value : 4.
- adp\_Krt\_RF            PIB attribute 0xD4: A weight factor for the number of active routes in the routing table to calculate link cost.  
Range : 0-31  
Default value : 0
- adp\_Kdc\_RF            PIB attribute 0xD5: A weight factor for duty cycle to calculate link cost.  
Range : 0-50  
Default value : 10

**4.13.8.2.7 adp\_use\_backup\_media**

PIB attribute 0xD6: This attribute controls if retransmission can use the backup medium in case of transmission failure.

## 4.14 Interface classes for setting up and managing DLMS/COSEM HS-PLC ISO/IEC 12139-1 neighbourhood networks

### 4.14.1 Overview

COSEM objects for data exchange using DLMS/COSEM HS-PLC ISO/IEC 12139-1 neighbourhood networks, if implemented, shall be located in the Management Logical Device of COSEM servers.

For setting up and managing DLMS/COSEM HS-PLC ISO/IEC 12139-1 neighbourhood networks the following ICs are specified:

- HS-PLC ISO/IEC 12139-1 MAC setup, see 4.14.2;
- HS-PLC ISO/IEC 12139-1 CPAS setup, see 4.14.3;
- HS-PLC ISO/IEC 12139-1 IP SSAS setup, see 4.14.4;
- HS-PLC ISO/IEC 12139-1 HDLC SSAS setup, see 4.14.5.

### 4.14.2 HS-PLC ISO/IEC 12139-1 MAC setup (class\_id = 140, version = 0)

#### 4.14.2.1 Overview

Instances of the "HS-PLC ISO/IEC 12139-1 MAC setup" IC hold parameters necessary to set up and manage the MAC layer of the HS-PLC ISO/IEC 12139-1 profile.

| HS-PLC ISO/IEC 12139-1 MAC setup   | 0...n           | class_id = 140 version = 0 |          |      |            |
|------------------------------------|-----------------|----------------------------|----------|------|------------|
| Attributes                         | Data type       | Min.                       | Max.     | Def. | Short name |
| 1. logical_name                    | octet-string    |                            |          |      | x          |
| 2. group_id (static)               | long64-unsigned | 0                          | $2^{46}$ |      | x + 0x08   |
| 3. secondary_group_id (static)     | long64-unsigned | 0                          | $2^{46}$ |      | x + 0x10   |
| 4. station_id (static)             | long64-unsigned | 0                          | $2^{48}$ |      | x + 0x18   |
| 5. parent_station_id (static)      | long64-unsigned | 0                          | $2^{48}$ |      | x + 0x20   |
| 6. repeater_status (static)        | boolean         |                            |          |      | x + 0x28   |
| 7. encryption_mode (static)        | enum            | 1                          | 2        | 1    | x + 0x30   |
| 8. initial_encryption_key (static) | octet-string    |                            |          |      | x + 0x38   |
| 9. rts/cts (static)                | boolean         | 0                          | 1        | 0    | x + 0x40   |
| Specific methods                   | m/o             |                            |          |      |            |

#### 4.14.2.2 Attribute description

##### 4.14.2.2.1 logical\_name

Identifies the "HS-PLC ISO/IEC 12139-1 MAC setup" object instance. See 6.2.31.

##### 4.14.2.2.2 group\_id

Holds the group identifier of HS-PLC ISO/IEC 12139-1. Refer to ISO/IEC 12139-1:2009, Clause 7 for detailed information.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 373/668 |
|-----------------------|------------|-----------------------------|---------|

**4.14.2.2.3 secondary\_group\_id**

Holds the secondary group identifier of HS-PLC ISO/IEC 12139-1. Refer to ISO/IEC 12139-1:2009, Clause 7 for detailed information.

**4.14.2.2.4 station\_id**

Holds the station identifier of HS-PLC ISO/IEC 12139-1. Refer to ISO/IEC 12139-1:2009, Clause 7 for detailed information.

**4.14.2.2.5 parent\_station\_id**

Holds the parent station identifier\* of HS-PLC ISO/IEC 12139-1. Refer to ISO/IEC 12139-1:2009, Clause 7 for detailed information.

\* The parent station means the directly neighbouring station from a station itself in a multi-stage HS-PLC ISO/IEC 12139-1 link from the station to NNAP (a source station – Repeater(s) –NNAP).

**4.14.2.2.6 repeater\_status**

Refer to ISO/IEC 12139-1:2009, Clause 7 for detailed information.

Holds the current repeater status of the device.

boolean:

|        |              |
|--------|--------------|
| FALSE: | no repeater, |
| TRUE:  | repeater     |

**4.14.2.2.7 encryption\_mode**

Holds the encryption mode the PLC station uses. Refer to ISO/IEC 12139-1:2009, Clause 7 for detailed information.

enum:

|     |           |
|-----|-----------|
| (0) | AES128,   |
| (1) | Reserved. |

**4.14.2.2.8 initial\_encryption\_key**

Encryption key which is used initially during PLC registration (cell join) period. Refer to ISO/IEC 12139-1:2009, Clause 7 for detailed information.

**4.14.2.2.9 rts/cts**

Holds the status of the RTS/CTS parameter. The RTS/CTS (Request To Send / Clear To Send) parameter is used to prevent collision caused by hidden HS-PLC ISO/IEC 12139-1 stations). Refer to ISO/IEC 12139-1:2009, Clause 7 for detailed information.

RTS/CTS enable/disable

boolean:

|        |          |
|--------|----------|
| FALSE: | Disable, |
| TRUE:  | Enable   |

#### 4.14.3 HS-PLC ISO/IEC 12139-1 CPAS setup (class\_id = 141, version = 0)

##### 4.14.3.1 Overview

Instances of the "HS-PLC ISO/IEC 12139-1 CPAS setup" IC hold parameters necessary to set up and manage the CPAS layer of the HS-PLC ISO/IEC 12139-1 profile.

| HS-PLC ISO/IEC 12139-1 CPAS setup       | 0...n           | class_id = 141 version = 0 |          |      |            |
|---|-----------------|----------------------------|----------|------|------------|
| Attributes                              | Data type       | Min.                       | Max.     | Def. | Short name |
| 1. logical_name                         | octet-string    |                            |          |      | x          |
| 2. cpas_address (static)                | long64-unsigned | 0                          | $2^{48}$ |      | x + 0x08   |
| 3. cpas_ether_type (static)             | long-unsigned   |                            |          |      | x + 0x10   |
| 4. master_station_cpas_address (static) | long64-unsigned | 0                          | $2^{48}$ |      | x + 0x18   |
| Specific methods                        | m/o             |                            |          |      |            |

##### 4.14.3.2 Attribute description

###### 4.14.3.2.1 logical\_name

Identifies the "HS-PLC ISO/IEC 12139-1 CPAS setup" object instance. See 6.2.31.

###### 4.14.3.2.2 cpas\_address

Holds the CPAS address of the HS-PLC ISO/IEC 12139-1 profile.

See IEC 62056-8-6: 2017, 5.4.2.

###### 4.14.3.2.3 cpas\_ether\_type

Holds the EtherType value of the CPAS sublayer.

See IEC 62056-8-6: 2017, 5.4.2.

###### 4.14.3.2.4 master\_station\_cpas\_address

Holds the master station's\* CPAS address of the HS-PLC ISO/IEC 12139-1 profile.

\* The master station means the destination station of the CPAS frame (i.e. typically NNAP) in a multi-stage HS-PLC ISO/IEC 12139-1 link (a source station – Repeater(s) – A destination station).

#### 4.14.4 HS-PLC ISO/IEC 12139-1 IP SSAS setup (class\_id = 142, version = 0)

##### 4.14.4.1 Overview

Instances of the "HS-PLC ISO/IEC 12139-1IP SSAS setup" IC hold parameters necessary to set up and manage the IP SSAS of the HS-PLC ISO/IEC 12139-1 profile.

## COSEM Interface Classes

|   |                  |                                    |             |             |                   |
|---|------------------|------------------------------------|-------------|-------------|-------------------|
| <b>HS-PLC ISO/IEC 12139-1 IP SSAS setup</b> | 0...n            | <b>class_id = 142, version = 0</b> |             |             |                   |
| <b>Attributes</b>                           | <b>Data type</b> | <b>Min.</b>                        | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 1. logical_name                             | octet-string     |                                    |             |             | x                 |
| 2. ip_header_comp_type (static)             | enum             |                                    |             |             | x + 0x08          |
| 3. ip_alive_time (static)                   | long-unsigned    |                                    | 0xFFFF      | 0           | x + 0x10          |
| <b>Specific methods</b>                     | <b>m/o</b>       |                                    |             |             |                   |

### 4.14.4.2 Attribute description

#### 4.14.4.2.1 logical\_name

Identifies the "HS-PLC ISO/IEC 12139-1 IP SSAS setup" object instance. See 6.2.31.

#### 4.14.4.2.2 ip\_header\_comp\_type

Holds the IP\_Header\_Comp\_Type value as specified in IEC 62056-8-6: 2017, Table 3.

enum:

- (0) General IPv4 packet (No compression),
- (1) General IPv6 packet (No compression),
- (2) Van Jacobson header compression (RFC 1144),
- (3) IP header compression (RFC 2508),
- (4) ROHC (RFC 3095).

#### 4.14.4.2.3 ip\_alive\_time

Holds the IP SSAS alive time value in seconds.

0: No expiry

### 4.14.5 HS-PLC ISO/IEC 12139-1 HDLC SSAS setup (class\_id = 143, version = 0)

#### 4.14.5.1 Overview

Instances of the "HS-PLC ISO/IEC 12139-1 HDLC SSAS setup" IC hold parameters necessary to set up and manage the HDLC SSAS of the HS-PLC ISO/IEC 12139-1 profile.

|   |                  |                                    |             |             |                   |
|---|------------------|------------------------------------|-------------|-------------|-------------------|
| <b>HS-PLC ISO/IEC 12139-1 HDLC SSAS setup</b> | 0...n            | <b>class_id = 143, version = 0</b> |             |             |                   |
| <b>Attributes</b>                             | <b>Data type</b> | <b>Min.</b>                        | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 1. logical_name                               | octet-string     |                                    |             |             | x                 |
| 2. master_station_id (static)                 | long64-unsigned  | 0                                  | $2^{48}$    |             | x + 0x08          |
| <b>Specific methods</b>                       | <b>m/o</b>       |                                    |             |             |                   |

**4.14.5.2 Attribute description**

**4.14.5.2.1 logical\_name**

Identifies the “HS-PLC ISO/IEC 12139-1 HDLC SSAS setup” object instance. See 6.2.31.

**4.14.5.2.2 master\_station\_id**

Holds the master station identifier of the HS-PLC ISO/IEC 12139-1 network.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 377/668 |
|-----------------------|------------|-----------------------------|---------|

## 4.15 ZigBee® setup classes

### 4.15.1 Overview

This subclause 4.15 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](http://www.zigbee.org).

**ZigBee® is a registered trademark of the ZigBee® Alliance.**

NOTE 1 A multi-part installation is one where the meter provides information and/or services to the householder on behalf of the utility. For example, the meter interacts with an in home display, and/or an external load control switch, and/or a smart appliance, to inform the customer of their usage in real time, to control heating devices, and possibly to disconnect peak loads when supply is constrained. While it is possible that the consumer will control the ZigBee® network, in normal operations the utility will control the radio system. This is to ensure that security is maintained for PAN, so that ZigBee® devices such as load switches controlled by the utility operate in a secure manner.

ZigBee® defines a local network of devices linked by radio, with routing and forwarding of messages and with encryption for privacy at network-level.

NOTE 2 Such a local network is known as a PAN – a Personal Area Network. This name is used in the ZigBee® community as ZigBee® is underpinned by the IEEE 802.15.4: 2006 standard, which uses the term PAN. This is broadly equivalent to a HAN (Home Area Network – name used in the context of smart metering in the UK) or PAN.

Each PAN has one device designated as ZigBee® coordinator, which has responsibility for creating and managing the network, and which normally acts also as a ZigBee® Trust Center for the management of ZigBee® network, PCLK's and APS Link keys.

There is a process of PAN creation (and corresponding destruction) which is performed by the coordinator; this declares the existence of the network without any devices apart from the coordinator forming part of it. Other ZigBee® devices can join a network created with cooperation of the coordinator, and equally can choose to leave, or can be invited to leave by the coordinator (this is not currently enforceable). Normally devices are members of the network indefinitely; they do not repeatedly join and leave. To create a PAN the coordinator has to receive an external trigger and needs to have setup information including:

- extended PAN ID;
- link keys or install code (for initial communication with new devices);
- radio channel information.

During the process of creating the PAN, the coordinator scans for nearby radio devices, “exchanges” keys, chooses the short addresses, and confirms use of radio channels. Details of the information available from the ZigBee® servers on each device are also exchanged.

NOTE 3 Full details of the joining process are documented in ZigBee® 053474 , the ZigBee® specification. More information on ZigBee® technology can be sought at <http://www.zigbee.org/>.

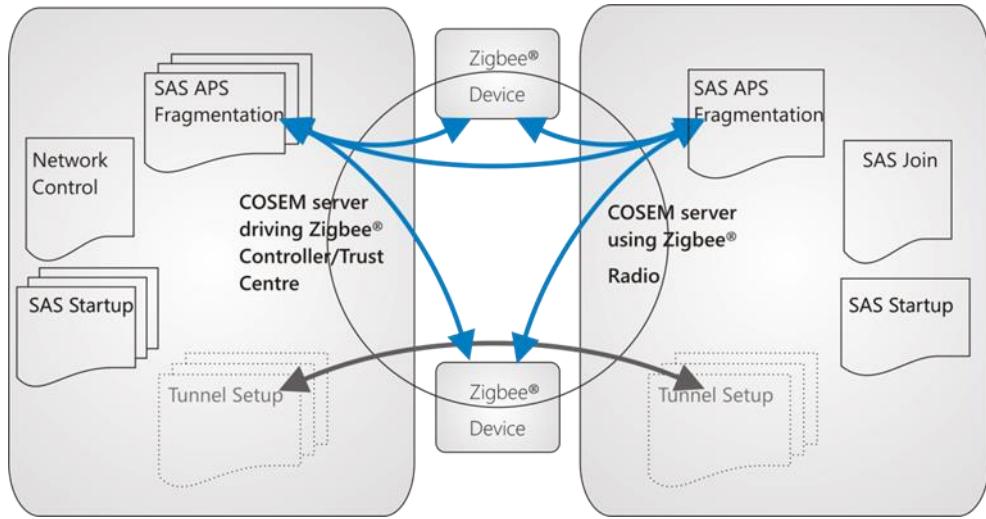
Figure 32 shows an example architecture with a Comms hub on the left, that comprises a DLMS server as well as the ZigBee® coordinator. The DLMS server has interface objects needed to set up and control the ZigBee® network and may have other COSEM objects to support a metering application. Further, there are two native ZigBee® devices and another DLMS server – an electricity meter or another device 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.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 378/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

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.



**Figure 32 – 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 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 de is merely the vehicle for controlling the ZigBee® network by an external manager (DLMS client).

The use of ZigBee® setup classes in the ZigBee® coordinator and other DLMS/COSEM ZigBee® devices is shown in Table 45.

**Table 45 – Use of ZigBee® setup COSEM interface classes**

| ZigBee® coordinator              | Other DLMS/COSEM ZigBee® devices | Reference |
|----------------------------------|----------------------------------|-----------|
| ZigBee® SAS startup              | ZigBee® SAS startup              | 4.15.2    |
| –                                | ZigBee® SAS join                 | 4.15.3    |
| ZigBee® SAS APS fragmentation    | ZigBee® SAS APS fragmentation    | 4.15.4    |
| ZigBee® network control          | –                                | 4.15.5    |
| Optionally: ZigBee® tunnel setup | Optionally: ZigBee® tunnel setup | 4.15.6    |

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.

NOTE 4 In the specification of the ZigBee® COSEM ICs, the length of the octet-strings is indicated for information.

#### 4.15.2 ZigBee® SAS startup (class\_id = 101, version = 0)

##### 4.15.2.1 Overview

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<br>(static)       | octet-string         |                             |      |        | x          |
| 2. short_address<br>(dyn.)        | long-unsigned        |                             |      | 0xFFFF | x + 0x08   |
| 3. extended_pan_id<br>(dyn.)      | octet-string         |                             |      | 0      | x + 0x10   |
| 4. pan_id<br>(dyn.)               | long-unsigned        |                             |      | 0xFFFF | x + 0x18   |
| 5. channel_mask<br>(dyn.)         | double-long-unsigned |                             |      | 0      | x + 0x20   |
| 6. protocol_version<br>(static)   | unsigned             | 0x02                        |      | 0x02   | x + 0x28   |
| 7. stack_profile<br>(static)      | enum                 |                             |      | 0x02   | x + 0x30   |
| 8. start_up_control<br>(dyn.)     | unsigned             |                             |      | 2      | x + 0x38   |
| 9. trust_center_address<br>(dyn.) | octet-string         |                             |      | 0      | x + 0x40   |
| 10. link_key<br>(dyn.)            | octet-string         |                             |      | 0      | x + 0x48   |
| 11. network_key<br>(dyn.)         | octet-string         |                             |      | 0      | x + 0x50   |
| 12. use_insecure_join<br>(static) | boolean              |                             |      | FALSE  | x + 0x58   |
| Specific methods                  | m/o                  |                             |      |        |            |

##### 4.15.2.2 Attribute description

###### 4.15.2.2.1 logical\_name

Identifies the “ZigBee® SAS startup” object instance. See 6.2.29.

###### 4.15.2.2.2 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 This short address will be issued to the device by the coordinator when the device joins the network.

###### 4.15.2.2.3 extended\_pan\_ID

Defines the unique address by which the network is known externally.

NOTE The length of the octet-string is 8 octets.

###### 4.15.2.2.4 pan\_ID

Defines the 16-bit address by which network is known locally.

**4.15.2.2.5 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.

**4.15.2.2.6 protocol\_version**

Defines the version of the ZigBee® protocol to be used on the network.

**4.15.2.2.7 stack\_profile**

Identifies the capabilities of the ZigBee ® stack.

- enum:  
 (1) ZigBee®,  
 (2) ZigBee® PRO

**4.15.2.2.8 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 extended\_pan\_id attribute. In this case it will not perform any explicit join or rejoin operation.

**NOTE** The “ZigBee® SAS startup” object reflects the state of the ZigBee® HAN to the WAN (or a diagnostic tool connected to a ZigBee® connected DLMS 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 32 is to express the idea that there may be multiple ZigBee® networks operated from a single Comms Hub.

**4.15.2.2.9 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.

**NOTE** The length of the octet-string is 8 octets.

**4.15.2.2.10 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.

**NOTE** The length of the octet-string is 16 octets.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 381/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.15.2.2.11 **network\_key**

Defines the key value used to secure general communications between devices. The way in which the raw key value is protected is not defined in this specification. The network key value may be set internally by the coordinator.

It is down to the implementation if this attribute is Hashed or in the clear or even if this is used.

If this is the TC then this is not usually implemented.

This is read only.

NOTE The length of the octet-string is 16 octets.

#### 4.15.2.2.12 **use\_insecure\_join**

Indicates whether the coordinator is permitted to allow insecure joining as defined by ZigBee® specification.

### 4.15.3 ZigBee® SAS join (**class\_id = 102, version = 0**)

#### 4.15.3.1 Overview

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 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         | <b>class_id = 102, version = 0</b> |      |      |            |
|--------------------------------------|---------------|------------------------------------|------|------|------------|
| Attributes                           | Data type     | Min.                               | Max. | Def. | Short name |
| 1. logical_name<br>(static)          | octet-string  |                                    |      |      | x          |
| 2. scan_attempts<br>(static)         | unsigned      |                                    |      | 3    | x + 0x08   |
| 3. time_between_scans<br>(static)    | long-unsigned |                                    |      | 1    | x + 0x10   |
| 4. rejoin_interval                   | long-unsigned |                                    |      | 60   | x + 0x18   |
| 5. rejoin_retry_interval<br>(static) | long-unsigned |                                    |      | 900  | x + 0x20   |
| <i>Specific methods</i>              | <i>m/o</i>    |                                    |      |      |            |

#### 4.15.3.2 Attribute description

##### 4.15.3.2.1 **logical\_name**

Identifies the “ZigBee® SAS join” object instance. See 6.2.29.

##### 4.15.3.2.2 **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.

**4.15.3.2.3 time\_between\_scans**

Defines the period in seconds between consecutive scans in a single attempt to rejoin a network.

**4.15.3.2.4 rejoin\_interval**

Defines the period in seconds that the device should wait after apparently becoming disconnected from a network, before attempting to rejoin.

**4.15.3.2.5 rejoin\_retry\_interval**

Defines the period in seconds for which the device should wait after a failed attempt to rejoin a network before trying again.

- Remarks on usage

At boot time or when instructed to join a network, the device should complete up to three (3) (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 (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).

- Remark on the rejoin\_retry\_interval

Imposes an upper bound on the *rejoin\_interval* parameter – this is restarted if device is touched by human user, i.e. by a button press. This parameter is intended to restrict how often a device will scan to find its network in case the network is no longer present and therefore a scan attempt by the device would always fail (i.e., if a device finds it has lost network connectivity, it will try to re-join the network, scanning all channels if necessary). If the scan fails to successfully re-join, the device will wait for 15 min before attempting to re-join again. To be network friendly, it would be recommended to adaptively extend this time period if successive re-joins fail. It would also be recommended the device should try a re-join when triggered (via a control, button, etc.) and fall back to the *rejoin\_retry\_interval* if attempts to re-join fail again.

**4.15.4 ZigBee® SAS APS fragmentation (class\_id = 103, version = 0)****4.15.4.1 Overview**

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 client).

Instances of this IC are present in all devices where a DLMS server controls the ZigBee® Radio behaviour.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 383/668 |
|-----------------------|------------|-----------------------------|---------|

| 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           |                             |      |      |            |

#### 4.15.4.2 Attribute description

##### 4.15.4.2.1 logical\_name

Identifies the “ZigBee® SAS APS fragmentation” object instance. See 6.2.29.

##### 4.15.4.2.2 aps\_interframe\_delay

Defines the delay in milliseconds between sending two blocks of a fragmented transmission.

##### 4.15.4.2.3 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.

#### 4.15.5 ZigBee® network control (class\_id = 104, version = 0)

##### 4.15.5.1 Overview

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 client. This class allows interaction between a DLMS 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             |                             |      |       |            |
| 2. unregister_device (data)      | m             |                             |      |       |            |
| 3. unregister_all_devices (data) | o             |                             |      |       |            |
| 4. backup_PAN (data)             | o             |                             |      |       |            |
| 5. restore_PAN (data)            | o             |                             |      |       |            |
| 6. identify_device (data)        | o             |                             |      |       |            |
| 7. remove_mirror (data)          | o             |                             |      |       |            |
| 8. update_network_key (data)     | o             |                             |      |       |            |
| 9. update_link_key (data)        | o             |                             |      |       |            |
| 10. create_PAN (data)            | m             |                             |      |       |            |
| 11. remove_PAN (data)            | m             |                             |      |       |            |

#### 4.15.5.2 Attribute description

##### 4.15.5.2.1 logical\_name

Identifies the “ZigBee® network control” object instance. See 6.2.29.

##### 4.15.5.2.2 enable\_disable\_joining

A flag controlling whether devices are allowed to join the ZigBee® network. The flag is normally FALSE (joining disabled). At certain times – when a new device is expected to join – the flag is set externally to TRUE and then remains set for the duration of the *join\_timeout*, or until externally reset if that happens sooner.

##### 4.15.5.2.3 join\_timeout

Defines the time period in seconds during which the coordinator device will permit joining of new devices, following setting of the *enable\_disable\_joining* flag.

##### 4.15.5.2.4 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,

```

## COSEM Interface Classes

```
    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
}
```

NOTE 1 The length of the mac\_address is 8 octets.

NOTE 2 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.

### 4.15.5.3 Method description

#### 4.15.5.3.1 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,
```

## COSEM Interface Classes

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 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.

device-type ::= enum

- (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

### 4.15.5.3.2 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.

### 4.15.5.3.3 unregister\_all\_devices (data)

This method is called externally to instruct the coordinator that all devices should leave the network.

data ::= integer (0)

NOTE The likely use of this function is to ensure a network is empty of devices prior to destroying it.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 387/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.15.5.3.3.1 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 1 The storage location of the back-up is not currently defined and is an internal function of the DLMS 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 4.6.1;
- 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 2 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.

#### 4.15.5.3.4 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 server.

```
data ::= structure
{
    extended_PAN_ID:   octet-string,
```

## COSEM Interface Classes

```
    devices_to_restore:          array device_to_restore
}
```

Where:

- extended\_PAN\_ID identifies the PAN. The length of the octet-string is 8;
- 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 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.

### **4.15.5.3.5 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.

### **4.15.5.3.6 remove\_mirror (data)**

This method causes the removal of a ZigBee® mirror that reflects the real device identified by the mac\_address parameter.

```
data ::= structure
{
  mac_address:      octet-string,
  mirror_control: bit-string
}
```

Where:

- MAC\_address holds the MAC address. The length of the octet-string is 8;
- the mirror\_control parameter is provided to support the execution of implementation-specific actions, which should be defined in a project specific companion specification.

EXAMPLE The following example is one of the possible options for the bit-string functionality.

0 = Force Gas Meter Removal.

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

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 389/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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 server should undertake are implementation-specific actions, which should be defined in a project specific companion specification.

### 4.15.5.3.7 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)

### 4.15.5.3.8 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.

### 4.15.5.3.9 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)

### 4.15.5.3.10 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)

## 4.15.6 ZigBee® tunnel setup (class\_id = 105, version = 0)

### 4.15.6.1 Overview

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.

## COSEM Interface Classes

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 client. The target device is implicitly identified by the COSEM addressing information.

**NOTE** See also the Gateway specification in DLMS UA 1000-2 Ed.11:2021, Annex C.

| <b>ZigBee® tunnel setup</b>       |          | <b>0...n</b>        | <b>class_id = 105, version = 0</b> |             |             |                   |
|-----------------------------------|----------|---------------------|------------------------------------|-------------|-------------|-------------------|
| <b>Attributes</b>                 |          | <b>Data type</b>    | <b>Min.</b>                        | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 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          |
| <b>Specific methods</b>           |          | <b>m/o</b>          |                                    |             |             |                   |

### 4.15.6.2 Attribute description

#### 4.15.6.2.1 logical\_name

Identifies the “ZigBee® tunnel setup” object instance. See 6.2.29.

#### 4.15.6.2.2 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.

#### 4.15.6.2.3 maximum\_outcoming\_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.

**NOTE** The word *outcoming* is unusual, but is adopted for commonality with ZigBee® specifications.

#### 4.15.6.2.4 protocol\_address

The *protocol\_address* is implementation-specific, which should be defined in a project specific companion specification.

The length of the octet-string is 6 octets.

#### 4.15.6.2.5 close\_tunnel\_timeout

Defines the time, in seconds that the ZigBee® server waits before closing an inactive tunnel on its own (without waiting for the *CloseTunnel* Command from the client within the ZigBee® specification) and freeing its resources

The timer is re-started with each reception of a command.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 391/668 |
|-----------------------|------------|-----------------------------|---------|

## 4.16 Interface classes for setting up and managing the DLMS/COSEM profile for LPWAN networks

### 4.16.1 General

This clause specifies ICs for setting up devices using the DLMS/COSEM LPWAN communication profile and to diagnose the LPWAN network:

- the ICs specified in 4.16.2 are used to define setup and diagnostic objects for the SCHC-LPWAN part;
- the ICs specified in 4.17 are used to define setup and diagnostic objects for the lower layers.

For each specific LPWAN technology, a setup and a diagnostic IC should be specified. ICs for the LoRaWAN 1.0.3 technology are specified in 4.16.3.2.

### 4.16.2 Generic interface classes

#### 4.16.2.1 SCHC-LPWAN setup (class\_id = 126, version = 0)

##### 4.16.2.1.1 Overview

Instances of this IC are available for configuring the parameters needed to set up a LPWAN device.

| SCHC-LPWAN setup                      |          | 0..n         | class_id = 126 , version = 0 |      |      |            |
|---------------------------------------|----------|--------------|------------------------------|------|------|------------|
| Attributes                            |          | Data type    | Min.                         | Max. | Def. | Short name |
| 1. logical_name                       | (static) | octet-string |                              |      |      | x          |
| 2. ipwan_reference                    | (static) | octet-string |                              |      |      | x + 0x08   |
| 3. schc_cd_rules                      | (static) | array        |                              |      |      | x + 0x10   |
| 4. schc_fr_param                      | (static) | structure    |                              |      |      | x + 0x18   |
| <b>Specific methods (if required)</b> |          | m/o          |                              |      |      |            |

##### 4.16.2.1.2 Attribute description

###### 4.16.2.1.2.1 logical\_name

Identifies the “LPWAN setup” object instance. See 6.2.23.

###### 4.16.2.1.2.2 ipwan\_reference

References an LPWAN technology specific setup object.

###### 4.16.2.1.2.3 schc\_cd\_rules

Contains the necessary parameters to support LPWAN Static Context Header Compression (SCHC) compression and decompression rules

```
array schc_cd_rules_element
schc_cd_rules_element ::= structure
```

## COSEM Interface Classes

```

{
    rule_id:          unsigned,
    field_descriptors: array field_descriptor
}

field_descriptor ::= structure

{
    field_id:          field_id_type,
    field_length:      unsigned,
    field_position:    unsigned,
    direction_indicator: direction_indicator_type,
    target_value:      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]
    }
}

matching_operator:      matching_operator_type,
compression_decompression_action:
compression_decompression_action_type
}

field_id_type ::= enum

{
    (0)          IPv6-version,
    (1)          IPv6-DiffServ,
    (2)          IPv6-FlowLabel,
    (3)          IPv6-Length,
    (4)          IPv6-NextHeader,
    (5)          IPv6-HopLimit,
}

```

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
|                       |            | 393/668                     |

## COSEM Interface Classes

```

(6)          IPv6-Dev-Prefix,
(7)          IPv6-DevlID,
(8)          IPv6-AppPrefix,
(9)          IPv6-AppIID,
(10)         UDP-DevPort,
(11)         UDP-AppPort,
(12)         UDP-Length,
(13)         UDP-C checksum,
(14)         DLMS-Wrapper-version,
(15)         DLMS-Wrapper-SSAP,
(16)         DLMS-Wrapper-CSAP,
(17)         DLMS-Wrapper-Length
}

direction_indicator_type ::= enum
{
    (0)          Uplink,
    (1)          Downlink,
    (2)          bidirectional
}

matching_operator_type ::= enum
{
    (0)          equal,
    (1)          ignore,
    (2)          MSBx,
    (3)          match-mapping
}

compression_decompression_action_type ::= enum
{
    (0)          not-sent,
    (1)          value-sent,
    (2)          mapping-sent,
    (3)          LSB,
    (4)          compute-length,
    (5)          compute-checksum,
    (6)          DevlID,
    (7)          AppIID,
}

```

### 4.16.2.1.2.4 schc\_fr\_param

Contains the necessary parameters to support LPWAN Static Context Header Compression (SCHC) fragmentation and reassembly parameters.

```

schc_fr_param ::= structure

{
    rule_id_scheme:      enum
    {
        (0)  fixed-size,
        (1)  variable-size
    },
    max_packet_size:     long-unsigned,
    padding_l2_word_size: unsigned,
    padding_bits_value:  unsigned,
    delay_after_transmission: long-unsigned,
    interleaved_packet_tran: boolean,
}

```

## COSEM Interface Classes

```

        window_size:           unsigned,
        rule_params:          array rule_param_element
    }

    rule_param_element ::= structure
    {
        rule_id:             unsigned,
        rule_id_len:         unsigned,
        dtag_len:            unsigned,
        w_len:               unsigned,
        fcn_len:             unsigned,
        rcs_algorithm:       enum
        {
            (0)  None,
            (1)  CRC32 using 0xEDB88320
        },
        reliability_mode:   enum
        {
            (0)  No-ACK,
            (1)  ACK-on-Error,
            (2)  ACK-Always
        },
        retransmission_timer: long-unsigned,
        inactivity_timer:    long-unsigned,
        max_ack_request:    unsigned
    }
}

```

Where:

- rule\_id\_scheme              The RuleID numbering scheme, fixed-size or variable-size Rule IDs, the way the Rule ID is transmitted;
- max\_packet\_size             The maximum packet size that should ever be reconstructed by SCHC Decompression [octets];
- padding\_l2\_word\_size        The size of the L2 Word [bits];
- padding\_bits\_value         The value of the padding bits (0 or 1);
- delay\_after\_transmission   The delay to be added after transmission [milliseconds];
- interleaved\_packet\_tran   The support for interleaved packet transmission;
- window\_size                 The WINDOW\_SIZE, for modes that use windows;
- rule\_id\_len                 The length in bits for the RuleID header field of SCHC F/R message [bits];
- dtag\_len                    The length in bits for the Datagram Tag (DTag) header field of SCHC F/R message [bits];
- w\_len                      The length in bits for the W header field of SCHC F/R message [bits];
- fcn\_len                    The length in bits for the Fragment Compressed Number (FCN) header field of SCHC F/R message [bits];
- rcs\_algorithm              The algorithm for the calculation of Reassembly Check Sequence (RCS) field of SCHC F/R message;
- reliability\_mode          The reliability mode used;
- retransmission\_timer      The Retransmission Timer duration [milliseconds];
- inactivity\_timer          The Inactivity Timer duration [milliseconds];

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
| 395/668               |            |                             |

## COSEM Interface Classes

- max\_ack\_request      The MAX\_ACK\_REQUEST value.

C/D rules defined for the DLMS over SCHC profile are listed in Table 58.

**Table 46 – C/D Rule 1**

| <b>field_id</b>   | <b>field_length</b> | <b>field_position</b> | <b>direction_indicator</b> | <b>target_value</b> | <b>matching_operator</b> | <b>compression_decompression_action</b> |
|---|---------------------|-----------------------|----------------------------|---------------------|--------------------------|---|
| IPv6-version  | 4                   | 1                     | bidirectional              | 6                   | equal                    | not-sent                                |
| IPv6-DiffServ   | 8                   | 1                     | bidirectional              | 0                   | equal                    | not-sent                                |
| IPv6-FlowLabel  | 20                  | 1                     | bidirectional              | 0                   | equal                    | not-sent                                |
| IPv6-Length   | 16                  | 1                     | bidirectional              |                     | ignore                   | comp-length                             |
| IPv6-NextHeader   | 8                   | 1                     | bidirectional              | 17                  | equal                    | not-sent                                |
| IPv6-HopLimit   | 8                   | 1                     | bidirectional              | 255                 | equal                    | not-sent                                |
| IPv6-Dev-Prefix   | 64                  | 1                     | bidirectional              | FE80::/64           | equal                    | not-sent                                |
| IPv6-DevIID   | 64                  | 1                     | bidirectional              |                     | ignore                   | DevIID                                  |
| IPv6-AppPrefix  | 64                  | 1                     | bidirectional              | FE80::/64           | equal                    | not-sent                                |
| IPv6-AppIID   | 64                  | 1                     | bidirectional              | 1                   | ignore                   | not-sent                                |
| UDP-DevPort   | 16                  | 1                     | bidirectional              | 4059                | equal                    | not-sent                                |
| UDP-AppPort   | 16                  | 1                     | bidirectional              | 4059                | equal                    | not-sent                                |
| UDP-Length  | 16                  | 1                     | bidirectional              |                     | ignore                   | comp-length                             |
| UDP-C checksum  | 16                  | 1                     | bidirectional              |                     | ignore                   | comp-chk                                |
| DLMS-Wrapper-version  | 16                  | 1                     | bidirectional              | 1                   | equal                    | not-sent                                |
| DLMS-Wrapper-SSAP   | 16                  | 1                     | bidirectional              | 1                   | equal                    | not-sent                                |
| DLMS-Wrapper-CSAP   | 16                  | 1                     | bidirectional              | 1                   | equal                    | not-sent                                |
| DLMS-Wrapper-Length   | 16                  | 1                     | bidirectional              |                     | ignore                   | comp-length                             |
| NOTE If the device has to manage a second client, a second rule can be defined where the only field-id that is different will be DLMS-Wrapper-CSAP. |                     |                       |                            |                     |                          |   |

### 4.16.2.2 SCHC-LPWAN diagnostic (class\_id = 127, version = 0)

#### 4.16.2.2.1 Overview

Instances of this IC are intended to provide diagnostic information regarding the transport of SCHC packets over the LPWAN network that can be useful at the application layer to investigate potential communications problems and service performance.

| <b>SCHC-LPWAN diagnostic</b> |           | <b>0..n</b>          | <b>class_id = 127 , version = 0</b> |             |             |                   |
|------------------------------|-----------|----------------------|-------------------------------------|-------------|-------------|-------------------|
| <b>Attributes</b>            |           | <b>Data type</b>     | <b>Min.</b>                         | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 1. logical_name.             | (static)  | octet-string         |                                     |             |             | x                 |
| 2. schc_packets_tx_counter   | (dynamic) | double-long-unsigned |                                     |             | 0           | x + 0x08          |
| 3. schc_packets_rx_counter   | (dynamic) | double-long-unsigned |                                     |             | 0           | x + 0x10          |
| 4. schc_fragments_tx_counter | (dynamic) | double-long-unsigned |                                     |             | 0           | x + 0x18          |
| 5. schc_fragments_rx_counter | (dynamic) | double-long-unsigned |                                     |             | 0           | x + 0x20          |

## COSEM Interface Classes

|                                       |           |                      |  |  |   |          |
|---------------------------------------|-----------|----------------------|--|--|---|----------|
| 6. schc_ack_tx_counter                | (dynamic) | double-long-unsigned |  |  | 0 | x + 0x28 |
| 7. schc_ack_rx_counter                | (dynamic) | double-long-unsigned |  |  | 0 | x + 0x30 |
| <b>Specific methods (if required)</b> |           | <i>m/o</i>           |  |  |   |          |
| 1. reset (data)                       |           | ...                  |  |  |   | x + 0x40 |

### 4.16.2.3 Attribute description

#### 4.16.2.3.1 logical\_name

Identifies the “LPWAN Diagnostic” object instance. See 6.2.23.

#### 4.16.2.3.2 schc\_packets\_tx\_counter

It counts the SCHC non fragmented packet transmitted by the device.

#### 4.16.2.3.3 schc\_packets\_rx\_counter

It counts the SCHC non fragmented packet received by the device.

#### 4.16.2.3.4 schc\_fragments\_tx\_counter

It counts the SCHC fragments transmitted by the device.

#### 4.16.2.3.5 schc\_fragments\_rx\_counter

It counts the SCHC fragments received by the device.

#### 4.16.2.3.6 schc\_ack\_tx\_counter

It counts the ACK frame transmitted by the device.

#### 4.16.2.3.7 schc\_ack\_rx\_counter

It counts the ACK frame received by the device.

### 4.16.2.4 Method description

#### 4.16.2.4.1 reset (data)

This method resets all counters to 0.

```
data ::= integer (0)
```

## 4.17 LPWAN specific interface classes

### 4.17.1 General

Interface classes specified in this clause are related to a given LPWAN technology. Currently, interface classes related to LoRaWAN 1.0.3 network are specified. ICs for other LPWAN technologies will be added later.

## 4.17.2 LoRaWAN® interface classes

### 4.17.2.1 General

These ICs define the LoRaWAN 1.0.3 parameters used during registration and deregistration process, needed at the device side.

NOTE 1 DEVEUI is defined using the MAC address setup interface class (class\_id = 43).

NOTE 2 ABP (Activation by Personalization is not permitted).

### 4.17.2.2 LoRaWAN setup (class\_id = 128, version = 0)

#### 4.17.2.2.1 Overview

Instances of this interface class hold the parameters needed to set up a LoRaWAN 1.0.3 device.

This interface class mirrors the parametrization published in LoRa Alliance document: *LoRaWAN-TR006 DLMS End Device Monitoring Guidelines.doc*.

| LoRaWAN® setup                           | 0..n                 | class_id = 128 , version = 0 |      |      |            |
|--|----------------------|------------------------------|------|------|------------|
| Attributes                               | Data type            | Min.                         | Max. | Def. | Short name |
| 1. logical_name<br>(static)              | octet-string         |                              |      |      | x          |
| 2. class<br>(dynamic)                    | enum                 |                              |      |      | x + 0x08   |
| 3. state<br>(dynamic)                    | enum                 |                              |      |      | x + 0x10   |
| 4. max_transmit_EIRP_setting<br>(static) | integer              |                              |      |      | x + 0x18   |
| 5. ADR_mode<br>(dynamic)                 | boolean              |                              |      |      | x + 0x20   |
| 6. regional_parameters<br>(dynamic)      | enum                 |                              |      |      | x + 0x28   |
| 7. device_operation<br>(dynamic)         | structure            |                              |      |      | x + 0x38   |
| 8. modem_versions<br>(static)            | structure            |                              |      |      | x + 0x40   |
| 9. devAddr<br>(dynamic)                  | double-long-unsigned |                              |      |      | x + 0x48   |
| 10. join_strategy<br>(dynamic)           | enum                 |                              |      |      | x + 0x50   |
| 11. multicasts_parameters<br>(dynamic)   | array                |                              |      |      | x + 0x58   |
| Specific methods (if required)           | m/o                  |                              |      |      |            |
| 1. disconnect_from_network()             | ...                  |                              |      |      | x + 060    |
| 2. change_class(data)                    | ...                  |                              |      |      | x + 068    |
| 3. change_region(data)                   | ...                  |                              |      |      | x + 0x70   |

#### 4.17.2.2.2 Attribute description

##### 4.17.2.2.2.1 logical\_name

Identifies the “LoRaWAN setup” object instance. See 6.2.23.

##### 4.17.2.2.2.2 class

Defines the class of the device.

Read only attribute, use change\_class method to change the value.

```
enum:
  (0)  Class A,
  (1)  Class B,
  (2)  Class C
```

#### 4.17.2.2.2.3 state

Shows the state of the network connection process of the device.

```
enum:
  (0)  not connected,
  (1)  connecting state (joining),
  (2)  not connected, connected at least one time,
  (3)  connected,
  (4)  connection error (join error)
```

#### 4.17.2.2.2.4 max\_transmit\_EIRP\_setting

Read only parameter. Maximum transmission capability of the device in dBm.

#### 4.17.2.2.2.5 ADR\_mode

**Adaptative Data Rate** is a LoRaWAN 1.0.3 parameter allowing the network server to control the device datarate and TX power. Recommended for static devices (devices expected to not roam between networks); it optimises energy consumption and network capacity.

boolean:

|        |                 |
|--------|-----------------|
| TRUE:  | ADR is enabled  |
| FALSE: | ADR is disabled |

#### 4.17.2.2.2.6 regional\_parameters

Identifies the list of region names that a network and a device need to know, to open a LoRaWAN service. Those regions are used to support local radio regulation specifics, like duty cycle (or LBT), max time over air per message, max power, ...

They also govern the selection of default radio channels, default radio settings, the channel mode (fixed or dynamic, ...)

```
regional_parameters := enum
{
  (0)  EU868,
  (1)  US915,
  (2)  CN779 (Deprecates 2021-01-01),
  (3)  EU433,
  (4)  AU915,
  (5)  CN470,
  (6)  AS923-1,
  (7)  AS923-2,
  (8)  AS923-3,
  (9)  KR920,
  (10) IN865,
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 399/668 |
|-----------------------|------------|-----------------------------|---------|

```
    } (11) RU864
```

#### 4.17.2.2.7 device\_operation

Counters that can help to identify issues.

```
device_operation ::= structure
{
    TotalJoinRequestCounter: long unsigned,
    TimeSinceLastJoinRequest: long unsigned,
    TimeSinceLastJoinAccept: long unsigned
}
```

Where:

- TotalJoinRequestCounter: Number of join requests since last reset.
- TimeSinceLastJoinRequest: Number of seconds elapsed since the device initiated the connection process.
- TimeSinceLastJoinAccept: Number of seconds elapsed since the device connected to the network.

#### 4.17.2.2.8 modem\_versions

Identifies the modem hardware, software and protocol versions and the version of the regional parameters.

```
versions ::= structure
{
    HardwareVersion: CHOICE
    {
        octet-string,
        visible-string
    },
    SoftwareVersion: CHOICE
    {
        octet-string,
        visible-string
    },
    RegionalParametersVersion: CHOICE
    {
        octet-string,
        visible-string
    },
    ProtocolVersion: CHOICE
    {
        octet-string,
        visible-string
    }
}
```

#### 4.17.2.2.9 devAddr

32 bits device address identifying the device inside the current network. Assigned by the Network Server of the device. Should be 0 when not joined.

**4.17.2.2.2.10 join\_strategy**

Defines the algorithm, timing/DR to apply during the joining.

To be documented by vendor. Vendor can assign a code to a Join strategy (where applying).

**4.17.2.2.2.11 multicasts\_parameters**

Both DLMS, IP and LoRaWAN are multicast technologies, LoRaWAN 1.0.3 supporting up to four multicast groups. One or more group(s) can be created for a set of DLMS devices then SCHC gateway will forward all IP multicast data corresponding to that group to the end devices.

LoRa Alliance has published a recommended application layer messaging definition allowing to remotely manage devices groups, and program class B or C distribution window for a given group.

```

multicasts_parameters ::= array multicast_parameters

multicast_parameters ::= structure

{
  mc_addr:           double-long-unsigned,
  mc_key:            octet-string,
  mc_min_fcount:    double-long-unsigned,
  mc_max_fcount:    double-long-unsigned,
  mc_start_time:    date-time,
  mc_duration:      long-unsigned,
  mc_class:          enum,
  mc_downlink_datarate: unsigned-integer
  mc_downlink_frequency: double-long-unsigned
}

```

Where:

- mc\_addr: multicast device address;  
NOTE 1 Should be 0 when multicast group is not used,32 bits
- mc\_key: key used to encode the payload of the multicast group;  
NOTE 2 128 bits
- mc\_min\_fcount: minimum multicast Fcounter, device should drop payload with lower FCount;
- mc\_max\_fcount: maximum multicast Fcounter, device should drop payload with higher FCount
- mc\_start\_time: Time to start the multicast session, if applicable;
- mc\_duration: Maximum duration of the multicast session;
- mc\_class: Device shall switch to the class during the length of the multicast window;  
enum:  
(0) Class B,  
(1) Class C  
All others reserved
- mc\_downlink\_datarate: Datarate of the multicast session;

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 401/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

- mc\_downlink\_frequency: Frequency of the multicast session, in Hz.

An end-device supports a maximum of 4 simultaneous groups, and a minimum of 0. Up to 4 elements may be used in the array.

### **4.17.2.2.3 Method description**

#### **4.17.2.2.3.1 disconnect\_from\_network()**

This method corresponds to a LoRaWAN 1.0.3 new join procedure.

```
data ::= integer (0)
```

#### **4.17.2.2.3.2 change\_class (data)**

This method is used to change the class attribute. Data is an enum.

#### **4.17.2.2.3.3 change\_region (data)**

This method is used to change locally the Regional Parameter. Can be used only in disconnected state.

### **4.17.2.3 LoRaWAN diagnostic (class\_id = 129, version = 0)**

#### **4.17.2.3.1 Overview**

Instances of this IC are intended to provide diagnostic information regarding the LoRaWAN 1.0.3 lower layers to investigate potential communications problems and service performance.

| <b>LoRaWAN diagnostic</b>             | <b>0..n</b>          | <b>class_id = 129 , version = 0</b> |             |             |                   |
|---------------------------------------|----------------------|-------------------------------------|-------------|-------------|-------------------|
| <b>Attributes</b>                     | <b>Data type</b>     | <b>Min.</b>                         | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 1. logical_name (static)              | octet-string         |                                     |             |             | x                 |
| 2. internal_error_code (dynamic)      | enum                 |                                     |             |             | x + 0x08          |
| 3. out_frames_u_counter (dynamic)     | double-long-unsigned |                                     |             | 0           | x + 0x10          |
| 4. out_frames_c_counter (dynamic)     | double-long-unsigned |                                     |             | 0           | x + 0x18          |
| 5. in_frames_u_counter (dynamic)      | double-long-unsigned |                                     |             | 0           | x + 0x20          |
| 6. in_frames_c_counter (dynamic)      | double-long-unsigned |                                     |             | 0           | x + 0x28          |
| 7. in_mac_command_counter (dynamic)   | double-long-unsigned |                                     |             | 0           | x + 0x30          |
| 8. in_mac_ans_error_counter (dynamic) | double-long-unsigned |                                     |             | 0           | x + 0x38          |
| 9. in_mac_ignored_counter (dynamic)   | double-long-unsigned |                                     |             | 0           | x + 0x40          |
| 10. in_per (dynamic)                  | integer              |                                     |             |             | x + 0x48          |
| 11. in_mean_rssi_rx1 (dynamic)        | integer              |                                     |             |             | x + 0x50          |
| 12. in_mean_snr_rx1 (dynamic)         | integer              |                                     |             |             | x + 0x58          |
| 13. in_mean_rssi_rx2 (dynamic)        | integer              |                                     |             |             | x + 0x60          |
| 14. in_mean_snr_rx2 (dynamic)         | integer              |                                     |             |             | x + 0x68          |
| <b>Specific methods (if required)</b> | <b>m/o</b>           |                                     |             |             |                   |
| 1. reset (data)                       | ...                  |                                     |             |             |                   |

**4.17.2.3.2 Attribute description****4.17.2.3.2.1 logical\_name**

Identifies the “LPWAN Diagnostic” object instance. See 6.2.23.

**4.17.2.3.2.2 internal\_error\_code**

Fault condition identifier.

To be documented by vendor. When an end-device is faulty, vendor can assign a code to the fault condition.

**4.17.2.3.2.3 out\_frames\_u\_counter**

Number of frames sent unconfirmed.

**4.17.2.3.2.4 out\_frames\_c\_counter**

Number of frames sent confirmed.

**4.17.2.3.2.5 in\_frames\_u\_counter**

Number of frames received unconfirmed.

**4.17.2.3.2.6 in\_frames\_c\_counter**

Number of frames received confirmed.

**4.17.2.3.2.7 In\_mac\_command\_counter**

Number of MAC command received from the network. Note : all commands are answered.

**4.17.2.3.2.8 in\_mac\_ans\_error\_counter**

Number of MAC answers replied with error.

**4.17.2.3.2.9 in\_mac\_ans\_ignored\_counter**

Number of MAC commands ignored.

**4.17.2.3.2.10 in\_per**

Per : packet error rate. Computed over the 5 received downlink.

Unit is percentage (56).

**4.17.2.3.2.11 in\_mean\_rssi\_rx1**

Computed over the last 5 received downlink on rx1.

Unit is dBm.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 403/668 |
|-----------------------|------------|-----------------------------|---------|

**4.17.2.3.2.12 in\_mean\_snr\_rx1**

Computed over the last 5 received downlink on rx1.

Unit is dB.

**4.17.2.3.2.13 in\_mean\_rssi\_rx2**

Computed over the last 5 received downlink on rx2.

Unit is dBm.

**4.17.2.3.2.14 in\_mean\_snr\_rx2**

Computed over the last 5 received downlink on rx2.

Unit is dB.

**4.17.2.3.3 Method description****4.17.2.3.3.1 reset (data)**

This method resets all counters to 0.

```
data ::= Integer (0)
```

**4.18 Interface classes for setting up and managing the DLMS/COSEM profile for Wi-SUN networks****4.18.1 Wi-SUN setup (class\_id = 95, version 0)****4.18.1.1 Overview**

This IC has the purpose of modelling the Wi-SUN network setup parameters.

All attributes in [FANSPEC] and [PHYSPEC] are written in camelCase and for the benefit of easy traceability these parameters are transposed into snake\_case (underscore notation) within the DLMS/COSEM specification.

| Wi-SUN setup                            | 0..n          | class_id = 95, version = 0 |       |      |            |  |
|---|---------------|----------------------------|-------|------|------------|--|
| Attributes                              | Data type     | Min.                       | Max.  | Def. | Short name |  |
| 1. logical_name<br>(static)             | octet-string  |                            |       |      | x          |  |
| 2. network_name<br>(static)             | octet-string  |                            |       |      | x + 0x08   |  |
| 3. routing_method<br>(dynamic)          | enum          | 0                          | 1     | 1    | x + 0x10   |  |
| 4. pan_id<br>(dynamic)                  | long-unsigned | 0                          | 65534 |      | x + 0x18   |  |
| 5. disc_imin<br>(static)                | unsigned      | 1                          | 255   |      | x + 0x20   |  |
| 6. disc_imax<br>(static)                | unsigned      | 1                          | 8     |      | x + 0x28   |  |
| 7. data_message_imin<br>(static)        | unsigned      | 1                          | 255   | 10   | x + 0x30   |  |
| 8. data_message_imax<br>(static)        | unsigned      | 1                          | 8     | 3    | x + 0x38   |  |
| 9. default_dio_interval_min<br>(static) | unsigned      | 1                          | 255   |      | x + 0x40   |  |

## COSEM Interface Classes

|                                       |           |            |   |   |  |          |
|---------------------------------------|-----------|------------|---|---|--|----------|
| 10. default_dio_interval_doublings    | (static)  | unsigned   | 1 | 8 |  | x + 0x48 |
| 11. channel_plan                      | (static)  | structure  |   |   |  | x + 0x50 |
| 12. channel_function                  | (static)  | enum       | 0 | 3 |  | x + 0x58 |
| 13. excluded_channels                 | (static)  | CHOICE     |   |   |  | x + 0x60 |
| 14. join_state                        | (dynamic) | enum       | 1 | 5 |  | x + 0x68 |
| 15. reg_channel_exclusions            | (static)  | structure  |   |   |  | x + 0x70 |
| <b>Specific methods (if required)</b> |           | <b>m/o</b> |   |   |  |          |
| 1. reset_join_state (data)            |           | o          |   |   |  | x + 0x78 |

### 4.18.1.2 Attribute description

#### 4.18.1.2.1 logical\_name

Identifies the “Wi-SUN setup” object instance. See 6.2.32.

#### 4.18.1.2.2 network\_name

Provides the name of the network that the device is allowed to join, or is a member of.

This is represented by ASCII characters.

NOTE The network name has a maximum of 32 octets.

#### 4.18.1.2.3 routing\_method

Provides the routing method employed by the device, two options are available:

enum:

- (0) Layer 2 routing,
- (1) Layer 3 routing

NOTE Default value according to [FANSPEC] is 1.

#### 4.18.1.2.4 pan\_id

Provides the PAN ID (configured at the Border Router and is known by every node on the network. This attribute is not controlled in the application layer.

NOTE 1 See [FANSPEC] , 6.3.1.1.

NOTE 2 IEEE 802.15.4 reserves the value 65 535 for the broadcast PAN ID.

#### 4.18.1.2.5 disc\_imin

Provides the time in seconds of the time-out for network discovery.

NOTE Typical values of 60s for large PANs (larger than 100 devices) or 15s for small PANs (less than 100 devices) may be employed.

#### 4.18.1.2.6 disc\_imax

Provides the number of doublings of the value of “imin” (1-8 doublings of disc\_imin).

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 405/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

NOTE Typical value of 4 for large PANs (approximately 16 minutes), value of 2 for small PANs (approximately 60 seconds).

### 4.18.1.2.7 data\_message\_imin

Provides the network broadcast timeout minimum value in seconds.

NOTE Typical default value of 10s.

### 4.18.1.2.8 data\_message\_imax

Provides the maximum broadcast timeout as a function of data\_message\_imin (1-8 doublings of data\_message\_imin).

NOTE Typical default value of 3 (80 seconds).

### 4.18.1.2.9 default\_dio\_interval\_min

Provides the ROLL RPL DIO minimum Interval (imin) The DIO trickle timer imin value is defined as 2 to the power of this value in ms (milliseconds).

NOTE Typical default value for a large network: 19 (an imin of approximately 8 minutes) and a small network: 15 (an imin of approximately 32 seconds).

### 4.18.1.2.10 default\_dio\_interval\_doublings

Provides the ROLL RPL DIO Interval IMAX and is a number of doublings of the value of default\_dio\_interval\_min (1-8 doublings). A value of 1 for large PANs (approximately 16 minutes i.e. 8 x 2 minutes), value of 2 for small PANs (approximately 128 seconds i.e 32 x 2 x 2).

### 4.18.1.2.11 channel\_plan

The channel plan sets the channel spacing, channel 0 frequency and number of channels plus the symbol rate.

Regional spectrum settings as specified in [PHYSPEC], Table 3. The values of regulatory\_domain and operating\_class uniquely identify the combination of operating parameters that the implementation is programmed to sustain. These parameters are subject to geographic regulation and may change from time to time depending on when regional spectrum authorities update their national frequency tables.

```
channel_plan ::= structure
{
    regulatory_domain_identifier:      unsigned,
    operating_class_designator:       unsigned
}
```

Where:

- regulatory\_domain\_identifier identifies the regulatory domain that the product is operating in, thereby identifying various PHY layer parameters including frequency bands, PHY modes, and channel spacing according to [PHYSPEC], Table 3,
- operating\_class\_designator identifies the operating class of the device according to [PHYSPEC], Table 3. Operating class is unique only within the regulatory domain.

### 4.18.1.2.12 channel\_function

Provides the channel function. There are currently 4 channel functions: fixed channel or a channel hopping mode where certain other parameters are specified as enumerated below.

enum:

| Value | Description    | Note   |
|-------|----------------|--|
| 0     | Fixed Channel  | The transmitting node uses a single, fixed channel from the available channels.  |
| 1     | TR51CF         | The transmitting node uses the channel function described in ANSI/TIA-4957.200 (TR51CF), 7.1.  |
| 2     | DH1CF          | The transmitting node uses the Direct Hash channel function (DH1CF described in [FANSPEC], 6.3.4.5).   |
| 3     | Vendor defined | The transmitting node uses a vendor defined channel function explicitly provided by the Hop Count and Hop List (defined in [FANSPEC] 6.3.2.3.2.1.4). |
| Other | reserved       |  |

#### 4.18.1.2.13 excluded\_channels

Provides a means via which a node can have either a range (excluded\_channel\_ranges) or individual channels (excluded\_channel\_mask) notched out (not visited) of the node's channel hop sequence. Excluded\_channel\_ranges and excluded\_channel\_mask are configured for device specific purposes and shall be a subset of the channel\_plan that fulfils regulatory channel restriction requirements. If no excluded channel ranges are set then the attribute shall be set to null-data.

Either excluded\_channel\_range OR excluded\_channel\_mask may be configured in a mutually exclusive way.

```

excluded_channels: CHOICE
{
    null-data [0],
    excluded_channel_ranges,
    excluded_channel_mask
}

null-data is selected when there are no excluded channels.

excluded_channel_ranges ::= structure
{
    number_of_ranges:    integer,
    ranges_list:          array channel_range
}
channel_range ::= structure
{
    start_channel_number:    long unsigned,
    end_channel_number:      long unsigned
}

```

Where:

- number\_of\_ranges is the number of ranges in the ranges\_list;
- ranges\_list is an array of start and end channel numbers representing one or multiple excluded channel ranges;

- channel\_range is a structure of start and end channel numbers that together represent an excluded channel range. The start and end channel shall also be considered to be included in the excluded range.

NOTE 1 Channels which are illegal or invalid may still not be used even if they are not included in a channel\_range.

excluded\_channel\_mask ::= bit-string

Excluded\_channel\_mask is a variable length bit\_string (varying in multiples of 8) representing the series of channels that are included or not included in the device's channel hop sequence. The Excluded Channel Mask field provides a means via which a node explicitly reports, for all channels within its channel hop sequence, which channels are and are not visited.

| Bit           | Channel |
|---------------|---------|
| 0 (first bit) | 0       |
| 1             | 1       |
| 2             | 2       |
| 3             | 3       |
| ....          | ....    |
| n             | n       |

NOTE 2 [FANSPEC] states "If a bit in the mask is set to 1, the corresponding channel MUST NOT be used. Bits that are not valid channels (unused following rounding to a multiple of 8) MUST be set to 1 on transmit and ignored on receipt."

This attribute is to be used for efficiently notching out individual channels rather than a range of channels within a band plan.

#### 4.18.1.2.14 join\_state

Provides the join state of the device.

enum:

- (0) Initialisation state,
- (1) Select PAN,
- (2) Authenticate,
- (3) Acquire PAN Config,
- (4) Configure Routing,
- (5) Operational,
- (6)...(255) reserved

NOTE Applications on a device should only send or receive traffic when the device is in join state 5.

#### 4.18.1.2.15 reg\_channel\_exclusions

Regional channel exclusions as designated in [FANSPEC]. The values of regulatory\_domain and operating\_class from channel\_plan uniquely identify the combination of operating parameters that the implementation is programmed to sustain. This field identifies any specific channels that must not be used when operating in a specific country.

```
reg_channel_exclusions ::= structure
{
    num_channels:      long_unsigned,
    excluded_channels: bit-string
}
```

Where:

- num\_channels is defined as the total number of channels, according to [PHYSPEC], Table 3 when operating with the regulatory\_domain and operating\_class defined in channel\_plan,
- excluded\_channels is a bit-string, sized as num\_channels, where the first bit, index 0 is Channel 0 and the last bit is “num\_channels – 1”. For each index into excluded\_channels, a value of “0” indicates the channel can be used and a value of “1” indicates the channel must not be used.

**NOTE** Unlike the Channel Exclusion in the [FANSPEC] , the regulatory channel exclusion is permanent and cannot be modified after configuration. These channels represent regulatory channel exclusions that apply to the particular country or region in that the configured device is deployed.

#### 4.18.1.3 Method description

##### 4.18.1.3.1 reset\_join\_state (data)

Resets join state attribute to 0.

### 4.18.2 Wi-SUN diagnostic (class\_id = 96, version 0)

#### 4.18.2.1 Overview

Instances of this IC are intended to provide diagnostic information regarding the Wi-SUN network that can be useful to investigate potential communication problems and service performance.

To facilitate management from the application, the following diagnostics are provided.

| Wi-SUN diagnostics                          | 0..n         | class_id = 96, version = 0 |      |      |            |
|---|--------------|----------------------------|------|------|------------|
| Attributes                                  | Data type    | Min.                       | Max. | Def. | Short name |
| 1. logical_name<br>(static)                 | octet-string |                            |      |      | x          |
| 2. sysdescr<br>(dynamic)                    | octet-string |                            |      |      | x + 0x08   |
| 3. ifnumber<br>(dynamic)                    | integer      |                            |      | 1    | x + 0x10   |
| 4. iftable<br>(dynamic)                     | array        |                            |      |      | x + 0x18   |
| 5. neighbour_table_information<br>(dynamic) | array        |                            |      |      | x + 0x20   |
| 6. transmission_information<br>(dynamic)    | array        |                            |      |      | x + 0x28   |
| <b>Specific methods (if required)</b>       | <b>m/o</b>   |                            |      |      |            |
| 1. reset                                    | o            |                            |      |      | x + 0x30   |

#### 4.18.2.2 Attribute description

##### 4.18.2.2.1 logical\_name

Identifies the “Wi-SUN diagnostic” object instance. See 6.2.32.

##### 4.18.2.2.2 sysdescr

A textual description of the entity. This value should include the full name and version identification of the system's hardware type, software operating-system, and networking software. It is mandatory that this only contains printable ASCII characters.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 409/668 |
|-----------------------|------------|-----------------------------|---------|

#### 4.18.2.2.3 ifnumber

The number of interfaces (ifentry) present within the physical device as per RFC 1213. Each interface shall have a value of ifentry defined in iftable.

#### 4.18.2.2.4 iftable

A list of interface entries as per RFC 1213. The number of entries in the ifentry array is given by the value of attribute ifnumber; normally this shall be 1.

```
array ifentry

ifentry ::=structure
{
    ifdescr          octet-string,
    ifphysaddress   octet-string,
    ifinoctets       double-long-unsigned,
    ifinucastpackets double-long-unsigned,
    ifinnucastpackets double-long-unsigned,
    ifinerrors       double-long-unsigned,
    ifinunknownprotos double-long-unsigned,
    ifoutoctets      double-long-unsigned,
    ifoutucastpackets double-long-unsigned,
    ifoutnucastpackets double-long-unsigned,
    ifouterrors      double-long-unsigned,
    ifoutQlen        double-long-unsigned,
}
```

Where:

| Item               | description   |
|--------------------|---|
| ifdescr            | An ASCII description of the interface as per RFC 1213. A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the hardware interface.  |
| ifphysaddress      | The interface's address at the protocol layer immediately 'below' the network layer in the protocol stack. For interfaces which do not have such an address (e.g., a serial line), this object should contain an empty octet string. In the case of Wi-SUN this entry shall be the EUI64. |
| ifinoctets         | The total number of octets received on the interface, including framing characters.   |
| ifinucastpackets   | The number of subnetwork-unicast packets delivered to a higher-layer protocol. In the case of Wi-SUN diagnostic this shall be the number of packets passed to the DLMS/COSEM application layer as per RFC 1213.   |
| ifinnucastpackets  | The number of non-unicast (i.e., subnetwork-broadcast or subnetwork-multicast) packets delivered to a higher-layer protocol as per RFC 1213.  |
| ifinerrors         | The number of inbound packets that contained errors preventing them from being deliverable to a higher layer protocol as per RFC 1213.  |
| ifunknowntprotos   | The number of packets received via the interface at the network layer which were discarded because of an unknown or unsupported protocol as per RFC 1213.   |
| ifoutoctets        | The total number of octets transmitted out of the interface at the network layer, including framing characters as per RFC 1213.   |
| ifoutucastpackets  | The total number of packets that higher-level protocols requested be transmitted to a subnetwork-unicast address, including those that were discarded or not sent. In the case of Wi-SUN diagnostic higher layer protocols shall be defined as the DLMS/COSEM application layer.          |
| ifoutnucastpackets | The total number of packets that higher-level protocols requested be transmitted to a non-unicast (i.e., a subnetwork-broadcast or subnetwork-multicast) address, including those that were discarded or not sent. In the case of Wi-SUN diagnostic higher layer                          |

## COSEM Interface Classes

|             |   |
|-------------|---|
|             | protocol shall be defined as the DLMS/COSEM application layer as per RFC 1213.  |
| ifouterrors | The number of outbound packets that could not be transmitted because of errors. |
| ifoutqlen   | The length of the output packet queue (in packets) at the network layer,        |

### 4.18.2.2.5 neighbour\_table\_information

The neighbour table keeps track of devices no more than 1 hop away.

```

array neighbour

neighbour ::= structure

{
    neighbour_id: octet-string,
    neighbour_type: enum,
    device_rank: unsigned,
    rssi: long-unsigned,
    etx_to_parent: double-long-unsigned
}

```

Where:

- neighbour\_id is the EUI 64 of the neighbour device. The EUI-64 shall be of the modified EUI-64 format described in RFC 4291;
- neighbour\_type is an enumeration defining whether the neighbour device is a parent, child or sibling.

```

enum:
    (0) Undetermined,
    (1) Parent,
    (2) Child,
    (3) Sibling

```

- device\_rank is a function of distance in hops of the device to the root device in the DODAG. The smaller the number the closer the device is to the DODAG root device;
- rssi is a measure of the signal strength of the neighbour;
- etx\_to\_parent is the Expected Number of transmissions required from the device to the destination via the parent in question required to transmit a packet. 1 is the perfect situation and FFFFFFFF is assumed to be equal to infinity and therefore a non-functioning link.

The first entry in the array of neighbours shall relate to the preferred parent and all subsequent entries shall be secondary parents.

NOTE Parent represents the node in the routing nearer to the Border Router (and may include the Border Router), child represents a node in the routing that has some dependency on this node, and sibling represents a node where communications is possible but is neither the parent or child of this node.

### 4.18.2.2.6 transmission\_information

Provides transmission information and quality of service per interface.

```

array      transmission_element

```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 411/668 |
|-----------------------|------------|-----------------------------|---------|

```

transmission_element ::= structure

{
    neighbour_id          octet-string,
    number_of_receptions   double-long-unsigned,
    number_of_sucessful_trans double-long-unsigned,
    number_of_failed_trans    double-long-unsigned
}

```

Where:

- neighbour\_id is the EUI-64 of the FAN interface. The EUI-64 shall be of the modified EUI-64 format described in RFC 4291;
- number\_of\_receptions is a simple counter of the total number of receptions from the device identified by interface\_id;
- number\_of\_sucessful\_trans is a simple counter of the total number of successful transmissions acknowledged by the device identified by interface\_id;
- number\_of\_failed\_trans is a simple counter of the total number of failed transmissions acknowledged by the device identified by interface\_id.

#### **4.18.2.3 Method description**

##### **4.18.2.3.1 reset (data)**

Clears attributes 2...4 of the object instance, that is:

```

- sysdescr,
- ifnumber,
- iftable
data ::= integer (0)

```

#### **4.18.3 RPL diagnostic (class\_id = 97, version 0)**

##### **4.18.3.1 Overview**

As per [FANSPEC] 6.2.3.1.8 RPL is supported for the dissemination of IPv6 multicast packets. Instances of this IC are intended to provide diagnostic information regarding the Wi-SUN network from RPL diagnostic parameters as per RFC 6550 that can be useful to investigate potential communication problems and service performance issues.

## COSEM Interface Classes

| RPL diagnostic                        | 0..n       | class_id = 97, version = 0 |      |      |            |
|---------------------------------------|------------|----------------------------|------|------|------------|
| Attributes                            | Data type  | Min.                       | Max. | Def. | Short name |
| 1. logical_name                       | (static)   | octet-string               |      |      | x          |
| 2. rpl_instance_id                    | (dynamic)  | unsigned                   | 0    | 191  | x + 0x08   |
| 3. dodag_version_number               | (dynamic)  | unsigned                   |      |      | x + 0x10   |
| 4. dodag_rank                         | (dynamic)  | long                       |      |      | x + 0x18   |
| 5. grounded                           | (dynamic)  | boolean                    |      |      | x + 0x20   |
| 6. mode_of_operation                  | (dynamic)  | enum                       |      |      | x + 0x28   |
| 7. dodag_prf                          | (dynamic)  | unsigned                   | 0    | 7    | x + 0x30   |
| 8. dodag_dtsn                         | (dynamic)  | unsigned                   |      |      | x + 0x38   |
| 9. dodag_id                           | (dynamic)  | octet-string               |      |      | x + 0x40   |
| <b>Specific methods (if required)</b> | <b>m/o</b> |                            |      |      |            |
| 1. reset (data)                       | o          |                            |      |      | x + 0x48   |

### 4.18.3.2 Attribute description

#### 4.18.3.2.1 logical\_name

Identifies the “RPL diagnostic” object instance. See 6.2.32.

#### 4.18.3.2.2 rpl\_instance\_id

Holds the RPL instance ID as defined in RFC 6550. The values below are defined in RFC 6550 to identify whether or not the RPL Instance ID relates to a Global Instance ID, as in the case of coordinated networks or Local Instance ID as issued by a DODAG in the case of unilateral networks.

Bits 0 and 1 of the unsigned value act as flags where bit 0 when clear defines that the rpl\_instance\_id is global and has 127 possible values. Bit 0 set means that the rpl\_instance\_id is local, and in such case bit 1 shall always be clear assuming that the server is the recipient of messages from the rpl\_instance\_id allocating DODAG.

| Instance ID Type   | Start value | End value |
|--------------------|-------------|-----------|
| Global Instance ID | 0           | 127       |
| Local Instance ID  | 128         | 191       |
| Not Used           | 192         | 255       |

#### 4.18.3.2.3 dodag\_version\_number

8-bit unsigned integer set by the DODAG root to the DODAGVersionNumber. RFC 6550, 8.2 describes the rules for DODAGVersionNumbers and how they affect DIO processing.

#### 4.18.3.2.4 dodag\_rank

Indicates the DODAG Rank of the node sending the DIO message. RFC 6550, 8.2 describes how Rank is set and how it affects DIO processing.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 413/668 |
|-----------------------|------------|-----------------------------|---------|

**4.18.3.2.5 grounded**

The Grounded 'G' flag indicates whether the DODAG advertised can satisfy the application-defined goal. If the flag is set, the DODAG is grounded. If the flag is cleared, the DODAG is floating.

**4.18.3.2.6 mode\_of\_operation**

The Mode of Operation (MOP) field identifies the mode of operation of the RPL instance as administratively provisioned at and distributed by the DODAG root. All nodes that join the DODAG shall honour the MOP in order to fully participate as a router, or else they must only join as a leaf. MOP is encoded as detailed below:

- enum: (0) No Downward routes maintained by RPL,
- (1) Non-Storing Mode of Operation,
- (2) Storing Mode of Operation with no multicast support,
- (3) Storing Mode of Operation with multicast support.

**4.18.3.2.7 dodag\_prf**

Defines how preferable the root of this DODAG is compared to other DODAG roots within the instance. DAG Preference ranges from 0x00 (least preferred) to 0x07 (most preferred). The default is 0 (least preferred). RFC 6550, 8.2 describes how DAG Preference affects DIO processing.

**4.18.3.2.8 dodag\_dtsn**

Destination Advertisement Trigger Sequence Number (DTSN) 8-bit unsigned integer set by the node issuing the DIO message. The Destination Advertisement Trigger Sequence Number (DTSN) flag is used as part of the procedure to maintain Downward routes. The details of this process are described in RFC 6550, 9.

**4.18.3.2.9 dodag\_id**

128-bit IPv6 address represented as octet-string (length 16) set by a DODAG root that uniquely identifies a DODAG. The DODAGID MUST be a routable IPv6 address belonging to the DODAG root.

**4.18.3.3 Method description****4.18.3.3.1 reset (data)**

Clears or (sets to default value) all dynamic fields of the object instance.

```
data ::= integer (0)
```

**4.18.4 MPL diagnostic (class\_id = 98, version 0)****4.18.4.1 Overview**

Instances of this IC are intended to provide diagnostic information regarding the Wi-SUN network from MPL diagnostic parameters as per RFC 7731 that can be useful at the application layer to investigate potential communication problems and service performance issues.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 414/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

| MPL diagnostic                        | 0..n          | class_id =98 , version = 0 |         |      |            |
|---------------------------------------|---------------|----------------------------|---------|------|------------|
| Attributes                            | Data type     | Min.                       | Max.    | Def. | Short name |
| 1. logical_name (static)              | octet-string  |                            |         |      | x          |
| 2. proactive_forwarding (dynamic)     | boolean       |                            |         |      | x + 0x08   |
| 3. z (dynamic)                        | unsigned      | 0                          | 0       | 0    | x + 0x18   |
| 4. tunit (dynamic)                    | unsigned      | 0x01                       | 0xFE    |      | x + 0x20   |
| 5. se_lifetime (dynamic)              | long-unsigned | 0x0001                     | 0xFFFFE |      | x + 0x28   |
| 6. dm_k (dynamic)                     | unsigned      |                            |         |      | x + 0x30   |
| 7. dm_imin (dynamic)                  | long-unsigned | 0x0001                     | 0xFFFFE |      | x + 0x38   |
| 8. dm_imax (dynamic)                  | unsigned      | 0x01                       | 0xFE    |      | x + 0x40   |
| 9. dm_t_exp (dynamic)                 | long-unsigned | 0x0001                     | 0xFFFFE |      | x + 0x48   |
| 10. c_k (dynamic)                     | unsigned      |                            |         |      | x + 0x50   |
| 11. c_imin (dynamic)                  | long-unsigned | 0x0001                     | 0xFFFFE |      | x + 0x58   |
| 12. c_imax (dynamic)                  | unsigned      | 0x01                       | 0xFE    |      | x + 0x60   |
| 13. c_t_exp (dynamic)                 | long-unsigned | 0x0001                     | 0xFFFFE |      | x + 0x68   |
| <b>Specific methods (if required)</b> | <b>m/o</b>    |                            |         |      |            |
| 1. reset (data)                       | o             |                            |         |      | x + 0x70   |

### 4.18.4.2 Attribute description

#### 4.18.4.2.1 logical\_name

Identifies the “MPL diagnostic” object instance. See 6.2.32.

#### 4.18.4.2.2 proactive\_forwarding

A flag to indicate PROACTIVE\_FORWARDING.

NOTE This flag is set if PROACTIVE\_FORWARDING = TRUE, as per RFC 7774, 2.1.

#### 4.18.4.2.3 z

Reserved for future use. Defined as per RFC 7774, 2.1.

NOTE This is set to zero in all DHCP Servers, DHCP Clients shall ignore any none zero values.

#### 4.18.4.2.4 tunit

Unit time of timer parameters (SE\_LIFETIME and \*\_IMIN) in this option.

NOTE Values 0 and 0xFF are reserved and are not be used, as per RFC 7774, 2.1.

#### 4.18.4.2.5 se\_lifetime

SEED\_SET\_ENTRY\_LIFETIME/TUNIT, in milliseconds (ms).

NOTE 0 and 0xFFFF are reserved and are not to be used, as per RFC 7774, 2.1.

#### 4.18.4.2.6 dm\_k

DATA\_MESSAGE\_K, as per RFC 7774, 2.1.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 415/668 |
|-----------------------|------------|-----------------------------|---------|

**4.18.4.2.7 dm\_imin**

DATA\_MESSAGE\_IMIN/TUNIT, in milliseconds (ms).

NOTE Values 0 and 0xFFFF are reserved and are not to be used, as per RFC 7774, 2.1.

**4.18.4.2.8 dm\_imax**

DATA\_MESSAGE\_IMAX. The actual maximum timeout is described as a number of doublings of DATA\_MESSAGE\_IMIN, as described in RFC 6206, 4.1.

NOTE Values 0 and 0xFF are reserved and MUST NOT be used, as per RFC 7774, 2.1.

**4.18.4.2.9 dm\_t\_exp**

DATA\_MESSAGE\_TIMER\_EXPIRATIONS, as per RFC 7774, 2.1.

NOTE Values 0 and 0xFFFF are reserved and are not to be used, as per RFC 7774, 2.1.

**4.18.4.2.10 c\_k**

CONTROL\_MESSAGE\_K, as per RFC 7774, 2.1.

**4.18.4.2.11 c\_imin**

CONTROL\_MESSAGE\_IMIN/TUNIT, in milliseconds (ms).

NOTE Values 0 and 0xFFFF are reserved and are not to be used, as per RFC 7774.

**4.18.4.2.12 c\_imax**

CONTROL\_MESSAGE\_IMAX. The actual maximum timeout is described as a number of doublings of CONTROL\_MESSAGE\_IMIN.

NOTE Values 0 and 0xFF are reserved and are not to be used, as per RFC 7774.

**4.18.4.2.13 c\_t\_exp**

CONTROL\_MESSAGE\_TIMER\_EXPIRATIONS.

NOTE Values 0 and 0xFFFF are reserved and are not to be used, as per RFC 7774.

**4.18.4.3 Method description****4.18.4.3.1 reset (data)**

Clears all dynamic fields of the object instance.

`data ::= integer (0)`

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 416/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## 4.19 Interface classes for setting up and managing the DLMS/COSEM profile for ISO/IEC 14908 PLC networks

### 4.19.1 General

The interface classes specified here are used in devices implementing the IEC 62056-8-8:2020 DLMS/COSEM communication profile for ISO/IEC 14908 series networks.

### 4.19.2 ISO/IEC 14908 identification (class\_id = 130, version = 0)

#### 4.19.2.1 Overview

Instances of the ISO/IEC 14908 identification IC allow the identification of the device and the network to which the device is connected.

| ISO/IEC14908 identification   | 0...n        | class_id = 130, version = 0 |      |      |            |  |
|-------------------------------|--------------|-----------------------------|------|------|------------|--|
| Attributes                    | Data type    | Min.                        | Max. | Def. | Short name |  |
| 1. logical_name<br>(static)   | octet-string |                             |      |      | x          |  |
| 2. node_ID<br>(static.)       | unsigned     | 1                           | 127  |      | x + 0x08   |  |
| 3. subnet_ID<br>(static)      | unsigned     | 1                           | 255  |      | x + 0x10   |  |
| 4. domain_ID<br>(static)      | octet-string |                             |      |      | x + 0x18   |  |
| 5. program_ID<br>(static)     | octet-string |                             |      |      | x + 0x20   |  |
| 6. unique_node_ID<br>(static) | octet-string |                             |      |      | x + 0x28   |  |
| Specific methods              | m/o          |                             |      |      |            |  |

#### 4.19.2.2 Attribute description

##### 4.19.2.2.1 logical\_name

Identifies the “ISO/IEC 14908 identification” object instance. See 6.2.30.

##### 4.19.2.2.2 node\_ID

Identification of the node.

##### 4.19.2.2.3 subnet\_ID

Identification of the subnet to which the device is connected.

##### 4.19.2.2.4 domain\_ID

Identification of the network. It shall be 3 bytes.

##### 4.19.2.2.5 program\_ID

Uniquely identifies the functionalities and version of the firmware running on the device. It shall be 8 bytes.

##### 4.19.2.2.6 unique\_node\_ID

Identification of the physical connection module connected to the ISO/IEC 14908-1:2012 network. Each compliant module is assigned a unique 48-bit identifier called

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 417/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

Unique\_Node\_ID. This ID is unique worldwide and is set at the time of manufacture. The value of this identifier does not change from the time of manufacture.

### 4.19.3 ISO/IEC 14908 protocol setup (class\_id = 131, version = 0)

#### 4.19.3.1 Overview

Instances of the ISO/IEC 14908 protocol setup IC allow the configuration of the ISO/IEC 14908 device.

| ISO/IEC 14908 protocol setup      | 0...n         | class_id = 131, version = 0 |      |      |            |
|-----------------------------------|---------------|-----------------------------|------|------|------------|
| Attributes                        | Data type     | Min.                        | Max. | Def. | Short name |
| 1. logical_name<br>(static)       | octet-string  |                             |      |      | x          |
| 2. inactivity_timeout<br>(static) | long-unsigned |                             |      |      | x + 0x08   |
| Specific methods                  | m/o           |                             |      |      |            |

#### 4.19.3.2 Attribute description

##### 4.19.3.2.1 logical\_name

Identifies the “ISO/IEC 14908 protocol setup” object instance. See 6.2.30.

##### 4.19.3.3 inactivity\_timeout

Minutes of absence of adaptation layer messages from the NNAP before the LNAP is considered to be out of communication; see IEC 62056-8-8:2020, F4.3.

### 4.19.4 ISO/IEC 14908 protocol status (class\_id = 132, version = 0)

#### 4.19.4.1 Overview

Instances of the ISO/IEC 14908 protocol status IC allow the status of the protocol in the ISO/IEC 14908 device to be determined.

## COSEM Interface Classes

| ISO/IEC 14908 protocol status   | 0...n                | class_id = 132, version = 0 |      |      |            |
|---------------------------------|----------------------|-----------------------------|------|------|------------|
| Attributes                      | Data type            | Min.                        | Max. | Def. | Short name |
| 1. logical_name<br>(static)     | octet-string         |                             |      |      | x          |
| 2. transmission_errors<br>(dyn) | long-unsigned        |                             |      |      | x + 0x08   |
| 3. transmit_tx_failure<br>(dyn) | long-unsigned        |                             |      |      | x + 0x10   |
| 4. transmit_tx_retries<br>(dyn) | long-unsigned        |                             |      |      | x + 0x18   |
| 5. receive_tx_full<br>(dyn)     | long-unsigned        |                             |      |      | x + 0x20   |
| 6. lost_messages<br>(dyn)       | long-unsigned        |                             |      |      | x + 0x28   |
| 7. missed_messages<br>(dyn)     | long-unsigned        |                             |      |      | x + 0x30   |
| 8. layer2_received<br>(dyn)     | long-unsigned        |                             |      |      | x + 0x38   |
| 9. layer3_received<br>(dyn)     | long-unsigned        |                             |      |      | x + 0x40   |
| 10. messages_received<br>(dyn)  | double-long-unsigned |                             |      |      | x + 0x48   |
| 11. messages_validated<br>(dyn) | double-long-unsigned |                             |      |      | x + 0x50   |
| Specific methods                | m/o                  |                             |      |      |            |
| 1. reset (data)                 | o                    |                             |      |      | X + 0x70   |

### 4.19.4.2 Attribute description

#### 4.19.4.2.1 logical\_name

Identifies the “ISO/IEC 14908 protocol status” object instance. See 6.2.30.

#### 4.19.4.2.2 transmission\_errors

This is a count of the number of transmission errors that have occurred on the network. A transmission error is detected via a CRC error during packet reception or as a packet that is less than 8 bytes long. This could result from a collision, a noisy medium, signal attenuation, etc.

See ISO/IEC 14908-1:2012, 13.8.2.

#### 4.19.4.2.3 transmit\_tx\_failure

The number of times that the node failed to receive expected acknowledgments or responses after retrying the configured number of times.

See ISO/IEC 14908-1:2012B.8.

#### 4.19.4.2.4 transmit\_tx\_retries

The number of retries sent by this node. This does not include retries used for messages sent with repeated service.

See ISO/IEC 14908-1:2012, B.8.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 419/668 |
|-----------------------|------------|-----------------------------|---------|

**4.19.4.2.5 receive\_tx\_full**

The number of times that an incoming packet was discarded because there was no room in the transaction database.

See ISO/IEC 14908-1:2012, B.8.

**4.19.4.2.6 lost\_messages**

This is the number of messages that were addressed to the node that were thrown away because there was no application buffer available for the message;

See ISO/IEC 14908-1:2012, 13.8.2.

**4.19.4.2.7 missed\_messages**

This is the number of messages that were on the network but could not be received because there was no network buffer (packet buffer) available for the message or the network buffer was too small to receive the message;

See ISO/IEC 14908-1:2012, 13.8.2.

**4.19.4.2.8 layer2\_received**

The number of Layer-2 messages received by this node. Layer-2 messages are those that have correct CRC and can be addressed to any node.

See ISO/IEC 14908-1:2012, B.8.

**4.19.4.2.9 layer3\_received**

The number of messages transmitted from layer 3 of the protocol processor. These can include network variable updates, explicit messages, acknowledgments, retries, reminders, service pin messages, and any other type of message.

See ISO/IEC 14908-1:2012, B.8.

**4.19.4.2.10 messages\_received**

Number of messages received, which are delivered to the Adaptation layer.

**4.19.4.2.11 messages\_validated**

Number of correct messages received at Adaptation layer level.

**4.19.4.3 Method description****4.19.4.3.1 reset (data)**

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

data ::= integer (0)

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 420/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

NOTE: IEC 62056-8-8:2020 specifies version = 0 which does not support the reset method.

### **4.19.5 ISO/IEC 14908 diagnostic (class\_id = 133, version = 0)**

#### **4.19.5.1 Overview**

Instances of the ISO/IEC 14908 diagnostic IC provides information about the device status inside the PLC network.

| ISO/IEC 14908 diagnostic           | 0...n         | class_id = 133, version = 0 |      |      |            |
|------------------------------------|---------------|-----------------------------|------|------|------------|
| Attributes                         | Data type     | Min.                        | Max. | Def. | Short name |
| 1. logical_name (static)           | octet-string  |                             |      |      | x          |
| 2. plc_signal_quality_status (dyn) | enum          |                             |      |      | x + 0x08   |
| 3. com_module_state (dyn)          | enum          |                             |      |      | x + 0x10   |
| 4. received_message_status (dyn)   | unsigned      |                             |      |      | x + 0x18   |
| 5. no_receive_buffer (dyn)         | long-unsigned |                             |      |      | x + 0x20   |
| 6. transmit_no_data (dyn)          | long-unsigned |                             |      |      | x + 0x28   |
| 7. backlog_overflows (dyn)         | long-unsigned |                             |      |      | x + 0x30   |
| 8. late_ack (dyn)                  | long-unsigned |                             |      |      | x + 0x38   |
| 9. frequency_invalid (dyn)         | long-unsigned |                             |      |      | x + 0x40   |
| <b>Specific methods</b>            | <b>m/o</b>    |                             |      |      |            |
| 1. reset (data)                    | o             |                             |      |      | X + 0x60   |

#### **4.19.5.2 Attribute description**

##### **4.19.5.2.1 logical\_name**

Identifies the “ISO/IEC 14908 diagnostic” object instance. See 6.2.30.

##### **4.19.5.2.2 plc\_signal\_quality\_status**

Provides the PLC signal quality status:

- enum:
- (0) No PLC traffic detected,
- (1) PLC traffic detected, but no traffic addressed to this device,
- (2) A packet addressed to this device has been received. The signal margin is less than or equal to 9 dB,
- (3) A packet addressed to this device has been received. The signal margin is between 12 dB or 15 dB,
- (4) A packet addressed to this device has been received. The signal margin is greater than or equal to 18 dB,
- (5) Meter is commissioned but its inactivity\_timeout as defined in 4.18.1 has expired. The last packet addressed to this meter had a signal margin is less than or equal to 9 dB,
- (6) Meter is commissioned but its inactivity\_timeout as defined in 4.18.1 has expired. The last packet addressed to this meter had a signal margin at 12 dB or 15 dB,

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 421/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

- (7) Meter is commissioned but its inactivity\_timeout as defined in 4.18.1 has expired. The last packet addressed to this meter had a signal margin greater than or equal to 18 dB.

NOTE. This parameter is updated every time a packet is received by the device.

### 4.19.5.2.3 com\_module\_state

Last received state of the device's power line IEC 62056-8-8:2020 communication module.

enum:

|       |                                 |
|-------|---------------------------------|
| (0)   | Invalid state,                  |
| (1)   | Invalid state,                  |
| (2)   | Unconfigured,                   |
| (3)   | Applicationless,                |
| (4)   | Configured,                     |
| (6)   | Hard-offline,                   |
| (12)  | Configured, soft offline,       |
| (140) | Configured, in bypass mode,     |
| (255) | No reply when status requested, |
| other | Invalid states.                 |

This field is updated on each device power-up, each re-synch request from the NNAP and each communication module reset. Any value which is not listed above is invalid state. See IEC 62056-8-8:2020,13.4.

NOTE Configured, soft offline state (12) and Configured, in bypass mode state (140) are sub states of Configured state (4). See IEC 62056-8-8:2020, 13.4 and 13.8.

### 4.19.5.2.4 received\_message\_status

Unsigned integer, constructed as:

| Bit 7 (MSB)       | b6 | b5       | b4 | b3 | b2 | b1        | Bit 0 (LSB) |
|-------------------|----|----------|----|----|----|-----------|-------------|
| Transceiver phase |    | reserved |    |    |    | See below |             |

Transceiver phase bits are used as below:

|                            | Bit 7   | b6      | Bit 5   |
|----------------------------|---------|---------|---------|
| No phase inversion present | Not set | Not set | Not set |
| In phase normal            | Not set | Not set | Set     |
| Plus 120 degrees inverted  | Not set | Set     | Not set |
| Minus 120 degrees normal   | Not set | Set     | Set     |
| Phase 180 degrees inverted | Set     | Not set | Not set |
| Plus 120 degrees normal    | Set     | Not set | Set     |
| Minus 120 degrees inverted | Set     | Set     | Not set |
| Reserved                   | Set     | Set     | Set     |

**4.19.5.2.5 no\_receive\_buffer**

Number of times the IEC 62056-8-8:2020 lower layers detected a network inbound packet addressed to it, but has no buffer to store it. Hence the message could not be made available to the adaptation layer.

**4.19.5.2.6 transmit\_no\_data**

Number of times the adaptation layer attempted to transmit data to the IEC 62056-8-8:2020 stack but no buffer is available. Updated on occurrence. IEC 62056-8-8:2020,13.4.

**4.19.5.2.7 backlog\_overflows**

Number of times the MAC layer predictive backlog counter overflowed. See IEC 62056-8-8:2020, B.8.

**4.19.5.2.8 late\_ack**

Number of times an ACK or response was received for a transaction that has been already completed. Typically, this occurs due to too short transaction timer values or network congestion.

**4.19.5.2.9 frequency\_invalid**

Number of times the IEC 62056-8-8:2020 stack detects an invalid frequency. Updated on occurrence.

**4.19.5.3 Method description****4.19.5.3.1 reset (data)**

Forces a reset of the object. By invoking this method, the values of attributes 5..9 are set to the default value. The default value is an instance specific constant.

```
data ::= integer (0)
```

NOTE IEC 62056-8-8:2020 specifies version = 0 which does not support the reset method.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 423/668 |
|-----------------------|------------|-----------------------------|---------|

## 5 Previous versions of interface classes

### 5.1 General

#### 5.1.1 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.

NOTE Previous versions of this Technical Report had additional text that applied a rule on the re-use and modification of attribute and method enumerators. This rule is no longer required as it placed an unnecessary constraint on interface classes with large numbers of attributes and/or methods.

Any modification of ICs will be recorded by moving the old version of an IC into the relevant clause below.

#### 5.1.2 New interface classes

The DLMS UA reserves the right to be the exclusive administrator of interface classes.

#### 5.1.3 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.

### 5.2 Previous versions of interface classes – general

The subsequent subclauses list 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.

### 5.3 Previous versions of interface classes for parameters and measurement data

#### 5.3.1 Profile generic (class\_id = 7, version = 0)

##### 5.3.1.1 Overview

The version listed here was valid in Edition 1 and replaced with effect from Edition 2.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 424/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

| 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   |
| <b>Specific methods</b>       | <b>m/o</b>           |                           |      |      |            |
| 1. reset (data)               |                      |                           |      |      | x + 0x58   |
| 2. capture (data)             |                      |                           |      |      | x + 0x60   |
| 3. get_buffer_by_range (data) |                      |                           |      |      | x + 0x68   |
| 4. get_buffer_by_index (data) |                      |                           |      |      | x + 0x70   |

### 5.3.1.2 Attribute description

#### 5.3.1.2.1 logical\_name

Identifies the “Profile generic” object instance. For examples, see 6.2.19, 6.2.21, 6.2.46, etc.

#### 5.3.1.2.2 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
Default    The buffer is empty at installation.
```

Remark: Reading the entire buffer delivers only those entries which are “in use”

#### 5.3.1.2.3 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 2<sup>nd</sup>, etc.)

#### 5.3.1.2.4 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

#### 5.3.1.2.5 sort\_method

If the profile is unsorted, it works as a „first in first out“ buffer (it is hence sorted by capturing, and not necessarily by the time maintained in the clock object). If the buffer is full, the next call to *capture ()* will push out the first (oldest) entry of the buffer to make space for the new entry.

If the profile is sorted, a call to *capture ()* will store the new entry at the appropriate position in the buffer, moving all following entries and probably losing the least interesting entry. If the new entry would enter the buffer after the last entry and if the buffer is already full, the new entry will not be retained at all.

enum:

- (1) fifo,
- (2) lifo (last in first out),
- (3) largest,
- (4) smallest,
- (5) nearest\_to\_zero,
- (6) farest\_from\_zero

Default fifo

#### 5.3.1.2.6 sort\_object

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

ObjectDefinition see 5.3.1.2.3

Default no object to sort by (only possible with *sort\_method* = fifo or lifo)

#### 5.3.1.2.7 entries\_in\_use

Counts the number of entries stored in the buffer. After a call of *reset ()* the buffer does not contain any entries, and this value is zero. Upon each subsequent call of *capture ()*, this value will be incremented up to the maximum number of entries that will get stored (see 5.3.1.2.8 *profile\_entries*).

double-long-unsigned 0...profile\_entries

Default 0

#### 5.3.1.2.8 profile\_entries

Specifies, how many entries should be retained in the buffer.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 426/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

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

Default 1

### 5.3.1.3 Method description

#### 5.3.1.3.1 reset (data)

Clears the buffer. The buffer has no valid entries afterwards; *entries\_in\_use* 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)

#### 5.3.1.3.2 capture (data)

Copies the values of the objects to capture into the buffer by calling *read (<object\_attribute>)* of each capture object. Depending on the *sort\_method* and the actual state of the buffer this produces a new entry or a replacement for the less significant entry. As long as not all entries are already used, the *entries\_in\_use* attribute will be incremented.

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

Note that only some attributes of the captured objects might be stored, not the complete object.

data ::= integer (0)

#### 5.3.1.3.3 write (...)

Any write access to one of the attributes 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 set to the maximum possible value (restricted by physical size, typically laid down in the firmware).

#### 5.3.1.3.4 get\_buffer\_by\_range (data)

Reads all entries between the given limits.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 427/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

---

|  |     |  |
|--|-----|--|
| restricting_object                         | in  | ObjectDefinition<br>Defines the register or clock restricting the range of entries to be retrieved.  |
| from_value                                 | in  | instance_specific<br>Oldest or smallest entry to retrieve.   |
| to_value                                   | in  | instance_specific<br>Newest or largest entry to retrieve.  |
| selected_values                            | in  | List of columns to retrieve:<br><br>array ObjectDefinition<br><br>If the array is empty (has no entries), all captured data is returned. Otherwise, only the columns specified in the array are returned. The type <i>ObjectDefinition</i> is specified above ( <i>Capture_Objects</i> ) |
| entries                                    | out | See 5.3.1.2.7 <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 instance\_specific\_value.

---

### 5.3.1.3.5 get\_buffer\_by\_index (data)

Read all entries between the given limits.

---

|   |     |  |
|---|-----|--|
| from_index                              | in  | double-long-unsigned<br>First entry to retrieve.                                   |
| to_index                                | in  | double-long-unsigned<br>Last entry to retrieve.                                    |
| from_selected_value                     | in  | long-unsigned<br>index of first value to retrieve                                  |
| to_selected_value                       | in  | long-unsigned<br>index of last value to retrieve.                                  |
| entries                                 | out | See 5.3.1.2.7 above.   |
| array of <i>instance_specific_value</i> | out | Sorted sequence of entries containing the requested values of the capture objects. |

data ::= structure {from\_index; to\_index; selected\_values}.

In the response “data” is an array of instance\_specific\_value.

---

### 5.3.2 Compact data (class\_id = 62, version = 0)

#### 5.3.2.1 Overview

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            |                            |      |      |            |
| 2. capture (data)              | o            |                            |      |      |            |

#### 5.3.2.2 Attribute description

##### 5.3.2.2.1 logical\_name

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

##### 5.3.2.2.2 compact\_buffer

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

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

##### 5.3.2.2.3 capture\_objects

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

The *template\_id* attribute shall be the first element in the *capture\_objects* array.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 429/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

Upon an explicit or implicit invocation of the *capture (data)* method the values of the selected attributes are captured into the *compact\_buffer*.

```
array      capture_object_definition

capture_object_definition ::= structure

{

    class_id:      long-unsigned,
    logical_name: octet-string,
    attribute_index: integer,
    data_index:      long-unsigned
}
```

Where:

- attribute\_index is a pointer to the attribute within the object, identified by class\_id and logical\_name: attribute\_index 1 refers to the 1<sup>st</sup> attribute (i.e. the *logical\_name*), attribute\_index 2 to the 2<sup>nd</sup> 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 9.

### 5.3.2.2.4 template\_id

Contains the identifier of the template.

It shall uniquely identify the instance of the “Compact data” IC and the *template\_description*.

### 5.3.2.2.5 template\_description

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

- 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 *capture\_objects* array – encoded as a variable length integer;

- this is followed by the data type of each attribute, in the same order as in the *capture\_object* array:
  - in the case of attributes with simple data type, the data type is represented by a single octet, carrying the tag of the data type. In the case of bit-string [4], octet-string [9], visible-string [10], utf8-string [12] the length of the string is part of the data held in the *compact\_buffer*;
  - in the case of an array [1], the data type is represented by a single octet 0x01. This is followed by the type description of the elements in the array. The number of elements in the array is part of the data held in the *compact\_buffer*;
  - in the case of a structure [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.

#### **5.3.2.2.6 capture\_method**

Defines the way the *compact\_buffer* is updated.

enum:

- (0) Capture upon invoking the *capture (data)* method. This may occur remotely or locally (explicit capturing),
- (1) Capture upon reading the *compact\_buffer* attribute (implicit capturing).

#### **5.3.2.3 Method description**

##### **5.3.2.3.1 reset (data)**

Clears the *compact\_buffer*. After invoking this method the *compact\_buffer* holds an octet-string of 0 length, until a new capture takes place.

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

data ::= integer(0)

##### **5.3.2.3.2 capture (data)**

Copies the values of the attributes into the *compact\_buffer* by reading each capture object.

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

data ::= integer(0)

#### **5.3.2.4 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*.

#### **5.3.2.5 Restrictions**

When defining the *capture\_object* attribute, circular references shall be avoided.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 431/668 |
|-----------------------|------------|-----------------------------|---------|

## 5.4 Previous versions of interface classes for access control and management

### 5.4.1 Association SN (class\_id = 12, version = 0)

#### 5.4.1.1 Overview

The version listed here was valid in Edition 1 and it is replaced with effect from Edition 2.

| Device Association View           | 0..1 <sup>a</sup> | class_id = 12, version = 0 |      |      |            |
|-----------------------------------|-------------------|----------------------------|------|------|------------|
| Attributes                        | Data type         | Min.                       | Max. | Def. | Short name |
| 1. logical_name (static)          | octet-string      |                            |      |      | x          |
| 2. object_list (static)           | objlist_type      |                            |      |      | x + 0x08   |
| Specific methods                  | m/o               |                            |      |      |            |
| 1. getlist_by_classid (data)      | o                 |                            |      |      |            |
| 2. getobj_by_logicalname (data)   | o                 |                            |      |      |            |
| 3. read_by_logicalname (data)     | o                 |                            |      |      |            |
| 4. get_attributes&services (data) | o                 |                            |      |      |            |
| 5. change_LLS_secret (data)       | o                 |                            |      |      |            |
| 6. change_HLS_secret (data)       | o                 |                            |      |      |            |
| 7. get_HLS_challenge (data)       | o                 |                            |      |      |            |
| 8. reply_to_HLS_challenge (data)  | o                 |                            |      |      |            |

#### 5.4.1.2 Attribute description

##### 5.4.1.2.1 logical\_name

Identifies the type of the client-server association. See 6.2.33.

##### 5.4.1.2.2 object\_list

Contains the list of all objects with their short\_name (DLMS ObjectName of the first attribute), class\_id, version<sup>b</sup> and logical\_name.

```

Objlist_type ::= array objlist_element

objlist_element ::= structure

{
    short_name: long,
    class_id: long-unsigned,
    version = unsigned,
    logical_name: octet-string
}

```

**5.4.1.3 Method description****5.4.1.3.1 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

**5.4.1.3.2 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

}

For the response: data ::= objlist\_element

**5.4.1.3.3 read\_by\_logicalname (data)**

Reads attributes for specific objects. The objects are specified by their class\_id and their logical\_name.

data ::= array Attributelidentification

Attributelidentification ::= structure

{

class\_id: long-unsigned,  
logical\_name: octet-string,  
attribute\_index: unsigned

}

Where:

- attribute\_index is a pointer (i.e. offset) to the attribute within the object;
- attribute\_index = 0 delivers all attributes<sup>c</sup>, attribute\_index = 1 delivers the first attribute (i.e. logical\_name), etc.

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

**5.4.1.3.4 get\_attributes&services (data)**

Delivers information about the access rights to the attributes and services within the actual association. The objects are specified by their class\_id and their logical\_name.

array ObjectIdentification

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 433/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

```
ObjectIdentification ::= structure
{
    class_id:      long-unsigned,
    logical_name:  octet-string
}
For the response
data ::= array      AccessDescription
AccessDescription ::= structure
{
    read_attributes: bit-string,
    write_attributes: bit-string,
    services:        bit-string
}
```

The position in the bit-string identifies the attribute/service (first position  $\leftrightarrow$  first attribute, first position  $\leftrightarrow$  first service) and the value of the bit specifies whether the attribute/service is available (bit set) or not available (bit clear).

### **5.4.1.3.5    change\_LLS\_secret (data)**

Changes the LLS secret (for example password).

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

### **5.4.1.3.6    change\_HLS\_secret (data)**

Changes the HLS secret (for example encryption key).

```
data ::= octet-stringd new HLS secret
```

### **5.4.1.3.7    get\_HLS\_challenge (data)**

Asks the server for the client “challenge” (for example random number).

```
data ::= octet-string      client challenge
```

### **5.4.1.3.8    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].

#### NOTES

a      Per client server association.

b      Within an client-server association, there are never two objects with the same class\_id and logical\_name differing only in the versions.

c      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, Clause A.12 (p. 221)).

d      The structure of the “new secret” depends on the security mechanism implemented. The “new secret” may contain additional checkbits and it may be encrypted.

## 5.4.2 Association SN (class\_id = 12, version = 1)

### 5.4.2.1 Overview

This IC allows modelling of AAs between a server and a client, using the application context *short name referencing*. A COSEM logical device may have one instance of this IC for each association the device is able to support.

The **short\_name** of the current “Association SN” object itself is fixed within the COSEM context. It is given in 4.1.3 as 0xFA00.

| Association SN                        | 0...n        | class_id = 12, version = 1 |      |      |            |
|---------------------------------------|--------------|----------------------------|------|------|------------|
| Attributes                            | Data type    | Min.                       | Max. | Def. | Short name |
| 1. logical_name (static)              | octet-string |                            |      |      | x          |
| 2. object_list (static)               | objlist_type |                            |      |      | x + 0x08   |
| Specific methods                      | m/o          |                            |      |      |            |
| 1. reserved from previous versions    | o            |                            |      |      |            |
| 2. reserved from previous versions    | o            |                            |      |      |            |
| 3. read_by_logicalname (data)         | o            |                            |      |      |            |
| 4. get_attributes&methods (data)      | o            |                            |      |      |            |
| 5. change_LLS_secret (data)           | o            |                            |      |      |            |
| 6. change_HLS_secret (data)           | o            |                            |      |      |            |
| 7. reserved from previous versions    |              |                            |      |      |            |
| 8. reply_to_HLS_authentication (data) | o            |                            |      |      |            |

### 5.4.2.2 Attribute description

#### 5.4.2.2.1 logical\_name

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

#### 5.4.2.2.2 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

}

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 435/668 |
|-----------------------|------------|-----------------------------|---------|

selective access (see 4.1.4) to the attribute `object_list` may be available (optional). The selective access parameters are as defined in Table 47.

**Table 47 – Parameters for selective access to the `object_list` attribute**

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

### 5.4.2.3 Method description

#### 5.4.2.3.1 `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 <sup>a</sup>, `attribute_index 1` delivers the first attribute (i.e. `logical_name`), etc.).

NOTE 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, A.12 (p. 221)).

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

#### 5.4.2.3.2 `get_attributes& methods (data)`

Delivers information about the access rights to the attributes and methods within the actual association. The objects are specified by their `class_id` and their `logical_name`. With this method, the parameterized access feature can also be used.

```
data ::= array object_identification

object_identification ::= structure
{
    class_id: long-unsigned,
    logical_name: octet-string
}
```

For the response

## COSEM Interface Classes

```
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  $\leftrightarrow$  first attribute, first position  $\leftrightarrow$  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).

### 5.4.2.3.3 change\_LLS\_secret (data)

Changes the LLS secret (for example password).

```
data ::= octet-string    new LLS secret change_HLS_secret (data)
```

### 5.4.2.3.4 change\_HLS\_secret (data)

Changes the HLS secret (for example encryption key).

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

NOTE 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.

### 5.4.2.3.5 reply\_to\_HLS\_authentication (data)

The remote invocation of this method delivers the client's “secretly” processed “challenge StoC” (f(StoC)) back to the server as the data service parameter of the 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.

## 5.4.3 Association SN (class\_id = 12, version = 2)

### 5.4.3.1 Overview

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 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 437/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| 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   |
| <b>Specific methods</b>               | <b>m/o</b>         |                            |      |      |            |
| 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   |

### 5.4.3.2 Attribute description

#### 5.4.3.2.1 logical\_name

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

#### 5.4.3.2.2 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 in Table 48.

#### 5.4.3.2.3 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,
}

```

## COSEM Interface Classes

```
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 in Table 48.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 439/668 |
|-----------------------|------------|-----------------------------|---------|

**Table 48 – 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.<br>For the response: data ::= objlist_type  |
| 2                     | structure<br>{<br>class_id: long-unsigned,<br>logical_name: octet-string<br>} | 2                        | Delivers the entry of the object_list for a specific class_id and logical_name.<br>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.<br>For the response: data ::= objlist_element<br>In the case of attribute 3, delivers the entry of the access_rights_list for a specific base_name.<br>For the response: data ::= access_rights_element |

#### 5.4.3.2.4 security\_setup\_ reference

References the “Security setup” object by its *logical\_name*. The referenced object manages security for a given “Association SN” object instance.

#### 5.4.3.3 Method description

##### 5.4.3.3.1 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 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. 10:2020, 9.5.

##### 5.4.3.3.2 change\_secret(data)

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

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

## COSEM Interface Classes

NOTE 1 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.

### 5.4.3.3.3 reply\_to\_HLS\_authentication (data)

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

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

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

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

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

## 5.4.4 Association SN (Class\_id = 12, version =3)

### 5.4.4.1 Overview

NOTE 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 441/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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

**Table 49 – 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.<br>For the response: data ::= objlist_type  |
| 2                     | structure<br>{<br>class_id: long-unsigned,<br>logical_name: octet-string<br>} | 2                        | Delivers the entry of the object_list for a specific class_id and logical_name.<br>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.<br>For the response: data ::= objlist_element<br>In the case of attribute 3, delivers the entry of the access_rights_list for a specific base_name.<br>For the response: data ::= access_rights_element |

### 5.4.4.2 Attribute description

#### 5.4.4.2.1 logical\_name

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

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 442/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

#### 5.4.4.2.2 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 in Table 49.

#### 5.4.4.2.3 access\_rights\_list

Contains the access rights to attributes and methods.

The link between the *object\_list* and the *access\_rights\_list* is the base\_name, present in both the *objlist\_element* structure and the *access\_right\_element* structure. Therefore, the base\_names on the two lists shall be the same. The number – and preferably, the order – of the elements in the array of *objlist\_element* and the array of *access\_right\_element* shall also be the same.

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

attribute_access_descriptor ::= array      attribute_access_item

attribute_access_item ::= structure
{
    attribute_id:      integer,
    access_mode:       enum:
                      (0) no_access,
                      (1) read_only,
                      (2) write_only,
                      (3) read_and_write,
                      (4) authenticated_read_only,
                      (5) authenticated_write_only,
                      (6) authenticated_read_and_write
    access_selectors: CHOICE
    {
        null-data      [0],
        array integer   [1]
    }
}

method_access_descriptor ::= array      method_access_item
method_access_item ::= structure
{
    method_id:      integer,
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 443/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

```
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 in Table 49.

### 5.4.4.2.4 security\_setup\_reference

References the “Security setup” object by its *logical\_name*. The referenced object manages security for a given “Association SN” object instance.

### 5.4.4.2.5 user\_list

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

```
array      user_list_entry  
  
user_list_entry ::= structure  
{  
    user_id:      unsigned,  
    user_name:    visible-string  
}
```

Where:

- *user\_id* is the identifier of the user (this value is carried by the calling-AE-invocation-id field of the AARQ);
- *user\_name* is the name of the user.

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

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

### 5.4.4.2.6 current\_user

Holds the identifier of the current user.

```
current_user ::= user_list_entry (see 5.4.4.2.5)
```

If the *user\_list* is empty, then *current\_user* shall be a structure {*user\_id*: unsigned 0, *user\_name*: visible string of 0 elements}

## 5.4.4.3 Method description

### 5.4.4.3.1 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** 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. 10:2020, 9.5.

#### 5.4.4.3.2 change\_secret (data)

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

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

**NOTE** 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.

#### 5.4.4.3.3 reply\_to\_HLS\_authentication (data)

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

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

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

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

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

#### 5.4.4.3.4 add\_user (data)

Adds a user to the user\_list.

```
data ::= user_list_entry (see 5.4.4.2.5)
```

#### 5.4.4.3.5 remove\_user (data)

Removes a user from the user\_list.

```
data ::= user_list_entry (see 5.4.4.2.5)
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 445/668 |
|-----------------------|------------|-----------------------------|---------|

### 5.4.5 Association LN (class\_id = 15, version = 0)

#### 5.4.5.1 Overview

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   |

#### 5.4.5.2 Attribute description

##### 5.4.5.2.1 logical\_name

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

##### 5.4.5.2.2 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

```

## COSEM Interface Classes

```

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 in 5.4.5.2.9.

### **5.4.5.2.3 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.

### **5.4.5.2.4 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
{

```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 447/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

```
    context_name_structure      [2],  
    octet-string                [9]  
}
```

The application context name is specified as OBJECT IDENTIFIER in DLMS UA 1000-2 Ed.11:2021, 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.11:2021, 9.4.2.2.2.

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).

### 5.4.5.2.5 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.11:2021 9.4.2.2.3);

- 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.11:2021, 9.4.2.2.3);
- 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.11:2021, 9.4.2.2.3).

#### 5.4.5.2.6 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.11:2021, 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.11:2021, Clause 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).

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 449/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

No mechanism\_name is required when no authentication is used.

### **5.4.5.2.7 LLS\_secret**

Contains the authentication value for the LLS authentication process.

### **5.4.5.2.8 association\_status**

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

enum: (0) non-associated,

- (1) association-pending,
- (2) associated

### **5.4.5.2.9 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 shown in Table 50:

**Table 50 – Parameters for selective access to the *object\_list* attribute**

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

#### 5.4.5.3 Method description

##### 5.4.5.3.1 reply\_to\_HLS\_authentication (data)

The remote invocation of this method delivers the client's "secretly" processed "challenge StoC" ( $f(StoC)$ ) back to the server as the *data* service parameter of the ACTION.request primitive invoked.

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

If the authentication is accepted, then the response (ACTION.confirm primitive) contains Result == OK and the server's "secretly" processed "challenge CtoS" ( $f(CtoS)$ ) back to the client in the *data* service parameter of the response service.

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

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

##### 5.4.5.3.2 change\_HLS\_secret (data)

Changes the HLS secret (for example encryption key).

data ::= octet-string new HLS secret

**NOTE** 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.

##### 5.4.5.3.3 add\_object (data)

Adds the referenced object to the object\_list.

data ::= object\_list\_element (see 5.4.5.2.2)

##### 5.4.5.3.4 remove\_object (data)

Removes the referenced object from the object\_list.

data ::= object\_list\_element (see 5.4.5.2.2)

#### 5.4.6 Association LN (class\_id = 15, version = 1)

##### 5.4.6.1 Overview

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.

A SET operation on an attribute of an association LN object becomes effective when this association object is used to establish a new association.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 451/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| 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   |
| <b>Specific methods</b>               | <b>m/o</b>               |                            |     |      |            |
| 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   |

### 5.4.6.2 Attribute description

#### 5.4.6.2.1 logical\_name

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

#### 5.4.6.2.2 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,
}

```

## COSEM Interface Classes

```
(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 in 5.4.5.2.9.

### 5.4.6.2.3 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.

### 5.4.6.2.4 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]
}
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 453/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

The application context name is specified as OBJECT IDENTIFIER in DLMS UA 1000-2 Ed. 10:2020, 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. 10:2020, 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).

### 5.4.6.2.5 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;
- 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. 10, 9.4.2.2.3.

### 5.4.6.2.6 authentication\_mechanism\_name

Contains the name of the authentication mechanism for the AA.

```
mechanism_name_type ::= CHOICE
```

## COSEM Interface Classes

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

The authentication mechanism name is specified as an OBJECT IDENTIFIER in DLMS UA 1000-2 Ed. 10:2020, 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. 10:2020, 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.

### **5.4.6.2.7 secret**

Contains the secret for the LLS or HLS authentication process.

NOTE In the case of HLS with GMAC, the (HLS\_)secret is held by the Security setup object referenced in attribute 9.

### **5.4.6.2.8 association\_status**

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

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

### **5.4.6.2.9 security\_setup\_reference**

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

### **5.4.6.2.10 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 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 455/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

- 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 shown in Table 51:

**Table 51 – Parameters for selective access to the object\_list attribute**

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

### 5.4.6.3 Method description

#### 5.4.6.3.1 reply\_to\_HLS\_authentication (data)

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

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

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

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

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

#### 5.4.6.3.2 change\_HLS\_secret (data)

Changes the HLS secret (for example encryption key).

data ::= octet-string     new HLS secret

## COSEM Interface Classes

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.

### **5.4.6.3.3 add\_object (data)**

Adds the referenced object to the *object\_list*.

data ::= object\_list\_element (see 5.4.6.2.2)

### **5.4.6.3.4 remove\_object (data)**

Removes the referenced object from the *object\_list*.

data ::= object\_list\_element (see 5.4.6.2.2)

## **5.4.7 Association LN (class\_id = 15, version = 2)**

### **5.4.7.1 Overview**

NOTE This 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 |            |             |                   |
|-----------------------------------|----------|---------------------|----------------------------|------------|-------------|-------------------|
| <i>Attributes</i>                 |          | <i>Data type</i>    | <i>Min.</i>                | <i>Max</i> | <i>Def.</i> | <i>Short name</i> |
| 1. logical_name                   | (static) | octet-string        |                            | -          |             | x                 |
| 2. object_list                    | (static) | object_list_type    |                            |            |             | x + 0x08          |
| 3. associated_partners_id         |          | associated_partners |                            |            |             | x + 0x10          |
| 4. application_context_name       |          | context_name_type   |                            |            |             | x + 0x18          |
| 5. xDLMS_context_info             |          | xDLMS_context_type  |                            |            |             | x + 0x20          |
| 6. authentication_0mechanism_name |          | mechanism_name_type |                            |            |             | x + 0x28          |
| 7. secret                         |          | octet-string        |                            |            |             | x + 0x30          |
| 8. association_status             |          | enum                |                            |            |             | x + 0x38          |
| 9. security_setup_reference       | (static) | octet-string        |                            |            |             | x + 0x40          |
| 10. user_list                     | (static) | array               |                            |            |             | x + 0x48          |
| 11. current_user                  |          | structure           |                            |            |             | x + 0x50          |
| <i>Specific methods</i>           |          | <i>m/o</i>          |                            |            |             |                   |
| 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          |

#### 5.4.7.2 Attribute description

##### 5.4.7.2.1 logical\_name

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

##### 5.4.7.2.2 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 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:
                    (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 in 5.4.5.2.9.

#### 5.4.7.2.3 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.

#### 5.4.7.2.4 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. 10, 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. 10, 11.4.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 459/668 |
|-----------------------|------------|-----------------------------|---------|

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).

#### 5.4.7.2.5 xDLMS\_context\_info

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

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

Where:

- the `conformance` element contains the xDLMS conformance block supported by the server. The length of the bit-string is 24 bits.
- the `max_receive_pdu_size` element contains the maximum length for an xDLMS APDU, expressed in bytes that the client may send. This is the same as the `server-max-receive-pdu-size` parameter of the xDLMS initiateResponse APDU;
- the `max_send_pdu_size` element, in an active AA contains the maximum length for an xDLMS APDU, expressed in bytes that the server may send. This is the same as the `client-max-receive-pdu-size` parameter of the xDLMS initiateRequest APDU;
- the `dlms_version_number` element contains the DLMS version number supported by the server;
- the `quality_of_service` element is not used;
- the `cyphering_info` element – in an active AA – contains the dedicated key parameter of the xDLMS initiateRequest APDU. See DLMS UA 1000-2 Ed. 10, 9.4.2.2.3.

#### 5.4.7.2.6 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. 10, 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. 10, 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.

#### **5.4.7.2.7 secret**

Contains the secret for the LLS or HLS authentication process.

#### **5.4.7.2.8 association\_status**

Contains the secret for the LLS or HLS authentication process.

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

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

#### **5.4.7.2.9 security\_setup\_reference**

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

#### **5.4.7.2.10 user\_list**

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

```
array    user_list_entry  
user_list_entry ::=    structure  
{  
    user_id:        unsigned,  
    user_name:      visible-string  
}
```

Where:

- user\_id is the identifier of the user (this value is carried by the calling-AE-invocation-id field of the AARQ);
- user\_name is the name of the user.

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

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 461/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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.

### **5.4.7.2.11 current\_user**

Holds the identifier of the current user.

`current_user ::= user_list_entry (see 5.4.7.2.10)`

If the *user\_list* is empty, then *current\_user* shall be a structure {*user\_id*: unsigned 0, *user\_name*: visible string of 0 elements}

### **5.4.7.3 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 shown in Table 56:

**Table 52 – Parameters for selective access to the *object\_list* attribute**

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

**5.4.7.4 Method description****5.4.7.4.1 reply\_to\_HLS\_authentication (data)**

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

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

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

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

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

**5.4.7.4.2 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.

**5.4.7.4.3 add\_object (data)**

Adds the referenced object to the object\_list.

data ::= object\_list\_element (see 5.4.7.2.2)

**5.4.7.4.4 remove\_object (data)**

Removes the referenced object from the object\_list.

data ::= object\_list\_element (see 5.4.7.2.2)

**5.4.7.4.5 add\_user (data)**

Adds a user to the user\_list.

data ::= user\_list\_entry (see 5.4.7.2.10)

**5.4.7.4.6 remove\_user (data)**

Removes a user from the user\_list.

data ::= user\_list\_entry (see 5.4.7.2.10)

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 463/668 |
|-----------------------|------------|-----------------------------|---------|

## 5.4.8 Security setup (class\_id = 64, version = 0)

### 5.4.8.1 Overview

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. 10:2020, Clause 5.

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

### 5.4.8.2 Attribute description

#### 5.4.8.2.1 logical\_name

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

#### 5.4.8.2.2 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

#### 5.4.8.2.3 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

#### 5.4.8.2.4 client\_system\_title

Carries the (current) client system title:

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

## COSEM Interface Classes

NOTE 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 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.

### 5.4.8.2.5 server\_system\_title

Carries the server system title.

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

This attribute shall be read only.

### 5.4.8.3 Method description

#### 5.4.8.3.1 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 The security policy can only be strengthened.

#### 5.4.8.3.2 global\_key\_transfer (data)

Updates one or more global keys. The *data* parameter includes wrapped key data. Key data include the key identifiers and the keys themselves.

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

The key wrapping algorithm is as specified by the security suite. The KEK is the master key.

The new key(s) are valid as soon as the method invocation has been confirmed with success.

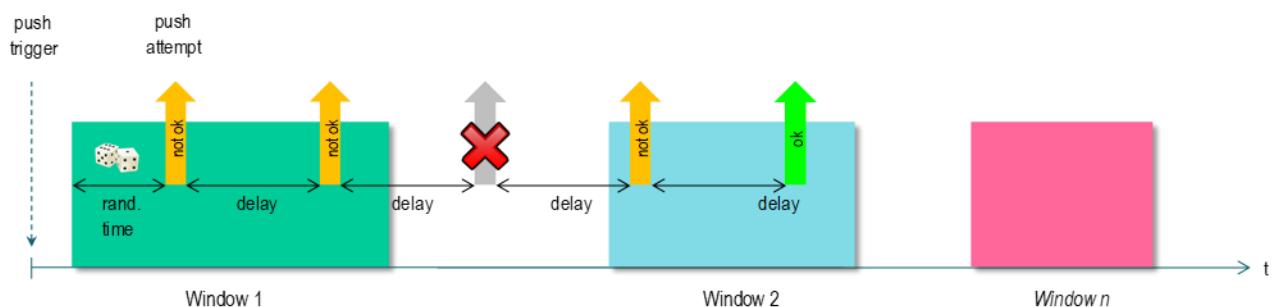
|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 465/668 |
|-----------------------|------------|-----------------------------|---------|

### 5.4.9 Push Setup (class\_id = 40, version = 0)

#### 5.4.9.1 Overview

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

The push takes place upon invoking the *push* method, triggered by a Push "Single action schedule" object, by an alarm "Register monitor" object, by a dedicated internal event or externally. After the push operation has been triggered, it is executed according to the settings made in the given "Push setup" object. Depending on the communication window settings, the push is executed immediately or as soon as a communication window becomes active, after a random delay. If the push was not successful, retries are made. Push windows, delays and retries are shown in Figure 33.



**Figure 33 – 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<br>(static)                 | octet-string  |                            |      |      | x          |  |
| 2. push_object_list<br>(static)             | array         |                            |      |      | x + 0x08   |  |
| 3. send_destination_and_method<br>(static)  | structure     |                            |      |      | x + 0x10   |  |
| 4. communication_window<br>(static)         | array         |                            |      |      | x + 0x18   |  |
| 5. randomisation_start_interval<br>(static) | long-unsigned |                            |      |      | x + 0x20   |  |
| 6. number_of_retries<br>(static)            | unsigned      |                            |      |      | x + 0x28   |  |
| 7. repetition_delay<br>(static)             | long-unsigned |                            |      |      | x + 0x30   |  |
| Specific methods                            | m/o           |                            |      |      |            |  |
| 1. push (data)                              | m             |                            |      |      |            |  |

#### 5.4.9.2 Attribute description

##### 5.4.9.2.1 logical\_name

Identifies the "Push setup" object instance. See 6.2.24.

#### 5.4.9.2.2 push\_object\_list

Defines the list of attributes to be pushed.

Upon invocation of the *push* (data) method the items are sent to the destination defined in the *send\_destination\_and\_method* attribute.

```

array      object_definition

object_definition ::=   structure

{
    class_id:          long-unsigned,
    logical_name:      octet-string,
    attribute_index:   integer,
    data_index:        long-unsigned
}

```

Where:

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

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

If the data\_type of the attribute is simple, then data\_index has no meaning.

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

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

- 0x0000 = identifies the whole attribute;
- 0x0001 to 0xFFFF = 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 9.

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.

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.

#### 5.4.9.2.3 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.

#### 5.4.9.2.4 communication\_window

Defines the time points when the communication window(s) for the push become(s) active (start\_time) and inactive (end\_time). See Figure 31.

```
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* including wildcards.

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

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

#### 5.4.9.2.5 randomisation\_start\_interval

To avoid simultaneous push operations by a lot of devices at exactly the same point in time, a randomisation interval – in seconds – can be defined. This means that the push operation is not started immediately after the invocation of the *push* method but randomly delayed within the interval.

The *randomisation\_start\_interval* attribute defines the maximum value which can be obtained from a randomizing algorithm. The resulting value is used to delay the first push operation. It is not used anymore in case of retries.

If no communication window is active when invoking the *push* method, the delay starts at the beginning of the next communication window. If the *push* method is invoked during an active communication window, the push operation is still delayed.

The *randomisation\_start\_interval* is only active for the first push attempt. If no *communication\_window* is defined, the *randomisation\_start\_interval* is active for every push attempt.

If the *randomisation\_start\_interval* is set to 0 no delay is active.

If the randomisation time is longer than the first communication window, the first push attempt will be made during any later window.

#### 5.4.9.2.6 number\_of\_retries

Defines the maximum number of retries in the case of unsuccessful or skipped push attempts. After a successful push operation no further push attempts are made until the push operation is triggered again. A push is treated as successful if a lower layer transmission confirmation has been received.

The conditions of detecting an unsuccessful push attempt may depend on the communication profile used and on the implementation and it is therefore out of the Scope of this document.

#### 5.4.9.2.7 repetition\_delay

The time delay, expressed in seconds until the next push attempt is started after an unsuccessful push.

**NOTE 1** The repetition delay itself is not influenced by the communication window. But a retry only can be made if a communication window is active at that time. Otherwise it is handled like an unsuccessful push attempt.

**NOTE 2** 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.

### 5.4.9.3 Method description

#### 5.4.9.3.1 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)

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 469/668 |
|-----------------------|------------|-----------------------------|---------|

### 5.4.10 Push Setup (class\_id = 40, version = 1)

#### 5.4.10.1 Overview

See 5.4.9.

In version 1 the possibility of data protection has been added offering the same options as defined in the “Data protection” IC.

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

| Push setup                                  | 0...n         | class_id = 40, version = 1 |      |      |            |
|---|---------------|----------------------------|------|------|------------|
| Attributes                                  | Data type     | Min.                       | Max. | Def. | Short name |
| 1. logical_name<br>(static)                 | octet-string  |                            |      |      | x          |
| 2. push_object_list<br>(static)             | array         |                            |      |      | x + 0x08   |
| 3. send_destination_and_method<br>(static)  | structure     |                            |      |      | x + 0x10   |
| 4. communication_window<br>(static)         | array         |                            |      |      | x + 0x18   |
| 5. randomisation_start_interval<br>(static) | long-unsigned |                            |      |      | x + 0x20   |
| 6. number_of_retries<br>(static)            | unsigned      |                            |      |      | x + 0x28   |
| 7. repetition_delay<br>(static)             | long-unsigned |                            |      |      | x + 0x30   |
| 8. port_reference<br>(static)               | octet-string  |                            |      |      | x + 0x38   |
| 9. push_client_SAP<br>(static)              | integer       |                            |      |      | x + 0x40   |
| 10. push_protection_parameters<br>(static)  | array         |                            |      |      | x + 0x48   |
| Specific methods                            | m/o           |                            |      |      |            |
| 1. push (data)                              | m             |                            |      |      |            |

#### 5.4.10.2 Attribute description

##### 5.4.10.2.1 logical\_name

Identifies the “Push setup” object instance. See 6.2.24.

##### 5.4.10.2.2 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.

Two, mutually exclusive selective access mechanisms are available :

## COSEM Interface Classes

- relative selective access, i.e. entries defined relative to current date or entry are returned: this mechanism is controlled by the *data\_index* element; or
- absolute selective access, i.e. entries in an explicitly defined date range or entry range are returned: entries selected by this mechanism are controlled by the *restriction\_element* and the columns are controlled by the lower nibble of the MS byte in the *data\_index*.

```

array push_object_definition

push_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 first attribute (i.e. the logical\_name), attribute\_index 2 to the second attribute etc.; attribute\_index 0 refers to all public attributes, noting DLMS UA 1000-2 Ed.11:2021, 4.2.4.3.7;
- data\_index is a pointer selecting a specific element 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 – other than the buffer of a Profile generic object – 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.

| <b>data_index</b> | <b>MS-Byte</b> |              | <b>LS-Byte</b> |
|-------------------|----------------|--------------|----------------|
|                   | Upper nibble   | Lower nibble |                |
|                   |                |              |                |

- 0x0000 = the whole attribute is pushed.
- 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 9.

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

```

restriction_element ::= structure

{
    restriction_type: enum:

```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 471/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

```

        (0) none,
        (1) range by date,
        (2) range 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 :
- MS Byte upper nibble set to 0x0;
- MS Byte lower nibble set to 0x0 to 0xF in accordance with Table 9;
- lower byte set to 0x00.
- 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.

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 also pushed to identify the data pushed.

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.

### **5.4.10.2.3 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®,
    (8)      DLMS Gateway
    (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.

#### 5.4.10.2.4 communication\_window

Defines the time points when the communication window(s) for the push become(s) active (start\_time) and inactive (end\_time). See Figure 32.

```
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 including wildcards.

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

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

#### 5.4.10.2.5 randomisation\_start\_interval

To avoid simultaneous push operations by a lot of devices at exactly the same point in time, a randomisation interval – in seconds – can be defined. This means that the push operation is not started immediately after the invocation of the *push* method but randomly delayed within the interval.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 473/668 |
|-----------------------|------------|-----------------------------|---------|

The *randomisation\_start\_interval* attribute defines the maximum value which can be obtained from a randomizing algorithm. The resulting value is used to delay the first push operation. It is not used anymore in case of retries.

If no communication window is active when invoking the *push* method, the delay starts at the beginning of the next communication window. If the *push* method is invoked during an active communication window, the push operation is still delayed.

The *randomisation\_start\_interval* is only active for the first push attempt. If no *communication\_window* is defined, the *randomisation\_start\_interval* is active for every push attempt.

If the *randomisation\_start\_interval* is set to 0 no delay is active.

If the randomisation time is longer than the first communication window, the first push attempt will be made during any later window.

#### **5.4.10.2.6 number\_of\_retries**

Defines the maximum number of retries in the case of unsuccessful or skipped push attempts. After a successful push operation no further push attempts are made until the push operation is triggered again. A push is treated as successful if a lower layer transmission confirmation has been received.

The conditions of detecting an unsuccessful push attempt may depend on the communication profile used and on the implementation and it is therefore out of the Scope of this specification.

#### **5.4.10.2.7 repetition\_delay**

The time delay, expressed in seconds until the next push attempt is started after an unsuccessful push.

NOTE 1      The repetition delay itself is not influenced by the communication window. But a retry can only be made if a communication window is active at that time. Otherwise it is handled like an unsuccessful push attempt.

NOTE 2      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.

#### **5.4.10.2.8 port\_reference**

Contains the logical name of a communication port setup object allowing the selection of a specific communication channel for the push based on the *transport\_service\_type*. This mainly applies in cases where several channels of the same type are supported.

If this information is not available or not needed, the attribute may be left empty (octet-string [0]).

#### **5.4.10.2.9 push\_client\_SAP**

Defines the client SAP where the push is directed to in the supporting layer of the DataNotification service. The push process takes place within the application context of the AA which is linked to the *push\_client\_SAP* attribute. The security context is determined by the "Security setup" object referenced in the related "Association" object.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 474/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

#### 5.4.10.2.10 push\_protection\_parameters

Specifies all protection parameters to be applied to the DataNotification APDU when the push data is sent. It offers the same options as the “Data Protection” IC but it is bound to the sending of the data defined in the *push\_object\_list* attribute.

```

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.

The fields *transaction-id* ....*other-information* are A-XDR encoded OCTET STRINGS. The length and the value of each field are included in the AAD when applicable.

```

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

    key_info_options: CHOICE

    {
        identified_key_info_options,
        wrapped_key_info_options,

```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 475/668 |
|-----------------------|------------|-----------------------------|---------|

```

        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 elements.

For the use of the various elements see Table 23 and Table 24 of the "Data protection" IC (class\_id = 30, version = 0).

#### 5.4.10.3 Method description

##### 5.4.10.3.1 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 class.

data ::= integer(0)

#### 5.4.11 Push setup (class\_id = 40, version = 2)

##### 5.4.11.1 Overview

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.

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

In version 2 six new possibilities are specified:

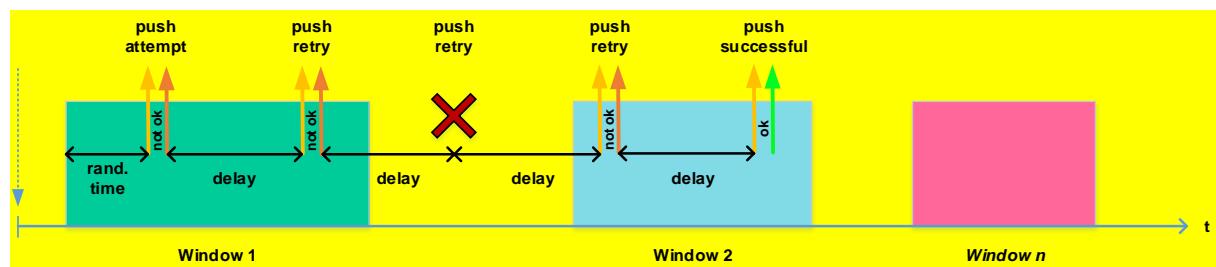
- a new entry selection mechanism allows selecting entries related to the last confirmed entry;
- a new column selection mechanism allows explicitly selecting columns of Profile generic object buffer attributes;
- the **repetition\_delay** attribute now provides an exponential formula that allows increasing the repetition delay with each retry attempt;

## COSEM Interface Classes

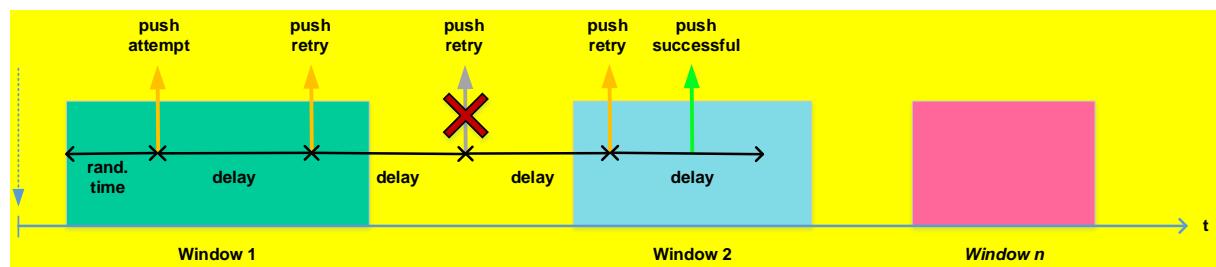
- a new **push\_operation\_method** attribute allows specifying if the DataNotification.request service primitive is invoked with Service\_Class == Unconfirmed or confirmed and how the push retry operation works;
- a new **confirmation\_parameters** attribute allows limiting the time interval in which entries related to last confirmed entry can be selected;
- a new **last\_confirmation\_date\_time** attribute holds the date\_time when the DataNotification service was last confirmed.

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

The push is started when the *push* method is invoked, triggered by a Push “Single action schedule” object, by an alarm “Register monitor” object, by a dedicated internal event or externally. After the push operation has been triggered, it is executed according to the settings made in the given “Push setup” object. Depending on the communication window settings, the push is executed immediately or as soon as a communication window becomes active, after a random delay. If the push was not successful, retries are made. Push retries may be made when supporting protocol layer failure is indicated or when confirmation is missing. Push operation with windows, delays and retries is shown in Figure 16.



Case a) Retry on supporting layer failure



Case b) Retry on missing confirmation

**Figure 34 – Push windows and delays**

## COSEM Interface Classes

| <b>Push setup</b>                        | <b>0...n</b>     | <b>class_id = 40, version = 2</b> |             |             |                   |
|--|------------------|-----------------------------------|-------------|-------------|-------------------|
| <b>Attribute (s)</b>                     | <b>Data type</b> | <b>Min.</b>                       | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 1. logical_name (static)                 | octet-string     |                                   |             |             | x                 |
| 2. push_object_list (static)             | array            |                                   |             |             | x + 0x08          |
| 3. send_destination_and_method (static)  | structure        |                                   |             |             | x + 0x10          |
| 4. communication_window (static)         | array            |                                   |             |             | x + 0x18          |
| 5. randomisation_start_interval (static) | long-unsigned    |                                   |             |             | x + 0x20          |
| 6. number_of_retries (static)            | unsigned         |                                   |             |             | x + 0x28          |
| 7. repetition_delay (static)             | structure        |                                   |             |             | x + 0x30          |
| 8. port_reference (static)               | octet-string     |                                   |             |             | x + 0x38          |
| 9. push_client_SAP (static)              | integer          |                                   |             |             | x + 0x40          |
| 10. push_protection_parameters (static)  | array            |                                   |             |             | x + 0x48          |
| 11. push_operation_method (static)       | enum             |                                   |             |             | x + 0x50          |
| 12. confirmation_parameters (static)     | structure        |                                   |             |             | x + 0x58          |
| 13. last_confirmation_date_time (dyn.)   | date-time        |                                   |             |             | x + 0x60          |
| <b>Specific methods</b>                  | <b>m/o</b>       |                                   |             |             |                   |
| 1. push (data)                           | m                |                                   |             |             | x + 0x38          |
| 2. reset (data)                          | o                |                                   |             |             | x + 0x70          |

### 5.4.11.2 Attribute description

#### 5.4.11.2.1 logical\_name

Identifies the “Push setup” object instance. See 6.2.24.

#### 5.4.11.2.2 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      push_object_definition
push_object_definition ::=   structure
{
    class_id:      long-unsigned,
    logical_name: octet-string,
    attribute_index: integer,
    data_index:      long-unsigned,
    restriction:    restriction_element,

```

## COSEM Interface Classes

```

        columns:      column_element
    }

    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
    }

    column_element ::= array capture_object_definition

```

Where:

- class\_id, logical\_name and attribute\_index reference a COSEM object attribute. For referencing all attributes of an object, see DLMS UA 1000-2 Ed. 10:2020, 9.1.4.3.7, attribute\_0 referencing.

If the attribute is simple or if attribute\_0 is referenced, then:

- data\_index has no meaning and shall be set to 0x0000;
- restriction\_element restriction\_type shall be 0 (none) and restriction\_element restriction\_value shall be null-data;
- column\_element shall be an array of 0 elements.

If the attribute is structure or an array – other than Profile generic object buffer then:

- data\_index points to one element in the structure or array and can take any value between 0x0001 and 0x0FFF. data\_index set to 0x0000 selects the whole attribute;
- restriction\_element restriction\_type shall be 0 (none) and restriction\_element restriction\_value shall be null-data;
- column\_element shall be an array of 0 elements.

If the attribute is a Profile generic object buffer, then data\_index, restriction and column define selective access parameters to select entries and columns.

There are three, mutually exclusive entry selection mechanisms available:

- 1) Relative selective access related to current date and time. Entries in last time intervals relative to current date and time or last entries as identified by data\_index are selected;

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
| 479/668               |            |                             |

- 2) Relative selective access related to last confirmed entry. Entries in next time intervals relative to last confirmed entry or next entries as identified by data\_index are selected;
- 3) Absolute selective access. Entries in an explicitly defined date range or entry range as identified by restriction are selected.

There are two, mutually exclusive column selection mechanisms available:

- a) a defined number of columns starting from column 1 are selected;
- b) an explicitly defined list of columns are selected.

In the case of entry selection mechanism (1) and (2) the selective access parameters are defined by data\_index interpreted as three different fields as shown below. The encoding is shown in Table 9.

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

Where:

- data\_index MS-Byte Upper nibble defines how the entries are selected;
- data\_index MS-Byte Lower nibble is used to specify the number of columns when applicable;
- data\_index LS-Byte defines the number of entries or time intervals.

In the case of entry selection mechanism (1) and (2), restriction\_element restriction\_type shall be 0 and restriction\_element restriction\_value shall be null\_data.

In the case of entry selection mechanism (3) the selective access parameters are defined by restriction\_element. In this case Data\_index MS\_byte Upper\_nibble shall be set to 0x0, and Data\_index LS\_byte shall be set to 0x00.

- for restriction by date range restriction\_type holds (1) range by date and restriction\_value holds restriction\_by\_date structure;
- for restriction by entries restriction\_type holds (2) range by entry and restriction\_value holds restriction\_by\_entry structure;
- if selective access is not required, restriction\_type holds (0) none and restriction\_value holds null-data. This choice shall be taken also if relative selective access is used.

In the case of column selection mechanism (a), Data\_index MS\_byte Lower\_nibble defines the number of columns to be selected, starting from column 1. In this case, column\_element shall be an empty array.

In the case of column selection mechanism (b), the columns to be selected are identified by column\_element. In this case, Data\_index MS\_byte Lower\_nibble shall be set to 0.

If relative selective access related to last confirmed entry is used, the last confirmed entry is updated when the DataNotification.confirm service primitive is invoked with Result == CONFIRMED.

The DataNotification APDU carrying the push data is protected as stipulated by the security suite, security policy and the security material held by the “Security setup” object referenced from the Association SN / Association LN object within the application context of the AA which is linked to the push\_client\_SAP attribute; as well as by the push\_protection\_parameters attribute. For attributes to which no access right is granted within this AA, or which cannot be accessed for any other reason, null-data should be returned.

## COSEM Interface Classes

- 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 also pushed to identify the data pushed.
- NOTE 3 Last confirmed entry is an internal variable maintained by the server AP

### 5.4.11.2.3 `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®  
(8)      DLMS Gateway  
(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.

### 5.4.11.2.4 `communication_window`

Defines the time points when the communication window(s) for the push become(s) active (`start_time`) and inactive (`end_time`). See Figure 16.

`array window_element`

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 481/668 |
|-----------------------|------------|-----------------------------|---------|

```
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* including wildcards.

If the end of a communication window is reached when a push has already started the operation will be completed.

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

#### 5.4.11.2.5 randomisation\_start\_interval

Defines a maximum delay, in seconds, of sending the first DataNotification APDU after invoking the push method. This is to avoid a situation where many servers may push simultaneously. It is used as follows:

- the delay is random from 0 up to the maximum value. A value of 0 deactivates the mechanism;
- if the push is invoked within an active communication window, the random delay is applied. It is not applied for retries;
- if the push is invoked outside a communication window, then the random delay is applied when the communication windows opens;
- if the random delay extends beyond the current communication window, then the DataNotification APDU will be sent in the next window if available;
- if no communication windows are defined, then the random delay is applied whenever the push method is invoked including all retries.

#### 5.4.11.2.6 number\_of\_retries

Defines the maximum number of retries in the case of unsuccessful or skipped push attempts. After a successful push operation no further push attempts are made until the push operation is triggered again. For further details on the retransmission operation see the *push\_operation\_method* attribute (5.4.11.2.11).

#### 5.4.11.2.7 repetition\_delay

Defines the elements for calculating the time delay until the next attempt after an unsuccessful push. See Figure 16.

```
repetition_delay ::= structure
{
    repetition_delay_min: long-unsigned,
    repetition_delay_exponent: long-unsigned,
    repetition_delay_max: long-unsigned
}
```

Where:

- `repetition_delay_min` is the minimum delay, in seconds, until a next push attempt is started;
- `repetition_delay_exponent` is used for calculating the next delay;

- repetition\_delay\_max is the maximum delay, in seconds. Any calculated time delay will be capped by this value.

The repetition delay is calculated using the following formula:

$$\text{repetition\_delay} = \text{repetition\_delay\_min} \times (\text{repetition\_delay\_exponent} \times 0.01)^{n-1}$$

where n refers to the ordinal number of the push retries with n = 1 for the first retry etc.

**NOTE 1** The repetition delay itself is not influenced by the communication window. But a push retry only can be made if a communication window is active at that time. Otherwise it is handled like an unsuccessful push attempt.

**NOTE 2** The push data is not stored in an intermediate buffer. In the case of push retries, the current values of the attributes may change with every push retry attempt.

#### 5.4.11.2.8 port\_reference

Contains the logical name of a communication port setup object allowing the selection of a specific communication channel for the push based on the transport\_service\_type. This mainly applies in cases where several channels of the same type are supported.

If this information is not available or not needed, the attribute may be left empty (octet-string [0]).

#### 5.4.11.2.9 push\_client\_SAP

Defines the client SAP where the push is directed to in the supporting layer of the DataNotification service. The push process takes place within the application context of the AA which is linked to the push\_client\_SAP attribute. The security context is determined by the Security setup object referenced in the related Association SN /LN object.

#### 5.4.11.2.10 push\_protection\_parameters

Specifies all protection parameters to be applied to the DataNotification APDU when the push data is sent. It offers the same options as the Data Protection IC but it is bound to the sending of the data defined in the push\_object\_list attribute.

##### 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:

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 483/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

- transaction\_id holds the identifier of the transaction;
- originator\_system\_title holds the system title of the originator that applies the protection;
- recipient\_system\_title holds the system title of the recipient which will check and remove the given protection;
- other\_information carries other information. Its contents may be specified in project specific companion specifications. An octet-string of length 0 indicates that this field is not used;
- key\_info holds the information necessary for the recipient to obtain the right key for checking and removing authentication and encryption. In the case of digital signature, key\_info is not necessary and it shall be a structure of 0 elements.

The fields transaction-id....other-information are A-XDR encoded OCTET STRINGS. The length and the value of each field are included in the AAD when applicable.

```

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

    key_info_options: CHOICE
    {
        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.

The use of the various elements is the same as specified in Table 23 and Table 24 of the Data protection IC (class\_id = 30, version = 0).

**5.4.11.2.11 push\_operation\_method**

Defines if the DataNotification.request service primitive is invoked with Service\_Class == Unconfirmed or Confirmed and how the push retry operates.

enum:

- (0) unconfirmed, retry on supporting protocol layer failure,
- (1) unconfirmed, retry on missing supporting protocol layer confirmation,
- (2) confirmed, retry on missing confirmation.

In case (0), the repetition delay for the next retry attempt is started upon supporting layer failure reported through the DataNotification.confirm service primitive.

In cases (1) and (2), the repetition delay for the next retry attempt is started when the DataNotification.request service primitive is invoked.

In cases (0) and (1), the push operation is deemed as successful when supporting layer confirmation is reported by invoking DataNotification.confirm service primitive with Result == CONFIRMED.

In case (2), the push operation is deemed as successful when the server AL receives the data-notification-confirm APDU and invokes DataNotification.confirm service primitive with Service\_Class == Confirmed and Result == CONFIRMED.

**5.4.11.2.12 confirmation\_parameters**

Defines the selection of entries defined by data\_index to avoid pushing data from too far in the past. If entries have not yet been confirmed then all entries are selected. This attribute is applicable only for relative selective access related to last confirmation.

```
confirmation_parameters ::= structure
{
    confirmation_start_date:      date-time,
    confirmation_interval:        double-long-unsigned
}
```

Where:

- confirmation\_start\_date is the starting date and time for selecting entries. Fields of date-time not specified are not used. If all fields are not specified the use of confirmation\_start\_date is disabled;
- confirmation\_interval is a time interval in seconds backwards from the current date and time. If confirmation\_interval is set to zero, the use of confirmation\_interval is disabled.

**5.4.11.2.13 last\_confirmation\_date\_time**

Holds the date and time when the AL most recently invoked the DataNotification.confirm service primitive with Result == CONFIRMED.

*date-time* is formatted as specified in 4.1.6.1.

**5.4.11.3 Method description****5.4.11.3.1 push (data)**

Activates the push process leading to an attempt to send a DataNotification APDU carrying the push data.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 485/668 |
|-----------------------|------------|-----------------------------|---------|

|                     |
|---------------------|
| data ::= integer(0) |
|---------------------|

#### 5.4.11.3.2 reset (data)

Resets the push process to initial state.

|                     |
|---------------------|
| data ::= integer(0) |
|---------------------|

## 5.5 Previous versions of interface classes for time- and event-bound control

### 5.5.1 Parameter monitor (class\_id = 65, version = 0)

#### 5.5.1.1 Overview

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 DLMS UA 1000-1 Ed 15 Part 1:2021, 6.5.

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

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

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

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

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

**5.5.1.2 Attribute description****5.5.1.2.1 logical\_name**

Identifies the “Parameter monitor” object instance. See 6.2.14.

**5.5.1.2.2 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
}
```

**5.5.1.2.3 capture\_time**

Provides data and time information showing when the value of the *changed\_parameter* attribute has been captured.

*date-time* is formatted as specified in 4.1.6.1.

**5.5.1.2.4 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 The list of parameters monitored may be changed by using the *add\_parameter* or *delete\_parameter* methods or writing this attribute.

**5.5.1.3 Method description****5.5.1.3.1 add\_parameter (data)**

Adds one parameter to the *parameter\_list*.

```
data ::= parameter_list_element
```

NOTE A parameter can be logged as soon as it is added to the list. Adding an element to the list does not affect the *buffer* of the “Profile generic” object capturing the *changed\_parameter* attribute.

**5.5.1.3.2 delete\_parameter (data)**

Deletes one parameter from the *parameter\_list*.

```
data ::= parameter_list_element
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 487/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

**NOTE** 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 *buffer* of the “Profile generic” object capturing the *changed\_parameter* attribute.

### 5.6 Previous versions of payment metering related interface classes

There are no previous versions to report.

### 5.7 Previous versions of interface classes for setting up data exchange via local ports and modems

#### 5.7.1 IEC local port setup (class\_id = 19, version = 0)

##### 5.7.1.1 Overview

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          |                            |      |      |            |

##### 5.7.1.2 Attribute description

###### 5.7.1.2.1 logical\_name

Identifies the “IEC local port setup” object instance. See 6.2.18.

###### 5.7.1.2.2 default\_mode

Defines the protocol used by the meter on the port.

enum:

- (0) protocol according to IEC 62056-21:2002 (modes A...E)
- (1) protocol according to IEC 62056-46:2002/AMD1:2006, Clause 8. Using this enumeration value all other attributes of this IC are not applicable.

**5.7.1.2.3 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

**5.7.1.2.4 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

**5.7.1.2.5 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

**5.7.1.2.6 device\_addr**

Device address according to IEC 62056-21:2002.

**5.7.1.2.7 pass\_p1**

Password 1 according to IEC 62056-21:2002.

**5.7.1.2.8 pass\_p2**

Password 2 according to IEC 62056-21:2002.

**5.7.1.2.9 pass\_w5**

Password W5 reserved for national applications.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 489/668 |
|-----------------------|------------|-----------------------------|---------|

## 5.7.2 IEC HDLC setup, (class\_id = 23, version = 0)

### 5.7.2.1 Overview

An instance of the “IEC HDLC setup” contains all data necessary to set up a communication channel according to IEC 62056-46:2002/AMD1:2006 . Several communication channels can be configured.

| IEC HDLC setup                    |          | 0...n         | class_id = 23, version = 0 |        |      |            |
|-----------------------------------|----------|---------------|----------------------------|--------|------|------------|
| Attributes                        |          | Data type     | Min.                       | Max.   | Def. | Short name |
| 1. logical_name                   | (static) | octet-string  |                            |        |      | x          |
| 2. comm_speed                     | (static) | enum          | 0                          | 9      | 5    | x + 0x08   |
| 3. window_size_transmit           | (static) | unsigned      | 1                          | 7      | 1    | x + 0x10   |
| 4. window_size_receive            | (static) | unsigned      | 1                          | 7      | 1    | x + 0x18   |
| 5. max_info_field_length_transmit | (static) | unsigned      | 32                         | 128    | 128  | x + 0x20   |
| 6. max_info_field_length_receive  | (static) | unsigned      | 32                         | 128    | 128  | x + 0x28   |
| 7. inter_octet_time_out           | (static) | long-unsigned | 20                         | 1000   | 25   | x + 0x30   |
| 8. inactivity_time_out            | (static) | long-unsigned | 0                          |        | 120  | x + 0x38   |
| 9. device_address                 | (static) | long-unsigned | 0x0010                     | 0x3FFD |      | x + 0x40   |
| Specific methods                  |          | m/o           |                            |        |      |            |

### 5.7.2.2 Attribute description

#### 5.7.2.2.1 logical\_name

Identifies the “IEC HDLC setup” object instance. See 6.2.20.

#### 5.7.2.2.2 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.

**5.7.2.2.3 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.

**5.7.2.2.4 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.

**5.7.2.2.5 max\_info\_length\_transmit**

The maximum information field length that a device can transmit. During logon, a smaller value can be negotiated.

**5.7.2.2.6 max\_info\_length\_receive**

The maximum information field length that a device can receive. During logon, a smaller value can be negotiated.

**5.7.2.2.7 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.

**5.7.2.2.8 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.

**5.7.2.2.9 device\_address**

Contains the physical device address of a device:

In the case of single byte addressing:

|             |                           |
|-------------|---------------------------|
| 0x00        | NO_STATION Address,       |
| 0x01...0x0F | Reserved for future use,  |
| 0x10...0x7D | Usable address space,     |
| 0x7E        | 'CALLING' device address, |
| 0x7F        | Broadcast address         |

In the case of double byte addressing:

|                 |                                    |
|-----------------|------------------------------------|
| 0x0000          | NO_STATION address,                |
| 0x0001...0x000F | Reserved for future use,           |
| 0x0010...0x3FFD | Usable address space,              |
| 0x3FFE          | 'CALLING' physical device address, |
| 0x3FFF          | Broadcast address                  |

### 5.7.3 IEC twisted pair (1) setup (class\_id = 24, version = 0)

#### 5.7.3.1 Overview

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   |
| <b>Specific methods</b>          | <b>m/o</b>                |                            |      |      |            |

#### 5.7.3.2 Attribute description

##### 5.7.3.2.1 logical\_name

Identifies the “IEC twisted pair setup” object instance. See 6.2.21.

##### 5.7.3.2.2 secondary\_address

*Secondary\_address* memorizes the ADS of the secondary station that corresponds to the real equipment.

octet-string (SIZE(6))

##### 5.7.3.2.3 primary\_address\_list

*Primary\_address\_list* memorizes the list of ADP or primary station physical addresses for which each logical device of the real equipment (the secondary station) has been programmed.

primary\_address\_list\_type ::= array primary\_address\_element

primary\_address\_element ::= octet-string

The length of the octet-string is one octet.

##### 5.7.3.2.4 tabi\_list

*tabi\_list* represents the list of the TAB(i) for which the real equipment (the secondary station) has been programmed in case of forgotten station call.

tabi\_list\_type ::= array tabi\_element

tabi\_element ::= integer

### 5.7.3.2.5 fatal\_error

*fatal\_error* represents the last occurrence of one of the fatal errors of the protocols described in IEC 62056-31:1999.

The initial default value of this variable is 0x00. Then, each fatal error is spotted.

enum:

|      |           |
|------|-----------|
| (0)  | No-error, |
| (1)  | t-EP-1F,  |
| (2)  | t-EP-2F,  |
| (3)  | t-EL-4F,  |
| (4)  | t-EL-5F,  |
| (5)  | eT-1F,    |
| (6)  | eT-2F,    |
| (7)  | e-EP-3F,  |
| (8)  | e-EP-4F,  |
| (9)  | e-EP-5F,  |
| (10) | e-EL-2F   |

### 5.7.3.2.6 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 53. This address shall be worldwide unique and it shall assigned by the manufacturer to the secondary station. The ADS assigned is valid through the lifetime of the secondary station.

**Table 53 – 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 DLMS UA 1000-1 Ed 15 Part 1:2021.             | 3             |      | 000001-999999<br>000000 is reserved |
| EXAMPLE              | 031267123456:   |               |      |                                     |
|                      | 03: ITRON International;  |               |      |                                     |
|                      | 12: Year 2012;  |               |      |                                     |
|                      | 67: Smart meter LINKY Single phase  |               |      |                                     |
|                      | 123456: Serial number beginning each year from 000001.  |               |      |                                     |

## 5.7.4 PSTN modem configuration (class\_id = 27, version = 0)

### 5.7.4.1 Overview

NOTE The name of this IC was changed to “Modem configuration” in Edition 2.

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          |                            |      |      |            |

### 5.7.4.2 Attribute description

#### 5.7.4.2.1 logical\_name

Identifies the “PSTN modem configuration” object instance. See 6.2.6.

#### 5.7.4.2.2 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

#### 5.7.4.2.3 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.

## COSEM Interface Classes

**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.

### 5.7.4.2.4 **modem\_profile**

This data defines the mapping from Hayes standard commands/responses to modem specific strings.

array       **modem\_profile\_element**

modem\_profile\_element: octet-string

The *modem\_profile* array shall contain the corresponding strings for the modem used in following order:

|             |                 |
|-------------|-----------------|
| Element 0:  | OK,             |
| Element 1:  | CONNECT,        |
| Element 2:  | RING,           |
| Element 3   | NO CARRIER,     |
| Element 4:  | ERROR,          |
| Element 5:  | CONNECT 1 200,  |
| Element 6   | NO DIAL TONE,   |
| Element 7:  | BUSY,           |
| Element 8:  | NO ANSWER,      |
| Element 9:  | CONNECT 600,    |
| Element 10: | CONNECT 2 400,  |
| Element 11: | CONNECT 4 800,  |
| Element 12  | CONNECT 9 600,  |
| Element 13: | CONNECT 14 400, |
| Element 14: | CONNECT 28 800, |
| Element 15: | CONNECT 36 600, |
| Element 16: | CONNECT 56 000  |

## 5.7.5 Auto answer (**class\_id = 28, version = 0**)

### 5.7.5.1 Overview

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.

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

**5.7.5.2 Attribute description****5.7.5.2.1 logical\_name**

Identifies the “Auto answer” object instance. See 6.2.6.

**5.7.5.2.2 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

**5.7.5.2.3 listening\_window**

Contains the start and end instant when the window becomes active (for the start instant), and inactive (for the end instant). The start\_date defines implicitly the period.

**EXAMPLE** When the day of month is not specified (equal to 0xFF) this means that we have a daily share line management. Daily, monthly ...window management can be defined.

```
array window_element
    window_element ::= structure
    {
        start_time: octet-string,
        end_time: octet-string
    }
```

start\_time and end\_time are formatted as specified in 4.6.1 for *date-time*.

**5.7.5.2.4 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,

## COSEM Interface Classes

- (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.

### **5.7.5.2.5 number\_of\_calls**

This number is the reference used in modes 1 and 2.

When set to 0, this means there is no limit.

### **5.7.5.2.6 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 *listening\_window* and number of rings outside the *listening\_window*.

```

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)
}

```

## **5.7.6 PSTN auto dial (class\_id = 29, version = 0)**

### **5.7.6.1 Overview**

NOTE The name of this IC was changed to "Auto connect" in Edition 2.

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.

| 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       |                            |      |      |            |  |

**5.7.6.2 Attribute description****5.7.6.2.1 logical\_name**

Identifies the “PSTN auto dial” object instance. See 6.2.6.

**5.7.6.2.2 mode**

Defines if the device can perform auto-dialling.

enum:

- (0) no auto dialling,
- (1) auto dialling allowed anytime,
- (2) auto dialling allowed within the validity time of the calling window,
- (3) “regular” auto dialling allowed within the validity time of the calling window; “alarm” initiated auto dialling allowed anytime,
- (200...255) manufacturer specific modes

**5.7.6.2.3 repetitions**

The maximum number of trials In the case of unsuccessful dialling attempts.

**5.7.6.2.4 repetition\_delay**

The time delay, expressed in seconds until an unsuccessful dial attempt can be repeated.

repetition\_delay 0 means delay is not specified

**5.7.6.2.5 calling\_window**

Contains the start and end date/time stamp when the window becomes active (for the start instant), or inactive (for the end instant). The start\_date defines implicitly the period.

**EXAMPLE** When day of month is not specified (equal to 0xFF) this means that we have a daily share line management. Daily, monthly ...window management can be defined.

```
array window_element
  window_element ::= structure
  {
    start_time: octet-string,
    end_time: octet-string
  }
```

start\_time and end\_time are formatted as specified in 4.6.1 for *date-time*.

**5.7.6.2.6 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

```

### 5.7.7 Auto connect (class\_id = 29, version = 1)

#### 5.7.7.1 Overview

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           |                            |      |      |            |

#### 5.7.7.2 Attribute description

##### 5.7.7.2.1 logical\_name

Identifies the “Auto connect” object instance. See 6.2.6.

##### 5.7.7.2.2 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

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 499/668 |
|-----------------------|------------|-----------------------------|---------|

**5.7.7.2.3 repetitions**

The maximum number of trials in the case of unsuccessful dialling attempts.

**5.7.7.2.4 repetition\_delay**

The time delay, expressed in seconds until an unsuccessful dial attempt can be repeated.

repetition\_delay 0 means delay is not specified

**5.7.7.2.5 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 date-time.

**5.7.7.2.6 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
```

**5.7.8 GSM diagnostic (class\_id = 47, version = 0)****5.7.8.1 Overview**

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 DLMS UA 1000-1 Ed 15 Part 1:2021, 6.5.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 500/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

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

### 5.7.8.2 Attribute description

#### 5.7.8.2.1 logical\_name

Identifies the “GSM Diagnostic” object instance. For logical names, see 6.2.23.

#### 5.7.8.2.2 operator

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

#### 5.7.8.2.3 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

#### 5.7.8.2.4 cs\_attachment

Indicates the current circuit switched status.

enum:

- (0) inactive,
- (1) incoming call,
- (2) active,
- (3) ... (255) reserved

#### 5.7.8.2.5 ps\_status

The ps\_status value field indicates the packet switched status of the modem.

enum:

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 501/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

(0) inactive,  
(1) GPRS,  
(2) EDGE,  
(3) UMTS,  
(4) HSDPA,  
(5) ... (255) reserved

### 5.7.8.2.6 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 Rate (BER) measurement in percent:
  - (0...7) as RXQUAL\_n values specified in ETSI GSM 05.08:1996,8.2.4.
  - (99) not known or not detectable.

### 5.7.8.2.7 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.

### 5.7.8.2.8 capture\_time

Holds the date and time when the data have been last captured.

*date-time* is formatted as specified in 4.6.1.

## 5.7.9 GSM diagnostic (class\_id: 47, version = 1)

### 5.7.9.1 Overview

The cellular 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, UMTS, CDMA or LTE 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 DLMS UA 1000-1 Ed 15 Part 1:2021, 6.5.

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

### 5.7.9.2 Attribute description

#### 5.7.9.2.1 logical\_name

Identifies the “GSM Diagnostic” object instance. For logical names, see 6.2.23.

#### 5.7.9.2.2 operator

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

#### 5.7.9.2.3 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,

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 503/668 |
|-----------------------|------------|-----------------------------|---------|

- (5) registered, roaming,

#### 5.7.9.2.4 cs\_attachment

Indicates the current circuit switched status.

```
enum:
(0)          inactive,
(1)          incoming call,
(2)          active,
(3) ... (255) reserved
```

#### 5.7.9.2.5 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)  LTE,
(6)  CDMA,
(7)...(255) reserved
```

#### 5.7.9.2.6 cell\_info

Represents the cell information:

```
cell_info_type ::= structure
{
    cell_ID:           double-long-unsigned,
    location_ID:      long-unsigned,
    signal_quality:   unsigned,
    ber:               unsigned,
    mcc:               long-unsigned,
    mnc:               long-unsigned,
    channel_number:   double-long-unsigned
}
```

Where:

- **cell\_ID:** Four-byte cell ID in hexadecimal format;
- **location\_ID:** Two-byte location area code (LAC) in the case of GSM networks or Tracking Area Code (TAC) in the case of UMTS, CDMA or LTE networks in hexadecimal format (e.g. "00C3" equals 195 in decimal);
- **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 Rate (BER) measurement in percent:
  - (0...7) as RXQUAL\_n values specified in ETSI GSM 05.08:1996, 8.2.4
  - (99) not known or not detectable.
- **mcc:** Mobile Country Code of the serving network, as defined in ITU-T E.212 (05.2008) SERIES E;
- **mnc:** Mobile Network Code of the serving network, as defined in ITU-T E.212 (05.2008) SERIES E;

- channel\_number: Represents the absolute radio-frequency channel number (ARFCN or eaRFCN for LTE network).

#### 5.7.9.2.7 adjacent\_cells

array adjacent\_cell\_info

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

Where:

- cell\_ID: Four-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.

#### 5.7.9.2.8 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.

### 5.7.10 LTE monitoring (class\_id: 151, version: 0)

#### 5.7.10.1 Overview

Instances of the ‘LTE monitoring’ IC allow monitoring LTE modems by handling all data necessary data for this purpose.

| LTE monitoring                   | 0...n        | class_id = 151, version = 0 |      |      |            |
|----------------------------------|--------------|-----------------------------|------|------|------------|
| Attributes                       | Data type    | Min.                        | Max. | Def. | Short name |
| 1. logical_name (static)         | octet-string |                             |      |      | x          |
| 2. lte_quality_of_service (dyn.) | LTE_qos_type |                             |      |      | x + 0x08   |
| Specific methods                 | m/o          |                             |      |      |            |

#### 5.7.10.2 Attribute description

##### 5.7.10.2.1 logical\_name

Identifies the “LTE monitoring” object instance. See 6.2.23.

##### 5.7.10.2.2 lte\_quality\_of\_service

Represents the quality of service for the LTE network

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 505/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

```
LTE_qos_type ::= structure
{
    T3402:          long-unsigned,
    T3412:          long-unsigned,
    RSRQ:           unsigned,
    RSRP:           unsigned,
    qRxlevMin:      integer
}
```

Where:

- T3402: timer in seconds, used on PLMN selection procedure and sent by the network to the modem. Refer to 3GPP TS 24.008 V13.7.0 (2016-10)3GPP TS 24.301 V13.11.0 (2018-01) for details;
- T3412: timer in seconds used to manage the periodic tracking area updating procedure and sent by the network to the modem. Refer to 3GPP TS 24.008 V13.7.0 (2016-10)3GPP TS 24.008 V13.7.0 (2016-10), *Technical Specification Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3*
- 3GPP TS 24.301 V13.4.0 (2016-01)3GPP TS 24.301 V13.4.0 (2016-01) for details;
- RSRQ: represents the signal quality as defined in 3GPP TS 24.008 V13.7.0 (2016-10)3GPP TS 24.008 V13.7.0 (2016-10), *Technical Specification Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3*
- 3GPP TS 24.301 V13.4.0 (2016-01):
  - (0) -19,5dB,
  - (1) -19 dB,
  - (2...31) -18,5...-3,5 dB,
  - (32) -3dB,
  - (99) Not known or not detectable;
- RSRP: represents the signal level as defined in 3GPP TS 24.008 V13.7.0 (2016-10), *Technical Specification Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3*
- 3GPP TS 24.301 V13.4.0 (2016-01):
  - (0) -140dBm,
  - (1) -139 dBm,
  - (2... 94) -138...-45 dBm,
  - (95) -44dBm,
  - (99) Not known or not detectable;
- qRxlevMin: specifies the minimum required Rx level in the cell in dBm as defined in 3GPP TS 24.008 V13.7.0 (2016-10), *Technical Specification Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3*
- 3GPP TS 24.301 V13.4.0 (2016-01).

## 5.8 Previous versions of interface classes for setting up data exchange via M-Bus

### 5.8.1 M-Bus client (class\_id = 72, version = 0)

#### 5.8.1.1 Overview

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 4.8.3.

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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 507/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| <b>M-Bus client</b>             | <b>0...n</b>         | <b>class_id = 72, version = 0</b> |             |             |                   |
|---------------------------------|----------------------|-----------------------------------|-------------|-------------|-------------------|
| <b>Attributes</b>               | <b>Data type</b>     | <b>Min.</b>                       | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 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          |
| <b>Specific methods</b>         | <b>m/o</b>           |                                   |             |             |                   |
| 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          |

### 5.8.1.2 Attribute description

#### 5.8.1.2.1 logical\_name

Identifies the “M-Bus client” object instance. See 6.2.22.

#### 5.8.1.2.2 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.

#### 5.8.1.2.3 capture\_definition

Provides the capture\_definition for M-Bus slave devices.

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

NOTE 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.

**5.8.1.2.4 capture\_period**

$\geq 1$ : Automatic capturing assumed. Specifies the capture period in seconds.

0: No automatic capturing: capturing is triggered externally or capture events occur asynchronously.

**5.8.1.2.5 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 *primary\_address* attribute. From this moment, the data exchange with the M-Bus slave device is possible.

Otherwise, the *slave\_install* method shall be used; see 5.8.1.3.1.

**NOTE** The *primary\_address* 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.

**5.8.1.2.6 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 is provided.

**5.8.1.2.7 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 *manufacturer\_id* 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.

**5.8.1.2.8 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.

**5.8.1.2.9 device\_type**

Carries the Device type identification element of the data header as specified in EN 13757-3:2004, 5.7, Table 3.

**5.8.1.2.10 access\_number**

Carries the Access Number element of the data header as specified in EN 13757-3:2004, 5.8.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 509/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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.

### 5.8.1.2.11 status

Carries the Status byte element of the data header as specified in

EN 13757-3:2004, 5.9, Table 4 and 5.

### 5.8.1.2.12 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.

### 5.8.1.3 Method description

#### 5.8.1.3.1 slave\_install (data)

Installs a slave device, which is yet unconfigured (its primary address is 0).

This method can be successfully invoked only if the value of the primary\_address attribute is 0. The following actions are performed:

the M-Bus address 0 is checked for presence of a new device.

- if no uninstalled M-Bus slave is found, the method invocation fails;
- if the *slave\_install* method is invoked with no parameter, then the primary address is assigned automatically. This is done by checking the *primary\_address* attribute of all M-Bus client objects in the DLMS/COSEM device and then selecting the first unused number. The *primary\_address* attribute is set to this address and it is then transferred to the M-Bus slave device;
- if the *slave\_install* method is invoked with a primary address as a parameter, then the *primary\_address* attribute is set to this value and it is then transferred to the M-Bus slave device.

data ::= unsigned (no data, or a valid primary address)

NOTE Unconfigured slave devices are configured with primary address as specified in EN 13757-3:2004, Annex E.5.

#### 5.8.1.3.2 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 A new M-Bus slave can be installed only, once the value of the *primary\_address* attribute is 0.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 510/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

```
data ::= integer (0)
```

#### **5.8.1.3.3 capture**

Capture values – as specified by the *capture\_definition* attribute – from the M-Bus slave device.

```
data ::= integer (0)
```

#### **5.8.1.3.4 reset\_alarm**

Reset alarm state of the M-Bus slave device.

```
data ::= integer (0)
```

#### **5.8.1.3.5 synchronize\_clock**

Synchronize the clock of the M-Bus slave device with that of the M-Bus client device.

```
data ::= integer (0)
```

#### **5.8.1.3.6 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]
    }
}
```

#### **5.8.1.3.7 set\_encryption\_key**

Sets encryption key to be used with the M-Bus slave device.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 511/668 |
|-----------------------|------------|-----------------------------|---------|

Changing the encryption key requires two steps: First, the key is sent to the M-Bus slave, encrypted with the master key, using the *transfer\_key* method. Second, the key is set in the M-Bus master using the *set\_encryption\_key* method.

```
data ::= octet-string (encryption_key)
```

After the installation of the M-Bus slave, the M-Bus client holds an empty encryption key. With this, encryption of M-Bus telegrams is disabled. Encryption can be disabled by invoking the *set\_encryption\_key* method with null-data as a parameter.

#### **5.8.1.3.8 transfer\_key**

Transfers an encryption key to the M-Bus slave.

```
data ::= octet-string (encrypted_key)
```

Each M-Bus slave device shall be delivered with a default encryption key.

Before encrypted M-Bus telegrams can be used, an operational encryption key has to be sent to the M-Bus slave, by invoking the *transfer\_key* method. The method invocation parameter is the operational encryption key encrypted with the default key of the M-Bus slave device. The M-Bus telegram sent is not encrypted. After the execution, the encryption is enabled and all further telegrams are encrypted.

A new encryption key can be set in the M-Bus client by invoking the *set\_encryption\_key* method with the new encryption key as a parameter.

With further invocations of the *transfer\_key* method, new encryption keys can be sent to the M-Bus slave. The method invocation parameter is the new encryption key encrypted with the default key. The M-Bus telegram is encrypted.

When an M-Bus slave is de-installed, the encryption key is destroyed, but the default key is not affected. Encryption remains disabled until a new encryption is transferred.

### **5.8.2 M-Bus client (class\_id = 72, version = 1)**

#### **5.8.2.1 Overview**

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

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 512/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

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   |

**5.8.2.2 Attribute description****5.8.2.2.1 logical\_name**

Identifies the “M-Bus client” object instance. See 6.2.22.

**5.8.2.2.2 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.

**5.8.2.2.3 capture\_definition**

Provides the *capture\_definition* for M-Bus slave devices.

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

```
array      capture_definition_element

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

**NOTE 2** The elements *data\_information\_block* and *value\_information\_block* correspond to Data Information Block (DIB) and Value Information Block (VIB) described in EN 13757-3:2013, 6.2 and Clause 7 respectively.

**5.8.2.2.4 capture\_period**

$\geq 1$ : Automatic capturing assumed. Specifies the capture period in seconds.

0: No automatic capturing: capturing is triggered externally or capture events occur asynchronously.

**5.8.2.2.5 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 1** The specification of the B field of the OBIS codes limits the range to 1...64 within one logical device. See DLMS UA 1000-1 Ed 15 Part 1:2021, 5.2.

If the slave device is already configured and thus, its primary address is different from 0, then this value shall be written to the *primary\_address* attribute. From this moment, the data exchange with the M-Bus slave device is possible.

Otherwise, the *slave\_install* method shall be used; see 5.8.2.3.1 below.

**NOTE 2** The *primary\_address* 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.

**5.8.2.2.6 identification\_number**

Carries the Identification Number element of the data header as specified in EN 13757-3:2013, 5.5.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 514/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

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.

#### **5.8.2.2.7 manufacturer\_id**

Carries the Manufacturer Identification element of the data header as specified in EN 13757-3:2013, 5.6.

#### **5.8.2.2.8 version**

Carries the Version element of the data header as specified in EN 13757-3:2013, 5.7.

#### **5.8.2.2.9 device\_type**

Carries the Device type identification element of the data header as specified in EN 13757-3:2013, 5.8, Table 6.

#### **5.8.2.2.10 access\_number**

Carries the Access Number element of the data header as specified in EN 13757-3:2013, 5.9.

#### **5.8.2.2.11 status**

Carries the Status byte element of the data header as specified in EN 13757-3:2013, 5.10, Tables 7 and 8.

It is updated with every readout of the M-Bus slave device.

#### **5.8.2.2.12 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.

#### **5.8.2.2.13 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.

#### **5.8.2.2.14 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 515/668 |
|-----------------------|------------|-----------------------------|---------|

**5.8.2.3 Method description****5.8.2.3.1 slave\_install (data)**

Installs a slave device, which is yet unconfigured (its primary address is 0).

data ::= unsigned

This method can be successfully invoked only if the current value of the *primary\_address* attribute is 0. The following actions are performed:

- the M-Bus address 0 is checked for presence of a new device;
- if no uninstalled M-Bus slave is found, the method invocation fails;
- if the *slave\_install* method is invoked with “0” as a parameter, then the primary address is assigned automatically. This is done by checking the *primary\_address* attribute of all M-Bus client objects in the DLMS/COSEM device and then selecting the first unused number. The *primary\_address* attribute is set to this address and it is then transferred to the M-Bus slave device;
- if the *slave\_install* method is invoked with a primary address (other than 0) as a parameter, then the *primary\_address* attribute is set to this value and it is then transferred to the M-Bus slave device.

**NOTE** Unconfigured slave devices are configured with primary address as specified in EN 13757-3:2013, Annex E.5.

**5.8.2.3.2 slave\_deinstall (data)**

De-installs the slave device. The main purpose of this service is to de-install the M-Bus slave device and to prepare the master for the installation of a new device. The following actions are performed:

- the M-Bus address is set to 0 in the M-Bus slave device;
- the encryption key transferred previously to the M-Bus slave device is destroyed; the default key is not affected;
- the *encryption\_key\_status* is set to (0): no encryption\_key;
- the attribute *primary\_address* is also set to 0.

**NOTE** A new M-Bus slave can be installed only once the value of the *primary\_address* attribute is 0.

data ::= unsigned (0)

**5.8.2.3.3 capture (data)**

Captures values – as specified by the *capture\_definition* attribute – from the M-Bus slave device.

data ::= integer (0)

**5.8.2.3.4 reset\_alarm (data)**

Resets alarm state of the M-Bus slave device.

data ::= integer (0)

**5.8.2.3.5 synchronize\_clock (data)**

Synchronize the clock of the M-Bus slave device with that of the M-Bus client device.

```
data ::= integer (0)
```

**5.8.2.3.6 data\_send (data) Sends 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], |
| bcd                  | [13]  |
| integer              | [15], |
| long                 | [16], |
| unsigned             | [17], |
| long-unsigned        | [18], |
| long64               | [20], |
| long64-unsigned      | [21], |
| float32              | [23], |
| float64              | [24]  |

```
}
```

```
}
```

**5.8.2.3.7 set\_encryption\_key (data)**

Sets the encryption key in the M-Bus client and enables encrypted communication with the M-Bus slave device.

```
data ::= octet-string (encryption_key)
```

After the installation of the M-Bus slave, the M-Bus client holds an empty encryption key. With this, encryption of M-Bus frames is disabled.

Encryption can be disabled by invoking the set\_encryption\_key method with an octet-string of zero length.

Changing the encryption key requires two steps:

- first, the key is sent to the M-Bus slave, encrypted with the default key, using the transfer\_key method;
- second, the key is set in the M-Bus master using the set\_encryption\_key method.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 517/668 |
|-----------------------|------------|-----------------------------|---------|

#### 5.8.2.3.8 transfer\_key (data)

Transfers an encryption key to the M-Bus slave device.

data ::= octet-string (encrypted\_key)

Before encrypted M-Bus frames can be used, an operational encryption key has to be sent to the M-Bus slave, by invoking the transfer\_key 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.

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 set\_encryption\_key method with the new encryption key as a parameter.

With further invocations of the transfer\_key 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.

When an M-Bus slave is de-installed, the encryption key is destroyed

but the default key is not affected. Encryption remains disabled until a new encryption key is transferred.

### 5.9 Previous versions of interface classes for setting up data exchange over the internet

There are no previous versions to report.

### 5.10 Previous versions of interface classes for data exchange using S-FSK PLC

#### 5.10.1 S-FSK Phy&MAC setup (class\_id = 50, version = 0)

##### 5.10.1.1 Overview

NOTE 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.

## COSEM Interface Classes

| <b>S-FSK Phy&amp;MAC setup</b> |          | <b>0...n</b>     | <b>class_id = 50, version = 0</b> |             |             |                   |
|--------------------------------|----------|------------------|-----------------------------------|-------------|-------------|-------------------|
| <b>Attributes</b>              |          | <b>Data type</b> | <b>Min.</b>                       | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 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          |
| <b>Specific methods</b>        |          | <i>m/o</i>       |                                   |             |             |                   |

### 5.10.1.2 Attribute description

#### 5.10.1.2.1 logical\_name

Identifies the “S-FSK Phy&MAC setup” object instance. See 6.2.24.

#### 5.10.1.2.2 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 This enumeration is different from that of IEC 61334-4-512:2001.

#### 5.10.1.2.3 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:

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 519/668 |
|-----------------------|------------|-----------------------------|---------|

- (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.

#### **5.10.1.2.4 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 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.

#### **5.10.1.2.5 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.

#### **5.10.1.2.6 search\_initiator\_gain**

This attribute is used in the intelligent search initiator process. If the value of the max\_receiving\_gain is below the value of this attribute, a fast synchronization process is possible.

#### **5.10.1.2.7 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.

#### **5.10.1.2.8 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 1 MAC addresses are expressed on 12 bits.

Contains the value of the address of the physical attachment (MAC address) associated to the local system. In the unconfigured state, the MAC address is "NEW-address".

This attribute is locally written by the CIASE when the system is registered (with a Register service). The value is used in each outgoing or incoming frame. The default value is "NEW-address".

This attribute is set to NEW:

- by the MAC sub-layer, once the time-out-not-addressed delay is exceeded;
- when a client system "resets" the server system. See the "S-FSK Active initiator" IC in Clause 4.10.4.

When this attribute is set to NEW:

- the system loses its synchronization (function of the MAC-sublayer);
- the mac\_group\_address attribute is reset (array of 0 elements);
- the system automatically releases all AAs which can be released.

NOTE 2 The second item is not present in IEC 61334-4-512:2001.

The predefined MAC addresses are shown in Table 40..

#### **5.10.1.2.9 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).

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
|                       |            | 521/668                     |

**5.10.1.2.10 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 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 (*repeater\_status* = TRUE).

**5.10.1.2.11 repeater\_status**

Holds the current repeater status of the device.

boolean:

- |        |              |
|--------|--------------|
| FALSE: | no repeater, |
| TRUE:  | repeater     |

**5.10.1.2.12 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 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.

**5.10.1.2.13 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 5.10.1.2.14). See also IEC 61334-5-1:2001, 3.5.3, 4.1.6.3 and 4.1.7.2.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 522/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

If the value NO-BODY is written then the server mac\_address (see the *mac\_address* attribute) has to be set to NEW.

#### 5.10.1.2.14 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 *initiator\_mac\_address* 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 *initiator\_mac\_address* attribute which remains at the NO-BODY value. The default value of this variable shall be specified in the implementation specifications.

**NOTE** 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 *initiator\_mac\_address* attribute.

### 5.10.2 S-FSK IEC 61334-4-32 LLC setup (class\_id = 55, version = 0)

#### 5.10.2.1 Overview

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<br>(static)     | octet-string |                            |      |      | x          |
| 2. max_frame_length<br>(static) | unsigned     | 26                         | 242  | 134  | x + 0x08   |
| 3. reply_status_list<br>(dyn.)  | array        |                            |      |      | x + 0x10   |
| <b>Specific methods</b>         | <b>m/o</b>   |                            |      |      |            |

#### 5.10.2.2 Attribute description

##### 5.10.2.2.1 logical\_name

Identifies the “S-FSK IEC 61334-4-32 LLC setup” object instance. See 6.2.25.

##### 5.10.2.2.2 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 523/668 |
|-----------------------|------------|-----------------------------|---------|

### 5.10.2.2.3 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.

```
array      reply_status

reply_status ::= structure
{
    L-SAP-selector:          unsigned,
    length-of-waiting-L-SDU: unsigned
}
```

length-of-waiting-LSDU in the case of the S-FSK profile is in number of sub-frames; valid values are 1 to 7.

## 5.11 Previous versions of interface classes for setting up the LLC layer for ISO/IEC 8802-2

There are no previous versions to report.

## 5.12 Previous versions of interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for PRIME networks

There are no previous versions to report.

## 5.13 Previous versions of interface classes for setting up and managing DLMS/COSEM narrowband OFDM PLC profile for G3-PLC networks

### 5.13.1 Overview

In terms of IEEE 802.15.4: 2006 , a meter is 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 NB OFDM PLC setup classes concern only the RFD (Reduced Function Device) and not the PAN coordinator.

Classes and attributes are mapped to specific versions of the ITU-T G9903 standard as described in 5.13.2 and 5.13.3 below.

### 5.13.2 Mapping of G3-PLC IB attributes to COSEM IC attributes (ITU-T G9903:2013 Amd. 1 version)

Table 54 shows the mapping of G3 NB OFDM PLC PIB attributes to attributes of COSEM interface classes.

## COSEM Interface Classes

**Table 54 – Mapping of G3-PLC IB attributes specified in ITU-T G.9903:2013 Amd. 1 to COSEM IC attributes**

| Name  | Identifier                       | Interface class  | class_id / attribute |
|---|----------------------------------|--|----------------------|
| <b>MAC counters – Read only PIB attributes that provide statistic information <sup>1</sup></b>    |                                  |  |                      |
| mac_Tx_data_packet_count  | 0x02000101                       | PRIME NB OFDM PLC<br>Physical layer<br>parameters (class_id = 90, version = 0)<br>(5.13.4)   | 90 / 2               |
| mac_Rx_data_packet_count  | 0x02000102                       |  | 90 / 3               |
| mac_Tx_cmd_packet_count   | 0x02000201                       |  | 90 / 4               |
| mac_Rx_cmd_packet_count   | 0x02000202                       |  | 90 / 5               |
| mac_CSMA_fail_count   | 0x02000103                       |  | 90 / 6               |
| mac_CSMA_no_ACK_count   | 0x02000104                       |  | 90 / 7               |
| mac_bad_CRC_count   | 0x02000108                       |  | 90 / 8               |
| mac_broadcast_count   | 0x02000106                       |  | 90 / 9               |
| mac_multicast_count   | 0x02000107                       |  | 90 / 10              |
| <b>MAC setup PIB attributes – Read only &amp; read-write variables <sup>1</sup></b>               |                                  |  |                      |
| mac_short_address   | 0x01000112                       | G3 NB OFDM PLC MAC<br>setup (class_id = 91, version = 0)<br>(5.13.5)                         | 91 / 2               |
| mac_coord_short_address   | 0x01000107                       |  | 91 / 3               |
| mac_PAN_id  | 0x0100010F                       |  | 91 / 4               |
| mac_max_orphan_timer  | 0x02000109                       |  | 91 / 5               |
| mac_security_enabled  | 0x01000111                       |  | 91 / 6               |
| mac_freq_notching   | 0x00000006D                      |  | 91 / 7               |
| mac_TMR_TTL   | 0x02000113                       |  | 91 / 8               |
| mac_max_frame_retries   | 0x0100010D                       |  | 91 / 9               |
| mac_neighbour_table_entry_TTL   | 0x02000114                       |  | 91 / 10              |
| mac_neighbour_table   | 0x00000006B                      |  | 91 / 11              |
| <b>6LoWPAN adaptation layer IB attributes – Read only &amp; read-write variables <sup>2</sup></b> |                                  |  |                      |
| adp_max_hops  | 0x10                             | G3 NB OFDM PLC<br>6LoWPAN adaptation<br>layer setup (Class_id = 92, version = 0)<br>(5.13.8) | 92 / 2               |
| adp_weak_LQI_value  | 0x1B                             |  | 92 / 3               |
| adp_tone_mask   | 0x0F                             |  | 92 / 4               |
| adp_discovery_attempts_wait_time  | 0x06                             |  | 92 / 5               |
| adp_routing_configuration   | 0x12 – 0x19, 0x1A,<br>0x1C, 0x26 |  | 92 / 6               |
| adp_broadcast_log_table_entry_TTL   | 0x02                             |  | 92 / 7               |
| adp_routing_table   | 0x0C                             |  | 92 / 8               |
| adp_context_information_table   | 0x07                             |  | 92 / 9               |
| adp_blacklist_table   | 0x25                             |  | 92 / 10              |
| adp_broadcast_log_table   | 0x0B                             |  | 92 / 11              |
| adp_group_table   | 0x0E                             |  | 92 / 12              |
| adp_max_join_wait_time  | 0x21                             |  | 92 / 13              |
| adp_path_discovery_time   | 0x22                             |  | 92 / 14              |
| adp_use_new_GMK_time  | 0x23                             |  | 92 / 15              |
| adp_exp_prec_GMK_time   | 0x24                             |  | 92 / 16              |

<sup>1)</sup> See ITU-T G.9903 Amd. 1:2013 clauses 9.3.6.2.2 and 9.3.6.2.3

<sup>2)</sup> See ITU-T G.9903 Amd. 1:2013 clause 9.4.1.1

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 525/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| Name  | Identifier | Interface class | class_id / attribute |
|---|------------|-----------------|----------------------|
| NOTE Whereas in ITU-T G.9903 Amd. 1:2013 the camel notation is used, in COSEM interface class specifications – and in this table – the underscore notation is used. |            |                 |                      |

### 5.13.3 Mapping of G3-PLC IB attributes to COSEM IC attributes (ITU-T G9903:2017 version)

Table 55 shows the mapping of G3-PLC PIB attributes to attributes of COSEM interface classes.

**Table 55 – Mapping of G3-PLC IB attributes specified in ITU-T G.9903:2017 to COSEM IC attributes**

| Name   | Identifier | Interface class   | class_id / attribute |
|--|------------|---|----------------------|
| <b>MAC counters – Read only PIB attributes that provide statistic information<sup>1</sup></b>        |            |   |                      |
| mac_Tx_data_packet_count   | 0x0101     | G3-PLC MAC layer counters<br>(class_id = 90, version = 1)<br>(4.13.3) | 90 / 2               |
| mac_Rx_data_packet_count   | 0x0102     |   | 90 / 3               |
| mac_Tx_cmd_packet_count  | 0x0103     |   | 90 / 4               |
| mac_Rx_cmd_packet_count  | 0x0104     |   | 90 / 5               |
| mac_CSMA_fail_count  | 0x0105     |   | 90 / 6               |
| mac_CSMA_no_ACK_count  | 0x0106     |   | 90 / 7               |
| mac_bad_CRC_count  | 0x0109     |   | 90 / 8               |
| mac_Tx_data_broadcast_count  | 0x0108     |   | 90 / 9               |
| mac_Rx_data_broadcast_count  | 0x0107     |   | 90 / 10              |
| <b>MAC setup PIB attributes – Read only &amp; read-write &amp; write only variables<sup>12</sup></b> |            |   |                      |
| mac_short_address  | 0x0053     | G3-PLC MAC setup<br>(class_id = 91, version = 2)<br>(5.13.7)          | 91 / 2               |
| mac_RC_coord   | 0x010F     |   | 91 / 3               |
| mac_PAN_id   | 0x0050     |   | 91 / 4               |
| mac_key_table  | 0x0071     |   | 91 / 5               |
| mac_frame_counter  | 0x0077     |   | 91 / 6               |
| mac_tone_mask  | 0x0110     |   | 91 / 7               |
| mac_TMR_TTL  | 0x010D     |   | 91 / 8               |
| mac_max_frame_retries  | 0x0059     |   | 91 / 9               |
| mac_POS_table_entry_TTL  | 0x010E     |   | 91 / 10              |
| mac_neighbour_table  | 0x010A     |   | 91 / 11              |
| mac_high_priority_window_size  | 0x0100     |   | 91 / 12              |
| mac_CSMA_fairness_limit  | 0x010C     |   | 91 / 13              |
| mac_beacon_randomization_window_length   | 0x0111     |   | 91 / 14              |
| mac_A  | 0x0112     |   | 91 / 15              |
| mac_K  | 0x0113     |   | 91 / 16              |
| mac_min_CW_attempts  | 0x0114     |   | 91 / 17              |
| mac_cenelec_legacy_mode  | 0x0115     |   | 91 / 18              |
| mac_FCC_legacy_mode  | 0x0116     |   | 91 / 19              |

## COSEM Interface Classes

| Name  | Identifier                                    | Interface class   | class_id / attribute |
|---|---|---|----------------------|
| mac_max_BE  | 0x0047  |   | 91 / 20              |
| mac_max_CSMA_backoffs   | 0x004E  |   | 91 / 21              |
| mac_min_BE  | 0x004F  |   | 91 / 22              |
| mac_broadcast_max_CW_enabled  | 0x011E  |   | 91 / 23              |
| mac_transmit_atten  | 0x011F  |   | 91 / 24              |
| mac_POS_table   | 0x0120  |   | 91 / 25              |
| <b>6LoWPAN adaptation layer IB attributes – Read only &amp; read-write variables<sup>3 4</sup></b>  |   |   |                      |
| adp_max_hops  | 0x0F  | G3-PLC 6LoWPAN adaptation layer setup (class_id = 92, version = 2)<br>(5.13.10) | 92 / 2               |
| adp_weak_LQI_value  | 0x1A  |   | 92 / 3               |
| adp_security_level  | 0x00  |   | 92 / 4               |
| adp_prefix_table  | 0x01  |   | 92 / 5               |
| adp_routing_configuration   | 0x09, 0x0A, 0x0D,<br>0x11-0x19, 0x1B,<br>0x1F |   | 92 / 6               |
| adp_broadcast_log_table_entry_TTL   | 0x02  |   | 92 / 7               |
| adp_routing_table   | 0x0C  |   | 92 / 8               |
| adp_context_information_table   | 0x07  |   | 92 / 9               |
| adp_blacklist_table   | 0x1E  |   | 92 / 10              |
| adp_broadcast_log_table   | 0x0B  |   | 92 / 11              |
| adp_group_table   | 0x0E  |   | 92 / 12              |
| adp_max_join_wait_time  | 0x20  |   | 92 / 13              |
| adp_path_discovery_time   | 0x21  |   | 92 / 14              |
| adp_active_key_index  | 0x22  |   | 92 / 15              |
| adp_metric_type   | 0x03  |   | 92 / 16              |
| adp_coord_short_address   | 0x08  |   | 92 / 17              |
| adp_disable_default_routing   | 0xF0  |   | 92 / 18              |
| adp_device_type   | 0x10  |   | 92 / 19              |
| adp_default_coord_route_enabled   | 0x24  |   | 92 / 20              |
| adp_destination_address_set   | 0x23  |   | 92 / 21              |
| <sup>1</sup> See ITU-T G.9903:2017, 9.3.6.2.2 and 9.3.6.2.3.  |   |   |                      |
| <sup>2</sup> The following attributes of the G3-PLC MAC sublayer IB attributes have been excluded as there is no need to expose them: macBSN, macDSN, macAckWaitDuration, macFreqNotching, macTimeStampSupported, macPromiscuousMode, macSecurityEnabled. |   |   |                      |
| <sup>3</sup> See ITU-T G.9903:2017, 9.4.1.1.  |   |   |                      |
| <sup>4</sup> The following attributes of the G3-PLC Adaptation sublayer IB attributes have been excluded as there is no need to expose them ; adpSoftVersion, adpSnifferMode.   |   |   |                      |
| NOTE Whereas in ITU-T G.9903 the camel-case notation is used, in COSEM interface class specifications – and in this table – the underscore notation is used.  |   |   |                      |

### 5.13.4 G3 NB OFDM PLC MAC layer counters (class\_id = 90, version = 0)

#### 5.13.4.1 Overview

An instance of the “G3 NB OFDM PLC MAC layer counters” IC stores counters related to the MAC layer exchanges. The objective of these counters is to provide statistical information for management purposes.

The attributes of instances of this IC shall be read only. They can be reset using the reset method.

| G3 NB OFDM PLC MAC layer counters  | 0...n                | class_id = 90, version = 0 |               |      |            |
|------------------------------------|----------------------|----------------------------|---------------|------|------------|
| Attributes                         | Data type            | Min.                       | Max.          | Def. | Short name |
| 1. logical_name (static)           | octet-string         |                            |               |      | x          |
| 2. mac_Tx_data_packet_count (dyn.) | double-long-unsigned | 0                          | 4 294 967 295 | 0    | x + 0x08   |
| 3. mac_Rx_data_packet_count (dyn.) | double-long-unsigned | 0                          | 4 294 967 295 | 0    | x + 0x10   |
| 4. mac_Tx_cmd_packet_count (dyn.)  | double-long-unsigned | 0                          | 4 294 967 295 | 0    | x + 0x18   |
| 5. mac_Rx_cmd_packet_count (dyn.)  | double-long-unsigned | 0                          | 4 294 967 295 | 0    | x + 0x20   |
| 6. mac_CSMA_fail_count (dyn.)      | double-long-unsigned | 0                          | 4 294 967 295 | 0    | x + 0x28   |
| 7. mac_CSMA_no_ACK_count (dyn.)    | double-long-unsigned | 0                          | 4 294 967 295 | 0    | x + 0x30   |
| 8. mac_bad_CRC_count (dyn.)        | double-long-unsigned | 0                          | 4 294 967 295 | 0    | x + 0x38   |
| 9. mac_broadcast_count (dyn.)      | double-long-unsigned | 0                          | 4 294 967 295 | 0    | x + 0x40   |
| 10. mac_multicast_count (dyn.)     | double-long-unsigned | 0                          | 4 294 967 295 | 0    | x + 0x48   |
| Specific methods                   | m/o                  |                            |               |      |            |
| 1. reset (data)                    | o                    |                            |               |      | x + 0x50   |

#### 5.13.4.2 Attribute description

##### 5.13.4.2.1 logical\_name

Identifies the “G3 NB OFDM PLC MAC layer counters” object instance. See 6.2.28.

##### 5.13.4.2.2 mac\_Tx\_data\_packet\_count

PIB attribute 0x02000101: Statistic counter of successfully transmitted data packets (MSDUs).

##### 5.13.4.2.3 mac\_Rx\_data\_packet\_count

PIB attribute 0x02000102: Statistic counter of successfully received data packets (MSDUs).

##### 5.13.4.2.4 mac\_Tx\_cmd\_packet\_count

PIB attribute 0x02000201: Statistic counter of successfully transmitted command packets.

##### 5.13.4.2.5 mac\_Rx\_cmd\_packet\_count

PIB attribute 0x02000202: Statistic counter of successfully received command packets.

**5.13.4.2.6 mac\_CSMA\_fail\_count**

PIB attribute 0x02000103: Counts the number of times when CSMA backoffs exceed macMaxCSMABackoffs.

**5.13.4.2.7 mac\_CSMA\_no\_ACK\_count**

PIB attribute 0x02000104: 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).

**5.13.4.2.8 mac\_bad\_CRC\_count**

PIB attribute 0x02000108: Statistic counter of the number of frames received with bad CRC.

**5.13.4.2.9 mac\_broadcast\_count**

PIB attribute 0x02000106: Statistic counter of the number of broadcast frames sent.

**5.13.4.2.10 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 is automatically rolled-over.

**5.13.4.3 Method description****5.13.4.3.1 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)

**5.13.5 G3 NB OFDM PLC MAC setup (class\_id = 91, version = 0)****5.13.5.1 Overview**

An instance of the “G3 NB OFDM PLC MAC setup” IC holds the necessary parameters to set up and manage the G3 NB OFDM PLC IEEE 802.15.4: 2006 MAC sub-layer.

These attributes influence the functional behaviour of an implementation. Implementations may allow changes to the attributes during normal running, i.e. even after the device start-up sequence has been executed.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 529/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| 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                |                            |               |        |            |
| <b>Specific methods</b>                 |          | <b>m/o</b>           |                            |               |        |            |
| 1. mac_get_neighbour_table_entry (data) |          | o                    |                            |               |        |            |

### 5.13.5.2 Attribute description

#### 5.13.5.2.1 logical\_name

Identifies the “G3 NB OFDM PLC MAC setup” object instance. See 6.2.28.

#### 5.13.5.2.2 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.

#### 5.13.5.2.3 mac\_coord\_short\_address

PIB attribute 0x01000107: Coordinator short address.

NOTE The short address assigned to the coordinator through which the device is associated.

#### 5.13.5.2.4 mac\_PAN\_id

PIB attribute 0x0100010F: PAN ID.

NOTE 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.

**5.13.5.2.5 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.

**5.13.5.2.6 mac\_security\_enabled**

PIB attribute 0x01000111: Security enabled.

```
boolean:  
    TRUE: security enabled,  
    FALSE: security disabled
```

**5.13.5.2.7 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).

**5.13.5.2.8 mac\_TMR\_TTL**

PIB attribute 0x02000113: Maximum valid time of tone map parameters in the neighbour table in seconds.

**5.13.5.2.9 mac\_max\_frame\_retries**

PIB attribute 0x0100010D: Maximum number of retransmissions.

**5.13.5.2.10 mac\_neighbour\_table\_entry\_TTL**

PIB attribute 0x02000114: Maximum valid time for an entry in the neighbour table in seconds.

**5.13.5.2.11 mac\_neighbour\_table**

PIB attribute 0x0000006B: See ITU-T G.9903 Amd. 1:2013, 9.3.7.2 for CENELEC and FCC bands.

The neighbour table contains information about all the devices within the POS of the device. One element of the table represents one PLC direct neighbour of the device.

```
array      mac_neighbour_table_element  
  
mac_neighbour_table_element ::= structure  
{  
    mac_short_address:          long-unsigned,  
    payload_modulation_scheme: boolean,  
    tone_map:                  bit-string,  
    modulation:                enum,  
    tx_gain:                   integer,  
    tx_res:                    boolean,  
    tx_coeff:                 array,  
    lqi:                      unsigned,  
    TMR_valid_time:            double-long-unsigned,  
    neighbour_valid_time:      double-long-unsigned  
}
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 531/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

NOTE This table is actualized each time any frame is received from a neighbour device, and each time a Tone Map Response is received.

|                           |  |
|---------------------------|--|
| mac_short_address         | The MAC Short Address of the node which this entry refers to.  |
| payload_modulation_scheme | 0: Differential,<br>1: Coherent  |
| tone_map                  | The Tone Map parameter defines which frequency sub-band can be used for communication with the device. A bit set to 1 means that the frequency sub-band can be used, and a bit set to 0 means that frequency sub-band shall not be used. |
| modulation                | <p>The modulation type to use for communicating with the device.</p> <p>enum:</p> <ul style="list-style-type: none"><li>(0) Robust Mode</li><li>(1) DBPSK</li><li>(2) DQPSK</li><li>(3) D8PSK</li><li>(4) 16-QAM</li></ul>               |

NOTE 4 The 16-QAM modulation is optional and only applicable for FCC band.

## COSEM Interface Classes

|                                   |   |
|-----------------------------------|---|
| <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.<br>0: 6 dB<br>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.<br><br>NOTE 5 One group of tones gathers 6 consecutive tones (or carriers) for CENELEC bands, and 3 consecutive tones for FCC band. |
| <code>lqi</code>                  | Link quality indicator<br><br>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.   |
| <code>TMR_valid_time</code>       | Remaining time in seconds until which the tone map response parameters in the neighbour table are considered valid. <ul style="list-style-type: none"><li>– When the entry is created, this value shall be set to the default value 0.</li><li>– 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.</li></ul>  |
| <code>neighbour_valid_time</code> | Remaining time in seconds until which this entry in the neighbour table is considered valid.<br><br>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.   |

### 5.13.5.3 Method description

#### 5.13.5.3.1 `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`

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 533/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

where `mac_neighbour_table-element` is as defined in the `mac_neighbour_table` attribute of the present IC.

### 5.13.6 G3-PLC MAC setup (`class_id = 91, version = 1`)

#### 5.13.6.1 Overview

An instance of the “G3-PLC MAC setup” IC holds the necessary parameters to set up and manage the G3-PLC IEEE 802.15.4: 2006 MAC sub-layer.

These attributes influence the functional behaviour of an implementation. Implementations may allow changes to the attributes during normal running, i.e. even after the device start-up sequence has been executed.

| G3-PLC MAC setup                           |          | 0...n                | <code>class_id = 91, version = 1</code> |                  |                                  |            |
|--|----------|----------------------|---|------------------|----------------------------------|------------|
| 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<br>967 295 | 0                                | x + 0x28   |
| 7. mac_tone_mask                           | (static) | bit-string           |   |                  | 0x00000<br>0000FF<br>FFFFFF<br>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   |
| <b>Specific methods</b>                    |          | <b>m/o</b>           |   |                  |                                  |            |
| 1. mac_get_neighbour_table_entry (data)    |          | o                    |   |                  |                                  | x + 0xB0   |

**5.13.6.2 Attribute description****5.13.6.2.1 logical\_name**

Identifies the “G3-PLC MAC setup” object instance. See 6.2.28.

**5.13.6.2.2 mac\_short\_address**

PIB attribute 0x0053: The 16-bit address the device is using to communicate through the PAN. Its value shall be equal to 0xFFFF when the device does not have a short address. An associated device necessarily has a short address, so that a device cannot be in the state where it is associated but does not have a short address.

**5.13.6.2.3 mac\_RC\_coord**

PIB attribute 0x010F: Route cost to coordinator, to be used in the beacon payload as RC\_COORD

**5.13.6.2.4 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.

**5.13.6.2.5 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
}
```

Where:

- 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.

**5.13.6.2.6 mac\_frame\_counter**

PIB attribute 0x0077: The outgoing frame counter for this device, used when ciphering frames at MAC layer.

**5.13.6.2.7 mac\_tone\_mask**

PIB attribute 0x0110: Defines the tone mask to use during symbol formation.

**5.13.6.2.8 mac\_TMR\_TTL**

PIB attribute 0x010D: Maximum time to live of tone map parameters entry in the neighbour table in minutes.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 535/668 |
|-----------------------|------------|-----------------------------|---------|

**5.13.6.2.9 mac\_max\_frame\_retries**

PIB attribute 0x0059: Maximum number of retransmissions.

**5.13.6.2.10 mac\_neighbour\_table\_entry\_TTL**

PIB attribute 0x010E: Maximum time to live for an entry in the neighbour table in minutes.

**5.13.6.2.11 mac\_neighbour\_table**

PIB attribute 0x010A: See ITU-T G.9903:2014 9.3.7.2 for CENELEC and FCC bands.

The neighbour table contains information about all the devices within the POS of the device. One element of the table represents one PLC direct neighbour of the device.

```
array neighbour_table

neighbour_table ::= structure

{
    short_address: long-unsigned,
    payload_modulation_scheme: boolean,
    tone_map: bit-string,
    modulation: enum,
    tx_gain: integer,
    tx_res: enum,
    tx_coeff: bit-string,
    lqi: unsigned,
    phase_differential: integer,
    TMR_valid_time: unsigned,
    neighbour_valid_time: unsigned
}
```

NOTE 1 This table is actualized each time any frame is received from a neighbour device, and each time a Tone Map Response is received.

**short\_address** The MAC Short Address of the node which this entry refers to.

**payload\_modulation\_scheme** Payload Modulation scheme to be used when transmitting to this neighbour.  
FALSE: Differential,  
TRUE: Coherent

**tone\_map** The Tone Map parameter defines which frequency sub-band can be used for communication with the device. A bit set to 1 means that the frequency sub-band can be used, and a bit set to 0 means that frequency sub-band shall not be used.

**modulation** The modulation type to use for communicating with the device.

```
enum:
    (0) Robust Mode,
    (1) DBPSK,
    (2) DQPSK,
    (3) D8PSK,
```

## COSEM Interface Classes

### (4) 16-QAM

NOTE 2 The 16-QAM modulation is optional and only applicable for FCC band.

**tx\_gain** Defines the Tx Gain to use to transmit frames to that device.

**tx\_res** Defines the Tx Gain resolution corresponding to one gain step.  
 0: 6 dB,  
 1: 3 dB

**tx\_coeff** A parameter that specifies transmitter gain for each group of tones represented by one valid bit of the tone map. The receiver measures the frequency-dependent attenuation of the channel and may request the transmitter to compensate for this attenuation by increasing the transmit power on sections of the spectrum that are experiencing attenuation in order to equalize the received signal. Each group of tones is mapped to a 4-bit value for CENELEC-A or a 2-bit value for FCC where a "0" in the most significant bit indicates a positive gain value, hence an increase in the transmitter gain scaled by TXRES is requested for that section and a "1" indicates a negative gain value, hence a decrease in the transmitter gain scaled by TXRES is requested for that section. Implementing this feature is optional and it is intended for frequency selective channels. If this feature is not implemented, the value zero shall be used.

*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'*

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

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
|                       |            | 537/668                     |

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.

#### **5.13.6.2.12 mac\_high\_priority\_window\_size**

PIB attribute 0x0100: The high priority contention window size in number of slots.

#### **5.13.6.2.13 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.

#### **5.13.6.2.14 mac\_beacon\_randomization\_window\_length**

PIB attribute 0x0111: Duration time in seconds for the beacon randomization.

#### **5.13.6.2.15 mac\_A**

PIB attribute 0x0112: This parameter controls the adaptive CW linear decrease.

#### **5.13.6.2.16 mac\_K**

PIB attribute 0x0113: Rate adaptation factor for channel access fairness limit. This attribute can take a value between 1 and macCSMAFairnessLimit.

#### **5.13.6.2.17 mac\_min\_CW\_attempts**

PIB attribute 0x0114: Number of consecutive attempts while using minimum CW.

#### **5.13.6.2.18 mac\_cenelec\_legacy\_mode**

PIB attribute 0x0115: This read only attribute indicates the capability of the node.

0: The following configuration is used (legacy mode):

- Elementary interleaving;
- Interleaver parameters  $n_i$  and  $n_j$  are not swapped when  $I(i,j) = 0$ .

1: The following configuration is used (non legacy mode):

- Full Block interleaving;
- Interleaver parameters  $n_i$  and  $n_j$  are swapped when  $I(i,j) = 0$ .

#### **5.13.6.2.19 mac\_FCC\_legacy\_mode**

PIB attribute 0x0116: This read only attribute indicates the capability of the node.

0: The following configuration is used (legacy mode):

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 538/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

- Differential FCH modulation;
- Elementary interleaving;
- Interleaver parameters  $n_i$  and  $n_j$  are not swapped when  $I(i,j) = 0$ ;
- Single RS block.

1: The following configuration is used (non legacy mode):

- Coherent FCH modulation;
- Full Block interleaving;
- Interleaver parameters  $n_i$  and  $n_j$  are swapped when  $I(i,j) = 0$ ;
- Two RS blocks.

#### **5.13.6.2.20 mac\_max\_BE**

PIB attribute 0x0047: Maximum value of backoff exponent. It should always be greater than macMinBE.

#### **5.13.6.2.21 mac\_max\_CSMA\_backoffs**

PIB attribute 0x004E: Maximum number of backoff attempts.

#### **5.13.6.2.22 mac\_min\_BE**

PIB attribute 0x004F: Minimum value of backoff exponent.

### **5.13.6.3 Method description**

#### **5.13.6.3.1 mac\_get\_neighbour\_table\_entry (data)**

This method is used to retrieve the mac neighbour table for one MAC short address. It may be used to perform topology monitoring by the client.

The method invocation parameter contains a mac\_short\_address.

data ::= long-unsigned

The response parameter includes the neighbour table for this mac\_short\_address.

data ::= array neighbour\_table

where mac\_neighbour\_table is as defined in the *mac\_neighbour\_table* attribute of the present IC.

### **5.13.7 G3-PLC MAC setup (class\_id = 91, version = 2)**

#### **5.13.7.1 Overview**

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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 539/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

| G3-PLC MAC setup                               |          | 0...n                | class_id = 91, version = 2 |                  |  |            |
|--|----------|----------------------|----------------------------|------------------|--|------------|
| 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                |                            |                  | Empty  | x + 0x20   |
| 6. mac_frame_counter                           | (dyn.)   | double-long-unsigned | 0                          | 4 294<br>967 295 | 0  | x + 0x28   |
| 7. mac_tone_mask                               | (static) | bit-string           |                            |                  | 'FFFFFFFFF0<br>00000000'H<br><br>for<br>CENELEC-A<br>bandplan;<br><br>'FFFFFFFFFF<br>FFFFFFF'H<br><br>for<br>FCC<br>bandplan | x + 0x30   |
| 8. mac_TMR_TTL                                 | (static) | unsigned             | 0                          | 255              | 10   | x + 0x38   |
| 9. mac_max_frame_retries                       | (static) | unsigned             | 0                          | 10               | 5  | x + 0x40   |
| 10. mac_POS_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<br>below               | 255              | 25   | x + 0x60   |
| 14. mac_beacon_randomization<br>_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<br>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_backoff                       | (static) | unsigned             | 0                          | 255              | 50   | x + 0xA0   |
| 22. mac_min_BE                                 | (static) | unsigned             | 0                          | 20               | 3  | x + 0xA8   |
| 23. mac_broadcast_max_CW<br>_enabled           | (static) | boolean              |                            |                  | FALSE  | x + 0xB0   |
| 24. mac_transmit_atten                         | (static) | unsigned             | 0                          | 25               | 0  | x + 0xB8   |
| 25. mac_POS_table                              | (dyn.)   | array                |                            |                  |  | x + 0xC0   |
| <b>Specific methods</b>                        |          | m/o                  |                            |                  |  |            |
| 1. mac_get_neighbour_table<br>_entry(data)     | O        |                      |                            |                  |  |            |

### 5.13.7.2 Attribute description

#### 5.13.7.2.1 logical\_name

Identifies the “G3-PLC MAC setup” object instance. See 6.2.28.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 540/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

**5.13.7.2.2 mac\_short\_address**

PIB attribute 0x0053: The 16-bit address the device is using to communicate through the PAN. Its value shall be equal to 0xFFFF when the device does not have a short address. An associated device necessarily has a short address, so that a device cannot be in the state where it is associated but does not have a short address.

**5.13.7.2.3 mac\_RC\_coord**

PIB attribute 0x010F: Route cost to coordinator, to be used in the beacon payload as RC\_COORD.

**5.13.7.2.4 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.

**5.13.7.2.5 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
}
```

Where:

- 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.

**5.13.7.2.6 mac\_frame\_counter**

PIB attribute 0x0077: The outgoing frame counter for this device, used when ciphering frames at MAC layer.

**5.13.7.2.7 mac\_tone\_mask**

PIB attribute 0x0110: Defines the tone mask to use during symbol formation. Each bit controls if a given carrier is used (value = 1) or notched (value = 0).

The bit-string is always 72 bits long, the number of meaningful bits depending on the device band-plan. The first bit in the bit-string (bit 0) bit represents the lowest frequency of the CENELEC -A or FCC band-plans.

The mapping in the CENELEC-A bandplan is given in the following table:

| <b>Bit number</b> | <b>Carrier number</b> | <b>Carrier Frequency (kHz)</b> |
|-------------------|-----------------------|--------------------------------|
| bit 0 (first bit) | 1                     | 35,937,5                       |
| bit 1             | 2                     | 37,5                           |

## COSEM Interface Classes

|                          |          |          |
|--------------------------|----------|----------|
| bit 2                    | 3        | 39,062 5 |
| bit 3                    | 4        | 40,625   |
| bit 4                    | 5        | 42,187 5 |
| bit 5                    | 6        | 43,75    |
| bit 6                    | 7        | 45,312 5 |
| bit 7                    | 8        | 46,875   |
| bit 8                    | 9        | 48,437 5 |
| bit 9                    | 10       | 50       |
| bit 10                   | 11       | 51,562 5 |
| bit 11                   | 12       | 53,125   |
| bit 12                   | 13       | 54,687 5 |
| bit 13                   | 14       | 56,25    |
| bit 14                   | 15       | 57,812 5 |
| bit 15                   | 16       | 59,375   |
| bit 16                   | 17       | 60,937 5 |
| bit 17                   | 18       | 62,5     |
| bit 18                   | 19       | 64,062 5 |
| bit 19                   | 20       | 65,625   |
| bit 20                   | 21       | 67,187 5 |
| bit 21                   | 22       | 68,75    |
| bit 22                   | 23       | 70,312 5 |
| bit 23                   | 24       | 71,875   |
| bit 24                   | 25       | 73,437 5 |
| bit 25                   | 26       | 75       |
| bit 26                   | 27       | 76,562 5 |
| bit 27                   | 28       | 78,125   |
| bit 28                   | 29       | 79,687 5 |
| bit 29                   | 30       | 81,25    |
| bit 30                   | 31       | 82,812 5 |
| bit 31                   | 32       | 84,375   |
| bit 32                   | 33       | 85,937 5 |
| bit 33                   | 34       | 87,5     |
| bit 34                   | 35       | 89,062 5 |
| bit 35                   | 36       | 90,625   |
| bits 36 to 71 (last bit) | not used | not used |

The mapping in the FCC bandplan is given in the following table:

| Bit number        | Carrier number | Carrier Frequency (kHz) |
|-------------------|----------------|-------------------------|
| bit 0 (first bit) | 1              | 154,687 5               |
| bit 1             | 2              | 159,375                 |
| bit 2             | 3              | 164,062 5               |

## COSEM Interface Classes

| Bit number | Carrier number | Carrier Frequency (kHz) |
|------------|----------------|-------------------------|
| bit 3      | 4              | 168,75                  |
| bit 4      | 5              | 173,437 5               |
| bit 5      | 6              | 178,125                 |
| bit 6      | 7              | 182,812 5               |
| bit 7      | 8              | 187,5                   |
| bit 8      | 9              | 192,187 5               |
| bit 9      | 10             | 196,875                 |
| bit 10     | 11             | 201,562 5               |
| bit 11     | 12             | 206,25                  |
| bit 12     | 13             | 210,937 5               |
| bit 13     | 14             | 215,625                 |
| bit 14     | 15             | 220,312 5               |
| bit 15     | 16             | 225                     |
| bit 16     | 17             | 229,687 5               |
| bit 17     | 18             | 234,375                 |
| bit 18     | 19             | 239,062 5               |
| bit 19     | 20             | 243,75                  |
| bit 20     | 21             | 248,437 5               |
| bit 21     | 22             | 253,125                 |
| bit 22     | 23             | 257,812 5               |
| bit 23     | 24             | 262,5                   |
| bit 24     | 25             | 267,187 5               |
| bit 25     | 26             | 271,875                 |
| bit 26     | 27             | 276,562 5               |
| bit 27     | 28             | 281,25                  |
| bit 28     | 29             | 285,937 5               |
| bit 29     | 30             | 290,625                 |
| bit 30     | 31             | 295,312 5               |
| bit 31     | 32             | 300                     |
| bit 32     | 33             | 304,687 5               |
| bit 33     | 34             | 309,375                 |
| bit 34     | 35             | 314,062 5               |
| bit 35     | 36             | 318,75                  |
| bit 36     | 37             | 323,437 5               |
| bit 37     | 38             | 328,125                 |
| bit 38     | 39             | 332,812 5               |
| bit 39     | 40             | 337,5                   |
| bit 40     | 41             | 342,187 5               |
| bit 41     | 42             | 346,875                 |
| bit 42     | 43             | 351,562 5               |
| bit 43     | 44             | 356,25                  |

## COSEM Interface Classes

| Bit number        | Carrier number | Carrier Frequency (kHz) |
|-------------------|----------------|-------------------------|
| bit 44            | 45             | 360,937.5               |
| bit 45            | 46             | 365,625                 |
| bit 46            | 47             | 370,312.5               |
| bit 47            | 48             | 375                     |
| bit 48            | 49             | 379,687.5               |
| bit 49            | 50             | 384,375                 |
| bit 50            | 51             | 389,062.5               |
| bit 51            | 52             | 393,75                  |
| bit 52            | 53             | 398,437.5               |
| bit 53            | 54             | 403,125                 |
| bit 54            | 55             | 407,812.5               |
| bit 55            | 56             | 412,5                   |
| bit 56            | 57             | 417,187.5               |
| bit 57            | 58             | 421,875                 |
| bit 58            | 59             | 426,562.5               |
| bit 59            | 60             | 431,25                  |
| bit 60            | 61             | 435,937.5               |
| bit 61            | 62             | 440,625                 |
| bit 62            | 63             | 445,312.5               |
| bit 63            | 64             | 450                     |
| bit 64            | 65             | 454,687.5               |
| bit 65            | 66             | 459,375                 |
| bit 66            | 67             | 464,062.5               |
| bit 67            | 68             | 468,75                  |
| bit 68            | 69             | 473,437.5               |
| bit 69            | 70             | 478,125                 |
| bit 70            | 71             | 482,812.5               |
| bit 71 (last bit) | 72             | 487,5                   |

### 5.13.7.2.8 mac\_TMR\_TTL

PIB attribute 0x010D: Maximum time to live for an entry in the neighbour table in minutes.

NOTE: This attribute was used with different definitions in earlier versions of the Interface Class. See 5.13.5 and 5.13.6.

### 5.13.7.2.9 mac\_max\_frame\_retries

PIB attribute 0x0059: Maximum number of retransmissions.

### 5.13.7.2.10 mac\_POS\_table\_entry\_TTL

PIB attribute 0x010E: Maximum time to live for an entry in the POS table in minutes

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 544/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

NOTE: This attribute was used with different name and different definitions in earlier versions of the Interface Class. See 5.13.3 and 5.13.6.

### 5.13.7.2.11 mac\_neighbour\_table

PIB attribute 0x010A: See ITU-T G.9903:2017, 9.3.7.2.2 for CENELEC and FCC bands.

Every device shall maintain a "neighbour table" which contains information about all the devices with which a unicast message has been exchanged. Each time a tone map response command is received the table is updated. In case the table is full, the entry corresponding to the shortest valid time is removed. This table shall be accessible by the adaptation, MAC sublayers and physical layer.

```
array neighbour_table
neighbour_table ::= structure
{
    short_address: long-unsigned,
    payload_modulation_scheme: boolean,
    tone_map: bit-string,
    modulation_type: enum,
    tx_gain: integer,
    tx_res: boolean,
    tx_coeff: bit-string,
    reverse_lqi: unsigned,
    phase_differential: integer,
    TMR_valid_time: unsigned,
    no_data: unsigned
}
```

Where:

- short\_address The MAC Short Address of the node which this entry refers to.
- payload\_modulation\_scheme Payload Modulation scheme to be used when transmitting to this neighbour.
  - 0: Differential,
  - 1: Coherent
- tone\_map The tone map to be used when transmitting to this neighbour.

NOTE 1 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 is not to be used.

The first bit in the bit-string (bit 0) corresponds to Tone Map bit TM[0].

In the CENELEC-A bandplan, each bit of the tone map field is associated with one sub-band consisting of a group of 6 tones.

| Bit number | TM field | Sub-band (kHz)     |
|------------|----------|--------------------|
| 0          | TM[0]    | 35,937.5 to 43,75  |
| 1          | TM[1]    | 45,312.5 to 53,125 |
| 2          | TM[2]    | 54,687.5 to 62,5   |

## COSEM Interface Classes

| Bit number | TM field | Sub-band (kHz)     |
|------------|----------|--------------------|
| 3          | TM[3]    | 64,062.5 to 71,875 |
| 4          | TM[4]    | 73,437.5 to 81,25  |
| 5          | TM[5]    | 82,812.5 to 90,625 |

In the FCC bandplan, each bit of the tone map field is associated with one sub-band consisting of a group of 3 tones.

| Bit number | TM field | Sub-band (kHz)         |
|------------|----------|------------------------|
| 0          | TM[0]    | 154,687.5 to 164,062.5 |
| 1          | TM[1]    | 168,75 to 178,125      |
| 2          | TM[2]    | 182,812.5 to 192,187.5 |
| 3          | TM[3]    | 196,875 to 206,25      |
| 4          | TM[4]    | 210,937.5 to 220,312.5 |
| 5          | TM[5]    | 225 to 234,375         |
| 6          | TM[6]    | 239,062.5 to 248,437.5 |
| 7          | TM[7]    | 253,125 to 262,5       |
| 8          | TM[8]    | 267,187.5 to 276,562.5 |
| 9          | TM[9]    | 281,25 to 290,625      |
| 10         | TM[10]   | 295,312.5 to 304,687.5 |
| 11         | TM[11]   | 309,375 to 318,75      |
| 12         | TM[12]   | 323,437.5 to 332,812.5 |
| 13         | TM[13]   | 337.5 to 346,875       |
| 14         | TM[14]   | 351,562.5 to 360,937.5 |
| 15         | TM[15]   | 365,625 to 375         |
| 16         | TM[16]   | 379,687.5 to 389,062.5 |
| 17         | TM[17]   | 393,75 to 403,125      |
| 18         | TM[18]   | 407,812.5 to 417,187.5 |
| 19         | TM[19]   | 421,875 to 431,25      |
| 20         | TM[20]   | 435,937.5 to 445,312.5 |
| 21         | TM[21]   | 450 to 459,375         |
| 22         | TM[22]   | 464,062.5 to 473,437.5 |
| 23         | TM[23]   | 478,125 to 487,5       |

– modulation\_type                          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.

## COSEM Interface Classes

|           |  |
|-----------|--|
| - tx_gain | Tx Gain to be used when transmitting to this neighbour.        |
| - tx_res  | Defines the Tx Gain resolution corresponding to one gain step. |
|           | 0 : 6 dB<br>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.

NOTE 3 One group of tones gathers 6 consecutive tones (or carriers) for CENELEC bands, and 3 consecutive tones for FCC band.

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 last 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.

In the CENELEC bandplan, the first bit in the bit-string (bit 0) corresponds to TXCOEF[0] and the last bit in the bit-string (bit 23) corresponds to TXCOEF[23], noting that the bits are ordered in groups as defined in ITU-T G.9903:2017, table 9-21.

In the FCC bandplan, the first bit in the bit-string (bit 0) corresponds to TXCOEF[0] and the last bit in the bit-string (bit 47) corresponds to TXCOEF[47], noting that the bits are ordered in groups as defined in ITU-T G.9903:2017, table 9-22.

NOTE 4 A description of tx\_coeff is included in G.9903:2017 tables 9-21 and 9-22

In the CENELEC-A bandplan, each group of 4 bits of the TXCOEF field is associated with one sub-band consisting of a group of 6 tones.

| Bit number | TXCOEF field | Sub-band (kHz)     |
|------------|--------------|--------------------|
| 0          | TXCOEF[0]    | 35,937.5 to 43,75  |
| 1          | TXCOEF[1]    |                    |
| 2          | TXCOEF [2]   |                    |
| 3          | TXCOEF [3]   |                    |
| 4          | TXCOEF [4]   |                    |
| 5          | TXCOEF [5]   | 45,312.5 to 53,125 |

## COSEM Interface Classes

| Bit number | TXCOEF field | Sub-band (kHz)     |
|------------|--------------|--------------------|
| 6          | TXCOEF[6]    | 54,6875 to 62,5    |
| 7          | TXCOEF[7]    |                    |
| 8          | TXCOEF[8]    |                    |
| 9          | TXCOEF[9]    |                    |
| 10         | TXCOEF[10]   |                    |
| 11         | TXCOEF[11]   |                    |
| 12         | TXCOEF[12]   | 64,062 5 to 71,875 |
| 13         | TXCOEF[13]   |                    |
| 14         | TXCOEF[14]   |                    |
| 15         | TXCOEF[15]   |                    |
| 16         | TXCOEF[16]   | 73,437 5 to 81,25  |
| 17         | TXCOEF[17]   |                    |
| 18         | TXCOEF[18]   |                    |
| 19         | TXCOEF[19]   |                    |
| 20         | TXCOEF[20]   | 82,812 5 to 90,625 |
| 21         | TXCOEF[21]   |                    |
| 22         | TXCOEF[22]   |                    |
| 23         | TXCOEF[23]   |                    |

In the FCC bandplan, each group of 2 bits of the TXCOEF field is associated with one sub-band consisting of a group of 3 tones

| Bit number | TXCOEF field | Sub-band (kHz)         |
|------------|--------------|------------------------|
| 0          | TXCOEF[0]    | 154,687 5 to 164,062 5 |
| 1          | TXCOEF[1]    |                        |
| 2          | TXCOEF [2]   | 168,75 to 178,125      |
| 3          | TXCOEF [3]   |                        |
| 4          | TXCOEF [4]   | 182,812 5 to 192,187 5 |
| 5          | TXCOEF [5]   |                        |
| 6          | TXCOEF[6]    | 196,875 to 206,25      |
| 7          | TXCOEF[7]    |                        |
| 8          | TXCOEF[8]    | 210,937 5 to 220,312 5 |
| 9          | TXCOEF[9]    |                        |
| 10         | TXCOEF[10]   | 225 to 234,375         |
| 11         | TXCOEF[11]   |                        |
| 12         | TXCOEF[12]   | 239,062 5 to 248,437 5 |
| 13         | TXCOEF[13]   |                        |
| 14         | TXCOEF[14]   | 253,125 to 262,5       |
| 15         | TXCOEF[15]   |                        |
| 16         | TXCOEF[16]   | 267,187 5 to 276,562 5 |
| 17         | TXCOEF[17]   |                        |
| 18         | TXCOEF[18]   | 281,25 to 290,625      |

## COSEM Interface Classes

| Bit number | TXCOEF field | Sub-band (kHz)         |
|------------|--------------|------------------------|
| 19         | TXCOEF[19]   |                        |
| 20         | TXCOEF[20]   | 295,312.5 to 304,687.5 |
| 21         | TXCOEF[21]   |                        |
| 22         | TXCOEF[22]   | 309,375 to 318,75      |
| 23         | TXCOEF[23]   |                        |
| 24         | TXCOEF[24]   | 323,437.5 to 332,812.5 |
| 25         | TXCOEF[25]   |                        |
| 26         | TXCOEF[26]   | 337.5 to 346,875       |
| 27         | TXCOEF[27]   |                        |
| 28         | TXCOEF[28]   | 351,562.5 to 360,937.5 |
| 29         | TXCOEF[29]   |                        |
| 30         | TXCOEF[30]   | 365,625 to 375         |
| 31         | TXCOEF[31]   |                        |
| 32         | TXCOEF[32]   | 379,687.5 to 389,062.5 |
| 33         | TXCOEF[33]   |                        |
| 34         | TXCOEF[34]   | 393,75 to 403,125      |
| 35         | TXCOEF[35]   |                        |
| 36         | TXCOEF[36]   | 407,812.5 to 417,187.5 |
| 37         | TXCOEF[37]   |                        |
| 38         | TXCOEF[38]   | 421,875 to 431,25      |
| 39         | TXCOEF[39]   |                        |
| 40         | TXCOEF[40]   | 435,937.5 to 445,312.5 |
| 41         | TXCOEF[41]   |                        |
| 42         | TXCOEF[42]   | 450 to 459,375         |
| 43         | TXCOEF[43]   |                        |
| 44         | TXCOEF[44]   | 464,062.5 to 473,437.5 |
| 45         | TXCOEF[45]   |                        |
| 46         | TXCOEF[46]   | 478,125 to 487,5       |
| 47         | TXCOEF[47]   |                        |

- reverse\_lqi Link Quality Indicator of the link to the neighbour (reverse LQI).

NOTE 5 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 when the parameters are considered valid. When an entry is created, this value shall be set to the default value.

- 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.

- When a MAC fail occurs (when mac\_max\_frame\_retries attempts have failed), this value shall be set to 0.

#### - no\_data

Value that may be used as needed for some implementations as determined by a companion specification.

### **5.13.7.2.12 mac\_high\_priority\_window\_size**

PIB attribute 0x0100: The high priority contention window size in number of slots.

### **5.13.7.2.13 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.

### **5.13.7.2.14 mac\_beacon\_randomization\_window\_length**

PIB attribute 0x0111: Duration time in seconds for the beacon randomization.

### **5.13.7.2.15 mac\_A**

PIB attribute 0x0112: This parameter controls the adaptive CW linear decrease.

### **5.13.7.2.16 mac\_K**

PIB attribute 0x0113: Rate Adaptation factor for channel access fairness limit. This attribute can take a value between 1 and macCSMAFairnessLimit.

### **5.13.7.2.17 mac\_min\_CW\_attempts**

PIB attribute 0x0114: Number of consecutive attempts while using minimum CW.

### **5.13.7.2.18 mac\_cenelec\_legacy\_mode**

PIB attribute 0x0115: This read only attribute indicates the capability of the node.

0: The following configuration is used :

- Elementary interleaving,
- Interleaver parameters  $n_i$  and  $n_j$  are not swapped when  $I(i,j) = 0$ .

1: The following configuration is used:

- Full Block interleaving,
- Interleaver parameters  $n_i$  and  $n_j$  are swapped when  $I(i,j) = 0$ .

### **5.13.7.2.19 mac\_FCC\_legacy\_mode**

PIB attribute 0x0116: This read only attribute indicates the capability of the node.

0: The following configuration is used:

- 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:

- Coherent FCH modulation,
- Full Block interleaving,
- Interleaver parameters  $n_i$  and  $n_j$  are swapped when  $I(i,j) = 0$ ,
- Two RS blocks.

#### **5.13.7.2.20 mac\_max\_BE**

PIB attribute 0x0047: Maximum value of backoff exponent. It should always be greater than mac\_min\_BE.

#### **5.13.7.2.21 mac\_max\_CSMA\_backoffs**

PIB attribute 0x004E: Maximum number of backoff attempts.

#### **5.13.7.2.22 mac\_min\_BE**

PIB attribute 0x004F: Minimum value of backoff exponent.

#### **5.13.7.2.23 mac\_broadcast\_max\_CW\_enabled**

PIB attribute 0x011E: If enabled, MAC uses maximum contention window.

#### **5.13.7.2.24 mac\_transmit\_atten**

PIB attribute 0x011F: Attenuation of the output level in dB.

#### **5.13.7.2.25 mac\_POS\_table**

PIB attribute 0x0120: The neighbour table contains some information about all the devices within the POS of the device.

```
array POS_table
POS_table ::= structure
{
    short_address: long-unsigned,
    LQI: unsigned,
    POS_valid_time: unsigned
}
```

NOTE 6 Each time a message is received (filtered according to IEEE 802.15.4, 7.5.6.2), an entry in the POS table is created or updated (if the entry is already present). In case the table is full, the entry corresponding to the shortest valid time is removed.

short\_address The MAC Short Address of the neighbour which this entry refers to.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 551/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

LQI Link quality indicator of the last received packet from this neighbour.

NOTE 7 LQI is referred to as forward\_LQI in some implementations.

POS\_valid\_time Remaining time in minutes until when this entry is considered valid.

Every time an entry is created, it is set to mac\_POS\_table\_entry\_TTL. When it reaches zero, this entry is no longer valid in the table and may be removed.

### 5.13.7.3 Method description

#### 5.13.7.3.1 mac\_get\_neighbour\_table\_entry (data)

This method is used to retrieve the mac neighbour table for one MAC short address. It may be used to perform topology monitoring by the client.

The method invocation parameter contains a mac\_short\_address.

data ::= long-unsigned

The response parameter includes the neighbour table for this mac\_short\_address.

data ::= array neighbour\_table

where mac\_neighbour\_table is as defined in the mac\_neighbour\_table attribute of the present IC.

### 5.13.8 G3 NB OFDM PLC 6LoWPAN adaptation layer setup (class\_id = 92, version = 0)

#### 5.13.8.1 Overview

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<br>5 for FCC | x + 0x18   |
| 4. adp_tone_mask                              | (static) | bit-string    |                            |       | All bits set to 1            | x + 0x20   |
| 5. adp_discovery_attempts_wait_time           | (static) | long-unsigned | 1                          | 3 600 | 60                           | x + 0x28   |
| 6. adp_routing_configuration                  | (static) | array         |                            |       |                              | x + 0x30   |
| 7. adp_broadcast_log_table_entry_TTL          | (static) | long-unsigned | 0                          | 3 600 | 90                           | x + 0x38   |
| 8. adp_routing_table                          | (dyn.)   | array         |                            |       |                              | x + 0x40   |
| 9. adp_context_information_table              | (dyn)    | array         |                            |       |                              | x + 0x48   |
| 10. adp_blacklist_table                       | (dyn)    | array         |                            |       |                              | x + 0x50   |
| 11. adp_broadcast_log_table                   | (dyn)    | array         |                            |       |                              | x + 0x58   |
| 12. adp_group_table                           | (dyn)    | array         |                            |       |                              | x + 0x60   |
| 13. adp_max_join_wait_time                    | (static) | long-unsigned | 0                          | 1 023 | 20                           | x + 0x68   |
| 14. adp_path_discovery_time                   | (static) | long-unsigned | 0                          | 65535 | 5 000                        | x + 0x70   |
| 15. adp_use_new_GMK_time                      | (static) | long-unsigned | 0                          | 65535 | 3 600                        | x + 0x78   |
| 16. adp_exp_prec_GMK_time                     | (static) | long-unsigned | 0                          | 3 600 | 3 600                        | X + 0x80   |
| Specific methods                              | m/o      |               |                            |       |                              |            |

#### 5.13.8.2 Attribute description

##### 5.13.8.2.1 logical\_name

Identifies the “G3 NB OFDM 6LoWPAN adaptation layer setup” object instance. See 6.2.28.

#### 5.13.8.2.2 adp\_max\_hops

PIB attribute 0x10: Defines the maximum number of hops to be used by the routing algorithm.

#### 5.13.8.2.3 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).

#### 5.13.8.2.4 adp\_tone\_mask

PIB attribute 0x0F: Defines the Tone Mask to use during symbol formation. The length of the bit-string is 70 bits.

#### 5.13.8.2.5 adp\_discovery\_attempts\_wait\_time

PIB attribute 0x06: Allows programming the maximum wait time between invocations of two consecutive network discovery primitives (in seconds).

#### 5.13.8.2.6 adp\_routing\_configuration

The routing configuration element specifies all parameters linked to the routing mechanism described in ITU-T G.9903 Amd. 1:2013. The elements are specified in 9.4.1.2 of that Recommendation.

NOTE 1 The Link cost calculation is provided in ITU-T G.9903 Amd. 1:2013 Annex B.

```
array routing_configuration

routing_configuration ::= structure
{
    adp_net_traversal_time:           long-unsigned,
    adp_routing_table_entry_TTL:      double-long-unsigned,
    adp_Kr:                          unsigned,
    adp_Km:                          unsigned,
    adp_Kc:                          unsigned,
    adp_Kq:                          unsigned,
    adp_Kh:                          unsigned,
    adp_Krt:                         unsigned,
    adp_RREQ_retries:                unsigned,
    adp_RREQ_RERR_wait:              long-unsigned,
    adp_blacklist_table_entry_TTL:   long-unsigned
}
```

---

|                        |  |
|------------------------|--|
| adp_net_traversal_time | PIB attribute 0x12: The Max duration between RREQ and the correspondent RREP (in seconds). |
| Range:                 | 0-65 535   |

Default value: 20

---

|                             |   |
|-----------------------------|---|
| adp_routing_table_entry_TTL | PIB attribute 0x13: The time to live of a route request table entry (in seconds). |
| Range:                      | 0-262 143   |

Default value: 90

## COSEM Interface Classes

---

|                               |   |
|-------------------------------|---|
| adp_Kr                        | PIB attribute 0x14: A weight factor for ROBO to calculate link cost.<br>Range: 0-31<br>Default value: 0   |
| adp_Km                        | PIB attribute 0x15: A weight factor for modulation to calculate link cost.<br>Range: 0-31<br>Default value: 0                                       |
| adp_Kc                        | PIB attribute 0x16: A weight factor for number of active tones to calculate link cost.<br>Range: 0-31<br>Default value: 0                           |
| adp_Kq                        | PIB attribute 0x17: A weight factor for LQI to calculate route cost.<br>Range: 0-31<br>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.<br>Range: 0-31<br>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.<br>Range: 0-31<br>Default value: 0 |
| adp_RREQ_retries              | PIB attribute 0x19: The number of RREQ re-transmission in case of RREP reception time out.<br>Range: 0-255<br>Default value: 0                      |
| adp_RREQ_RERR_wait            | PIB attribute 0x1A: The number of seconds to wait between two consecutive RREQ – RERR generations.<br>Range: 0-65 535<br>Default value: 30          |
| adp_blacklist_table_entry_TTL | PIB attribute 0x26: Time to live of a blacklisted neighbour set entry in minutes.<br>Range: 0-65 535<br>Default value: 10                           |

---

### 5.13.8.2.7 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).

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
|                       |            | 555/668                     |

**5.13.8.2.8 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
}
```

|                     |  |
|---------------------|--|
| 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.<br>Range: 0-15<br><br>NOTE 3      Practically the maximum allowed value is limited by adp_max_hops. |
| weak_link_count     | Number of weak links to destination.<br>Range : 0-15<br><br>NOTE   Practically the maximum allowed value is limited by adp_max_hops.                         |
| status              | Status of the routing table entry.<br>enum :        (0)    Invalid route,<br>(1)    Valid route,<br>(2)    Route under construction                          |
| valid_time          | Remaining time in seconds until which this entry in the routing table is considered valid.   |

**5.13.8.2.9****5.13.8.2.10 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).<br>Range: 0x00-0x0F   |
| context_length | Indicates the length of the carried context (up to 128-bit contexts may be carried).<br>Range: 0-128   |
| context        | Corresponds to the carried context used for compression/decompression purposes.<br>Range: 0x0000:0000:0000:0000:0000:0000:0000:0000 – 0xFFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF  |
| C              | Indicates if the context is valid for use in compression<br>0: Only decompression is allowed,<br>1: Compression and decompression are allowed<br><br>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.<br>Range: 0-65 535   |

---

**5.13.8.2.11 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. |

---

**5.13.8.2.12 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,
```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 557/668 |
|-----------------------|------------|-----------------------------|---------|

```
    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.                 |

---

#### 5.13.8.2.13 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.
```

#### 5.13.8.2.14 adp\_max\_join\_wait\_time

PIB attribute 0x21: Network join timeout in seconds for LBD.

#### 5.13.8.2.15 adp\_path\_discovery\_time

PIB attribute 0x22: Timeout for path discovery in msec.

#### 5.13.8.2.16 adp\_use\_new\_GMK\_time

PIB attribute 0x23: The wait time in seconds for a device to use new GMK after rekeying.

#### 5.13.8.2.17 adp\_exp\_prec\_GMK\_time

PIB attribute 0x24: The time in seconds to keep PrecGMK after switching to a new GMK.

### 5.13.9 G3-PLC 6LoWPAN adaptation layer setup (class\_id = 92, version = 1)

#### 5.13.9.1 Overview

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.

## COSEM Interface Classes

| <b>G3-PLC 6LoWPAN adaptation layer setup</b> |            | <b>0...n</b>     | <b>class_id = 92, version = 1</b> |             |             |                   |
|--|------------|------------------|-----------------------------------|-------------|-------------|-------------------|
| <b>Attributes</b>                            |            | <b>Data type</b> | <b>Min.</b>                       | <b>Max.</b> | <b>Def.</b> | <b>Short name</b> |
| 1. logical_name                              | (static)   | octet-string     |                                   |             |             | x                 |
| 2. adp_max_hops                              | (static)   | unsigned         | 1                                 | 14          | 8           | x + 0x08          |
| 3. adp_weak_LQI_value                        | (static)   | unsigned         | 0                                 | 255         | 52          | x + 0x10          |
| 4. adp_security_level                        | (static)   | unsigned         | 0                                 | 7           | 5           | x + 0x18          |
| 5. adp_prefix_table                          | (dyn.)     | array            |                                   |             |             | x + 0x20          |
| 6. adp_routing_configuration                 | (static)   | array            |                                   |             |             | x + 0x28          |
| 7. adp_broadcast_log_table_entry_TTL         | (static)   | long-unsigned    | 0                                 | 65535       | 2           | x + 0x30          |
| 8. adp_routing_table                         | (dyn.)     | array            |                                   |             |             | x + 0x38          |
| 9. adp_context_information_table             | (dyn)      | array            |                                   |             |             | x + 0x40          |
| 10. adp_blacklist_table                      | (dyn)      | array            |                                   |             |             | x + 0x48          |
| 11. adp_broadcast_log_table                  | (dyn)      | array            |                                   |             |             | x + 0x50          |
| 12. adp_group_table                          | (dyn)      | array            |                                   |             |             | x + 0x58          |
| 13. adp_max_join_wait_time                   | (static)   | long-unsigned    | 0                                 | 1 023       | 20          | x + 0x60          |
| 14. adp_path_discovery_time                  | (static)   | unsigned         | 0                                 | 255         | 40          | x + 0x68          |
| 15. adp_active_key_index                     | (static)   | unsigned         | 0                                 | 1           | 0           | x + 0x70          |
| 16. adp_metric_type                          | (static)   | unsigned         | 0x00                              | 0x0F        | 0x0F        | x + 0x78          |
| 17. adp_coord_short_address                  | (static)   | long-unsigned    | 0x0000                            | 0x7FFF      | 0x0000      | x + 0x80          |
| 18. adp_disable_default_routing              | (static)   | boolean          |                                   |             | FALSE       | x + 0x88          |
| 19. adp_device_type                          | (static)   | enum             | 0                                 | 2           | 2           | x + 0x90          |
| <b>Specific methods</b>                      | <b>m/o</b> |                  |                                   |             |             |                   |

### 5.13.9.2 Attribute description

#### 5.13.9.2.1 logical\_name

Identifies the “G3-PLC OFDM 6LoWPAN adaptation layer setup” object instance. See 6.2.28.

#### 5.13.9.2.2 adp\_max\_hops

PIB attribute 0x0F: Defines the maximum number of hops to be used by the routing algorithm.

#### 5.13.9.2.3 adp\_weak\_LQI\_value

PIB attribute 0x1A: The weak link value defines the LQI value below which a link to a neighbour is considered as a weak link. A value of 52 represents an SNR of 3 dB.

#### 5.13.9.2.4 adp\_security\_level

PIB attribute 0x00: The minimum security level to be used for incoming and outgoing adaptation frames. Only values 0 (no ciphering) and 5 (ciphering with 32 bits integrity code) are supported.

**5.13.9.2.5 adp\_prefix\_table**

PIB attribute 0x01: Contains the list of prefixes defined on this PAN.

**5.13.9.2.6 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.

```
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
}
```

Where

|                             |   |
|-----------------------------|---|
| 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.<br>Range : 0-255<br>Default value : 20 |
| adp_routing_table_entry_TTL | PIB attribute 0x12: Maximum time-to-live of a routing table entry (in minutes).<br>Range : 0-65 535<br>Default value : 60                             |
| adp_Kr                      | PIB attribute 0x13: A weight factor for the Robust Mode to calculate link cost.<br>Range : 0-31<br>Default value : 0                                  |
| adp_Km                      | PIB attribute 0x14: A weight factor for modulation to calculate link cost.<br>Range : 0-31<br>Default value : 0                                       |
| adp_Kc                      | PIB attribute 0x15: A weight factor for number of active tones to calculate link cost.<br>Range : 0-31<br>Default value : 0                           |

## COSEM Interface Classes

|                               |   |
|-------------------------------|---|
| adp_Kq                        | PIB attribute 0x16: A weight factor for LQI to calculate route cost.<br>Range : 0-50<br>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.<br>Range : 0-31<br>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.<br>Range : 0-31<br>Default value : 0   |
| adp_RREQ_retries              | PIB attribute 0x18: The number of RREQ re-transmission in case of RREP reception time out.<br>Range : 0-255<br>Default value : 0  |
| adp_RREQ_RERR_wait            | PIB attribute 0x19: The number of seconds to wait between two consecutive RREQ – RERR generations.<br>Range : 0-255<br>Default value : 30   |
| adp_blacklist_table_entry_TTL | PIB attribute 0x1F: Maximum time-to-live of a blacklisted neighbour entry (in minutes).<br>Range : 0-65 535<br>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'.<br>If FALSE, the RREQ shall be generated with its "unicast RREQ" flag set to '0'.<br>Default value : TRUE |
| adp_RLC_enabled               | PIB attribute 0x09: Enable the sending of RLCREQ frame by the device.<br>Default value : FALSE  |
| adp_add_rev_ink_cost          | PIB attribute 0x0A: It represents an additional cost to take into account a possible asymmetry in the link.<br>Range : 0-255<br>Default value : 0   |

### 5.13.9.2.7 adp\_broadcast\_log\_table\_entry\_TTL

PIB attribute 0x02: Maximum time to live of an adpBroadcastLogTable entry (in minutes).

### 5.13.9.2.8 adp\_routing\_table

PIB attribute 0x0C: Contains the routing table.

array routing\_table

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
|                       |            | 561/668                     |

```

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.

Where

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.

#### **5.13.9.2.9 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
}

```

Where

CID Corresponds to the 4-bit context information used for source and destination addresses (SCI, DCI).  
Range: 0x00-0x0F

## COSEM Interface Classes

|                |   |
|----------------|---|
| context_length | Indicates the length of the carried context (up to 128-bit contexts may be carried).<br>Range: 0-128  |
| context        | Corresponds to the carried context used for compression/decompression purposes.<br>Range: 0x0000:0000:0000:0000:0000:0000:0000 – 0xFFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF   |
| C              | <p>Indicates if the context is valid for use in compression.</p> <p>FALSE: Only decompression is allowed,</p> <p>TRUE: 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 | <p>Remaining time in minutes during which the context information table is considered valid. It is updated upon reception of the advertised context.</p> <p>Range: 0-65 535</p>   |

### **5.13.9.2.10 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
}

```

Where

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.

### **5.13.9.2.11 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,

```

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 563/668 |
|-----------------------|------------|-----------------------------|---------|

```

sequence_number:         unsigned,
valid_time:             long-unsigned
}

```

Where

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.

valid\_time      Remaining time in minutes until when this entry in the broadcast log table is considered valid.

#### **5.13.9.2.12 adp\_group\_table**

PIB attribute 0x0E: Contains the group addresses to which the device belongs.

```
array group_address
```

```
group_address ::= long-unsigned
```

group\_address      Group address to which this node has been subscribed.

#### **5.13.9.2.13 adp\_max\_join\_wait\_time**

PIB attribute 0x20: Network join timeout in seconds for LBD.

#### **5.13.9.2.14 adp\_path\_discovery\_time**

PIB attribute 0x21: Timeout for path discovery in seconds.

#### **5.13.9.2.15 adp\_active\_key\_index**

PIB attribute 0x22: Index of the active GMK to be used for data transmission

#### **5.13.9.2.16 adp\_metric\_type**

PIB attribute 0x03: Metric Type to be used for routing purposes

#### **5.13.9.2.17 adp\_coord\_short\_address**

PIB attribute 0x08: Defines the short address of the coordinator.

#### **5.13.9.2.18 adp\_disable\_default\_routing**

PIB attribute 0xF0: If TRUE, the default routing (LOADng) is disabled. If FALSE, the default routing (LOADng) is enabled.

#### **5.13.9.2.19 adp\_device\_type**

PIB attribute 0x10: Defines the type of the device connected to the modem:

enum:

- (0) PAN device,
- (1) PAN coordinator,

(2) Not Defined

**5.13.10 G3-PLC 6LoWPAN adaptation layer setup (class\_id = 92, version = 2)****5.13.10.1 Overview**

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 = 2 |        |        |            |
|---------------------------------------|-----------|------------------------------|----------------------------|--------|--------|------------|
| Attribute(s)                          | 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                          | 5      | 5      | x + 0x18   |
| 5. adp_prefix_table                   | (dyn)     | array                        |                            |        |        | x + 0x20   |
| 6. adp_routing_configuration          | (static)  | array                        |                            |        |        | x + 0x28   |
| 7. adp_broadcast_log_table            | (static)  | long- unsigned<br>_entry_TTL | 0                          | 65535  | 2      | x + 0x30   |
| 8. adp_routing_table                  | (dyn)     | array                        |                            |        |        | x + 0x38   |
| 9. adp_context_information_table      | (dyn)     | array                        |                            |        |        | x + 0x40   |
| 10. adp_blacklist_table               | (dyn)     | array                        |                            |        |        | x + 0x48   |
| 11. adp_broadcast_log_table           | (dyn)     | array                        |                            |        |        | x + 0x50   |
| 12. adp_group_table                   | (dyn)     | array                        |                            |        |        | x + 0x58   |
| 13. adp_max_join_wait_time            | (static)  | long- unsigned               | 0                          | 1023   | 20     | x + 0x60   |
| 14. adp_path_discovery_time           | (static)  | unsigned                     | 0                          | 255    | 40     | x + 0x68   |
| 15. adp_active_key_index              | (static)  | unsigned                     | 0                          | 1      | 0      | x + 0x70   |
| 16. adp_metric_type                   | (static)  | unsigned                     | 0x00                       | 0x0F   | 0x0F   | x + 0x78   |
| 17. adp_coord_short_address           | (static)  | long- unsigned               | 0x0000                     | 0x7FFF | 0x0000 | x + 0x80   |
| 18. adp_disable_default_routing       | (static)  | boolean                      |                            |        | FALSE  | x + 0x88   |
| 19. adp_device_type                   | (static)  | enum                         | 0                          | 2      | 2      | x + 0x90   |
| 20. adp_default_coord_route_enabled   | (static)  | boolean                      |                            |        | FALSE  | x + 0x98   |
| 21. adp_destination_address_set       | (dyn)     | array                        |                            |        |        | x + 0xA0   |
| <b>Specific methods</b>               |           | m/o                          |                            |        |        |            |

**5.13.10.2 Attribute description****5.13.10.2.1 logical\_name**

Identifies the “G3-PLC OFDM 6LoWPAN adaptation layer setup” object instance. See 6.2.28.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 565/668 |
|-----------------------|------------|-----------------------------|---------|

**5.13.10.2.2 adp\_max\_hops**

PIB attribute 0x0F: Defines the maximum number of hops to be used by the routing algorithm.

**5.13.10.2.3 adp\_weak\_LQI\_value**

PIB attribute 0x1A: The weak link value defines the LQI value below which a link to a neighbour is considered as a weak link. A value of 52 represents an SNR of 3 dB.

**5.13.10.2.4 adp\_security\_level**

PIB attribute 0x00: The minimum security level to be used for incoming and outgoing adaptation frames. Only values 0 (no ciphering) and 5 (ciphering with 32 bits integrity code) are supported.

**5.13.10.2.5 adp\_prefix\_table**

PIB attribute 0x01: Contains the list of prefixes defined on this PAN.

NOTE 1 it is assumed that the link local IPv6 address exists independently and is not affected by the prefixes defined in the prefix table.

**5.13.10.2.6 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 2 The Link cost calculation is provided in ITU-T G.9903:2014 Annex B.

```
array routing_configuration
routing_configuration ::= structure
{
    adp_net_traversal_time:           unsigned,
    adp_routing_table_entry_TTL:      long-unsigned,
    adp_Kr:                          unsigned,
    adp_Km:                          unsigned,
    adp_Kc:                          unsigned,
    adp_Kq:                          unsigned,
    adp_Kh:                          unsigned,
    adp_Krt:                         unsigned,
    adp_RREQ_retries:                unsigned,
    adp_RREQ_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
}
```

Where

|                               |  |
|-------------------------------|--|
| <b>adp_net_traversal_time</b> | 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   |

|                                    |   |
|------------------------------------|---|
| <b>adp_routing_table_entry_TTL</b> | PIB attribute 0x12: Maximum time-to-live of a routing table entry (in minutes). |
|------------------------------------|---|

## COSEM Interface Classes

|  |  |
|--|--|
|  | Range : 0-65 535<br>Default value : 60   |
| adp_Kr   | PIB attribute 0x13: A weight factor for the Robust Mode to calculate link cost.<br>Range : 0-31<br>Default value : 0   |
| adp_Km   | PIB attribute 0x14: A weight factor for modulation to calculate link cost.<br>Range : 0-31<br>Default value : 0  |
| adp_Kc   | PIB attribute 0x15: A weight factor for number of active tones to calculate link cost.<br>Range : 0-31<br>Default value : 0                                  |
| adp_Kq   | PIB attribute 0x16: A weight factor for LQI to calculate route cost.<br>Range : 0-50<br>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.<br>Range : 0-31<br>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.<br>Range : 0-31<br>Default value : 0        |
| adp_RREQ_retries   | PIB attribute 0x18: The number of RREQ re-transmission in case of RREP reception time out.<br>Range : 0-255<br>Default value : 0                             |
| adp_RREQ_wait  | PIB attribute 0x19: The number of seconds to wait between two consecutive RREQ – RERR generations.<br>Range : 0-255<br>Default value : 30                    |
| NOTE 3 : The title and definition of this item has been modified from previous version |  |
| adp_blacklist_table_entry_TTL  | PIB attribute 0x1F: Maximum time-to-live of a blacklisted neighbour entry (in minutes).<br>Range : 0-65 535<br>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'.<br>If FALSE, the RREQ shall be generated with its "unicast |

## COSEM Interface Classes

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\_ink\_cost** PIB attribute 0x0A: It represents an additional cost to take into account a possible asymmetry in the link.  
Range : 0-255  
Default value : 0

### 5.13.10.2.7 adp\_broadcast\_log\_table\_entry\_TTL

PIB attribute 0x02: Maximum time to live of an adpBroadcastLogTable entry (in minutes).

### 5.13.10.2.8 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 4 This table is actualized each time a route is built or updated (triggered by data traffic) and each time the TTL timer expires.

Where:

**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 5 Practically the maximum allowed value is limited by adp\_max\_hops.

**weak\_link\_count** Number of weak links to destination.  
Range: 0-14

NOTE 6 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.

**5.13.10.2.9 adp\_context\_information\_table**

PIB attribute 0x07: Contains the context information associated to each CID extension field. See ITU-T G.9903:2017. The elements are specified in Table 9-30 of that Recommendation.

```
array context_information_table

context_information_table ::= structure

{
  CID:          bit-string,
  context_length: unsigned,
  context:       octet-string,
  C:             boolean,
  valid_lifetime: long-unsigned
}
```

Where:

|                |  |
|----------------|--|
| CID            | Corresponds to the 4-bit context information used for source and destination addresses (SCI, DCI).<br>The first bit in the bit-string (bit 0) corresponds to the least significant bit of CID in G.9903:2017, Table 9-30.<br>Range: 0x00-0x0F  |
| context_length | Indicates the length of the carried context (up to 128-bit contexts may be carried).<br>Range: 0-128   |
| context        | Corresponds to the carried context used for compression/decompression purposes.  |
| C              | Indicates if the context is valid for use in compression.<br><br>FALSE: Only decompression is allowed,<br><br>TRUE: Compression and decompression are allowed<br><br>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.<br><br>Range: 0-65 535   |

**5.13.10.2.10 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
}
```

Where:

**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.

#### 5.13.10.2.11 adp\_broadcast\_log\_table

PIB attribute 0x0B: Contains the broadcast log table.

NOTE 7 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
}
```

Where:

**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.

**valid\_time** Remaining time in minutes until when this entry in the broadcast log table is considered valid.

#### 5.13.10.2.12 adp\_group\_table

PIB attribute 0x0E: Contains the group addresses to which the device belongs.

```
array group_address

group_address ::= long-unsigned

group_address Group address to which this node has been subscribed.
```

#### 5.13.10.2.13 adp\_max\_join\_wait\_time

PIB attribute 0x20: Network join timeout in seconds for LBD.

#### 5.13.10.2.14 adp\_path\_discovery\_time

PIB attribute 0x21: Timeout for path discovery in seconds.

**5.13.10.2.15 adp\_active\_key\_index**

PIB attribute 0x22: Index of the active GMK to be used for data transmission

**5.13.10.2.16 adp\_metric\_type**

PIB attribute 0x03: Metric Type to be used for routing purposes

**5.13.10.2.17 adp\_coord\_short\_address**

PIB attribute 0x08: Defines the short address of the coordinator.

**5.13.10.2.18 adp\_disable\_default\_routing**

PIB attribute 0xF0: If TRUE, the default routing (LOADng) is disabled. If FALSE, the default routing (LOADng) is enabled.

**5.13.10.2.19 adp\_device\_type**

PIB attribute 0x10: Defines the type of the device connected to the modem:

enum:

- (0) PAN device,
- (1) PAN coordinator,
- (2) Not Defined

**5.13.10.2.20 adp\_default\_coord\_route\_enabled**

PIB attribute 0x24: If TRUE, the adaptation layer adds a default route to the coordinator after successful completion of the bootstrapping procedure. If FALSE no default route will be created.

**5.13.10.2.21 adp\_destination\_address\_set**

PIB attribute 0x23: Contains the list of the addresses of the devices for which this LOADng router is providing connectivity.

array D\_address

D\_address ::= long-unsigned

Where:

|           |   |
|-----------|---|
| D_Address | The 16-bit address of a destination node attached to this LOADng router and for which this LOADng router provides connectivity. |
|-----------|---|

## **5.14 Previous versions of interface classes for setting up and managing DLMS/COSEM HS-PLC ISO/IEC 12139-1 neighbourhood networks**

There are no previous versions to report.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 571/668 |
|-----------------------|------------|-----------------------------|---------|

### 5.15 Previous versions of ZigBee® setup classes

There are no previous versions to report.

## 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. The detailed OBIS code allocations for all media / energy types are specified in DLMS UA 1000-1 Ed 15 Part 1:2021.

The following rules for object instantiations are applicable with interface classes:

- if the use of IC “Data” is specified but it is not available in a given implementation, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used;
- when, instead of a “Data” object, a “Register” or “Extended register” object is used, then the data types allowed for the value attribute of the “Data” IC are allowed.

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;
- 6.5 specifies the use and logical names for heat cost allocator (HCA) related COSEM objects;
- 6.6 specifies the use and logical names for thermal energy related COSEM objects;
- 6.7 specifies the use and logical names for gas related COSEM objects;
- 6.8 specifies the use and logical names for water related COSEM objects;

**NOTE** The use and the logical names of COSEM objects related to other media / energy types are under consideration.

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 DLMS UA 1000-1 Ed 15 Part 1:2021 .

All codes that are not explicitly listed, but 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 56 shows the use of value group C for abstract objects in the COSEM context. See also DLMS UA 1000-1 Ed 15 Part 1:2021, Table 5.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 573/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

**Table 56 – Use of value group C for abstract objects in the COSEM context**

| Value group C<br>Abstract objects (A = 0) |  |
|---|--|
| <b>0</b>                                  | General purpose COSEM objects  |
| <b>1</b>                                  | Instances of IC "Clock"  |
| <b>2</b>                                  | Instances of IC "Modem configuration" and related IC-s   |
|   |  |
| <b>10</b>                                 | Instances of IC "Script table"   |
| <b>11</b>                                 | Instances of IC "Special days table"   |
| <b>12</b>                                 | Instances of IC "Schedule"   |
| <b>13</b>                                 | Instances of IC "Activity calendar"  |
| <b>14</b>                                 | Instances of IC "Register activation"  |
| <b>15</b>                                 | Instances of IC "Single action schedule"   |
| <b>16</b>                                 | Instances of IC "Register monitor", "Parameter monitor"  |
| <b>17</b>                                 | Instances of IC "Limiter"  |
| <b>18</b>                                 | Instances of IC "Array manager"  |
| <b>19</b>                                 | COSEM objects related to payment metering: "Account", "Credit", "Charge", "Token gateway", "IEC 62055-41 attributes"   |
|   |  |
| <b>20</b>                                 | Instances of IC "IEC local port setup"   |
| <b>21</b>                                 | Standard readout definitions   |
| <b>22</b>                                 | Instances of IC "IEC HDLC setup"   |
| <b>23</b>                                 | Instances of IC "IEC twisted pair (1) setup"   |
| <b>24</b>                                 | COSEM objects related to M-Bus   |
| <b>25</b>                                 | Instances of IC "TCP-UDP setup", "IPv4 setup", "IPv6 setup", "MAC address setup", "PPP setup", "GPRS modem setup", "SMTP setup", "GSM diagnostic", "FTP setup", "Push setup", "NTP setup", "LTE monitoring", "SCHC-LPWAN setup", "SCHC-LPWAN diagnostic", "SCHC-LoRaWAN setup", "SCHC-LoRaWAN diagnostic", "CoAP setup", "CoAP diagnostic" |
| <b>26</b>                                 | COSEM objects for data exchange using S-FSK PLC networks   |
| <b>27</b>                                 | COSEM objects for ISO/IEC 8802-2 LLC layer setup   |
| <b>28</b>                                 | COSEM objects for data exchange using narrow-band OFDM PLC for PRIME networks  |
| <b>29</b>                                 | COSEM objects for data exchange using narrow-band OFDM PLC for G3-PLC networks   |
| <b>30</b>                                 | COSEM objects for data exchange using ZigBee®  |
| <b>31</b>                                 | Instances of IC "Wireless Mode Q" (M-Bus)  |
| <b>32</b>                                 | Instances of IC "ISO/IEC 14908 identification", "ISO/IEC 14908 protocol setup", "ISO/IEC protocol status", "ISO/IEC 14908 diagnostic"  |
| <b>33</b>                                 | COSEM objects for data exchange using HS-PLC ISO/IEC 12139-1 networks  |
| <b>34</b>                                 | Instances of IC "Wi-SUN setup", "Wi-SUN diagnostic", "RPL diagnostic", "MPL diagnostic"  |
| <b>40</b>                                 | Instances of IC "Association SN/LN"  |
| <b>41</b>                                 | Instances of IC "SAP assignment"   |
| <b>42</b>                                 | COSEM logical device name  |
| <b>43</b>                                 | COSEM objects related to security: Instances of IC "Security setup" and "Data protection"  |
| <b>44</b>                                 | Instances of IC "Image transfer", "Function control" and "Communication port protection" objects   |
|   |  |
| <b>65</b>                                 | Instances of IC "Utility tables"   |

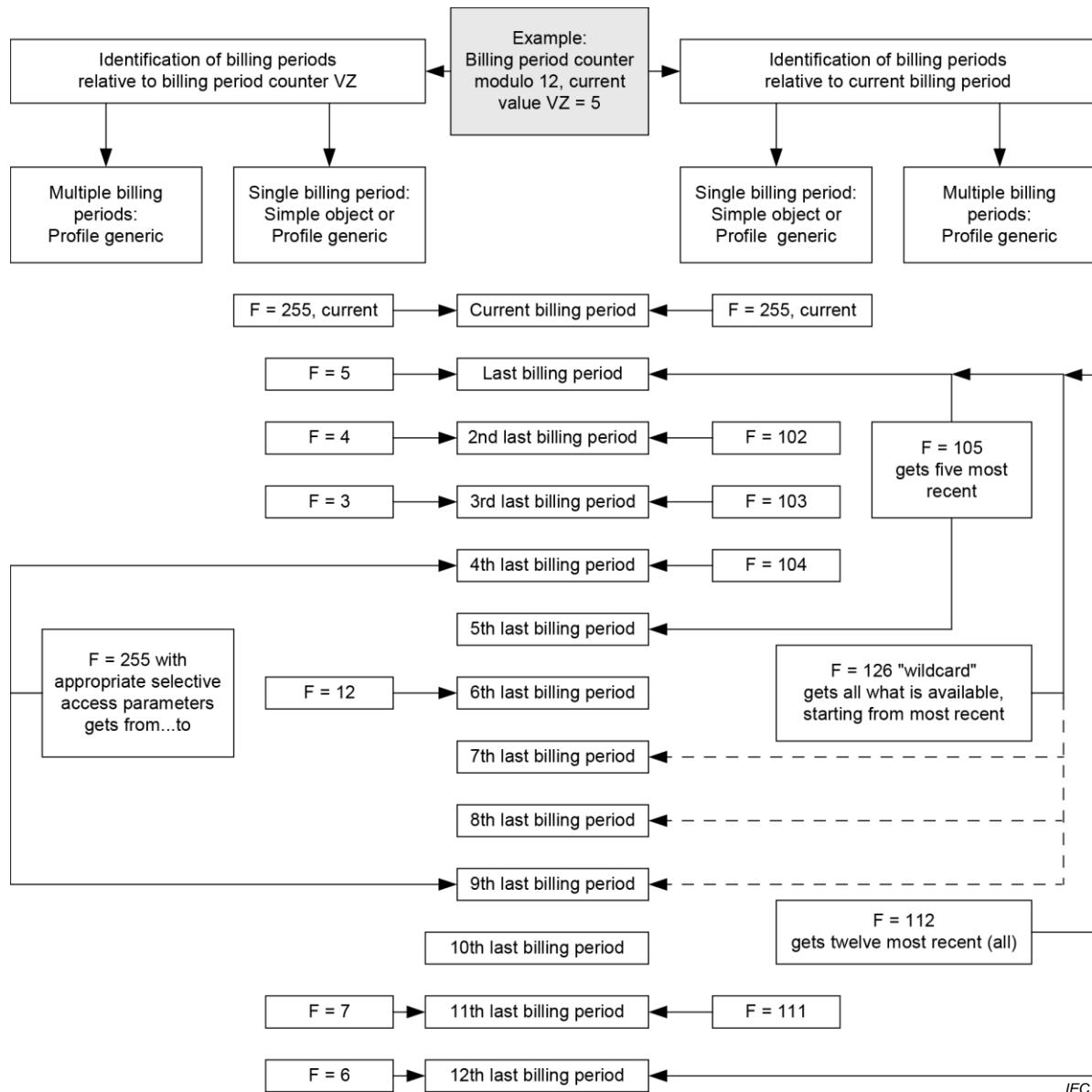
## COSEM Interface Classes

| <b>Value group C</b>            |   |
|---------------------------------|---|
| <b>Abstract objects (A = 0)</b> |   |
| <b>66</b>                       | Instances of "Compact data"   |
| <b>96</b>                       | Instances of "Device ID", "Metering point ID", "Parameter changes and calibration", "I/O control signal", "Disconnect control", "Arbitrator", "Status of internal control signals", "internal operating status", "Battery entities", "Power failure monitoring", "operating time", Environment related parameters", "Status register", "Event code", "Communication port log parameter", "Consumer message", "Currently active tariff", "event counter", "Profile entry digital signature", "Meter tamper event related", "Profile entry counter" objects |
| <b>97</b>                       | Instances of "Error register", "Alarm register", "Alarm filter", Alarm descriptor" objects  |
| <b>98</b>                       | Instances of "General list" objects   |
| <b>99</b>                       | Instances of "Event log" objects  |
| <b>127</b>                      | Instances of "Inactive objects"   |
| <b>128...199</b>                | Manufacturer specific COSEM related abstract objects  |
| <b>All other</b>                | 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 35 and is described below:

## COSEM Interface Classes



**Figure 35 – 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 and channels, see 6.3.3 and DLMS UA 1000-1 Ed 15 Part 1:2021, Table 20.

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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 1, Data <sup>a</sup> | 0         | b | 0 | 1 | e | 255 |

<sup>a</sup> 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.

## COSEM Interface Classes

For detailed OBIS codes, see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8.

| Abstract general purpose OBIS codes  | IC                   | OBIS code |   |   |   |   |     |
|--|----------------------|-----------|---|---|---|---|-----|
|  |                      | A         | B | C | D | E | F   |
| Program entries  | 1, Data <sup>a</sup> | 0         | b | 0 | 2 | e | 255 |
| Time entries   |                      | 0         | b | 0 | 9 | e | 255 |
| <sup>a</sup> 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.

NOTE: DLMS uses *double-long-unsigned* rather than the UNIX standard *double-long-signed* data type to extend the roll over date to 2106.

"Microseconds clock" objects are instances of the IC "Data", with data type *long64-unsigned*. They hold the number of microseconds since 1970-01-01 00:00:00 UTC.

NOTE Previously, the name of these objects was "High resolution clock".

"Minutes clock" objects are instances of the IC "Data", with data type *double-long-unsigned*. They hold the whole number of minutes since 1970-01-01 00:00:00 UTC.

"Hours clock" objects are instances of the IC "Data", with data type *double-long-unsigned*. They hold the whole number of the hours since 1970-01-01 00:00:00 UTC.

"Days clock" objects are instances of the IC "Data", with data type *long-unsigned*. They hold the whole number of the days since 1970-01-01 00:00:00 UTC.

"Weeks clock" objects are instances of the IC "Data", with data type *long-unsigned*. They hold the whole number of weeks since 1970-01-01 00:00:00 UTC.

These objects are all inter-related in that it is anticipated that there is a single notion of time within the device.

| Clock objects  | IC                   | OBIS code |   |   |   |   |     |
|--|----------------------|-----------|---|---|---|---|-----|
|  |                      | A         | B | C | D | E | F   |
| Clock  | 8, Clock             | 0         | b | 1 | 0 | e | 255 |
| UNIX clock   | 1, Data <sup>a</sup> | 0         | b | 1 | 1 | e | 255 |
| Microseconds clock   | 1, Data <sup>a</sup> | 0         | b | 1 | 2 | e | 255 |
| Minutes clock  | 1, Data <sup>a</sup> | 0         | b | 1 | 3 | e | 255 |
| Hours clock  | 1, Data <sup>a</sup> | 0         | b | 1 | 4 | e | 255 |
| Days clock   | 1, Data <sup>a</sup> | 0         | b | 1 | 5 | e | 255 |
| Weeks clock  | 1, Data <sup>a</sup> | 0         | b | 1 | 6 | e | 255 |
| <sup>a</sup> 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;
- Push* “Script table” objects hold scripts to activate the push operation. Normally every entry in the array of scripts calls the push method of one “Push setup” object instance;
- Load profile control* “Script table” allow to change attributes of “Profile generic” objects e.g. to change the capture period and thus allow extended time control;
- M-Bus profile control* “Script table” allow to change attributes of M-Bus related “Profile generic” objects e.g. to change the capture period and thus allow extended time control;
- Function control* “Script table” objects allow making changes to “Function control” objects;
- Broadcast* “Script table” objects allow standardising utility wide the entry point into regularly needed functionality.

## COSEM Interface Classes

| Script table objects  | IC              | OBIS code |   |    |   |     |     |
|---|-----------------|-----------|---|----|---|-----|-----|
|   |                 | A         | B | C  | D | E   | F   |
| Global meter reset <sup>a</sup> Script table                | 9, Script table | 0         | b | 10 | 0 | 0   | 255 |
| MDI reset / End of billing period <sup>a</sup> Script table |                 | 0         | b | 10 | 0 | 1   | 255 |
| Tarification Script table                                   |                 | 0         | b | 10 | 0 | 100 | 255 |
| Activate test mode <sup>a</sup> Script table                |                 | 0         | b | 10 | 0 | 101 | 255 |
| Activate normal mode <sup>a</sup> Script table              |                 | 0         | b | 10 | 0 | 102 | 255 |
| Set output signals Script table                             |                 | 0         | b | 10 | 0 | 103 | 255 |
| Switch optical test output <sup>b, c</sup> Script table     |                 | 0         | b | 10 | 0 | 104 | 255 |
| Power quality measurement management Script table           |                 | 0         | b | 10 | 0 | 105 | 255 |
| Disconnect control Script table                             |                 | 0         | b | 10 | 0 | 106 | 255 |
| Image activation Script table                               |                 | 0         | b | 10 | 0 | 107 | 255 |
| Push Script table   |                 | 0         | b | 10 | 0 | 108 | 255 |
| Load profile control Script table                           |                 | 0         | b | 10 | 0 | 109 | 255 |
| M-Bus profile control Script table                          |                 | 0         | b | 10 | 0 | 110 | 255 |
| Function control Script table                               |                 | 0         | b | 10 | 0 | 111 | 255 |
|   |                 |           |   |    |   |     |     |
| Broadcast Script table                                      |                 | 0         | b | 10 | 0 | 125 | 255 |

<sup>a</sup> The activation of these scripts is performed by calling the execute() method to the script identifier 1 of the corresponding script object.  
<sup>b</sup> The optical test output is switched to measuring quantity Y and the test mode is activated by calling the execute method of the script table object 0.x.10.0.104.255 using Y as parameter; where Y is given by see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 13. The default value of A is 1 (Electricity).  
 EXAMPLE In the case of electricity meters, A = 1, default, execute (21) switches the test output to display the active power + of phase 1.  
<sup>c</sup> The optical test output is also switched back to its default value when this script is activated.

### 6.2.8 Special days table objects (class\_id = 11)

Instances of the IC “Special days table” – see 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.

Load profile control “Single action schedule” objects activate scripts in Load profile control “Script table” objects and thus allow extended time control.

M-Bus profile control “Single action schedule” objects activate scripts in M-Bus profile control “Script table” objects and thus allow extended time control.

Function control “Single action schedule” objects activate scripts in Function control “Script table” objects.

| Single action schedule objects               | IC                         | OBIS code |   |    |   |   |     |
|--|----------------------------|-----------|---|----|---|---|-----|
|  |                            | A         | B | C  | D | E | F   |
| End of billing period Single action schedule | 22, Single action schedule | 0         | b | 15 | 0 | 0 | 255 |
| Disconnect control Single action schedule    |                            | 0         | b | 15 | 0 | 1 | 255 |
| Image activation Single action schedule      |                            | 0         | b | 15 | 0 | 2 | 255 |
| Output control Single action schedule        |                            | 0         | b | 15 | 0 | 3 | 255 |
| Push Single action schedule                  |                            | 0         | b | 15 | 0 | 4 | 255 |
| Load profile control Single action schedule  |                            | 0         | b | 15 | 0 | 5 | 255 |
| M-Bus profile control Single action schedule |                            | 0         | b | 15 | 0 | 6 | 255 |
| Function control Single action schedule      |                            | 0         | b | 15 | 0 | 7 | 255 |

### 6.2.13 Register monitor objects (class\_id = 21)

Instances of the IC “Register monitor” – see 4.5.6 – control the registerand alarm 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 5.5.1 – 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 Array manager objects (class\_id = 123)

Instances of the IC “Array manager” – see 4.4.11 – allow managing COSEM interface object attributes of type *array*.

| Array manager objects | IC  | OBIS code |   |    |   |   |     |
|-----------------------|-----|-----------|---|----|---|---|-----|
|                       |     | A         | B | C  | D | E | F   |
| Array manager         | 123 | 0         | b | 18 | 0 | e | 255 |

### 6.2.17 Payment metering related objects

Payment accounting can be applied to any commodity.

## COSEM Interface Classes

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 related objects  | IC                           | OBIS code |   |    |        |   |     |
|-----------------------------------|------------------------------|-----------|---|----|--------|---|-----|
|                                   |                              | A         | B | C  | D      | E | F   |
| Account                           | 111, Account                 | 0         | b | 19 | 0..9   | 0 | 255 |
| Credit                            | 112, Credit                  | 0         | b | 19 | 10..19 | e | 255 |
| Charge                            | 113, Charge                  | 0         | b | 19 | 20..29 | e | 255 |
| Token gateway                     | 115, Token gateway           | 0         | b | 19 | 40..49 | e | 255 |
| <i>Configurable limit objects</i> |                              |           |   |    |        |   |     |
| Max credit limit                  | 01, Data                     | 0         | b | 19 | 50..59 | 1 | 255 |
| Max vend limit                    | 01, Data                     | 0         | b | 19 | 50..59 | 2 | 255 |
| IEC 62055-41 attributes           | 116, IEC 62055-41 attributes | 0         | b | 19 | 60..69 | 0 | 255 |

### 6.2.18 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 |

### 6.2.19 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 |

### 6.2.20 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 |

### 6.2.21 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.

## COSEM Interface Classes

| 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)<br>setup<br><br>43, MAC address setup<br><br>1, Data<br><br>7, Profile generic | 0         | b | 23 | 0 | 0 | 255 |
| IEC twisted pair (1) MAC address setup         |   | 0         | b | 23 | 1 | 0 | 255 |
| IEC twisted pair (1) Fatal Error register      |   | 0         | b | 23 | 2 | 0 | 255 |
| IEC 62056-3-1 Short readout                    |   | 0         | b | 23 | 3 | 0 | 255 |
| IEC 62056-3-1 Long readout                     |   | 0         | b | 23 | 3 | 1 | 255 |
| IEC 62056-3-1 Alternate readout profile 0      |   | 0         | b | 23 | 3 | 2 | 255 |
| IEC 62056-3-1 Additional readout profile 1     |   | 0         | b | 23 | 3 | 3 | 255 |
| IEC 62056-3-1 Additional readout profile 2     |   | 0         | b | 23 | 3 | 4 | 255 |
| IEC 62056-3-1 Additional readout profile 7     |   | 0         | b | 23 | 3 | 9 | 255 |

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 |

### 6.2.22 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;

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 585/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

- instances of the IC "DLMS server M-Bus port setup" are used in DLMS 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 <sup>a</sup> | 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        |
| <b>M-Bus image transfer</b>                 | <b>18, Image transfer</b>        | <b>0</b>  | <b>b</b> | <b>24</b> | <b>7</b> | <b>0</b>       | <b>255</b> |
| Wireless Mode Q channel                     | 73, Wireless Mode Q channel      | 0         | b        | 31        | 0        | 0              | 255        |
| DLMS server M-Bus port setup                | 76, DLMS server M-Bus port setup | 0         | b        | 24        | 8        | e <sup>b</sup> | 255        |
| M-Bus diagnostic                            | 77, M-Bus diagnostic             | 0         | b        | 24        | 9        | e <sup>b</sup> | 255        |

<sup>a</sup> "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.

<sup>b</sup> 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.

### 6.2.23 Objects to set up data exchange over the Internet

In this group, the following objects are available:

- Instances of the IC "TCP-UDP setup" – see 4.9.1 – handle all information related to the setup of the TCP and UDP layer of the **TCP-UDP/IP based** 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;

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 586/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

- Instances of the IC “SMTP setup” – see 4.9.6 – handle all information related to the setup of the SMTP service;
- Instances of the IC “CoAP setup” – see 4.9.8 – handle all information related to the setup of the CoAP sublayer of the DLMS/COSEM CoAP transport layer of a DLMS/COSEM CoAP based communication profile and point to the UDP setup object(s) handling the setup of the UDP layer of the DLMS/COSEM CoAP transport layer;
- Instances of the IC “CoAP diagnostic” – see 4.9.9 – handle all diagnostic information related to CoAP sub-layer of the DLMS/COSEM CoAP transport layer;

NOTE The following objects have internet related OBIS codes, although they are not strictly related to use over the internet only.

- Instances of the IC “GSM diagnostic” – see 5.7.9 – 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;
- Instances of the IC “NTP setup” – 4.9.7 – see handle all information related to the setup of the NTP time synchronisation service;
- Instances of the IC “LTE monitoring” – see 4.7.9 – allow monitoring LTE modems.
- Instances of the IC “SCHC-LPWAN setup” – see 4.16.2.1 – handle all information related to the setup of the SCHC-LPWAN layer.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 587/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

Instances of the IC “SCHC-LPWAN diagnostic” – see 4.16.2.2 – handle all diagnostic information related to the SCHC-LPWAN network.

| Objects to set up data exchange over the Internet | IC                             | OBIS code |   |    |    |   |     |
|---|--------------------------------|-----------|---|----|----|---|-----|
|   |                                | A         | B | C  | D  | E | F   |
| TCP-UDP setup                                     | 41, TCP-UDP setup              | 0         | b | 25 | 0  | 0 | 255 |
| IPv4 setup  | 42, IPv4 setup                 | 0         | b | 25 | 1  | 0 | 255 |
| MAC address setup                                 | 43, MAC address setup          | 0         | b | 25 | 2  | 0 | 255 |
| PPP setup   | 44, PPP setup                  | 0         | b | 25 | 3  | 0 | 255 |
| GPRS modem setup                                  | 45, GPRS modem setup           | 0         | b | 25 | 4  | 0 | 255 |
| SMTP setup  | 46, SMTP setup                 | 0         | b | 25 | 5  | 0 | 255 |
| GSM diagnostic                                    | 47, GSM diagnostic             | 0         | b | 25 | 6  | 0 | 255 |
| IPv6 setup  | 48, IPv6 setup                 | 0         | b | 25 | 7  | 0 | 255 |
| <i>Reserved for FTP setup</i>                     |                                |           |   |    |    |   |     |
| Push setup  | 40, Push setup                 | 0         | b | 25 | 9  | 0 | 255 |
| NTP setup   | 100, NTP setup                 | 0         | b | 25 | 10 | 0 | 255 |
| LTE monitoring                                    | 151, LTE monitoring            | 0         | b | 25 | 11 | 0 | 255 |
| SCHC-LPWAN setup                                  | 126, SCHC-LPWAN setup          | 0         | b | 25 | 12 | 0 | 255 |
| SCHC-LPWAN diagnostic                             | 127, SCHC-LPWAN diagnostic     | 0         | b | 25 | 13 | 0 | 255 |
| SCHC-LoRaWAN setup                                | 128, SCHC – LoRaWAN setup      | 0         | b | 25 | 14 | 0 | 255 |
| SCHC-LoRaWAN diagnostic                           | 129, SCHC – LoRaWAN diagnostic | 0         | b | 25 | 15 | 0 | 255 |
| CoAP setup  | 152, CoAP setup                | 0         | b | 25 | 16 | 0 | 255 |
| CoAP diagnostic                                   | 153, CoAP diagnostic           | 0         | b | 25 | 17 | 0 | 255 |

### 6.2.24 Push Setup objects (class\_id = 40)

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 by which the data should be pushed.

| Push Setup | IC             | OBIS code |   |    |   |   |     |
|------------|----------------|-----------|---|----|---|---|-----|
|            |                | A         | B | C  | D | E | F   |
| Push setup | 40, Push setup | 0         | b | 25 | 9 | 0 | 255 |

### 6.2.25 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.

## COSEM Interface Classes

- 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 |

### 6.2.26 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 |

### 6.2.27 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;

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 589/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

- 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 PRIME NB OFDM PLC | IC  | OBIS code |   |    |   |   |     |
|---|---|-----------|---|----|---|---|-----|
|   |   | A         | B | C  | D | E | F   |
| 61334-4-32 LLC SSCS setup                         | 80, 61334-4-32 LLC SSCS setup                         | 0         | b | 28 | 0 | 0 | 255 |
| PRIME NB OFDM PLC Physical layer counters         | 81, PRIME NB OFDM PLC Physical layer counters         | 0         | b | 28 | 1 | 0 | 255 |
| PRIME NB OFDM PLC MAC setup                       | 82, PRIME NB OFDM PLC MAC setup                       | 0         | b | 28 | 2 | 0 | 255 |
| PRIME NB OFDM PLC MAC functional parameters       | 83, PRIME NB OFDM PLC MAC functional parameters       | 0         | b | 28 | 3 | 0 | 255 |
| PRIME NB OFDM PLC MAC counters                    | 84, PRIME NB OFDM PLC MAC counters                    | 0         | b | 28 | 4 | 0 | 255 |
| PRIME NB OFDM PLC MAC network administration data | 85, PRIME NB OFDM PLC MAC network administration data | 0         | b | 28 | 5 | 0 | 255 |
| PRIME NB OFDM PLC MAC address setup               | 43, MAC address setup                                 | 0         | b | 28 | 6 | 0 | 255 |
| PRIME NB OFDM PLC Application identification      | 86, PRIME NB OFDM PLC Application identification      | 0         | b | 28 | 7 | 0 | 255 |

### 6.2.28 Objects for data exchange using narrow-band OFDM PLC for G3-PLC networks

For setting up and managing data exchange using G3-PLC profile, one instance of each following classes shall be implemented for each interface:

- an instance of the IC “G3-PLC MAC layer counters” – see 4.13.3 – to store counters related to the MAC layer exchanges;
- an instance of the IC “G3-PLC MAC setup” – see 4.13.4 – to hold the necessary parameters to set up the G3-PLC MAC IEEE 802.15.4: 2006 layer;
- an instance of the IC “G3-PLC 6LoWPAN adaptation layer setup” – see 4.13.5– to hold the necessary parameters to set up the Adaptation layer.
- An instance of the IC “G3-PLC Hybrid RF MAC layer counters (class\_id = 160, version = 0)” – see 4.13.6– to store counters related to the MAC layer exchanges

## COSEM Interface Classes

- An instance of the IC “G3-PLC Hybrid RF MAC setup (class\_id = 161, version = 0)” – see 4.13.7– to store the necessary additional parameters to set up and manage the G3-PLC Hybrid PLC & RF IEEE 802.15.4:2015 RF MAC sub-layer.
- An instance of the IC “G3-PLC Hybrid 6LoWPAN adaptation layer setup (class\_id = 162, version = 0)” – see 4.13.8– to store the necessary additional parameters to set up and manage the G3-PLC Hybrid PLC & RF 6LoWPAN 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 |
| G3-PLC Hybrid RF MAC layer counters          | 160, G3-PLC Hybrid RF MAC layer counters          | 0         | b | 29 | 2 | 0 | 255 |
| G3-PLC Hybrid RF MAC setup                   | 161, G3-PLC Hybrid RF MAC setup                   | 0         | b | 29 | 2 | 0 | 255 |
| G3-PLC Hybrid 6LoWPAN adaptation layer setup | 162, G3-PLC Hybrid 6LoWPAN adaptation layer setup | 0         | b | 29 | 2 | 0 | 255 |

### 6.2.29 ZigBee® setup objects

The following objects are available for setting up and managing a ZigBee® network; see also 4.15.

| Objects set up and manage ZigBee® networks | IC                                 | OBIS code |   |    |   |   |     |
|--|------------------------------------|-----------|---|----|---|---|-----|
|  |                                    | A         | B | C  | D | E | F   |
| ZigBee® SAS startup                        | 101, ZigBee® SAS Startup           | 0         | b | 30 | 0 | e | 255 |
| ZigBee® SAS join                           | 102, ZigBee® SAS join              | 0         | b | 30 | 1 | e | 255 |
| ZigBee® SAS APS fragmentation              | 103, ZigBee® SAS APS fragmentation | 0         | b | 30 | 2 | e | 255 |
| ZigBee® network control                    | 104, ZigBee® network control       | 0         | b | 30 | 3 | e | 255 |
| ZigBee® tunnel setup                       | 105, ZigBee® tunnel setup          | 0         | b | 30 | 4 | e | 255 |

### 6.2.30 Objects for setting up and managing data exchange using ISO/IEC 14908 PLC networks

For setting up and managing data exchange using ISO/IEC 14908 PLC networks at least one instance of each of the following interface classes shall be implemented, see also 4.16

| Objects to set up and manage ISO/IEC 14908 PLC netwrks | IC                                 | OBIS identification |   |    |   |   |     |
|--|------------------------------------|---------------------|---|----|---|---|-----|
|  |                                    | A                   | B | C  | D | E | F   |
| ISO/IEC 14908 identification                           | 130, ISO/IEC 14908 identification  | 0                   | b | 32 | 0 | 0 | 255 |
| ISO/IEC 14908 protocol setup                           | 131, ISO/IEC 14908 protocol setup  | 0                   | b | 32 | 1 | 0 | 255 |
| ISO/IEC 14908 protocol status                          | 132, ISO/IEC 14908 protocol status | 0                   | b | 32 | 2 | 0 | 255 |
| ISO/IEC 14908 diagnostic                               | 133, ISO/IEC 14908 diagnostic      | 0                   | b | 32 | 3 | 0 | 255 |

### 6.2.31 Objects for data exchange using HS-PLC ISO/IEC 12139-1 ISO/EC 12139-1 networks

For setting up and managing data exchange using HS-PLC ISO/IEC 12139-1 networks one instance of each following classes shall be implemented for each interface:

- an instance of the IC “HS-PLC ISO/IEC 12139-1 MAC setup” – see 4.14.2 – holds the necessary parameters for setting up the MAC layer;
- an instance of the IC “HS-PLC ISO/IEC 12139-1 CPAS setup” – see 4.14.3 – holds the necessary parameters for setting up the CPAS;
- an instance of the IC “HS-PLC ISO/IEC 12139-1 IP SSAS setup” – see 4.14.4 – holds the necessary parameters for setting up the IP SSAS;
- an instance of the IC “HS-PLC ISO/IEC 12139-1 HDLC SSAS setup” – see 4.14.5 – holds the necessary parameters for setting up the HDLC SSAS.

| Objects for data exchange using HS-PLC ISO/IEC 12139-1 | IC  | OBIS code |   |    |   |   |     |
|--|---|-----------|---|----|---|---|-----|
|  |   | A         | B | C  | D | E | F   |
| HS-PLC ISO/IEC 12139-1 MAC setup                       | 140, HS-PLC ISO/IEC 12139-1 MAC setup       | 0         | b | 33 | 0 | 0 | 255 |
| HS-PLC ISO/IEC 12139-1 CPAS setup                      | 141, HS-PLC ISO/IEC 12139-1 CPAS setup      | 0         | b | 33 | 1 | 0 | 255 |
| HS-PLC ISO/IEC 12139-1 IP SSAS setup                   | 142, HS-PLC ISO/IEC 12139-1 IP SSAS setup   | 0         | b | 33 | 2 | 0 | 255 |
| HS-PLC ISO/IEC 12139-1 HDLC SSAS setup                 | 143, HS-PLC ISO/IEC 12139-1 HDLC SSAS setup | 0         | b | 33 | 3 | 0 | 255 |

### 6.2.32 Objects for data exchange using Wi-SUN networks

For setting up and managing data exchange using Wi-SUN networks one instance of each following classes shall be implemented for each interface:

- An instance of the IC “IPv6 setup” – see 4.9.3 – handle all information related to the setup of the IPv6 layer of the Internet based communication profile(s) and point to the data link layer setup object(s) handling the setup of the data link layer on which the IP connections is (are) used;
- An instance of the IC “Wi-SUN setup” – see 4.18.1 – handle all information related to the setup of the SMTP service.
- An instance of the IC “Interface diagnostic” – see 4.18.2 – handle all diagnostic information related to the network.
- An instance of the IC “RPL diagnostic” – see 4.18.3 – handle all diagnostic information related to RPL.
- An instance of the IC “MPL diagnostic” – see 4.18.4 – handle all diagnostic information related to MPL.

| Objects to set up data exchange over the Wi-SUN FAN | IC                    | OBIS code |   |    |   |   |     |
|---|-----------------------|-----------|---|----|---|---|-----|
|   |                       | A         | B | C  | D | E | F   |
| Wi-SUN setup  | 95, Wi-SUN setup      | 0         | b | 34 | 0 | 0 | 255 |
| Wi-SUN diagnostic                                   | 96, Wi-SUN diagnostic | 0         | b | 34 | 1 | 0 | 255 |
| RPL diagnostic                                      | 97, RPL diagnostic    | 0         | b | 34 | 2 | 0 | 255 |
| MPL diagnostic                                      | 98, MPL diagnostic    | 0         | b | 34 | 3 | 0 | 255 |

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 592/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

### 6.2.33 Association objects (class\_id = 12, 15)

A series of Association SN / LN objects – see 4.4.3, 4.4.4 – are available to model application associations between a DLMS client and server.

| Association objects            | IC                                       | OBIS code |   |    |   |   |     |
|--------------------------------|--|-----------|---|----|---|---|-----|
|                                |  | A         | B | C  | D | E | F   |
| Current association            | 12, Association SN<br>15, Association LN | 0         | 0 | 40 | 0 | 0 | 255 |
| Association, instance 1        |  | 0         | 0 | 40 | 0 | 1 | 255 |
| .....                          |  |           |   |    |   |   |     |
| Association, instance <i>n</i> |  | 0         | 0 | 40 | 0 | n | 255 |

### 6.2.34 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 |

### 6.2.35 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 <sup>a</sup> | 0         | 0 | 42 | 0 | 0 | 255 |

<sup>a</sup> If the IC “Data” is not available, “Register” (with scaler = 0, unit = 255) may be used.

### 6.2.36 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 5.4.2 and 5.4.3. Value group E numbers the instances.

| Security setup objects | IC                 | OBIS code |   |    |   |   |     |
|------------------------|--------------------|-----------|---|----|---|---|-----|
|                        |                    | A         | B | C  | D | E | F   |
| Security setup         | 64, Security setup | 0         | 0 | 43 | 0 | e | 255 |

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.

## COSEM Interface Classes

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 <sup>a</sup> | 0         | b | 43 | 1 | e | 255 |
| NOTE In earlier versions of this standard, these objects were called Frame counter objects.           |                      |           |   |    |   |   |     |
| <sup>a</sup> 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 |

### 6.2.37 Image transfer objects (class\_id = 18)

Instances of the IC “Image transfer” – see 4.4.6 – control the Image transfer process.

| Image transfer related objects | IC                 | OBIS code |   |    |   |   |     |
|--------------------------------|--------------------|-----------|---|----|---|---|-----|
|                                |                    | A         | B | C  | D | E | F   |
| Image transfer                 | 18, Image transfer | 0         | 0 | 44 | 0 | e | 255 |

### 6.2.38 Function control objects (class\_id = 122)

Instances of the IC “Function control” – see 4.4.10 – allow enabling and disabling functions in the server.

| Function control related objects | IC                    | OBIS code |   |    |   |   |     |
|----------------------------------|-----------------------|-----------|---|----|---|---|-----|
|                                  |                       | A         | B | C  | D | E | F   |
| Function control                 | 122, Function control | 0         | 0 | 44 | 1 | e | 255 |

### 6.2.39 Communication port protection objects (class\_id = 124)

Instances of the “Communication port protection” IC – see 4.4.12 – control the communication port protection mechanism.

| Communication port protection objects | IC  | OBIS code |   |    |   |   |     |
|---------------------------------------|-----|-----------|---|----|---|---|-----|
|                                       |     | A         | B | C  | D | E | F   |
| Communication port protection objects | 124 | 0         | b | 44 | 2 | e | 255 |

#### 6.2.40 Utility table objects (class\_id = 26)

Instances of the IC “Utility tables” – see 4.3.7 – allow representing ANSI utility tables. The Utility table IDs are mapped to OBIS codes as follows:

- value group A: use value of 0 to specify abstract object;
- value group B: instance of table set;
- value group C: use value 65 – signifies utility tables specific definitions;
- value group D: table group selector;
- value group E: table number within group;
- value group F: use value 0xFF for data of current billing period.

| 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 |

#### 6.2.41 Compact data objects (class\_id = 62)

“Compact data” objects – see 4.3.10 – store data and metadata separated, thus they allow reducing overhead.

| Compact data objects | IC               | OBIS code |   |    |   |   |     |
|----------------------|------------------|-----------|---|----|---|---|-----|
|                      |                  | A         | B | C  | D | E | F   |
| Compact data         | 62, Compact data | 0         | b | 66 | 0 | e | 255 |

#### 6.2.42 Device ID objects

A series of objects are used to hold ID numbers of the device. These ID numbers can be defined by the manufacturer (e.g. manufacturing number) or by the user.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 595/668 |
|-----------------------|------------|-----------------------------|---------|

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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8.

| Device ID objects                              | IC                   | OBIS code |   |    |   |       |     |
|--|----------------------|-----------|---|----|---|-------|-----|
|  |                      | A         | B | C  | D | E     | F   |
| Device ID 1...10 object (manufacturing number) | 1, Data <sup>a</sup> | 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 |

<sup>a</sup> If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

#### 6.2.43 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 <sup>a</sup> | 0         | b | 96 | 1 | 10 | 255 |

<sup>a</sup> If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

#### 6.2.44 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 1, Data <sup>a</sup> | 0         | b | 96 | 2 | e | 255 |

<sup>a</sup> If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

#### 6.2.45 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 4.3.9, which holds both the status word and the mapping of its bits to the reference table. If there are several I/O control status objects used, it is allowed to combine them into an instance of the IC "Profile generic" or "Register table", using the OBIS code of the global state of I/O control signals object. See also DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8.

## COSEM Interface Classes

| I/O control signal objects                                       | IC                                       | OBIS code |   |    |   |       |     |
|--|--|-----------|---|----|---|-------|-----|
|  |  | A         | B | C  | D | E     | F   |
| I/O control signal objects, contents manufacturer specific       | 1, Data <sup>a</sup>                     | 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 |

<sup>a</sup> If the IC “Data” is not available, “Register” or “Extended register” (with scaler = 0, unit = 255) may be used.

### 6.2.46 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 6.2.22.

| Disconnect control objects | IC                     | OBIS code |   |    |   |    |     |
|----------------------------|------------------------|-----------|---|----|---|----|-----|
|                            |                        | A         | B | C  | D | E  | F   |
| Disconnect control         | 70, Disconnect control | 0         | b | 96 | 3 | 10 | 255 |

### 6.2.47 Arbitrator objects (class\_id = 68)

Instances of the IC “Arbitrator” – see 4.5.12 – are used

| Arbitrator Objects         | IC             | OBIS code |   |    |   |             |     |
|----------------------------|----------------|-----------|---|----|---|-------------|-----|
|                            |                | A         | B | C  | D | E           | F   |
| General-purpose Arbitrator | 68, Arbitrator | 0         | b | 96 | 3 | 20...<br>29 | 255 |

### 6.2.48 Status of internal control signals objects

A series of objects are available to hold the status of internal control signals.

The status carries binary information from a bitmap, and it shall be held by the *value* attribute of a “Data” object, with data type *bit-string*, *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-unsigned* or *octet-string*. Alternatively, the status is held by a “Status mapping” object, see 4.3.9, which holds both the status word and the mapping of its bits to the reference table. If there are several status of internal control signals objects used, it is allowed to combine them into an instance of the IC “Profile generic” or “Register table”, using the OBIS code of the global “Internal control signals” object. See also DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8.

| Internal control signals objects                               | IC                                       | OBIS code |   |    |   |       |     |
|--|--|-----------|---|----|---|-------|-----|
|  |  | A         | B | C  | D | E     | F   |
| Internal control signals, contents manufacturer specific       | 1, Data <sup>a</sup>                     | 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 |

|  |
|--|
| <sup>a</sup> If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used. |
|--|

### 6.2.49 Internal operating status objects

A series of objects are available to hold internal operating statuses.

The status carries binary information from a bitmap, and it shall be held by the value attribute of a "Data" object, with data type *bit-string*, *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-unsigned* or *octet-string*. Alternatively, the status is held by a "Status mapping" object, see 4.3.9, which holds both the status word and the mapping of its bits to the reference table. If there are several status of internal control signals objects used, it is allowed to combine them into an instance of the IC "Profile generic" or "Register table", using the OBIS code of the global "Internal operating status" object. See also DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8.

| Internal operating status objects                                       | IC                                       | OBIS code |   |    |   |       |     |
|---|--|-----------|---|----|---|-------|-----|
|   |  | A         | B | C  | D | E     | F   |
| Internal operating status objects, contents manufacturer specific       | 1, Data <sup>a</sup>                     | 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 |

<sup>a</sup> 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.

### 6.2.50 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 1, Data, 3, Register or 4, Extended register | 0         | b | 96 | 6 | 0...6 | 255 |

### 6.2.51 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 23.

### 6.2.52 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 1, Data, 3, Register or 4, Extended register | 0         | b | 96 | 8 | 0...63 | 255 |

### 6.2.53 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 3, Register or 4, Extended register | 0         | b | 96 | 9 | 0...2 | 255 |

### 6.2.54 Status register objects

A series of objects are available to hold statuses that can be captured in load profiles. See also see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8.

| Status register objects                         | IC                   | OBIS code |   |    |    |        |     |
|---|----------------------|-----------|---|----|----|--------|-----|
|   |                      | A         | B | C  | D  | E      | F   |
| Status register, contents manufacturer specific | 1, Data <sup>a</sup> | 0         | b | 96 | 10 | 1...10 | 255 |

## COSEM Interface Classes

|  |                    |   |          |    |    |            |     |
|--|--------------------|---|----------|----|----|------------|-----|
| Status register, contents mapped to reference table  | 63, Status mapping | 0 | <i>b</i> | 96 | 10 | 1...<br>10 | 255 |
| <sup>a</sup> 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.

### 6.2.55 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 6.2.66.

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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 1, Data, 3, Register, or 4, Extended register | 0         | <i>b</i> | 96 | 11 | 0...99 | 255 |

Events may also set flags in error registers and alarm registers. See also 6.2.64.

### 6.2.56 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 1, Data, 3, Register, or 4, Extended register | 0         | <i>b</i> | 96 | 12 | 0...6 | 255 |

### 6.2.57 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 1, Data, 3, Register, or 4, Extended register | 0         | <i>b</i> | 96 | 13 | 0, 1 | 255 |

### 6.2.58 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 1, Data, 3, Register, or 4, Extended register | 0         | b | 96 | 14 | 0...15 | 255 |

### 6.2.59 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 1, Data, 3, Register, or 4, Extended register | 0         | b | 96 | 15 | 0...99 | 255 |

### 6.2.60 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 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 1, Data, 3, Register, or 4, Extended register | 0         | b | 96 | 16 | 0...9 | 255 |

### 6.2.61 Profile entry counter objects

A series of objects are available to keep track of the total number of entries captured in a “Profile generic” object. These objects are instances of the IC “Data”, Register” or “Extended register”. The data type shall be *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned* with appropriate scaler and unit. If the *capture\_objects* attribute of a “Profile generic” object contains a reference to a “Profile entry counter” object, then the counter is monotonically increased by one and captured together with the other attribute values when the *capture* method is invoked or when auto capture takes place.

If a “Profile entry counter” object reaches its maximum value, it will wrap-around and the count will re-start at zero.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 601/668 |
|-----------------------|------------|-----------------------------|---------|

| Profile entry counter objects   | IC  | OBIS code |   |    |    |         |     |
|---|---|-----------|---|----|----|---------|-----|
|   |   | A         | B | C  | D  | E       | F   |
| For names and OBIS codes see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 1, Data, 3, Register, or 4, Extended register | 0         | b | 96 | 17 | 0...127 | 255 |

### 6.2.62 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8. | 1, Data, 3, Register, or 4, Extended register | 0         | b | 96 | 20 | e | 255 |

### 6.2.63 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 6.2.55. 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 DLMS UA 1000-1 Ed 15 Part 1:2021, 6.2 and 7.5.2.

| Error register objects       | IC                   | OBIS code |   |    |    |       |     |
|------------------------------|----------------------|-----------|---|----|----|-------|-----|
|                              |                      | A         | B | C  | D  | E     | F   |
| Error register 1...10 object | 1, Data <sup>a</sup> | 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 |

<sup>a</sup> If the IC "Data" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

### 6.2.64 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 <sup>a</sup> | 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 <sup>a</sup> | 0         | b | 97 | 98 | 10...19 | 255 |
| Alarm descriptor objects 1...10                           | 1, Data <sup>a</sup> | 0         | b | 97 | 98 | 20...29 | 255 |

<sup>a</sup> If the IC "Dtaata" is not available, "Register" or "Extended register" (with scaler = 0, unit = 255) may be used.

### 6.2.65 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 DLMS UA 1000-1 Ed 15 Part 1:2021, 6.3 and 7.5.3                  | 7, Profile generic | 0         | b | 98 | d | e | 255 <sup>a</sup> |
| <sup>a</sup> F = 255 means a wildcard here. See DLMS UA 1000-1 Ed 15 Part 1:2021, Clause A.3. |                    |           |   |    |   |   |                  |

### 6.2.66 Event log objects (class\_id 7)

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 DLMS UA 1000-1 Ed 15 Part 1:2021, 6.5 and 7.5.4.

| Event log objects   | IC                 | OBIS code |   |    |    |   |                  |
|---|--------------------|-----------|---|----|----|---|------------------|
|   |                    | A         | B | C  | D  | E | F                |
| Event log   | 7, Profile generic | a         | b | 99 | 98 | e | 255 <sup>a</sup> |
| <sup>a</sup> F = 255 means a wildcard here. See DLMS UA 1000-1 Ed 15 Part 1:2021, Clause A.3.   |                    |           |   |    |    |   |                  |
| NOTE 1 Event logs may capture for example the time of occurrence of the event, the event code and other relevant data.  |                    |           |   |    |    |   |                  |
| NOTE 2 Project specific companion specifications may specify a more precise meaning of the instances of the different event logs, i.e. the data captured and the number of events captured. |                    |           |   |    |    |   |                  |

### 6.2.67 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 DLMS UA 1000-1 Ed 15 Part 1:2021, 5.3.2.

| Inactive objects | IC  | OBIS code |   |     |   |   |     |
|------------------|-----|-----------|---|-----|---|---|-----|
|                  |     | A         | B | C   | D | E | F   |
| Inactive objects | Any | 0         | b | 127 | 0 | e | 255 |

### 6.3 AC 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 DLMS UA 1000-1 Ed 15 Part 1:2021, 7.2.1 – are modelled as shown in Table 57.

**Table 57 – 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.<br>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).<br>For display purposes: Instances of IC "Register" or "Demand register". The logical name is the OBIS code of current average (D = 4, 14 or 24) or last average (D = 5, 15 or 25) as appropriate. |
| instantaneous values                 | Instances of IC "Register".  |
| time integral values                 | Instances of IC "Register" or "Extended register".   |
| occurrence counters                  | Instances of IC "Data" or "Register".  |
| contracted values                    | Instances of IC "Register" or "Extended register".   |
| Under/Over limit thresholds          | Instances of IC "Register" or "Extended register".   |
| Over/Under limit occurrence counters | Instances of IC "Register" or "Extended register".   |
| Under/Over limit durations           | Instances of IC "Register" or "Extended register".   |
| Over/Under limit magnitudes          | Instances of IC "Register" or "Extended register".   |

#### 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 20.

| Electricity ID objects       | IC                   | OBIS code |   |   |   |       |     |
|------------------------------|----------------------|-----------|---|---|---|-------|-----|
|                              |                      | A         | B | C | D | E     | F   |
| Electricity ID 1...10 object | 1, Data <sup>a</sup> | 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 |

<sup>a</sup> 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 20. | 1, Data <sup>a</sup> | 1         | b | 0 | 1 | e | 255 |

<sup>a</sup> 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*.

|                               |       |     |                             |
|-------------------------------|-------|-----|-----------------------------|
| <b>Synchronization method</b> | 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 20.

| Electricity related general purpose objects                         | IC   | OBIS code |   |   |    |   |     |
|---|--|-----------|---|---|----|---|-----|
|   |  | A         | B | C | D  | E | F   |
| Program entries   | 1, Data <sup>a</sup>                           | 1         | b | 0 | 2  | e | 255 |
| Output pulse values or constants                                    | 1, Data  | 1         | b | 0 | 3  | e | 255 |
| Reading factor and CT/VT ratio                                      | 3, Register<br>4, Extended Register            | 1         | b | 0 | 4  | e | 255 |
| Nominal values  | 3, Register<br>4, Extended Register            | 1         | b | 0 | 6  | e | 255 |
| Input pulse values or constants                                     | 1, Data<br>3, Register<br>4, Extended Register | 1         | b | 0 | 7  | e | 255 |
| Measurement period- / recording interval- / billing period duration | 1, Data<br>3, Register                         | 1         | b | 0 | 8  | e | 255 |
| Time entries  | 4, Extended Register                           | 1         | b | 0 | 9  | e | 255 |
| Transformer and line loss coefficients                              | 3, Register<br>4, Extended Register            | 1         | b | 0 | 10 | e | 255 |

<sup>a</sup> 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 <sup>a</sup> | 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 |

<sup>a</sup> In cases where the IC “Data” is not available, “Register” (with scaler = 0, unit = 255) may be used.

The enumerated values are specified in Table 58:

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 607/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

**Table 58 – Measuring algorithms – enumerated values**

| <b>Measuring algorithm for active power and energy</b>     |   |
|--|---|
| (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   |
| <b>Measuring algorithm for reactive power and energy</b>   |   |
| (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   |
| <b>Measurement algorithm for apparent power and energy</b> |   |
| (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)   | $S = \sum \sqrt{P_i^2 + Q_i^2}$ , 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)   | $S = \sum \sqrt{P_i^2 + Q_i^2}$ , 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)   | $S = \sum \sqrt{P_i^2 + Q_i^2}$ , 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                                   |
| <b>Measurement algorithm for power factor calculation</b>  |   |
| (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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 20.

| Metering point ID objects                      | IC                   | OBIS code |   |    |   |       |     |
|--|----------------------|-----------|---|----|---|-------|-----|
|  |                      | A         | B | C  | D | E     | F   |
| Metering point ID 1...10 (electricity related) | 1, Data <sup>a</sup> | 1         | b | 96 | 1 | 0...9 | 255 |
| Metering point ID-s object                     | 7, Profile generic   | 1         | b | 96 | 1 | 255   | 255 |
| Metering point ID-s object                     | 61, Register table   | 1         | b | 96 | 1 | 255   | 255 |

<sup>a</sup> If the IC “Data” is not available, “Register” (with scaler = 0, unit = 255) may be used.

### 6.3.7 Electricity related status objects

A number of electricity related objects are available to hold information about the internal operating status, the starting of the meter and the status of voltage and current circuits.

The status is held by the *value* attribute of a “Data” object, with data type *bit-string*, *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-unsigned* or *octet-string*.

Alternatively, the status is held by a “Status mapping” object, which holds both the status word and the mapping of its bits to the reference table.

If there are several electricity related internal operating status objects used, it is allowed to combine them into an instance of the IC “Profile generic” or “Register table”, using the OBIS code of the global internal operating status. For detailed OBIS codes, see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 20.

| Electricity related status objects   | IC                   | OBIS code |   |    |    |       |     |
|--|----------------------|-----------|---|----|----|-------|-----|
|  |                      | A         | B | C  | D  | E     | F   |
| Internal operating status signals, electricity related, contents manufacturer specific     | 1, Data <sup>a</sup> | 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 <sup>a</sup> | 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 |

<sup>a</sup> 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 DLMS UA 1000-1 Ed 15 Part 1:2021, 7.5.3.

| List objects – Electricity  | IC                 | OBIS code |   |    |   |   |                  |
|---|--------------------|-----------|---|----|---|---|------------------|
|   |                    | A         | B | C  | D | E | F                |
| For names and OBIS codes see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 22.                      | 7, Profile generic | 1         | b | 98 | d | e | 255 <sup>a</sup> |
| <sup>a</sup> F = 255 means a wildcard here. See DLMS UA 1000-1 Ed 15 Part 1:2021, Clause A.3. |                    |           |   |    |   |   |                  |

### 6.3.9 Threshold values

A number of objects are available for representing thresholds for instantaneous quantities. The thresholds may be “under limit”, “over limit”, “missing” and “time thresholds”. Time thresholds are used to detect “under limit”, “over limit” and “missing” conditions.

Objects are also available to represent the number of occurrences when these thresholds are exceeded, the duration of such events and the magnitude of the quantity during such events.

These values are represented by instances of IC “Data”, “Register” or “Extended register”.

All these quantities may be related to tariffs.

As defined in DLMS UA 1000-1 Ed 15 Part 1:2021, 7.4.2, value group F may be used to identify multiple thresholds.

For OBIS codes, see Table 59 below and DLMS UA 1000-1 Ed 15 Part 1:2021, Table 14.

**Table 59 – Threshold objects, electricity**

| Threshold objects                          | IC   | OBIS code |   |  |   |        |                |
|--|--|-----------|---|--|---|--------|----------------|
|  |  | A         | B | C  | D   | E      | F              |
| Threshold objects for instantaneous values | 1, Data,<br>3, Register,<br>4, Extended register | 1         | b | 1...10,<br>13, 14,<br>16...20,<br>21...30,<br>33, 34,<br>36...40,<br>41...50,<br>53, 54,<br>56...60,<br>61...70,<br>73, 74,<br>76...80,<br>82,<br>84...89<br>100...1<br>07 | 31...34,<br>35...38,<br>39...42,<br>43...45 | 0...63 | 0...99.<br>255 |

|  |  |   |   |  |  |                           |  |
|--|--|---|---|--|--|---------------------------|--|
| Threshold objects for harmonics of voltage, current and active power |  | 1 | b | 11, 12,<br>15, 31,<br>32, 35,<br>51, 52,<br>55, 71,<br>72, 75,<br>90...92<br><br>124...1<br>26 |  | 0...120,<br>124...<br>127 |  |
|--|--|---|---|--|--|---------------------------|--|

For monitoring the supply voltage, a more sophisticated functionality is also available, that allows counting the number of occurrences classified by the duration of the event and the depth of the voltage dip. For OBIS codes, see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 8.

### 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.

See Table 60.

**Table 60 – 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,<br>35,<br>39           | 0-63                  | 0-99,<br>255 |  |  |  |
|   |                      | 1         | b | c2 |                            | 0-120,<br>124-<br>127 |              |  |  |  |
|   |                      | 1         | b | c1 | 4, 5,<br>14, 15,<br>24, 25 | 0-63                  |              |  |  |  |
|   |                      | 1         | b | c2 |                            | 0-120,<br>124-<br>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, 100-107. |                      |           |   |    |                            |                       |              |  |  |  |
| c2 = 11, 12, 15, 31, 32, 35, 51, 52, 55, 71, 72, 75, 90-92, 124-126.  |                      |           |   |    |                            |                       |              |  |  |  |

For the use of value group D, see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 14.

For the use of value group E, see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 15 and Table 16.

For the use of value group F, see DLMS UA 1000-1 Ed 15 Part 1:2021, 7.4.2.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 611/668 |
|-----------------------|------------|-----------------------------|---------|

## 6.4 DC electricity related COSEM objects

### 6.4.1 Value group D definitions

The different ways of processing measurement values as defined by value group D are modelled as shown in Table 61, see also 7.5.2.1.

**Table 61 – 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.<br>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).<br>For display purposes: Instances of IC "Register" or "Demand register". The logical name is the OBIS code of current average (D = 4, 14 or 24) or last average (D= 5, 15 or 25) as appropriate. |
| instantaneous values                 | Instances of IC "Register".  |
| time integral values                 | Instances of IC "Register" or "Extended register".   |
| occurrence counters                  | Instances of IC "Data" or "Register".  |
| contracted values                    | Instances of IC "Register" or "Extended register".   |
| Under/Over limit thresholds          | Instances of IC "Register" or "Extended register".   |
| Over/Under limit occurrence counters | Instances of IC "Register" or "Extended register".   |
| Under/Over limit durations           | Instances of IC "Register" or "Extended register".   |
| Over/Under limit magnitudes          | Instances of IC "Register" or "Extended register".   |

### 6.4.2 ID numbers – DC electricity

The different DC 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 DC electricity ID "Data" objects, the capture period is 1 to have only actual values, the sort method is FIFO, the profile entries are limited to 1. Alternatively, a "Register table" object can be used. See also DLMS UA 1000-1 Ed 15 Part 1:2021

| ID objects – DC electricity     | IC                 | OBIS code |   |   |   |       |     |
|---------------------------------|--------------------|-----------|---|---|---|-------|-----|
|                                 |                    | A         | B | C | D | E     | F   |
| DC electricity ID 1...10 object | 1, Data            | 2         | b | 0 | 0 | 0...9 | 255 |
| DC electricity ID-s object      | 7, Profile generic | 2         | b | 0 | 0 | 255   | 255 |
| DC electricity ID-s object      | 61, Register table | 2         | b | 0 | 0 | 255   | 255 |

### 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                         | IC      | OBIS code |   |   |   |   |     |
|---|---------|-----------|---|---|---|---|-----|
|   |         | A         | B | C | D | E | F   |
| For item names and OBIS codes see<br>DLMS UA 1000-1 Ed 15 Part 1:2021 | 1, Data | 2         | b | 0 | 1 | e | 255 |

### 6.4.4 Other DC 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, sensor<sup>2</sup> 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*, *double-long-unsigned* or *long64-unsigned* can also be used where appropriate.

---

<sup>2</sup> This refers to Transformers within A.C .electrical parameters, and sensor has been used within this text, noting that Low Power Instrument Transformers are referenced within IEC 62053-41 for D.C. electricity meters.

## COSEM Interface Classes

The *Clock synchronization method* shall be represented by an instance of an IC “Data” with data type enum.

| <b>Synchronization method</b> | 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 DLMS UA 1000-1 Ed 15 Part 1:2021.

| <b>DC electricity related general purpose objects</b>               | <b>IC</b>            | <b>OBIS code</b> |          |          |          |          |          |
|---|----------------------|------------------|----------|----------|----------|----------|----------|
|   |                      | <b>A</b>         | <b>B</b> | <b>C</b> | <b>D</b> | <b>E</b> | <b>F</b> |
| Program entries   | 1, Data              | 2                | b        | 0        | 2        | e        | 255      |
| Output pulse values or constants                                    | 1, Data              | 2                | b        | 0        | 3        | e        | 255      |
| Reading factor and sensor ratio                                     | 3, Register          | 2                | b        | 0        | 4        | e        | 255      |
|   | 4, Extended Register |                  |          |          |          |          |          |
| Nominal values  | 3, Register          | 2                | b        | 0        | 6        | e        | 255      |
|   | 4, Extended Register |                  |          |          |          |          |          |
| Input pulse values or constants                                     | 1, Data              | 2                | b        | 0        | 7        | e        | 255      |
|   | 3, Register          |                  |          |          |          |          |          |
|   | 4, Extended Register |                  |          |          |          |          |          |
| Measurement period- / recording interval- / billing period duration | 1, Data              | 2                | b        | 0        | 8        | e        | 255      |
|   | 3, Register          |                  |          |          |          |          |          |
| Time entries  | 4, Extended Register | 2                | b        | 0        | 9        | e        | 255      |
|   |                      |                  |          |          |          |          |          |
| Line loss coefficients  | 3, Register          | 2                | b        | 0        | 10       | e        | 255      |
|   | 4, Extended Register |                  |          |          |          |          |          |

### 6.4.5 Measurement algorithm

These values are represented by instances of the IC “Data”, with data type enum.

| <b>Measurement algorithm objects</b> | <b>IC</b> | <b>OBIS code</b> |          |          |          |          |          |
|--------------------------------------|-----------|------------------|----------|----------|----------|----------|----------|
|                                      |           | <b>A</b>         | <b>B</b> | <b>C</b> | <b>D</b> | <b>E</b> | <b>F</b> |
| Measuring algorithm for power        | 1, Data   | 2                | b        | 0        | 11       | 1        | 255      |
| Measurement algorithm for energy     |           | 2                | b        | 0        | 11       | 2        | 255      |

The enumerated values are specified in Table 62:

**Table 62 – Measuring algorithms – enumerated values**

| <b>Measuring algorithm for power and energy</b> |   |
|---|---|
| (0)   | not specified                                   |
| (1)   | reserved  |
| (2)   | reserved  |
| (3)   | only the DC part of voltage and current is used |
| (4)   | reserved  |

**6.4.6 Metering point ID (DC electricity related)**

A series of objects are available to hold DC 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 DC 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 DLMS UA 1000-1 Ed 15 Part 1:2021.

| <b>Metering point ID objects</b>                  | <b>IC</b>          | <b>OBIS code</b> |          |          |          |          |          |
|---|--------------------|------------------|----------|----------|----------|----------|----------|
|   |                    | <b>A</b>         | <b>B</b> | <b>C</b> | <b>D</b> | <b>E</b> | <b>F</b> |
| Metering point ID 1...10 (DC electricity related) | 1, Data            | 2                | b        | 96       | 1        | 0...9    | 255      |
| Metering point ID-s object                        | 7, Profile generic | 2                | b        | 96       | 1        | 255      | 255      |
| Metering point ID-s object                        | 61, Register table | 2                | b        | 96       | 1        | 255      | 255      |

**6.4.7 DC electricity related status objects**

A number of DC 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 *boolean*, *bit-string*, *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-unsigned* or *octet-string*.

Alternatively, the status is held by a “Status mapping” object, which holds both the status word and the mapping of its bits to the reference table.

If there are several DC 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 DLMS UA 1000-1 Ed 15 Part 1:2021.

| <b>DC electricity related status objects</b>  | <b>IC</b>          | <b>OBIS code</b> |          |          |          |          |          |
|---|--------------------|------------------|----------|----------|----------|----------|----------|
|   |                    | <b>A</b>         | <b>B</b> | <b>C</b> | <b>D</b> | <b>E</b> | <b>F</b> |
| Internal operating status signals, DC electricity related, contents manufacturer specific     | 1, Data            | 2                | b        | 96       | 5        | 0...5    | 255      |
| Internal operating status signals, DC electricity related, contents mapped to reference table | 63, Status mapping | 2                | b        | 96       | 5        | 0...5    | 255      |

## COSEM Interface Classes

|  |                    |   |   |    |    |       |     |
|--|--------------------|---|---|----|----|-------|-----|
| DC electricity related status data, contents manufacturer specific     | 1, Data            | 2 | b | 96 | 10 | 0...3 | 255 |
| DC electricity related status data, contents mapped to reference table | 63, Status mapping | 2 | b | 96 | 10 | 0...3 | 255 |
| DC electricity related status data, reversed polarity                  | 1, Data            | 2 | B | 96 | 11 | 0     | 255 |

### 6.4.8 List objects – DC 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 – DC electricity                                  | IC                 | OBIS code |   |    |   |   |                  |
|--|--------------------|-----------|---|----|---|---|------------------|
|  |                    | A         | B | C  | D | E | F                |
| For names and OBIS codes see DLMS UA 1000-1 Ed 15 Part 1:2021. | 7, Profile generic | 2         | b | 98 | d | e | 255 <sup>a</sup> |
| <sup>a</sup> F = 255 means a wildcard here. See 7.11.3.        |                    |           |   |    |   |   |                  |

### 6.4.9 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 7.5.4.2, value group F may be used to identify multiple thresholds.

For OBIS codes, see Table 63 below and DLMS UA 1000-1 Ed 15 Part 1:2021.

**Table 63 – Threshold objects, DC electricity**

| Threshold objects                          | IC   | OBIS code |   |     |   |        |                |
|--|--|-----------|---|-----|---|--------|----------------|
|  |  | A         | B | C   | D   | E      | F              |
| Threshold objects for instantaneous values | 1, Data,<br>3, Register,<br>4, Extended register | 2         | b | 1.2 | 31...34,<br>35...38,<br>39...42,<br>43...45 | 0...63 | 0...99,<br>255 |

For monitoring the supply voltage, a more sophisticated functionality is also available, that allows counting the number of occurrences classified by the duration of the event and the depth of the voltage dip. For OBIS codes, see DLMS UA 1000-1 Ed 15 Part 1:2021.

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 616/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

**6.4.10 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.

See Table 64.

**Table 64 – Register monitor objects, DC electricity**

| Register monitor objects                                 | IC                   | OBIS code |   |              |                            |                       |                       |
|--|----------------------|-----------|---|--------------|----------------------------|-----------------------|-----------------------|
|  |                      | A         | B | C            | D                          | E                     | F                     |
| Instantaneous values, under limit / over limit / missing | 21, Register monitor | 2         | b | 1,2          | 31,<br>35,<br>39           | 0-63                  | 0-99,<br>255          |
|  |                      | 2         | b | 11,12,<br>92 |                            | 0-120,<br>124-<br>127 |                       |
| Current average and last average values                  |                      | 2         | b | 1,2          | 4, 5,<br>14, 15,<br>24, 25 | 0-63                  | 0-120,<br>124-<br>127 |
|  |                      | 2         | b | 11,12,<br>92 |                            | 0-120,<br>124-<br>127 |                       |

For the use of value group D, see DLMS UA 1000-1 Ed 15 Part 1:2021, 7.2.2.

For the use of value group E, see DLMS UA 1000-1 Ed 15 Part 1:2021, 7.2.3.

For the use of value group F, see DLMS UA 1000-1 Ed 15 Part 1:2021, 7.2.5.

## 6.5 HCA related COSEM objects

### 6.5.1 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.

### 6.5.2 ID numbers – HCA

The different HCA ID numbers are held by instances of the IC "Data", with data type *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-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 the HCA 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 36.

| HCA ID               | 3IC                | OBIS code |   |   |   |       |     |
|----------------------|--------------------|-----------|---|---|---|-------|-----|
|                      |                    | A         | B | C | D | E     | F   |
| HCA ID 1...10 object | 1, Data            | 4         | b | 0 | 0 | 0...9 | 255 |
| HCA ID-s object      | 7, Profile generic | 4         | b | 0 | 0 | 255   | 255 |
| HCA ID-s object      | 61, Register table | 4         | b | 0 | 0 | 255   | 255 |

### 6.5.3 Billing period values / reset counter entries - HCA

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*, *date* 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 "Storage information"          | IC      | OBIS code |   |   |   |                   |     |
|--|---------|-----------|---|---|---|-------------------|-----|
|  |         | A         | B | C | D | E                 | F   |
| For item names and OBIS codes see<br>See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 36. | 1, Data | 4         | b | 0 | 1 | 1,2,<br>10,<br>11 | 255 |

#### 6.5.4 General purpose objects – HCA

The use of ICs shall be as specified below:

- *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;
- *Device measuring principle* values are represented by instances of the IC "Data" with data type *unsigned* or *enumerated* for the *value* attribute;

| Enum  | Value                        |
|-------|------------------------------|
| (0)   | Single sensor                |
| (1)   | Single sensor + start sensor |
| (2)   | Dual sensor                  |
| (3)   | Triple sensor                |
| Other | reserved                     |

- *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;
- *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 *double-long-unsigned* (in the case of UNIX time), *octet-string*, *date* or *date-time* formatted as specified in 4.1.6.1.

| General purpose objects – HCA related <sup>a</sup> | IC   | OBIS code |   |   |   |           |     |
|--|--|-----------|---|---|---|-----------|-----|
|  |  | A         | B | C | D | E         | F   |
| Configuration objects                              | 1, Data  | 4         | b | 0 | 2 | 0-2       | 255 |
| Device measuring principle                         | 1, Data  | 4         | b | 0 | 2 | 3         | 255 |
| Conversion factors                                 | 1, Data or<br>3, Register or<br>4, Extended register | 4         | b | 0 | 4 | 0-6       | 255 |
| Threshold values                                   | 3, Register or<br>4, Extended register               | 4         | b | 0 | 5 | 10-11     | 255 |
| Period information                                 | 1, Data or<br>3, Register or<br>4, Extended register | 4         | b | 0 | 8 | 0,4,<br>6 | 255 |
| Time entries                                       | 1, Data or<br>3, Register or<br>4, Extended register | 4         | b | 0 | 9 | 1-3       | 255 |

<sup>a</sup> For item names and OBIS codes see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 36

#### 6.5.5 Measured Values – HCA

##### 6.5.5.1 Consumption – HCA

The use of ICs shall be as specified below:

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 619/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

Consumption values are represented by instances of the IC "Register", "Extended register" or "Profile generic" with data types *double-long*, *double-long-unsigned*, *octet-string*, *visible-string*, *integer*, *long-unsigned*, *long-unsigned*, *long64*, *long64-unsigned*, *float32* or *float64*.

| Consumption   | IC                                     | OBIS code |   |                |                |                |                       |
|---|--|-----------|---|----------------|----------------|----------------|-----------------------|
|   |  | A         | B | C <sup>a</sup> | D <sup>b</sup> | E <sup>c</sup> | F                     |
| Integral value, current   | 3, Register or<br>4, Extended register | 4         | b | 1, 2           | 0              | 0              | 255                   |
| Integral value over measurement periods (periodical value)                      |  |           |   |                | 1              |                | 0...99                |
| Integral value relative to billing periods (Set date value, Billing date value) |  |           |   |                | 2,3            |                | 101...<br>125,<br>255 |
| Minimum, Maximum of integral value over billing periods                         |  |           |   |                | 4,5            |                |                       |
| Integral test value   |  |           |   |                | 6              |                | 255                   |

<sup>a</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 33 – Value group C codes - HCA  
<sup>b</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 34 – Value group D codes – HCA  
<sup>c</sup> No further classification in value group E is made. Therefore, E shall be 0.

### 6.5.5.2 Temperature Values – HCA

The use of ICs shall be as specified below:

The objects are used to represent temperature information as measurand values.

Temperature values are represented by instances of the IC "Register", "Extended register" or "Profile generic".

| Temperature                     | IC                                     | OBIS code |   |                |                |                |     |
|---------------------------------|--|-----------|---|----------------|----------------|----------------|-----|
|                                 |  | A         | B | C <sup>a</sup> | D <sup>b</sup> | E <sup>c</sup> | F   |
| Temperature value, current      | 3, Register or<br>4, Extended register | 4         | b | 3..7           | 0              | 255            | 255 |
| Minimum, Maximum of temperature |  |           |   |                | 4,5            |                |     |
| Temperature test value          |  |           |   |                | 6              |                |     |

<sup>a</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 33 – Value group C codes - HCA  
<sup>b</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 34 – Value group D codes – HCA  
<sup>c</sup> Value group E not used. Therefore, E shall be 255.

### 6.5.6 Error register objects – HCA

A series of objects are used to communicate error indications of the device. See also DLMS UA 1000-1 Ed 15 Part 1:2021, Table 40

The different error registers are held by the *value* attribute of “Data”, “Register” or “Extended register” objects, with data type *bit-string*, *octet-string*, *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*.

| Error register objects - HCA | IC   | OBIS code |   |    |    |   |     |
|------------------------------|--|-----------|---|----|----|---|-----|
|                              |  | A         | B | C  | D  | E | F   |
| Error register objects       | 1, Data or<br>3, Register or<br>4, Extended register | 4         | b | 97 | 97 | e | 255 |

### 6.5.7 List objects – HCA

The use of ICs shall be as specified below:

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. Examples are present in DLMS UA 1000-1 Ed 15 Part 1:2021, 8.1.5.3.

| List objects – HCA                          | IC                 | OBIS code |   |    |   |   |                  |
|---|--------------------|-----------|---|----|---|---|------------------|
|   |                    | A         | B | C  | D | E | F                |
| For names and OBIS codes see XX             | 7, Profile generic | 4         | b | 98 | d | e | 255 <sup>a</sup> |
| <sup>a</sup> F = 255 means a wildcard here. |                    |           |   |    |   |   |                  |

### 6.5.8 Data profile objects – HCA

The use of ICs shall be as specified below:

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. See also 7.6.4.4.

| Data profile objects - HCA | IC                 | OBIS code |   |    |   |   |     |
|----------------------------|--------------------|-----------|---|----|---|---|-----|
|                            |                    | A         | B | C  | D | E | F   |
| Data profile objects       | 7, Profile generic | 4         | b | 99 | 1 | e | 255 |

## 6.6 Thermal energy meter related COSEM objects

### 6.6.1 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.

### 6.6.2 ID numbers – Thermal energy meter

The different thermal energy meter ID numbers are held by instances of the IC "Data", with data type *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-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 the thermal meter 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 44.

| Thermal energy meter ID               | IC                 | OBIS code |   |   |   |       |     |
|---------------------------------------|--------------------|-----------|---|---|---|-------|-----|
|                                       |                    | A         | B | C | D | E     | F   |
| Thermal energy meter ID 1...10 object | 1, Data            | 5/6       | b | 0 | 0 | 0...9 | 255 |
| Thermal energy meter ID-s object      | 7, Profile generic | 5/6       | b | 0 | 0 | 255   | 255 |
| Thermal energy meter ID-s object      | 61, Register table | 5/6       | b | 0 | 0 | 255   | 255 |

### 6.6.3 Billing period values / reset counter entries - Thermal energy meter

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 "Storage information"  | IC      | OBIS code |   |   |   |                   |             |
|--|---------|-----------|---|---|---|-------------------|-------------|
|  |         | A         | B | C | D | E                 | F           |
| For item names and OBIS codes see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 44 | 1, Data | 5/6       | b | 0 | 1 | 1,2,<br>10,<br>11 | 1,2,<br>255 |

#### 6.6.4 General purpose objects – Thermal energy meter

The use of ICs shall be as specified below:

- *Configuration entries* are represented by instances of the IC "Data" with data type *unsigned*, *long-unsigned* or *octet-string*. Configuration entries can also be related to a channel;
- *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;
- *Timing information for averaging, measurement, billing periods and recording interval* is presented 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 *double-long-unsigned* (in the case of UNIX time), *octet-string* or *date-time* formatted as specified in 4.1.6.1.

| General purpose objects – Thermal energy meter related <sup>a</sup> | IC                                     | OBIS code |   |   |   |                                   |     |
|---|--|-----------|---|---|---|-----------------------------------|-----|
|   |  | A         | B | C | D | E                                 | F   |
| Configuration objects   | 1, Data                                | 5/6       | b | 0 | 2 | 0-4,<br>10 -13                    | 255 |
| Conversion factors  | 1, Data                                | 5/6       | b | 0 | 4 | 1-3                               | 255 |
| Threshold values  | 3, Register or<br>4, Extended register | 5/6       | b | 0 | 5 | 1-9,<br>21-24                     | 255 |
| Timing information  | 1, Data                                | 5/6       | b | 0 | 8 | 0-7,<br>11-14,<br>21-25,<br>31-34 | 255 |
| Time entries  | 1, Data                                | 5/6       | b | 0 | 9 | 1-3                               | 255 |

<sup>a</sup> For item names and OBIS codes see See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 44.

#### 6.6.5 Measured values - Thermal energy meterConsumption – Thermal energy meter

The use of ICs shall be as specified below:

Consumption values are represented by instances of the IC "Register", "Extended register" or "Profile generic" with data types *double-long*, *double-long-unsigned*, *octet-string*, *visible-string*, *integer*, *long,unsigned*, *long-unsigned*, *long64*, *long64-unsigned*, *float32* or *float64*.

## COSEM Interface Classes

| Consumption   | IC   | OBIS code |   |                |                |                |                              |
|---|--|-----------|---|----------------|----------------|----------------|------------------------------|
|   |  | A         | B | C <sup>a</sup> | D <sup>b</sup> | E <sup>c</sup> | F                            |
| Energy, volume, mass values                             | 3, Register or 4, Extended register, or 7, Profile generic | 5/6       | b | 1...7          | 0...3, 7       | 0,<br>1...9    | 255                          |
| Integral value relative to billing periods              | 3, Register or 4, Extended register, or 7, Profile generic |           |   |                | 3, 8, 9        |                | 0...99,<br>101...125,<br>255 |
| Periodical value  | 3, Register or 4, Extended register, or 7, Profile generic |           |   |                | 1,12,13        |                | 0...99,<br>101...125         |
| Set date value  | 3, Register or 4, Extended register, or 7, Profile generic |           |   |                | 2              |                |                              |
| Minimum, Maximum of integral value over billing periods | 3, Register or 4, Extended register, or 7, Profile generic |           |   |                | 4, 5, 14, 15   |                |                              |
| Integral test value                                     | 3, Register, 4, Extended register                          |           |   |                | 6              |                | 255                          |

<sup>a</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 41 - Value group C codes – thermal energy meters  
<sup>b</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 42– Value group D codes - thermal energy meters  
<sup>c</sup> All other values reserved for further use (tariff rate E = 0 means Total)

### 6.6.5.2 Monitoring values – Thermal energy meter

The use of ICs shall be as specified below:

The objects are to be used to represent information as monitored values.

*Monitoring values* are held by the *value* attribute of “Register” or “Extended register” objects with data types *double-long*, *double-long-unsigned*, *octet-string*, *visible-string*, *integer*, *long*, *unsigned*, *long-unsigned*, *long64*, *long64-unsigned*, *float32* or *float64*.

## COSEM Interface Classes

| Monitoring values                                    | IC  | OBIS code           |         |                |                     |                |                               |
|--|---|---------------------|---------|----------------|---------------------|----------------|-------------------------------|
|  |   | A                   | B       | C <sup>a</sup> | D <sup>b</sup>      | E <sup>c</sup> | F                             |
| Energy, volume, mass<br>Maximum of integral value    | 4, Extended register, or<br>7, Profile generic                    | 5/6<br><br><i>b</i> | 10...13 | 1...7          | 5,15                | 0,<br>1..9     | 0...99,<br>101...125<br>, 255 |
| Power or flow rate values                            | 3, Register, or<br>4, Extended register, or<br>7, Profile generic |                     |         | 8,9            | 1, 4, 5,<br>12...15 |                |                               |
| Temperature or pressure<br>value, current            | 3, Register or<br>4, Extended register, or<br>7, Profile generic  |                     |         |                | 0                   |                | 255                           |
| Minimum, Maximum of<br>temperature or pressure       | 4, Extended register, or<br>7, Profile generic                    |                     |         |                | 4, 5, 14, 15        |                |                               |
| Temperature or pressure<br>test, instantaneous value | 3, Register, or<br>4, Extended register                           |                     | 1...13  |                | 6,7                 |                | 255                           |
| Average values of<br>temperature or pressure         | 5, Demand register, or<br>7, Profile generic                      |                     |         |                | 10,11               |                |                               |
| Occurrence counters                                  | 3, Register, or<br>4, Extended register, or<br>7, Profile generic |                     |         |                | 20, 22, 24          |                |                               |
| Occurrence durations                                 | 3, Register, or<br>4, Extended register, or<br>7, Profile generic |                     |         |                | 21, 23, 25          |                |                               |

<sup>a</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 41 – Value group C codes – Thermal energy meter

<sup>b</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 42 – Value group D codes - Thermal energy meter

<sup>c</sup> All other values reserved for further use (tariff rate E = 0 means Total)

### 6.6.6 Error register objects – Thermal energy meter

A series of objects are used to communicate error indications of the device. See also DLMS UA 1000-1 Ed 15 Part 1:2021, Table 45.

The different error registers are held by the value attribute of “Data”, “Register” or “Extended register” objects, with data type *bit-string*, *octet-string*, *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*.

| – Error register objects – Thermal energy meters | IC   | OBIS code |          |    |    |   |     |
|--|--|-----------|----------|----|----|---|-----|
|  |  | A         | B        | C  | D  | E | F   |
| Overall error status <sup>a</sup>                | 1, Data or<br>3, Register or<br>4, Extended register                           | 5/6       | <i>b</i> | 97 | 97 | 0 | 255 |
| Subsystem where error has occurred <sup>b</sup>  |  | 5/6       | <i>b</i> | 97 | 97 | 1 | 255 |
| Duration of error condition <sup>c</sup>         |  | 5/6       | <i>b</i> | 97 | 97 | 2 | 255 |
| <sup>a</sup>                                     | This object is a 'mirror' of the object 0.x.97.97.0.                           |           |          |    |    |   |     |
| <sup>b</sup>                                     | A further subdivision of error information.                                    |           |          |    |    |   |     |
| <sup>c</sup>                                     | This is the time during which the meter has not been able to calculate energy. |           |          |    |    |   |     |

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 625/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

### 6.6.7 List objects – Thermal energy meter

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.

| List objects – Thermal energy meter   | IC                 | OBIS code |   |    |   |   |                  |
|---|--------------------|-----------|---|----|---|---|------------------|
|   |                    | A         | B | C  | D | E | F                |
| For names and OBIS codes see XX   | 7, Profile generic | 5/6       | b | 98 | d | e | 255 <sup>a</sup> |
| <sup>a</sup> F = 255 means a wildcard here. Also note DLMS UA 1000-1 Ed 15 Part 1:2021 7.7.4.3. |                    |           |   |    |   |   |                  |

### 6.6.8 Data profile objects – Thermal energy meter

Thermal energy meter 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.

| Data profile objects – Thermal energy meters   | IC                 | OBIS code |   |    |    |     |     |
|--|--------------------|-----------|---|----|----|-----|-----|
|  |                    | A         | B | C  | D  | E   | F   |
| Data profile objects. For names and OBIS codes see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 47. | 7, Profile generic | 5/6       | b | 99 | 1  | 1-3 | 255 |
|  |                    |           |   |    | 2  | 1-3 |     |
|  |                    |           |   |    | 3  | 1   |     |
|  |                    |           |   |    | 99 | e   |     |

## 6.7 Gas related COSEM objects

NOTE The reader is advised to refer to 7.8.1, General introduction to gas measurement before reading this section.

### 6.7.1 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.

### 6.7.2 ID numbers – Gas

The different gas ID numbers are represented 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 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 60.

| Gas ID               | IC                 | OBIS code |   |   |   |       |     |
|----------------------|--------------------|-----------|---|---|---|-------|-----|
|                      |                    | A         | B | C | D | E     | F   |
| Gas ID 1...10 object | 1, Data            | 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 |

### 6.7.3 Billing period values / reset counter entries – Gas

Billing period 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 60. | 1, Data | 7         | b | 0 | 1 | e | 255 |

### 6.7.4 Other general purpose objects – Gas

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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 627/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

For the digital and analogue output configuration objects the enumerated values are specified in Table 65 below.

**Table 65 – Digital / Analogue output configurations – enumerated values**

| <b>Digital output configuration</b>  |  |
|--------------------------------------|--|
| (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)                                     |
| <b>Analogue output configuration</b> |  |
| (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;

## COSEM Interface Classes

- 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, DLMS UA 1000-1 Ed 15 Part 1:2021, Table 50. 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 DLMS UA 1000-1 Ed 15 Part 1:2021, Table 60.

| Gas related general purpose objects   | IC   | OBIS code |   |   |           |   |     |
|---|--|-----------|---|---|-----------|---|-----|
|   |  | A         | B | C | D         | E | F   |
| Configuration objects   | 1, Data                                      | 7         | b | 0 | 2         | e | 255 |
| Output pulse constants converted / unconverted                                      | 1, Data                                      | 7         | b | 0 | 3         | e | 255 |
| Conversion factors  | 1, Data                                      | 7         | b | 0 | 4         | e | 255 |
| Threshold values  | 3, Register or<br>4, Extended register       | 7         | b | 0 | 5         | e | 255 |
| Nominal values volume sensor  | 3, Register or<br>4, Extended register       | 7         | b | 0 | 6         | e | 255 |
| Input pulse constants   | 1, Data                                      | 7         | b | 0 | 7         | e | 255 |
| Intervals and periods   | 1, Data                                      | 7         | b | 0 | 8         | e | 255 |
| Time entries  | 1, Data                                      | 7         | b | 0 | 9         | e | 255 |
| Station management information  | 3, Register or<br>4, Extended register       | 7         | b | 0 | 10,<br>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,<br>14 | e | 255 |
| Sensor manager objects  | 67, Sensor manager                           | 7         | b | 0 | 15        | e | 255 |

### 6.7.5 Internal operating status objects – Gas

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*, *boolean* 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            | 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 |

### 6.7.6 Measured values – Gas

#### 6.7.6.1 Indexes and index differences – Gas

Table 66 shows the objects to be used to represent indexes and index differences of volume, mass and energy.

**Table 66 – Indexes and index differences**

| Indexes and index differences                     | IC                                    | OBIS code |   |   |   |                |                             |
|---|---------------------------------------|-----------|---|---|---|----------------|-----------------------------|
|   |                                       | A         | B | C <sup>a</sup>  | D <sup>b</sup>                              | E <sup>c</sup> | F                           |
| Index   |                                       |           |   |   | 0...3                                       |                | 255                         |
| Index relative to billing periods                 | Register or Extended register         |           |   |   | 24...26<br>42...44<br>63...65<br>81...83    |                | 0...99<br>101...126         |
| Index difference over measurement periods         | 7                                     |           | b | 1...8,<br>11...16,<br>21...26,<br>31...36,<br>61...66 | 6....23                                     | 0,<br>1...63   | 255                         |
| Index difference over billing periods             |                                       |           |   |   | 27...32,<br>45...50,<br>66...71,<br>84...89 |                | 255                         |
| Maximum of index differences over billing periods | Extended register, or Profile generic |           |   |   | 33...41,<br>51...62,<br>72...80,<br>90...98 |                | 0...99<br>101...126,<br>255 |

<sup>a</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 50 – Value group C codes – Gas

<sup>b</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021 Table 51 – Value group D codes – Gas – Indexes and index differences

<sup>c</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 56 – Value group E codes – Gas – Indexes and index differences – Tariff rates

### 6.7.6.2 Flow rate – Gas

Table 67 shows the objects to be used to represent flow rate values.

**Table 67 – Flow rate**

| Flow rate  | IC   | OBIS code |   |                |  |   |                        |
|--|--|-----------|---|----------------|--|---|------------------------|
|  |  | A         | B | C <sup>a</sup> | D <sup>b</sup>   | 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 |

<sup>a</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 50 – Value group C codes – Gas  
<sup>b</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021 Table 53 – Value group D codes – Gas – Flow rate

### 6.7.6.3 Process values – Gas

Table 68 shows the objects to be used to represent process values.

**Table 68 – Process values**

| Process values   | IC                            | OBIS code |   |                 |                              |   |     |
|--|-------------------------------|-----------|---|-----------------|------------------------------|---|-----|
|  |                               | A         | B | C <sup>a</sup>  | D <sup>b</sup>               | 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 |

<sup>a</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 50 – Value group C codes – Gas  
<sup>b</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 53 – Value group D codes – Gas – Process values

### 6.7.7 Conversion related factors and coefficients – Gas

Table 69 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.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 631/668 |
|-----------------------|------------|-----------------------------|---------|

**Table 69 – Conversion related factors and coefficients**

| Conversion related factors and coefficients   | IC                            | OBIS code |   |                |                    |                |     |  |  |  |  |  |  |
|---|-------------------------------|-----------|---|----------------|--------------------|----------------|-----|--|--|--|--|--|--|
|   |                               | A         | B | C <sup>a</sup> | D <sup>b</sup>     | E <sup>c</sup> | F   |  |  |  |  |  |  |
| Process independent current value or weighted value   | Register or Extended register | 7         | b | 51...55        | 0, 2, 3,<br>10, 11 | 0,1            | 255 |  |  |  |  |  |  |
| Current average and last average values   |                               |           |   |                |                    | 11...28        |     |  |  |  |  |  |  |
| <sup>a</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 50 – Value group C codes – Gas   |                               |           |   |                |                    |                |     |  |  |  |  |  |  |
| <sup>b</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 54 – Value group D codes – Gas – Conversion related factors and coefficients |                               |           |   |                |                    |                |     |  |  |  |  |  |  |
| <sup>c</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 57 – Value group E codes – Gas – Conversion related factors and coefficients |                               |           |   |                |                    |                |     |  |  |  |  |  |  |

### 6.7.8 Calculation methods – Gas

Table 70 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 70 – Calculation methods**

| Calculation methods   | IC                                  | OBIS code |   |                |                |                |     |
|---|-------------------------------------|-----------|---|----------------|----------------|----------------|-----|
|   |                                     | A         | B | C <sup>a</sup> | D <sup>b</sup> | E <sup>c</sup> | F   |
| Calculation methods   | Data, Register or Extended register | 7         | b | 51...55        | 12             | 0...20         | 255 |
| <sup>a</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 50 – Value group C codes – Gas   |                                     |           |   |                |                |                |     |
| <sup>b</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 54 – Value group D codes – Gas – Conversion related factors and coefficients |                                     |           |   |                |                |                |     |
| <sup>c</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 58 – Value group E codes – Gas – Calculation methods                         |                                     |           |   |                |                |                |     |

### 6.7.9 Natural gas analysis

Table 71 shows the objects to be used to represent values of natural gas analysis.

**Table 71 – Natural gas analysis**

| Natural gas analysis                                  | IC                            | OBIS code |   |                |                     |                |     |  |
|---|-------------------------------|-----------|---|----------------|---------------------|----------------|-----|--|
|   |                               | A         | B | C <sup>a</sup> | D <sup>b</sup>      | E <sup>c</sup> | F   |  |
| Reference values of gas analysis process              | Register or Extended register | 7         | b | 70             | 8,9                 | 0              | 255 |  |
| Gas characteristics                                   |                               |           |   |                | 10...20,<br>60...84 | 0,1            |     |  |
| Process independent current value, and weighted value |                               |           |   |                |                     | 11...28        |     |  |
| Gas characteristics                                   |                               |           |   |                |                     |                |     |  |
| Average values for current and last intervals         |                               |           |   |                |                     |                |     |  |

<sup>a</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 50 – Value group C codes – Gas  
<sup>b</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 55 – Value group D codes – Gas – Natural gas analysis values  
<sup>c</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 59 – Value group E codes – Gas – Natural gas analysis values – Averages

### 6.7.10 List objects – Gas

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 DLMS UA 1000-1 Ed 15 Part 1:2021, 8.3.6.3.

One standard object per billing period scheme is defined. See also DLMS UA 1000-1 Ed 15 Part 1:2021, 8.3.6.3.

| List objects - Gas  | IC                 | OBIS code |   |    |   |   |                  |
|---|--------------------|-----------|---|----|---|---|------------------|
|   |                    | A         | B | C  | D | E | F                |
| For names and OBIS codes see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 62.          | 7, Profile generic | 7         | b | 98 | d | e | 255 <sup>a</sup> |
| <sup>a</sup> F = 255 means a wildcard here. DLMS UA 1000-1 Ed 15 Part 1:2021, A.3 |                    |           |   |    |   |   |                  |

## 6.8 Water meter related COSEM objects

### 6.8.1 General

The use of interface classes to represent various data is described in the following subclauses.

Grouping the data by major categories supports the link between the data and the OBIS value groups associated with them.

### 6.8.2 ID numbers – water meter

The different water meter ID numbers are instances of the IC "Data", with data type *unsigned*, *long-unsigned*, *double-long-unsigned*, *long64-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 the water meter 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 DLMS UA 1000-1 Ed 15 Part 1:2021,Table 67.

| Water meter ID               | IC                 | OBIS code |   |   |   |       |     |
|------------------------------|--------------------|-----------|---|---|---|-------|-----|
|                              |                    | A         | B | C | D | E     | F   |
| Water meter ID 1...10 object | 1, Data            | 8/9       | b | 0 | 0 | 0...9 | 255 |
| Water meter ID-s object      | 7, Profile generic | 8/9       | b | 0 | 0 | 255   | 255 |
| Water meter ID-s object      | 61, Register table | 8/9       | b | 0 | 0 | 255   | 255 |

### 6.8.3 Billing period values / reset counter entries – water meter

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 "Storage information"   | IC      | OBIS code |   |   |   |                    |     |
|---|---------|-----------|---|---|---|--------------------|-----|
|   |         | A         | B | C | D | E                  | F   |
| For item names and OBIS codes see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 67. | 1, Data | 8/9       | b | 0 | 1 | 1,2,<br>10..<br>12 | 255 |

#### 6.8.4 General purpose objects – water meter

The use of ICs shall be as specified below:

- *Program entries* are represented by instances of the IC "Data" with data type *unsigned*, *long-unsigned* or *octet-string*. Program entries can also be related to a channel;
- *Threshold values* are represented by instances of the IC "Register" or "Extended register".
- *Input pulse constants* are represented by instances of the IC "Data", "Register" or "Extended register" with data type *unsigned*, *long-unsigned* or *octet-string*. *Input pulse constants* entries can also be related to a channel;
- *Timing information for measurement and registration periods* is presented 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 *double-long-unsigned* (in the case of UNIX time), *octet-string* or *date-time* formatted as specified in 4.1.6.1.

| General purpose objects – Water meter related <sup>a</sup> | IC  | OBIS code |   |   |   |     |     |
|--|---|-----------|---|---|---|-----|-----|
|  |   | A         | B | C | D | E   | F   |
| Program entries  | 1, Data   | 8/9       | b | 0 | 2 | 0,3 | 255 |
| Threshold values   | 3, Register or<br>4, Extended register            | 8/9       | b | 0 | 5 | 1   | 255 |
| Input pulse constants                                      | 3, Register or<br>4, Extended register            | 8/9       | b | 0 | 7 | 1   | 255 |
| Measurement-/registration period duration                  | 1, Data<br>3, Register or<br>4, Extended register | 8/9       | b | 0 | 8 | 1,6 | 255 |
| Time entries   | 1, Data<br>3, Register or<br>4, Extended register | 8/9       | b | 0 | 9 | 1-3 | 255 |

<sup>a</sup> For item names and OBIS codes see DLMS UA 1000-1 Ed 15 Part 1:2021, Table 67

#### 6.8.5 Measured values – water meter

##### 6.8.5.1 Consumption – water meter

The use of ICs shall be as specified below:

Consumption values are held by the *value* attribute of "Register" or "Extended register" objects with data types *double-long*, *double-long-unsigned*, *octet-string*, *visible-string*, *integer*, *long*, *unsigned*, *long-unsigned*, *long64*, *long64-unsigned*, *float32* or *float64*.

## COSEM Interface Classes

| Consumption   | IC   | OBIS code |   |                |                |                |                      |
|---|--|-----------|---|----------------|----------------|----------------|----------------------|
|   |  | A         | B | C <sup>a</sup> | D <sup>b</sup> | E <sup>c</sup> | F                    |
| Accumulated volume                                      | 3, Register or<br>4, Extended register                         | 8/9       | b | 1              | 0-3            | 0-12           | 255                  |
| Minimum, Maximum of integral value over billing periods | 4, Extended register, or 7, Profile generic                    |           |   |                | 1-3            |                | 0-99,<br>101-<br>125 |
| Integral test value                                     | 3, Register,<br>4, Extended register,<br>or 7, Profile generic |           |   |                | 4,5            |                | 255                  |
|   |  |           |   |                | 6              |                |                      |

<sup>a</sup> See.DLMS UA 1000-1 Ed 15 Part 1:2021, Table 64 – Value group C codes - water meters

<sup>b</sup> See.DLMS UA 1000-1 Ed 15 Part 1:2021, Table 65 – Value group D codes - water meters

<sup>c</sup> All other values reserved for further use (tariff rate E = 0 means Total)

### 6.8.5.2 Monitoring values – water meter

The use of ICs shall be as specified below:

*Monitoring values* are held by the *value* attribute of “Register” or “Extended register” objects with data types *double-long,double-long-unsigned, octet-string, visible-string, integer, long,unsigned, long-unsigned, long64, long64-unsigned, float32* and *float64*.

| Monitoring values             | IC   | OBIS code |   |                |                |                |                      |
|-------------------------------|--|-----------|---|----------------|----------------|----------------|----------------------|
|                               |  | A         | B | C <sup>a</sup> | D <sup>b</sup> | E <sup>c</sup> | F                    |
| Flow rate values              | 3, Register or<br>4, Extended register                         | 8/9       | b | 2              | 0-3            | 0-12           | 255                  |
| Minimum, Maximum of Flow rate | 4, Extended register, or 7, Profile generic                    |           |   |                | 1-3            |                | 0-99,<br>101-<br>125 |
| Flow rate test value          | 3, Register,<br>4, Extended register,<br>or 7, Profile generic |           |   |                | 4,5            |                | 255                  |
|                               |  |           |   |                | 6              |                |                      |

## COSEM Interface Classes

| Monitoring values               | IC   | OBIS code |   |                |                |                |                       |
|---------------------------------|--|-----------|---|----------------|----------------|----------------|-----------------------|
|                                 |  | A         | B | C <sup>a</sup> | D <sup>b</sup> | E <sup>c</sup> | F                     |
| Temperature values              | 3, Register or<br>4, Extended register                         | 8/9       | b | 3              | 0-3            | 0-12           | 255                   |
| Minimum, Maximum of temperature | 4, Extended register, or<br>7, Profile generic                 |           |   |                | 1-3            |                | 0..99,<br>101-<br>125 |
| Temperature test value          | 3, Register,<br>4, Extended register, or<br>7, Profile generic |           |   |                | 4,5            |                | 255                   |
|                                 |  |           |   |                | 6              |                |                       |

<sup>a</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 64 – Value group C codes - water meters

<sup>b</sup> See DLMS UA 1000-1 Ed 15 Part 1:2021, Table 65 – Value group D codes - water meters

<sup>c</sup> All other values reserved for further use (tariff rate E = 0 means Total)

### 6.8.6 Error register objects – water meter

A series of objects are used to communicate error indications of the device. See also DLMS UA 1000-1 Ed 15 Part 1:2021, Table 68.

The different error registers are held by the *value* attribute of “Data”, “Register” or “Extended register” objects, with data type *bit-string*, *octet-string*, *unsigned*, *long-unsigned*, *double-long-unsigned* or *long64-unsigned*.

| Error register objects – Water meters | IC   | OBIS code |   |    |    |   |     |
|---------------------------------------|--|-----------|---|----|----|---|-----|
|                                       |  | A         | B | C  | D  | E | F   |
| Error register objects                | 1, Data or<br>3, Register or<br>4, Extended register | 8/9       | b | 97 | 97 | e | 255 |

### 6.8.7 List objects – water meter

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

| List objects – Water meters | IC                 | OBIS code |   |    |   |   |                  |
|-----------------------------|--------------------|-----------|---|----|---|---|------------------|
|                             |                    | A         | B | C  | D | E | F                |
| List objects – water        | 7, Profile generic | 8/9       | b | 98 | d | e | 255 <sup>a</sup> |

<sup>a</sup> F = 255 means a wildcard here.

### 6.8.8 Data profile objects – water meter

Water meter 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.

| Data profile objects – Water meters | IC                 | OBIS code |   |    |   |   |     |
|-------------------------------------|--------------------|-----------|---|----|---|---|-----|
|                                     |                    | A         | B | C  | D | E | F   |
| Data profile objects                | 7, Profile generic | 8/9       | b | 99 | 1 | e | 255 |

### 6.9 Coding of OBIS identifications

To identify different instances of the same IC, their logical\_name shall be different. In COSEM, the logical\_name is taken from the OBIS definition (see 6.2, 6.3 and DLMS UA 1000-1 Ed 15 Part 1:2021).

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 DLMS UA 1000-1 Ed 15 Part 1:2021. | 0 0 0 0                                  |
| Reserved for future use                     | 0 0 0 1<br>...<br>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 DLMS UA 1000-1 Ed 15 Part 1:2021, 4.2) is used, then the whole OBIS code shall be considered as manufacturer specific, and the value of the other groups does not necessarily carry a meaning defined neither by Clause 5 of this standard nor by DLMS UA 1000-1 Ed 15 Part 1:2021.

## Annex A (informative)

### Additional information on Auto answer and Auto connect ICs

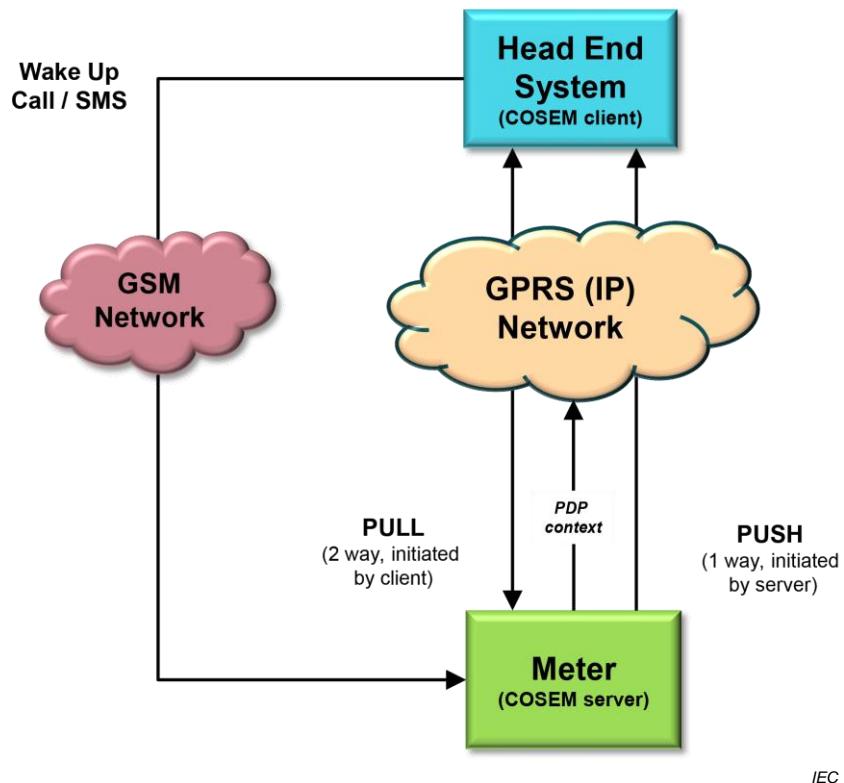
**NOTE** This information is related to the "Auto answer" (class\_id = 28, version = 2, see 4.7.5) and "Auto connect" (class\_id = 29, version = 2, see 4.7.6) interface classes.

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 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.



IEC

**Figure A.1 – Network connectivity example for a GSM/GPRS network**

The basic network connectivity in the case of a mobile network (GPRS or equivalent service) is modelled by the "Auto connect" IC. Depending on the mode the connection can be 'always on', 'always on in a time window' or 'only on after a wake-up'. If the device is connected to the network it has the PDP context attached and it is accessible from by the HES via its IP address. If necessary the current IP address of the server (meter) can be sent to the client (HES) using the DataNotification xDLMS service.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 639/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

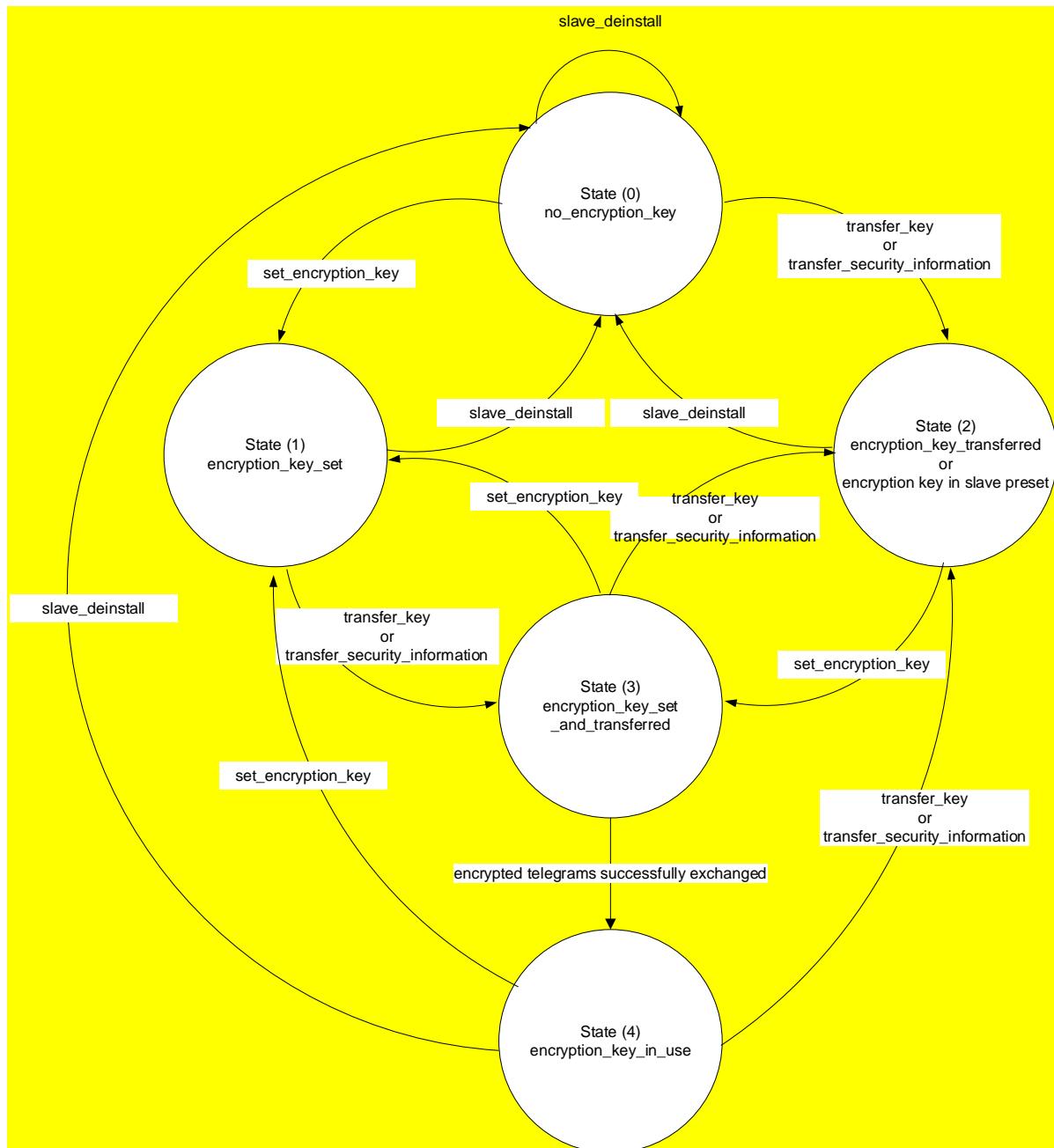
The wake-up process is modelled using an instance of the “Auto answer” IC which provides additional security (check calling number) compared to today's solution.

Also note that the “Auto answer” class is fully decoupled from the xDLMS application layer services. The main reason is to have a clear separation between the communication layers as well as to avoid creating an unsecured backdoor to execute application layer services with almost no protection. The execution of xDLMS services via SMS should be handled by sending ciphered xDLMS APDUs in a pre-established AA.

**Annex B**  
(informative)

**Additional information to M-Bus client (class\_id = 72, version 1 & 2)**

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**

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 641/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

State transitions of the *encryption\_key\_status* attribute for different use cases are given below. At the time of installation of the slave four cases are possible:

- a) Encryption key is preset in the slave and cannot be changed, see Table B.1;
- b) Encryption key is preset in the slave and new key is set after installation, see Table B.2;
- c) Encryption key is not preset in the slave, but can be set, see Table B.3 and Table B.4;
- d) The slave is used without encryption. In this case, the *encryption\_key\_status* stays in state (0).

**Table B.1 – Encryption key is preset in the slave and cannot be changed**

| Step | State | Condition for state transition<br>(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<br>(event or successfully invoked method) |
|------|-------|--|
| 1    | (2)   | set_encryption_key   |
| 2    | (3)   | transfer_key <b>or transfer_security_information</b>                     |
| 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)**

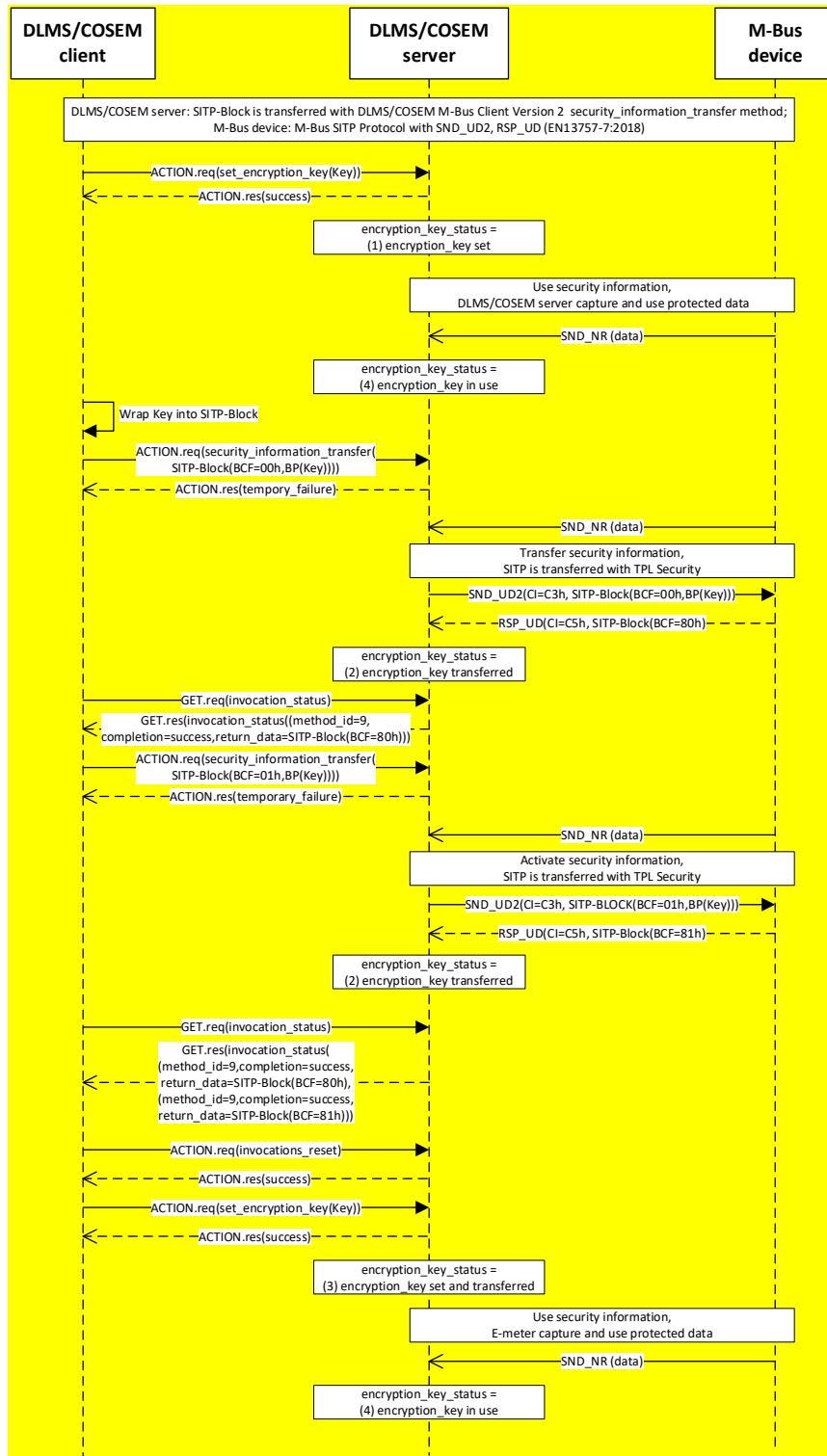
| Step | State | Condition for state transition<br>(event or successfully invoked method) |
|------|-------|--|
| 1a   | (0)   | set_encryption_key   |
| 2a   | (1)   | transfer_key <b>or transfer_security_information</b>                     |
| 3a   | (3)   | encrypted telegrams successfully exchanged                               |
| 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<br>(event or successfully invoked method) |
|------|-------|--|
| 1b   | (0)   | transfer_key <b>or transfer_security_information</b>                     |
| 2b   | (2)   | set_encryption_key   |
| 3b   | (3)   | encrypted telegrams successfully exchanged                               |
| 4b   | (4)   | –  |

## COSEM Interface Classes

Example of SITP transfer (EN 13757-7:2018 Annex A) with *transfer\_security\_information* method invocation with single M-Bus transfers shown in Figure B.2.



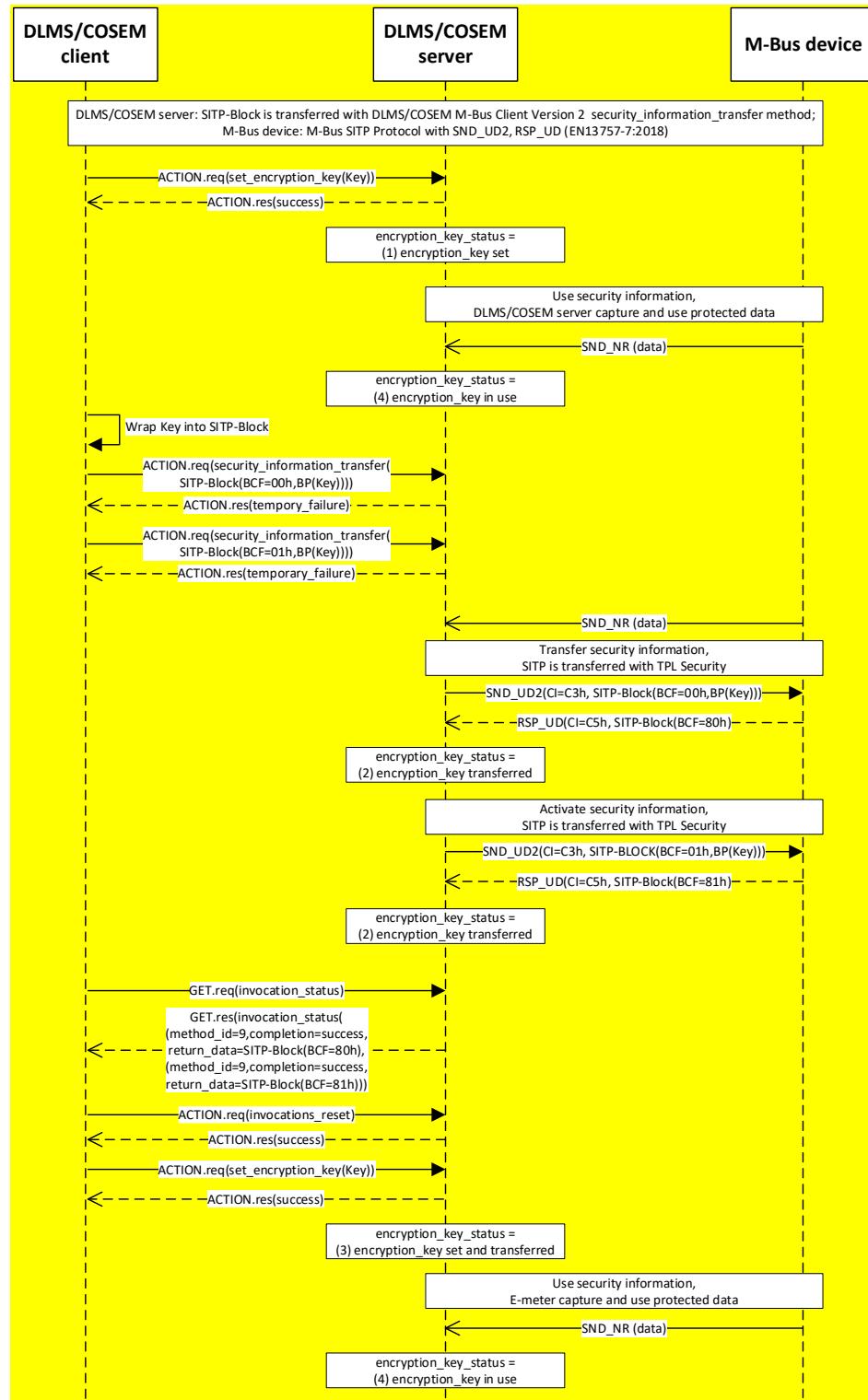
**Figure B.2 – Example of SITP transfer with single M-Bus transfer**

NOTE 1 Single in this context refers to one M-Bus transfer per Frequent Access Cycle.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 643/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

SITP transfer (EN 13757-7:2018 Annex A) with *transfer\_security\_information* method invocation with multiple M-Bus transfers shown in Figure B.3.



**Figure B.3 – SITP transfer with multiple M-Bus transfers diagram**

**NOTE 2** Multiple in this context refers to multiple M-Bus transfers per Frequent Access Cycle.

**Annex C**  
(informative)**Additional information on IPv6 setup class (class\_id = 48, version = 0)****C.1 General**

In most regards, IPv6 is a conservative extension of IPv4. Most transport and application-layer protocols need little or no change to operate over IPv6; exceptions are application protocols that embed internet-layer addresses, such as FTP or NTPv3.

IPv6 specifies a new packet format, designed to minimize packet-header processing. Since the headers of IPv4 packets and IPv6 packets are significantly different, the two protocols are not interoperable.

**C.2 IPv6 addressing**

The most important feature of IPv6 is a much larger address space than that of IPv4: addresses in IPv6 are 128 bits long, compared to 32-bit addresses in IPv4. Furthermore, compared to IPv4, IPv6 supports multi-addressing on one physical interface (global, unique or link local IPv6 addresses).

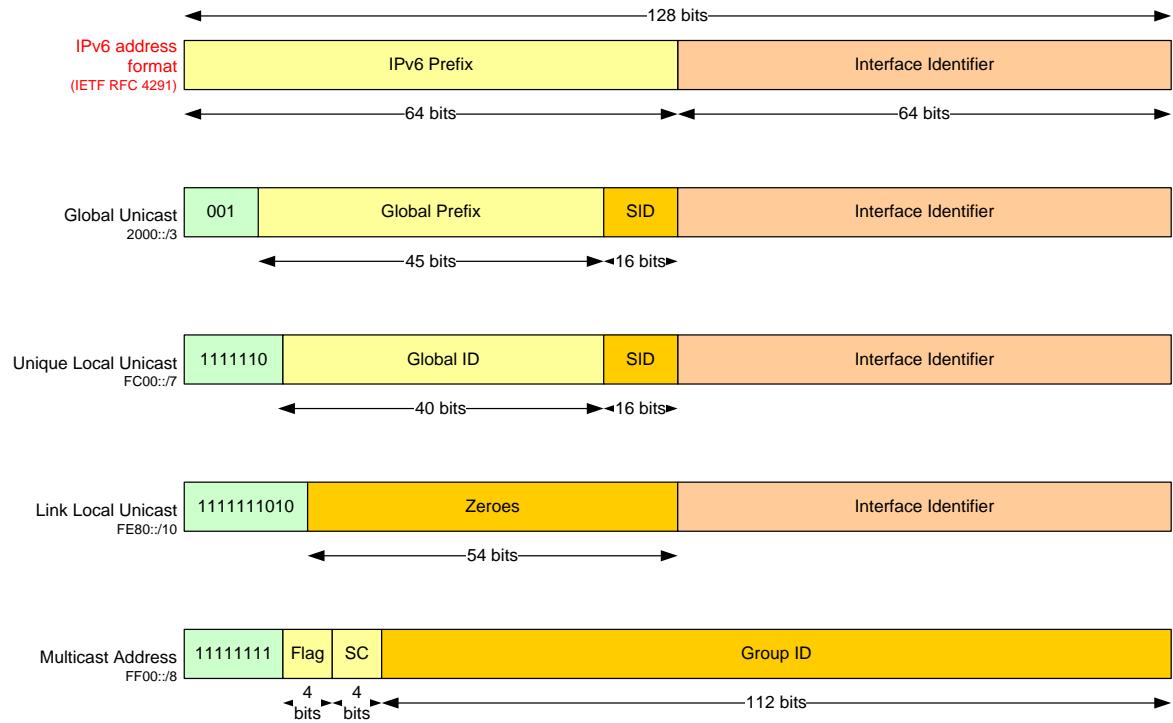
IPv6 addresses are typically composed of two logical parts: a 64-bit (sub-)network prefix used for routing, and a 64-bit host part used to identify a host within the network.

The formats allowed for an IPv6 address are shown in Figure C.1.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 645/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

(see <http://www.iana.org/assignments/ipv6-address-space/>). Note that to facilitate the IPv6 address writing, a specific notation defined in RFC 4291 has been specified by IETF (e.g. FF00::/8).



**Figure C.1 – IPv6 address formats**

Where:

- *Global Unicast* is a routable address in the whole internet network and is composed as follows:
  - Global prefix assigned by IANA (see <http://www.iana.org/assignments/ipv6-unicast-address-assignments/>);
  - Subnet ID (SID) allocated by the network administrator; and
  - Interface Identifier either generated from the interface's MAC address (using modified EUI-64 format), or obtained from a DHCPv6 server, or assigned manually;
- *Unique Local Unicast* is an address only applicable to local network. This type of address is not routable outside the local network. The Global ID and the Subnet ID (SID) are allocated by the network administrator;
- *Link Local Unicast* is a unicast address allowed for a link local (without router). This type of address is not routable outside a local link;
- *Multicast* is an address assigned to different devices of the network. Following the scope (SC) of the address, the multicast group may be either Interface-local, Link-local, Admin-local, Site-local, Organization-local or global. For more information about Flag and SC (scope) parameters, see RFC 4291, 2.7.

## COSEM Interface Classes

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:

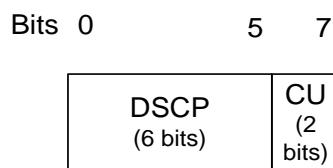


**Figure C.2 – IPv6 header format**

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;

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 647/668 |
|-----------------------|------------|-----------------------------|---------|

- *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 used;
- *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, etc.) 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.2.

Table C.2 summarizes the options managed / not managed by the IPv6 setup objects.

**Table C.2 – Optional IPv6 header extensions vs. IPv6 IC**

| Extensions   | IPv6 setup IC | See subclause |
|--|---------------|---------------|
| Hop-by-Hop options   | Not managed   | C.4.2         |
| Destination options  | Not managed   | C.4.3         |
| Routing options  | Not managed   | C.4.4         |
| Fragment options   | Not managed   | C.4.5         |
| Security options (Authentication and Encapsulating Security Payload)   | Not managed   | C.4.6         |
| NOTE The options are listed in the order as they appear in the packet. |               |               |

#### 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 2460RFC 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, etc.) (see RFC 3775); and
- Tunnelling options which permit to communicate between two IPv6 networks separated by an IPv4 network (6To4 mechanism) (see RFC 2460).

NOTE The Mobility parameters and the Tunnelling options have been defined separately from RFC 2460RFC 2460 and are not mentioned in this document. See appropriate RFCs for more details.

These optional elements are directly managed by the IPv6 protocol stack. These optional fields are not managed by the COSEM IPv6 setup class.

#### C.4.4 Routing options

Routing options element is used to specify some routing parameters (see **RFC 2460, 4.4**). Currently, only one type is defined by IETF (type 2), which is used in case of mobility IPv6.

This notion is out of scope of the COSEM IPv6 setup class.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 649/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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 4291 *IP Version 6 Addressing Architecture*

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 4291 *IP Version 6 Addressing Architecture*)
- 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.

**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 61334-4-32:1996;
- a convergence layer for IPv4;
- a convergence layer for IPv6;

The DLMS/COSEM communication profiles for narrow-band OFDM PLC PRIME neighbourhood networks are specified in draft IEC 62056-8-4,13/1749/CDV.

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 651/668 |
|-----------------------|------------|-----------------------------|---------|

## 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 IEC 62056-8-5, *Electricity metering data exchange –The DLMS/COSEM suite –*

*Part 8-5: Narrow-band OFDM G3-PLC communication profile*

*for neighbourhood networks*

ITU-T E.212 (05.2008), *Series E: Overall network operation, telephone service, service operation and human factors – International operation – Maritime mobile service and public land mobile service – The international identification plan for public networks and subscriptions*

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: 2006 type (extended), well suited to low data rates;

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 652/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

- IPv6, the new generation of IP (Internet Protocol), which widely opens the range of potential applications and services; and
- to allow good IPv6 and MAC interoperability, an Adaptation sublayer taken from the Internet world (IETF) and called 6LoWPAN (RFC 4944 extended and RFC 6282). The adaptation sub layer also embeds the LOADng routing algorithm to allow multi-hop mesh connectivity.

For more information about the extensions of IEEE 802.15.4: 2006 , RFC 4944 and RFC 6282 (AKA 6LoWPAN) standards, and LOADng, see ITU-T G.9903:2014.

The DLMS/COSEM narrow-band G3-PLC communication profile is specified in IEC 62056-8-5:2017.

## Bibliography

ANSI/TIA-4957.200     *Layer 2 Standard Specification for the Smart Utility Network*

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

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

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

IEC 62056-4-7:2015, *Electricity metering data exchange – The DLMS/COSEM suite – Part 4-7: DLMS/COSEM transport layer for IP networks*

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

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

ISO/IEC 10646:2014, *Information technology – Universal Coded Character Set (UCS)*

draft IEC 62056-8-4: 21XX, 13/1749/CDV, *Electricity metering data exchange – The DLMS/COSEM suite – Part 8-4: Communication profiles for narrow-band OFDM PLC PRIME neighbourhood networks*

IEC 62056-8-5, *Electricity metering data exchange –The DLMS/COSEM suite –*

*Part 8-5: Narrow-band OFDM G3-PLC communication profile*

*for neighbourhood networks*

ITU-T E.212 (05.2008), *Series E: Overall network operation, telephone service, service operation and human factors – International operation – Maritime mobile service and public land mobile service – The international identification plan for public networks and subscriptions*

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 653/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

ITU-T G.9901:2014, Series G: *Transmission systems and media, digital systems and networks – Access Networks – In premises networks – Narrow-band orthogonal frequency division multiplexing power line communication transceivers – Power spectral density specification*

ITU Recommendation X.217:1995, *Information technology – Open Systems Interconnection – Service definition for the association control service element*

ITU Recommendation X.227:1995, *Information technology – Open Systems Interconnection Connection-oriented protocol for the association control service element: Protocol specification*

IETF STD 5, *Internet Protocol*, 1981. (Also IETF RFC 0791, RFC 0792, RFC 0919, RFC 0922, RFC 0950, RFC 1112)

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

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

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

3GPP TS 36.214 V13.0.0 (2015-12), *Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer; Measurements*

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

RFC 768 *User Datagram Protocol*

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

RFC 793, *Transmission Control Protocol (Also IETF STD 0007)*, 1981, Updated by: RFC 3168

RFC 940, *Toward an Internet Standard Scheme for Subnetting*, 1985

RFC 950, *Internet Standard Subnetting Procedure*, 1985

RFC 1144, *Compressing TCP/IP Headers for Low-Speed Serial Links*, 1990

RFC 1213 *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*

RFC 1332, *The PPP Internet Protocol Control Protocol (IPCP)*, 1992, Updated by: RFC 3241. Obsoletes: RFC 1172

RFC 1570, *PPP LCP Extensions*, 1994

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

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 654/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

RFC 2433, *PPP CHAP Extension*, 1998

RFC 2460, *Internet Protocol, Version 6 (IPv6)*, 1998

RFC 2473, *Generic Packet Tunneling in IPv6*, 1998

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

RFC 2507, *IP Header Compression*, 1999

RFC 2508, *Compressing IP/UDP/RTP Headers for Low-Speed Serial Links*, 1999

RFC 2675, *IPv6 Jumbograms*, 1999

RFC 2711, *IPv6 Router Alert Option*, 1999

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

RFC 2986, *PKCS #10 v1.7: Certification Request Syntax Standard*

RFC 3095, *RObust Header Compression (ROHC): Framework and four profiles: RTP, UDP, ESP, and uncompressed*, 2001

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

RFC 3315 *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*

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

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

RFC 3748, *Extensible Authentication Protocol (EAP)*, 2004

RFC 3775, *Mobility Support in IPv6*, 2004

RFC 4291 *IP Version 6 Addressing Architecture*

RFC 4302, *IP Authentication Header*, 2005

RFC 4303, *IP Encapsulating Security Payload (ESP)*, 2005

RFC 4443 *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*

RFC 4944 *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*

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

RFC 4944, *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*, 2007

RFC 5095, *Deprecation of Type 0 Routing Headers in IPv6*, 2007

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 655/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

RFC 5216 *The EAP-TLS Authentication Protocol*

RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, 2008

RFC 5905, *Network Time Protocol Version 4: Protocol and Algorithms Specification*, 2010

RFC 6206, *The Trickle Algorithm*

RFC 6282, *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks* [online]. Edited by J. Hui, Ed. September 2011

RFC 6550 *RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks*

RFC 6775, *Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)*, 2012

RFC 7252, *The Constrained Application Protocol*.

RFC 7731, *Multicast Protocol for Low-Power and Lossy Networks (MPL)*

RFC 7774, *Multicast Protocol for Low-Power and Lossy Networks (MPL) Parameter Configuration Option for DHCPv6*

RFC 7959, *Block-Wise Transfers in the Constrained Application Protocol (CoAP)*

RFC 7967, *Constrained Application Protocol (CoAP) Option for No Server Response*

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

DLMS UA 1000-1, the “Blue Book” Ed. 12.1:2015, *COSEM interface classes and OBIS identification system*

DLMS UA 1000-2, the “Green Book” Ed. 8.1:2015, *DLMS/COSEM Architecture and Protocols*

DLMS UA 1001-1, the “Yellow Book”, Ed. 5.0:2015, *DLMS/COSEM Conformance test and certification process*

DLMS UA 1002, the “White Book”, Ed. 1.0:2003, *COSEM Glossary of terms*

3GPP TS 24.008 V13.7.0 (2016-10), *Technical Specification Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; Mobile radio interface Layer 3 specification; Core network protocols; Stage 3*

3GPP TS 24.301 V13.4.0 (2016-01), *Technical Specification Group Core Network and Terminals; Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3*

3GPP TS 24.301 V13.11.0 (2018-01), *Technical Specification Group Core Network and Terminals; Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS);Stage 3*

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 656/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

3GPP TS 27.007 V16.1.0 (2019-06) *Technical Specification Group Core Network and Terminals; AT command set for User Equipment (UE)*

3GPP TS 36.101 V15.4.0 (2019-01) *Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception*

3GPP TS 36.133 V13.11.0 (2018-04) *Technical Specification LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Requirements for support of radio resource management*

3GPP TS 36.133 V14.4.0 (2017-07) *Technical Specification LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Requirements for support of radio resource management*

3GPP TS 36.213 V15.5.0 (2019-05) *Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures*

3GPP TS 36.304 V13.0.0 (2015-12), *Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) procedures in idle mode*

3GPP TS 36.304 V13.8.0 (2018-01) *Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) procedures in idle mode*

3GPP TS 36.321 V15.5.0 (2019-05) *Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification*

3GPP TS 36.331 V15.5.1 (2019-05) *Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification*

ITU-T G.9903 Amd. 1:2013, *Series G: Transmission systems and media, digital systems and networks – Access networks – In premises networks – Narrow-band orthogonal frequency division multiplexing power line communication transceivers for G3-PLC networks*

NOTE This Recommendation is referenced in version 0 of the G3-PLC setup classes.

ITU-T G.9903:2014, *Series G: Transmission systems and media, digital systems and networks – Access networks – In premises networks – Narrow-band orthogonal frequency division multiplexing power line communication transceivers for G3-PLC networks*

NOTE This Recommendation is referenced in version 1 of the G3-PLC setup classes.

**ITU-T G.9903 Amd. 1:2021, *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***

**NOTE This Recommendation is referenced in version 2 of the G3-PLC setup classes.**

ITU-T G.9904:2012, *Series G: Transmission systems and media, digital systems and networks – Access networks – In premises networks – Narrow-band orthogonal frequency division multiplexing power line communication transceivers for PRIME networks*

EN 13757-1:2014, *Communication system for meters – Part 1: Data exchange*

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 657/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

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

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

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

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

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

EN 13757-3:2018, *Communication systems for and remote reading of meters – Part 3: Dedicated application layer*

NOTE This standard is referenced in the M-Bus client setup interface class version 2.

EN 13757-4:2013, *Communication system for and remote reading of meters – Part 4: Wireless meter (Radio meter reading for operation in SRD bands)*

EN 13757-4:2019, *Communication system for and remote reading of meters – Part 4: Wireless meter (Radio meter reading for operation in SRD bands)*

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

EN 13757-7:2018, *Communication systems for meters - Part 7: Transport and security services*

IEEE 802.15.4:2006, *Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*

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

**802.15.4:2015 IEEE Standard for Low-Rate Wireless Networks**

**802.15.4:2020 IEEE Standard for Low-Rate Wireless Networks**

ETSI GSM 05.08:1996, *Digital cellular telecommunications system (Phase 2+); Radio subsystem link control*

**ETSI EN 303 204 V2.1.2 (2016-09), Network Based Short Range Devices (SRD); Radio equipment to be used in the 870 MHz to 876 MHz frequency range with power levels ranging up to 500 mW; Harmonised Standard covering the essential requirements of article 3.2 of the Directive 2014/53/EU**

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

ZigBee® 053474 ZigBee® Specification. *The specification can be downloaded free of charge from <http://www.zigbee.org/Standards/ZigBeeSmartEnergy/Specification.aspx>*

[FANSPEC] Wi-SUN Alliance: *Field Area Network Working Group (FANWG):Technical Profile Specification:Field Area Network:Version 1v26.*

|         |            |                              |                       |
|---------|------------|------------------------------|-----------------------|
| 658/668 | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed. 15 | DLMS User Association |
|---------|------------|------------------------------|-----------------------|

## COSEM Interface Classes

[PHYSPEC] Wi-SUN Alliance: PHY Working Group (PHYWG) Wi-SUN PHY Specification Revision 1V02

LoRaWAN 1.0.3 LoRaWAN® Specification v1.0.3

<https://lora-alliance.org/resource-hub/lorawanr-specification-v103>

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

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 659/668 |
|-----------------------|------------|-----------------------------|---------|

## Index

- 61334-4-32 LLC SSCS setup ..... 326  
Abstract object ..... 594  
Access rights ..... 53  
Access selector ..... 435  
Account ..... 30, 207, 209, 573, 582  
**account\_mode\_and\_status** ..... 211  
actions ..... 203  
activate\_account ..... 218  
Active initiator ..... 25  
Active power ..... 607, 614  
Activity calendar ..... 179, 181, 573, 580  
Actor, Arbitrator ..... 202  
Advanced power failure monitoring ..... 597  
AES-GCM-128 ..... 125  
AES-GCM-256 ..... 125  
Agreed\_key ..... 149  
*Alarm descriptor* ..... 131, 602  
Alarm filter ..... 602  
Alarm monitor ..... 131, 581  
*Alarm register* ..... 131, 602  
Algorithm ..... 606, 607, 613  
All configured ..... 305  
All Physical ..... 305  
Apparent power ..... 607  
Application context ..... 53  
Arbitrator ..... 34, 596  
Array manager ..... 573  
Association ..... 573, 592  
Association LN ..... 98, 104, 445, 450, 456  
Association object ..... 324  
Association SN ..... 41, 98, 99, 431, 434, 436  
Association view ..... 52, 53  
Attribute ..... 17, 39, 41, 44, 111, 239, 391, 395, 397, 401, 403, 408, 412, 414  
Authenticated request ..... 125, 151  
Authenticated response ..... 125, 151  
Authentication key ..... 127  
Authentication mechanism ..... 53  
Auto answer ..... 250, 494  
Auto connect ..... 253, 498  
Auto dial ..... 496  
Base node ..... 26, 325  
base\_name ..... 41, 100, 434, 437, 442  
Battery ..... 597  
Beacon slot ..... 26  
Billing period ..... 574, 579, 605, 612, 617, 621, 626, 633  
Billing period counter ..... 576  
Billing period, end of ..... 580  
Block demand ..... 72  
Broadcast ..... 578  
CAD ..... 28  
Calculation methods, Gas ..... 631  
Calendar ..... 181  
Calling line identification ..... 252  
Capture period ..... 595, 601, 604, 608, 611, 614, 617, 621, 626, 633  
capture\_tim ..... 259, 504  
Captured object ..... 604, 608, 611, 614, 617, 621, 626, 633  
Certificate ..... 143  
Certificate Signing Request ..... 128  
Certificate, export ..... 129  
Certificate, import ..... 129

## COSEM Interface Classes

|  |                        |  |  |
|--|------------------------|--|--|
| Challenge .....                            | 450                    | COSEM physical device.....               | 302, 323   |
| change_HLS_secret.....                     | 436                    | Credit .....                             | 207, 219, 223, 573, 582  |
| Charge.....                                | 30, 207, 230, 573, 582 | credit mode .....                        | 30   |
| class_id .....                             | 105                    | Credit states.....                       | 219  |
| Client user identification .....           | 98                     | credit token .....                       | 34   |
| client_SAP.....                            | 107, 458               | Credit, emergency.....                   | 30   |
| Clock .....                                | 54, 171, 573, 577      | Credit, enabled.....                     | 219  |
| <i>Clock synchronization method</i> .....  | 605, 613               | Credit, exhausted.....                   | 219  |
| collect .....                              | 30                     | Credit, in use.....                      | 219  |
| Commodity .....                            | 30, 33                 | Credit, priority.....                    | 219  |
| Communication port log parameter .....     | 599                    | Credit, selectable .....                 | 219  |
| Communication tamper event.....            | 601                    | Credit, selected/invoked.....            | 219  |
| Compact data .....                         | 574                    | CT/VT ratio.....                         | 606, 613   |
| Compressibility .....                      | 630                    | Cumulative value .....                   | 604, 611   |
| Configuration.....                         | 268, 514, 595          | Current and last average values .....    | 604, 611   |
| Conformance block .....                    | 108                    | Current association .....                | 53   |
| Consumer message .....                     | 599                    | current_average_value .....              | 72   |
| <b>consumption_based_collection</b> .....  | 208                    | Currently active tariff .....            | 600  |
| <b>consumption_based_credit</b> .....      | 207                    | Data .                                   | 54, 64, 86, 183, 595, 598, 601, 605, 606, 609, 612, 613, 615, 618, 622, 627, 634 |
| <b>Contracted value</b> .....              | 604, 611               | Data format .....                        | 44, 46   |
| Convergence layer, PRIME NB OFDM PLC ..... | 650                    | Data model.....                          | 17   |
| Conversion .....                           | 630                    | Data profile objects – HCA.....          | 620, 625   |
| Conversion factor .....                    | 618, 622, 627          | Data profile objects – water meter ..... | 637  |
| Correction.....                            | 630                    | Data protection, COSEM.....              | 593  |
| COSEM Conformance Test Process.....        | 17                     | Data security, access .....              | 54   |
| COSEM Glossary of Terms .....              | 18                     | Data security, transport .....           | 54   |
| COSEM logical device name.....             | 52, 53                 | Data type.....                           | 44   |
| COSEM objects, Abstract.....               | 572                    | data_index .....                         | 80   |
| COSEM objects, Electricity.....            | 604, 611               | Date and time format .....               | 46   |
| COSEM objects, Gas .....                   | 626                    | Daylight saving.....                     | 171, 179   |
| COSEM objects, HCA .....                   | 617                    | Demand register .....                    | 54, 72, 76, 604, 611   |

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
|                       |            | 661/668                     |

## COSEM Interface Classes

|   |                              |   |  |
|---|------------------------------|---|--|
| Demotion .....                          | 28                           | Error registers –water .....                    | 636  |
| Device ID.....                          | 594                          | Event code.....                                 | 599  |
| <i>Device measuring principle</i> ..... | 618                          | Event counter.....                              | 600  |
| Device start-up .....                   | 329                          | Event log.....                                  | 603  |
| device_serial_number .....              | 492                          | execute .....                                   | 176  |
| Digital signature, profile entry .....  | 600                          | Extended register....                           | 70, 76, 86, 183, 196, 584,<br>597, 598, 599, 604, 605, 609, 611, 612, 615,<br>618, 619, 622, 627, 628, 634 |
| Digitally signed request .....          | 125, 151                     | Fatal error register .....                      | 248  |
| Digitally signed response.....          | 125                          | Firmware.....                                   | 576  |
| Disconnect control.....                 | 186, 578, 580, 596           | Floating point format .....                     | 49   |
| Disconnected state .....                | 27                           | Flow rate .....                                 | 630  |
| DLMS version.....                       | 108                          | Frequency notching, G3-PLC.....                 | 651  |
| ECDH P-384.....                         | 125                          | Friendly credit .....                           | 31   |
| ECDSA P-256 .....                       | 125                          | Full Function Device .....                      | 339  |
| Electrical port.....                    | 582                          | Function control .....                          | 160, 573, 578, 580, 593  |
| Electricity breaker.....                | 596                          | Fundamental .....                               | 607  |
| Emergency activation time .....         | 190                          | Fundamental component .....                     | 607  |
| Emergency credit .....                  | 30                           | G3-PLC .....                                    | 651  |
| Emergency duration .....                | 190                          | G3-PLC 6LoWPAN adaptation layer.....            | 589  |
| Emergency profile .....                 | 190                          | <b>G3-PLC 6LoWPAN adaptation layer setup</b> .. | 62,<br>357, 369, 526, 557, 564, 590  |
| Emergency profile, Limiter.....         | 191                          | <b>G3-PLC MAC layer counters</b>                | 62, 343, 364, 525,<br>589  |
| Emergency threshold.....                | 190                          | <b>G3-PLC MAC setup</b> ..                      | 62, 340, 344, 366, 525,<br>533, 538, 589   |
| <b>emergency_credit</b> .....           | 207                          | Gas ID number.....                              | 626, 633   |
| Enabled .....                           | 30                           | Gas law deviation.....                          | 630  |
| Encrypted request .....                 | 125, 151                     | Gas station management.....                     | 627  |
| Encrypted response .....                | 125, 151                     | Gas valve .....                                 | 596  |
| Encryption key.....                     | 110, 268, 436, 455, 506, 514 | <i>get_protected_attributes</i> .....           | 144, 151   |
| Enterprise Resource Planning.....       | 31                           | Global broadcast encryption key .....           | 127  |
| Environmental related parameters .....  | 598                          | Global unicast encryption key .....             | 127  |
| Ephemeral Unified Model .....           | 127                          | <i>global_broadcast_encryption_key</i> .....    | 150  |
| equipment_id.....                       | 492                          |   |  |
| Error register .....                    | 583, 601, 602, 620, 624, 636 |   |  |
| Error registers – HCA .....             | 620, 624                     |   |  |

## COSEM Interface Classes

|  |                                   |
|--|-----------------------------------|
| global_unicast_encryption_key .....                  | 150                               |
| GPRS modem setup ..                                  | 255, 259, 504, 573, 585           |
| GSM diagnostic.                                      | 256, 499, 502, 573, 586, 587, 591 |
| Harmonics .....                                      | 607, 610                          |
| Hayes standard .....                                 | 249                               |
| HCA ID number .....                                  | 617                               |
| HDLC setup.....                                      | 573, 583                          |
| HDLC setup class.....                                | 244, 489                          |
| Head End System .....                                | 31                                |
| High resolution clock .....                          | 577                               |
| HLS secret.....                                      | 110, 436, 455                     |
| HS-PLC ISO/IEC 12139-1 neighbourhood network .....   | 372                               |
| HS-PLC ISO/IEC 12139-1 networks.....                 | 573                               |
| I/O control signal .....                             | 595                               |
| ID numbers, Electricity.....                         | 604, 611                          |
| Identification number .....                          | 506                               |
| Identification system.....                           | 637                               |
| Identified_key .....                                 | 149                               |
| IEC 61334-4-32 LLC setup.....                        | 588                               |
| IEC local port setup .....                           | 573                               |
| IEC twisted pair (1) Fatal Error register .....      | 584                               |
| IEC twisted pair (1) MAC address setup.....          | 584                               |
| IEC twisted pair (1) setup ...                       | 246, 491, 583, 584                |
| IEEE address .....                                   | 29                                |
| IHD .....  | 28                                |
| Image .....  | 24, 119                           |
| <i>Image activation</i> .....                        | 578, 580                          |
| Image transfer .....                                 | 98, 112, 113, 573, 593            |
| Image, activation .....                              | 120                               |
| Image, completeness .....                            | 119                               |
| Image, initiate transfer.....                        | 119                               |
| ImageBlock .....                                     | 24                                |
| ImageBlockNumber .....                               | 25                                |
| ImageBlocks, transfer .....                          | 119                               |
| ImageBlockSize .....                                 | 24, 116                           |
| ImageSize.....                                       | 24                                |
| In Home Display.....                                 | 28                                |
| Inactive objects .....                               | 603                               |
| Index difference, Gas.....                           | 629, 634, 635                     |
| Index, Gas.....                                      | 629                               |
| Information security.....                            | 54                                |
| Information, measured .....                          | 40                                |
| Information, static.....                             | 40                                |
| Initiator .....                                      | 25, 305                           |
| Input pulse constant.....                            | 627                               |
| insert_entry .....                                   | 166                               |
| Install code.....                                    | 28                                |
| Instantaneous value...604, 609, 610, 611, 615, 616   |                                   |
| Instantiation.....                                   | 40, 64, 65, 184, 423              |
| Interface class .....                                | 39, 54, 58                        |
| Interface object .....                               | 39                                |
| Internal control signals .....                       | 596, 597                          |
| Internal operating status....596, 597, 608, 614, 629 |                                   |
| Internet.....  | 585                               |
| Interval .....                                       | 627                               |
| Invocation counter.....                              | 592                               |
| invoke_credit.....                                   | 227                               |
| <i>invoke_protected_method</i> .....                 | 145, 152                          |
| IPv4 setup.....                                      | 281, 573, 585                     |
| IPv6 addressing .....                                | 644                               |
| IPv6 header.....                                     | 646                               |
| IPv6 setup .....                                     | 573, 585, 591                     |

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 663/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

|                                      |  |                                       |                              |
|--------------------------------------|--|---------------------------------------|------------------------------|
| ISO/IEC 8802-2 LLC layer setup ..... | 573  | MAC sub-layer .....                   | 302                          |
| ISO/IEC 8802-2 LLC Type 1 setup .... | 319, 588   | Managed payment mode .....            | 32                           |
| ISO/IEC 8802-2 LLC Type 2 setup .... | 320, 588   | Management Logical Device.....        | 52, 53, 323                  |
| ISO/IEC 8802-2 LLC Type 3 setup .... | 321, 588   | manual_disconnect.....                | 188                          |
| Key agreement .....                  | 125  | manual_reconnect .....                | 188                          |
| Key information .....                | 154  | Manufacturer ID .....                 | 506                          |
| key_info .....                       | 149  | manufacturer_id .....                 | 492                          |
| last_average_value .....             | 72   | Master key .....                      | 127, 143                     |
| Limiter.....                         | 190, 573, 581  | master_key .....                      | 150                          |
| Link key .....                       | 29   | Max credit limit.....                 | 582                          |
| List objects – Electricity .....     | 609, 615, 636  | Max vend limit.....                   | 582                          |
| List objects – Gas .....             | 632  | Maximum and minimum values.....       | 604, 611                     |
| List objects – General.....          | 603  | M-Bus.....                            | 573                          |
| List objects – HCA.....              | 620  | M-Bus client .....                    | 506, 527, 528, 552, 584, 640 |
| List objects – Thermal Energy .....  | 625  | M-Bus control log object.....         | 584                          |
| LLC layer.....                       | 317, 522   | M-Bus diagnostics.....                | 585                          |
| LLC sub-layer.....                   | 302  | M-Bus disconnect control object ..... | 584                          |
| LLS secret .....                     | 103, 436, 439, 444                                     | M-Bus master port setup.....          | 273, 584, 585                |
| Load limiting .....                  | 31   | M-Bus port setup.....                 | 506                          |
| <i>Load profile control</i> .....    | 578, 580   | <i>M-Bus profile control</i> .....    | 578, 580                     |
| Local port setup.....                | 242, 487, 582  | M-Bus profile generic object.....     | 506, 584                     |
| local_disconnect.....                | 188  | M-Bus slave .....                     | 506, 584                     |
| local_reconnect .....                | 188  | M-Bus slave port setup .....          | 263, 584                     |
| Logical device....                   | 51, 52, 53, 99, 104, 111, 112, 434, 436, 440, 450, 456 | M-Bus value object .....              | 506                          |
| Logical Device Name....              | 41, 302, 324, 573, 592                                 | Measurement period .....              | 605, 612                     |
| Logical name .....                   | 40   | Medium access control layer.....      | 650                          |
| Logical name referencing .....       | 104, 450, 456  | Messaging.....                        | 17                           |
| logical_name .....                   | 39   | Meter open event .....                | 601                          |
| LTE monitoring .....                 | 573, 586   | Meter reset.....                      | 579                          |
| MAC address.....                     | 29   | Meter tamper event.....               | 601                          |
| MAC address setup.....               | 573, 585, 589  | Metering point ID.....                | 595, 608, 614                |
|                                      |  | Method .....                          | 17, 39, 44                   |

## COSEM Interface Classes

|   |                                  |
|---|----------------------------------|
| metrological_ .....                                   | 198                              |
| Metrology tamper event.....                           | 601                              |
| MIB variables.....                                    | 302                              |
| Modelling.....  | 17                               |
| Modem .....   | 248, 249, 496, 578               |
| Modem configuration.....                              | 248, 573, 578                    |
| Natural gas analysis .....                            | 631                              |
| NB OFDM PLC for G3-PLC networks.....                  | 573                              |
| NB OFDM PLC for PRIME networks .....                  | 573                              |
| NB OFDM PLC profile for PRIME networks                | 323                              |
| NEW .....   | 305                              |
| New system.....                                       | 25                               |
| New system title .....                                | 25                               |
| NO-BODY.....  | 305                              |
| Node.....   | 26                               |
| Nominal value.....                                    | 627                              |
| Nonce .....   | 154                              |
| Normal mode.....                                      | 579                              |
| Normal threshold.....                                 | 190                              |
| Notation .....  | 41                               |
| NTP protocol .....                                    | 294                              |
| NTP setup .....                                       | 573, 586                         |
| number_of_periods .....                               | 72                               |
| OBIS.....   | 40, 637                          |
| Object Identification System.....                     | 572                              |
| <i>object_list</i> .....                              | 53, 106, 107, 111, 449, 454, 461 |
| Occurrence counter.....                               | 604, 611                         |
| One-Pass Diffie-Hellman .....                         | 154                              |
| Operating time.....                                   | 598                              |
| Optical port.....                                     | 582                              |
| Optical test output .....                             | 579                              |
| originator_system_title.....                          | 149                              |
| Orthogonal Frequency Division Multiplexing .....      | 650                              |
| other_information .....                               | 149                              |
| Output control .....                                  | 580                              |
| Output pulse constant.....                            | 605, 612, 627                    |
| Output signals .....                                  | 579                              |
| Packet switched network.....                          | 254                              |
| PAN coordinator.....                                  | 339                              |
| Parameter changes.....                                | 595                              |
| Parameter monitor .....                               | 193, 485, 573, 581               |
| Parameter monitor log .....                           | 193, 485                         |
| Password .....  | 103, 436, 439, 444               |
| Payment metering.....                                 | 207                              |
| <b>payment_event_based_collection</b> .....           | 208                              |
| Period .....  | 72                               |
| Permission .....                                      | 201                              |
| Personal Area Network .....                           | 377                              |
| Physical device .....                                 | 51, 111, 592                     |
| Physical layer, PRIME NB OFDM PLC .....               | 650                              |
| PLC OFDM Type 1 Application identification .....      | 589                              |
| PLC OFDM Type 1 MAC counters.....                     | 589                              |
| PLC OFDM Type 1 MAC functional parameters .....       | 589                              |
| PLC OFDM Type 1 MAC network administration data ..... | 589                              |
| PLC OFDM Type 1 MAC setup .....                       | 589                              |
| PLC OFDM Type 1 physical layer counters               | 588                              |
| Post-payment.....                                     | 32                               |
| Power factor.....                                     | 607                              |
| Power factor, displacement.....                       | 607                              |
| Power factor, true.....                               | 607                              |
| <b>Power failure</b> .....                            | 179, 597                         |
| Power failure duration .....                          | 598                              |

|                       |            |                             |
|-----------------------|------------|-----------------------------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 |
|                       |            | 665/668                     |

## COSEM Interface Classes

|  |  |
|--|--|
| PPP setup .....  | 289, 573, 585                                    |
| <b>Preferred readout</b> .....   | <b>83</b>  |
| Primary address .....  | 506  |
| PRIME NB OFDM PLC Applications identification.....   | 337  |
| PRIME NB OFDM PLC MAC counters.....  | 333  |
| PRIME NB OFDM PLC MAC functional parameters.....   | 331  |
| PRIME NB OFDM PLC MAC setup.....   | 329  |
| PRIME NB OFDM PLC Physical layer counters .....  | 328  |
| <i>priority</i> .....  | 219  |
| Process value.....   | 65, 630  |
| Profile .....  | 423  |
| Profile entries ....   | 595, 601, 604, 608, 611, 614, 617, 621, 626, 633 |
| Profile generic 54, 77, 575, 576, 595, 601, 604, 608, 611, 614, 617, 621, 626, 633               |  |
| Program entries.....   | 605, 612   |
| Promotion.....   | 27   |
| Proprietary.....   | 40   |
| Protection parameters .....  | 143  |
| <i>protection_buffer</i> .....   | 144, 145, 147                                    |
| PSTN modem configuration .....   | 493  |
| Public client .....  | 53   |
| Push .....   | 130, 578, 580                                    |
| Push setup .....   | 132, 475, 573, 586, 587                          |
| Push window .....  | 133, 476   |
| random delay.....  | 133, 465, 476                                    |
| Reactive power.....  | 607  |
| Reading factor .....   | 606, 613   |
| Readout.....   | 83, 573  |
| recipient_system_title.....  | 149  |
| Reduced Function Device .....  | 339  |
| Register40, 65, 71, 76, 86, 183, 598, 604, 605, 609, 611, 612, 615, 618, 619, 622, 627, 628, 634 |  |
| Register activation .....  | 76, 573, 580                                     |
| Register monitor..   | 183, 196, 573, 581, 610, 616                     |
| Register table.....  | 85   |
| <i>register_assignment</i> .....   | 76   |
| Registered system .....  | 26   |
| Registration.....  | 27   |
| remote_disconnect.....   | 188, 189   |
| remote_reconnect.....  | 188, 190   |
| remove_entries .....   | 167  |
| repayable .....  | 32   |
| repayment mode .....   | 32   |
| Repetition phase .....   | 313  |
| Reporting system .....   | 26   |
| Reporting system list.....   | 318  |
| Reserved base_names .....  | 41   |
| Reserved credit.....   | 32   |
| <b>reserved_credit</b> .....   | <b>207</b>                                       |
| reset .....  | 199  |
| Reset counter.....   | 617, 621, 626, 633                               |
| reset_account .....  | 219  |
| retrieve_entries .....   | 165  |
| SAP .....  | 111, 112   |
| SAP assignment .....   | 41, 53, 98, 111, 573, 592                        |
| Schedule.....  | 176, 179, 181, 573, 579                          |
| Script.....  | 174, 179   |
| Script table .....   | 41, 174, 181, 184, 573, 578                      |
| Script, null .....   | 175  |
| Secret .....   | 103, 111, 440, 444, 450                          |
| Secret, new .....  | 436  |
| Security mechanisms.....   | 54   |

## COSEM Interface Classes

|  |                    |  |                              |
|--|--------------------|--|------------------------------|
| Security setup.....  | 98, 592            | Standard readout profile.....                            | 583                          |
| Security suite.....  | 143                | Static Unified Model.....                                | 154                          |
| Selectable.....  | 33                 | Status mapping .....                                     | 87                           |
| Selected/Invoked.....  | 33                 | Status register.....                                     | 598                          |
| <b>Selective access</b> 44, 80, 81, 84, 90, 100, 106, 147, 442, 443, 449, 454, 461 |                    | Status value .....                                       | 65                           |
| Selective access parameters.....   | 44                 | Strong DC magnetic field event .....                     | 601                          |
| Sensor manager.....  | 196, 628           | Subnetwork.....  | 27                           |
| Sensor, pressure .....   | 199                | Sub-slot.....  | 26                           |
| Serial number, sensor .....  | 197                | superior calorific value .....                           | 630                          |
| Server model.....  | 51                 | Switch node .....  | 325                          |
| server_SAP .....   | 107, 458           | Switch state.....  | 27                           |
| Service node .....   | 27                 | Synchronization .....                                    | 179                          |
| Service Specific Convergence Sublayer....  | 326                | Tamper.....  | 601                          |
| set_amount_to_value .....  | 227                | Tarification .....                                       | 76, 579, 580                 |
| set_protected_attributes .....   | 144, 151           | TCP-UDP setup .....                                      | 280, 573, 585                |
| S-FSK Active initiator.....  | 587                | Temporary debt.....                                      | 33                           |
| S-FSK MAC counters .....   | 588                | Terminal cover open event.....                           | 601                          |
| S-FSK MAC synchronization timeouts.....  | 587                | Terminal node .....                                      | 325                          |
| S-FSK Phy&MAC setup.....   | 587                | Terminal state .....                                     | 27                           |
| S-FSK Physical layer.....  | 302                | Test mode .....  | 579                          |
| S-FSK PLC setup .....  | 573                | Thermal Energy meter ID number.....                      | 621                          |
| S-FSK Reporting system list.....   | 588                | Threshold value . 190, 609, 615, 618, 619, 622, 627, 634 |                              |
| Shared line .....  | 251                | Threshold, missing.....                                  | 609, 615                     |
| Short name.....  | 41                 | Threshold, over limit.....                               | 609, 615                     |
| Single action schedule.....  | 185, 573, 580      | Threshold, under limit .....                             | 609, 615                     |
| Sliding demand.....  | 72                 | Tilt event .....   | 601                          |
| SMTP setup.....  | 293, 586, 591      | <b>Time changes</b> .....                                | 179                          |
| Social credit.....   | 33                 | Time entries .....                                       | 605, 612, 618, 622, 627, 634 |
| Sort method595, 601, 604, 608, 611, 614, 617, 621, 626, 633                        |                    | Time integral .....                                      | 604, 611                     |
| Special days .....   | 176                | <b>Time setting backward</b> .....                       | 179                          |
| Special days table .....   | 179, 181, 573, 579 | <b>Time setting forward</b> .....                        | 179                          |

|                       |            |                             |         |
|-----------------------|------------|-----------------------------|---------|
| DLMS User Association | 2021-12-21 | DLMS UA 1000-1 Part 2 Ed 15 | 667/668 |
|-----------------------|------------|-----------------------------|---------|

## COSEM Interface Classes

|                                    |               |                                    |                            |
|------------------------------------|---------------|------------------------------------|----------------------------|
| <b>Time synchronization</b> .....  | 179           | Version, previous .....            | 423                        |
| <b>time_based_collection</b> ..... | 208           | Wake-up.....                       | 638                        |
| <b>time_based_credit</b> .....     | 207           | Wake-up request.....               | 252                        |
| Timeslot.....                      | 26            | Weighting.....                     | 201                        |
| Token .....                        | 33            | Wireless Mode Q .....              | 573, 584                   |
| Token carrier .....                | 33            | Wireless Mode Q channel.....       | 272                        |
| Token Carrier Interface.....       | 236           | Wrapped_key.....                   | 149                        |
| Token gateway .....                | 236, 573, 582 | xDLMS context.....                 | 53                         |
| <b>token_credit</b> .....          | 207           | year_of_manufacture .....          | 492                        |
| Top-up.....                        | 34            | ZigBee® .....                      | 28, 29, 137, 377, 573, 590 |
| transaction_id .....               | 149           | ZigBee® 053474 .....               | 657                        |
| Transmission phase .....           | 313           | ZigBee® 2007 .....                 | 378                        |
| Transporting .....                 | 17            | ZigBee® client.....                | 29                         |
| Twisted pair .....                 | 246, 491      | ZigBee® cluster .....              | 29                         |
| Twisted pair setup .....           | 573           | ZigBee® Communications hub .....   | 377                        |
| Unit.....                          | 67            | ZigBee® coordinator .....          | 29, 377                    |
| UNIX clock.....                    | 577           | ZigBee® mirror.....                | 29                         |
| update_amount .....                | 227           | ZigBee® network control.....       | 383, 590                   |
| update_entry .....                 | 166           | ZigBee® PAN creation .....         | 377                        |
| UTC .....                          | 171, 172      | ZigBee® Pro .....                  | 29, 378                    |
| Utility tables .....               | 83, 573, 594  | ZigBee® router .....               | 29                         |
| V.44 compression.....              | 125           | ZigBee® SAS APS fragmentation..... | 382, 383, 590              |
| Value .....                        | 40            | ZigBee® SAS join .....             | 381, 590                   |
| Value group C .....                | 573           | ZigBee® SAS startup .....          | 379, 590                   |
| Value group D .....                | 604, 611      | ZigBee® server .....               | 29, 377                    |
| Value group F.....                 | 609, 615      | ZigBee® Trust Center .....         | 30, 377                    |
| Vend .....                         | 34            | ZigBee® tunnel setup .....         | 389, 590                   |
| Version .....                      | 105, 423, 637 |                                    |                            |