

DIN EN 62056-6-2

ICS 91.140.50

Einsprüche bis 2011-12-17
Vorgesehen als Ersatz für
DIN EN 62056-62:2007-08

Entwurf

**Datenkommunikation der elektrischen Energiemessung –
DLMS/COSEM –
Teil 6-2: COSEM Interface classes (IEC 13/1466/CDV:2010);
Englische Fassung FprEN 62056-6-2:2010**

Electricity Metering Data Exchange –
The DLMS/COSEM Suite –
Part 6.2: COSEM Interface classes (IEC 13/1466/CDV:2010);
English version FprEN 62056-6-2:2010

Anwendungswarnvermerk

Dieser Norm-Entwurf mit Erscheinungsdatum 2011-10-10 wird der Öffentlichkeit zur Prüfung und Stellungnahme vorgelegt.

Weil die beabsichtigte Norm von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses Entwurfes besonders zu vereinbaren.

Stellungnahmen werden erbeten

- vorzugsweise als Datei per E-Mail an dke@din.de in Form einer Tabelle. Die Vorlage dieser Tabelle kann im Internet unter www.din.de/stellungnahme oder für Stellungnahmen zu Norm-Entwürfen der DKE unter www.dke.de/stellungnahme abgerufen werden;
- oder online im Norm-Entwurfs-Portal des DIN unter www.entwuerfe.din.de, sofern dort wiedergegeben;
- oder in Papierform an die DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (Hausanschrift: Stresemannallee 15, 60596 Frankfurt am Main).

Die Empfänger dieses Norm-Entwurfs werden gebeten, mit ihren Kommentaren jegliche relevanten Patentrechte, die sie kennen, mitzuteilen und unterstützende Dokumentationen zur Verfügung zu stellen.

Gesamtumfang 186 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

Nationales Vorwort

Die Englische Fassung des europäischen Dokuments FprEN 62056-6-2:2010 „Datenkommunikation der elektrischen Energiemessung – DLMS/COSEM – Teil 6-2: COSEM Interface classes“ (Entwurf in der Umfrage) ist unverändert in diesen Norm-Entwurf übernommen worden.

Die Internationale Elektrotechnische Kommission (IEC) und das Europäische Komitee für Elektrotechnische Normung (CENELEC) haben vereinbart, dass ein auf IEC-Ebene erarbeiteter Entwurf für eine Internationale Norm zeitgleich (parallel) bei IEC und CENELEC zur Umfrage (CDV-Stadium) und Abstimmung als FDIS (en: Final Draft International Standard) bzw. Schluss-Entwurf für eine Europäische Norm gestellt wird, um eine Beschleunigung und Straffung der Normungsarbeit zu erreichen. Dem entsprechend ist das internationale Dokument IEC 13/1466/CDV:2010 „Electricity Metering Data Exchange – The DLMS/COSEM Suite – Part 6.2: COSEM Interface classes“ unverändert in den Entwurf FprEN 62056-6-2:2010 übernommen worden.

Das internationale Dokument wurde vom TC 13 „Equipment for electrical energy measurement and load control“ der Internationalen Elektrotechnischen Kommission (IEC) erarbeitet.

Das DKE/K 461 hat auf seiner Sitzung am 2011-01-12 festgestellt, dass die Bedingungen des DIN-Präsidialbeschlusses 1/2004 für den vorliegenden Norm-Entwurf erfüllt sind, den vorliegenden Norm-Entwurf in englischer Sprache zu veröffentlichen. Auch die mit diesem Norm-Entwurf beabsichtigte Norm soll in englischer Sprache veröffentlicht werden. Sie steht nicht in unmittelbarem Zusammenhang mit Rechtsvorschriften und ist nicht als Sicherheitsnorm anzusehen. Es kann davon ausgegangen werden, dass die Anwender über hinreichende englische Sprachkenntnisse verfügen. Für viele der verwendeten Benennungen existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender auf dem Gebiet der „Datenkommunikation der elektrischen Energiemessung“ in der Regel ebenfalls der englischen Benennungen bedienen.

Bei der Abstimmung zu dem Europäischen Schluss-Entwurf bei CENELEC und dem Internationalen Schluss-Entwurf bei IEC [Final Draft International Standard (FDIS)] sind jeweils nur „JA/NEIN“-Entscheidungen möglich, wobei „NEIN“-Entscheidungen fundiert begründet werden müssen. Dokumente, die bei CENELEC als Europäische Norm angenommen und ratifiziert werden, sind unverändert als Deutsche Normen zu übernehmen.

Für diesen Norm-Entwurf ist das nationale Arbeitsgremium K 461 „Elektrizitätszähler“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (www.dke.de) zuständig.

Für den Fall einer undatierten Verweisung im normativen Text (Verweisung auf eine Norm ohne Angabe des Ausgabedatums und ohne Hinweis auf eine Abschnittsnummer, eine Tabelle, ein Bild usw.) bezieht sich die Verweisung auf die jeweils neueste gültige Ausgabe der in Bezug genommenen Norm.

Für den Fall einer datierten Verweisung im normativen Text bezieht sich die Verweisung immer auf die in Bezug genommene Ausgabe der Norm.

Der Zusammenhang der zitierten Normen mit den entsprechenden Deutschen Normen ergibt sich, soweit ein Zusammenhang besteht, grundsätzlich über die Nummer der entsprechenden IEC-Publikation. Beispiel: IEC 60068 ist als EN 60068 als Europäische Norm durch CENELEC übernommen und als DIN EN 60068 ins Deutsche Normenwerk aufgenommen.

Änderungen

Gegenüber DIN EN 62056-6-2:2007-08 wurden folgende Änderungen vorgenommen:

- a) Dieser Teil der IEC 62056 legt ein Modell für einen Zähler aus der Sicht seiner Kommunikationsschnittstelle(n) fest. Generische Gebäudeblocks werden definiert und dabei objektorientierte Verfahren angewendet, in der Form von Schnittstellenklassen zu Modellzählern von einfacher bis hin zu sehr komplexer Funktionalität.
- b) Diese dritte Ausgabe ersetzt die zweite Ausgabe, die 2006 veröffentlicht wurde. Sie ist eine technische Überarbeitung.
- c) Der Titel der überarbeiteten und neuen Teile der Reihe IEC 62056 wurde geändert.

- d) Die Benummerung wurde geändert von IEC 62056-XY in IEC 62056-X-Y. IEC 62056-62 wird z. B. IEC 62056-6-2;
- e) Der Text der Einleitung wurde geändert;
- f) In Abschnitt 3 wurden neue Abkürzungen hinzugefügt;
- g) Der Abschnitt 4.1 Allgemeines wurde geändert;
- h) Neu 4.2, Referencing methods wurde ergänzt, basierend auf der Einleitung für den früheren Anhang C;
- i) Neu 4.3, Reserved base_names for special COSEM objects wurde ergänzt, basierend auf dem früheren Anhang C.3;
- j) In 4.4, Class description notation: das Schnittstellenklassen-Template – und die Spezifikation jeder Schnittstellenklasse (IC) – enthält nun die Zuordnung zu Kurznamen;
- k) In 4.8.4 Mandatory contents of a COSEM logical device wurde eine Anmerkung ergänzt;
- l) In 4.5 Common data types wurde ein neuer Datentyp, UTF 8-string [12] ergänzt. Dieser Typ wurde gegebenenfalls zu den CHOICES hinzugefügt;
- m) Die Beschreibung der Funktion der ICs wurde an mehreren Stellen geändert.
- n) In 5.2 Tabelle 3 wurden neue Größen ergänzt: 65 Spezifische Energie und 67 Signalstärke;
- o) Abschnitt 6 Relation to OBIS enthält nun den Inhalt des früheren Anhangs D, ergänzt um die neuen Schnittstellenklassendefinitionen, OBIS-Code-Zuweisungen und Datentypen;

This page is left intentionally empty.

CONTENTS

FOREWORD.....	7
INTRODUCTION	11
1 Scope.....	12
2 Normative references.....	12
3 Terms, definitions and abbreviations	14
4 Basic principles	15
4.1 General.....	15
4.2 Referencing methods.....	16
4.3 Reserved base_names for special COSEM objects.....	17
4.4 Class description notation	17
4.5 Common data types	19
4.6 Data formats.....	20
4.6.1 Date and time formats	20
4.6.2 Floating point number formats	22
4.7 The COSEM server model.....	24
4.8 The COSEM logical device.....	24
4.8.1 General.....	24
4.8.2 COSEM logical device name	25
4.8.3 The “association view” of the logical device	25
4.8.4 Mandatory contents of a COSEM logical device.....	25
4.8.5 Management logical device	25
4.9 Data security	26
5 The COSEM interface classes	26
5.1 Overview.....	26
5.2 Interface classes for parameters and measurement data.....	29
5.2.1 Data (class_id: 1, version: 0).....	29
5.2.2 Register (class_id: 3, version: 0).....	29
5.2.3 Extended register (class_id: 4, version: 0)	33
5.2.4 Demand register (class_id: 5, version: 0).....	34
5.2.5 Register activation (class_id: 6, version: 0).....	37
5.2.6 Profile generic (class_id: 7, version: 1).....	38
5.2.7 Utility tables (class_id: 26, version: 0)	43
5.2.8 Register table (class_id: 61, version: 0).....	44
5.2.9 Status mapping (class_id: 63, version: 0).....	46
5.3 Interface classes for access control and management.....	47
5.3.1 Association SN (class_id: 12, version: 2)	47
5.3.2 Association LN (class_id: 15, version: 1)	50
5.3.3 SAP assignment (class_id: 17, version: 0)	55
5.3.4 Image transfer (class_id: 18, version: 0).....	56
5.3.4.1 Introduction.....	56
5.3.4.2 Definitions related to the Image transfer process	56
5.3.4.3 The Image transfer mechanism	57
5.3.4.3.1 General	57
5.3.4.4 The Image transfer IC definition	62
5.3.5 Security setup (class_id: 64, version: 0)	64
5.4 Interface classes for time- and event bound control	

5.4.1	Clock (class_id: 8, version: 0).....	65
5.4.2	Script table (class_id: 9, version: 0)	68
5.4.3	Schedule (class_id: 10, version: 0).....	69
5.4.4	Special days table (class_id: 11, version: 0).....	72
5.4.5	Activity calendar (class_id: 20, version: 0)	73
5.4.6	Register monitor (class_id: 21, version: 0)	75
5.4.7	Single action schedule (class_id: 22, version: 0)	76
5.4.8	Disconnect control (class_id: 70, version: 0)	77
5.4.9	Limiter (class_id: 71, version: 0)	80
5.4.10	Sensor manager interface class (class_id:67, version: 0)	82
	5.4.10.1 Introduction.....	82
	5.4.10.2 The Sensor manager interface class specification	82
	5.4.10.3 Example for absolute pressure sensor	85
5.5	Interface classes for setting up data exchange via local ports and modems.....	86
5.5.1	IEC local port setup (class_id: 19, version: 1).....	86
5.5.2	IEC HDLC setup (class_id: 23, version: 1)	87
5.5.3	IEC twisted pair (1) setup (class_id: 24, version: 0)	89
5.5.4	Modem configuration (class_id: 27, version: 1).....	90
5.5.5	Auto answer (class_id: 28, version: 0)	91
5.5.6	Auto connect (class_id: 29, version: 1)	92
5.6	Interface classes for setting up data exchange via M-Bus.....	94
5.6.1	M-Bus slave port setup (class_id: 25, version: 0).....	94
5.6.2	M-Bus client (class_id: 72, version: 0)	94
5.6.3	Wireless Mode Q channel (class_id: 73, version: 1)	98
5.6.4	M-Bus master port setup (class_id: 74, version: 0).....	99
5.7	Interface classes for setting up data exchange over the Internet	100
5.7.1	TCP-UDP setup (class_id: 41, version: 0)	100
5.7.2	IPv4 setup (class_id: 42, version: 0).....	101
5.7.3	MAC address setup (class_id: 43, version: 0).....	104
5.7.4	PPP setup (class_id: 44, version: 0).....	104
5.7.5	GPRS modem setup (class_id: 45, version: 0).....	108
5.7.6	SMTP setup (class_id: 46, version: 0)	109
5.8	Interface classes for setting up data exchange using S-FSK PLC.....	110
5.8.1	Introduction	110
5.8.2	Definitions and abbreviations related to the S-FSK PLC profile	110
5.8.3	Overview	111
5.8.4	S-FSK Phy&MAC set-up (class_id: 50, version: 1)	113
5.8.5	S-FSK Active initiator (class_id: 51, version: 0)	117
5.8.6	S-FSK MAC synchronization timeouts (class_id: 52, version: 0)	119
5.8.7	S-FSK MAC counters (class_id: 53, version: 0)	121
5.8.8	IEC 61334-4-32 LLC setup (class_id: 55, version: 1).....	124
5.8.9	S-FSK Reporting system list (class_id: 56, version: 0)	125
5.9	Interface classes for setting up the LLC layer for ISO/IEC 8802-2	127
5.9.1	General.....	127
5.9.2	Definitions related to the ISO/IEEE 8802-2 LLC layer.....	127
5.9.3	ISO/IEC 8802-2 LLC Type 1 setup (class_id: 57, version: 0).....	127
5.9.4	ISO/IEC 8802-2 LLC Type 2 setup (class_id: 58, version: 0).....	127
5.9.5	ISO/IEC 8802-2 LLC Type 3 setup (class_id: 59, version: 0).....	129
5.10	Maintenance of the interface classes.....	

5.10.1	New versions of interface classes	131
5.10.2	New interface classes.....	131
5.10.3	Removal of interface classes.....	131
6	Relation to OBIS	132
6.1	General.....	132
6.2	Abstract COSEM objects.....	132
6.2.1	Use of value group C.....	132
6.2.2	Data of historical billing periods	133
6.2.3	Billing period values / reset counter entries	134
6.2.4	Clock objects (class_id: 8)	134
6.2.5	Modem configuration and related objects.....	135
6.2.6	Script table objects (class_id: 9)	135
6.2.7	Special days table objects (class_id: 11)	136
6.2.8	Schedule objects (class_id: 10)	136
6.2.9	Activity calendar objects (class_id: 20).....	136
6.2.10	Register activation objects (class_id: 6).....	136
6.2.11	Single action schedule objects (class_id: 22)	136
6.2.12	Register monitor objects (class_id: 21).....	137
6.2.13	Limiter objects (class_id: 71).....	137
6.2.14	IEC local port setup objects (class_id: 19)	137
6.2.15	Standard readout profile objects (class_id: 7).....	137
6.2.16	IEC HDLC setup objects (class_id: 23).....	138
6.2.17	IEC twisted pair (1) setup objects (class_id: 24)	138
6.2.18	Objects related to data exchange over M-Bus	138
6.2.19	Objects to set up data exchange over the Internet	139
6.2.20	Objects for setting up data exchange using S-FSK PLC	139
6.2.21	Objects for setting up the ISO/IEC 8802-2 LLC layer.....	140
6.2.22	Association objects (class_id: 12, 15).....	140
6.2.23	SAP assignment object (class_id: 17).....	140
6.2.24	COSEM logical device name object.....	141
6.2.25	Security setup and frame counter objects (class_id: 64)	141
6.2.26	Image transfer objects (class_id: 18).....	141
6.2.27	Utility table objects (class_id: 26).....	141
6.2.28	Device ID objects	142
6.2.29	Metering point ID objects	142
6.2.30	Parameter changes and calibration objects	143
6.2.31	I/O control signal objects.....	143
6.2.32	Disconnect control objects (class_id: 70)	143
6.2.33	Status of internal control signals objects.....	143
6.2.34	Internal operating status objects	144
6.2.35	Battery entries objects	144
6.2.36	Power failure monitoring objects	144
6.2.37	Operating time objects.....	145
6.2.38	Environment related parameters objects	145
6.2.39	Status register objects	145
6.2.40	Event code objects.....	146
6.2.41	Communication port log parameter objects	146
6.2.42	Consumer message objects	146
6.2.43	Currently active tariff objects.....	146

6.2.44	Event counter objects	147
6.2.45	Error register objects	147
6.2.46	Alarm registers and alarm filters objects	147
6.2.47	General list objects (class_id: 7)	148
6.2.48	Event log objects	148
6.2.49	Inactive objects	148
6.3	Electricity related COSEM objects	150
6.3.1	Value group D definitions	150
6.3.2	Electricity ID numbers	150
6.3.3	Billing period values / reset counter entries	150
6.3.4	Other electricity related general purpose objects	150
6.3.5	Measurement algorithm	151
6.3.6	Metering point ID (electricity related)	153
6.3.7	Electricity related status objects	153
6.3.8	Electricity related list objects (class_id: 7)	153
6.3.9	Threshold values	154
6.3.10	Register monitor objects (class_id: 21)	154
6.4	Coding of OBIS identifications	155
7	Previous versions of interface classes	156
7.1	Profile generic (class_id: 7, version: 0)	156
7.2	Association SN (class_id: 12, version: 0)	159
7.3	Association SN (class_id: 12, version: 1)	161
7.4	Association LN (class_id: 15, version: 0)	163
7.5	IEC local port setup (class_id: 19, version: 0)	168
7.6	IEC HDLC setup, (class_id: 23, version: 0)	169
7.7	PSTN modem configuration (class_id: 27, version: 0)	170
7.8	PSTN auto dial (class_id: 29, version: 0)	171
7.9	S-FSK Phy&MAC setup (class_id: 50, version: 0)	172
7.10	S-FSK IEC 61334-4-32 LLC setup (class_id: 55, version: 0)	176
	Bibliography	177

List of figures

Figure 1	An interface class and its instances	16
Figure 2	The COSEM server model	24
Figure 3	Combined metering device	24
Figure 4	Overview of the interface classes – Part 1	27
Figure 5	Overview of the interface classes – Part 2	28
Figure 6	The time attributes when measuring sliding demand	34
Figure 7	The attributes in the case of block demand	34
Figure 8	The attributes in the case of sliding demand (number of periods = 3)	35
Figure 9	The meaning of the definitions concerning the Image	57
Figure 10	The Image Read and the Image Transfer services	58
Figure 11	Image transfer process flow chart	61
Figure 12	The generalized time concept	66
Figure 13	State diagram of the Disconnect control IC	78
Figure 14	Definition of upper and lower thresholds	85
Figure 15	Object model of DLMS/COSEM servers	

Figure 16 – Data of historical billing periods – example with module 12, VZ = 5 134

List of tables

Table 1 – Reserved base_names for SN referencing	17
Table 2 – Common data types	19
Table 3 – Enumerated values for physical units	31
Table 4 – Examples for scaler_unit.....	32
Table 5 – Schedule.....	69
Table 6 – Special days table	69
Table 7 – Disconnect control IC – states and state transitions.....	78
Table 8 – Explicit presentation of threshold value arrays	85
Table 9 – Explicit presentation of action_sets.....	85
Table 10 – Mapping IEC 61334-5-512 MIB variables to COSEM IC attributes / methods	112
Table 11 – MAC addresses in the S-FSK profile	117
Table 12 – Use of value group C for abstract objects in the COSEM context.....	132
Table 13 – Representation of various values by appropriate ICs.....	150
Table 14 – Measuring algorithms – enumerated values	152
Table 15 – Threshold objects, electricity	154
Table 16 – Register monitor objects, electricity.....	155

INTERNATIONAL ELECTROTECHNICAL COMMISSION

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

FOREWORD

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

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this International Standard may involve the use of a maintenance service concerning the stack of protocols on which the present Standard IEC 62056-6-2 is based.

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

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

DLMS¹ User Association
Geneva/Switzerland
www.dlms.ch

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

¹ Device Language Message Specification.

This third edition cancels and replaces the second edition published in 2006. It constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

- The title of the revised and new parts of the IEC 62056 suite has changed from “ELECTRICITY METERING – DATA EXCHANGE FOR METER READING, TARIFF AND LOAD CONTROL” to “ELECTRICITY METERING DATA EXCHANGE – THE DLMS/COSEM SUITE”;
- The numbering scheme has changed from IEC 62056-XY to IEC 62056-X-Y. For example IEC 62056-62 becomes IEC 62056-6-2;
- The title of this part 6-6-2 became COSEM interface classes;
- The text of the Introduction has been amended;
- In 3, new abbreviations have been added;
- 4.1 General has been amended;
- New 4.2, Referencing methods has been added based on the Introduction of the former Annex C;
- New 4.3, Reserved base_names for special COSEM objects has been added based on former Annex C.3;
- In 4.4, Class description notation: the interface class template – and the specification of each IC – includes now the mapping to short names;
- In 4.8.4 Mandatory contents of a COSEM logical device a note has been added;
- In 4.5 Common data types a new data type, UTF 8-string [12] has been added. This type has been added to the CHOICES where appropriate;
- Clause 4.9 Data security just a short overview is provided, detailed specification can be found now in IEC 62056-5-3 4.2;
- In 5.1 Overview (of COSEM interface classes) new IC have been added. Figure 4 and Figure 5 provide the overview. The ICs are classified as:
 - Interface classes for parameters and measurement data, see 5.2; _____
 - Interface classes for access control and management, see 5.3; _____
 - Interface classes for time- and event bound control; see 5.4; _____
 - Interface classes for setting up data exchange via local ports and modems, see 5.5; _____
 - Interface classes for setting up data exchange via M-Bus, see 5.6; _____
 - Interface classes for setting up data exchange over the Internet, see 5.7; _____
 - Interface classes for setting up data exchange using S-FSK PLC, see 5.8; _____
 - Interface classes for setting up the LLC layer for ISO/IEC 8802-2, see 5.9. _____

The description of the function of the ICs has been amended in several places.

- In 5.2 Table 3 new quantities have been added: 65 *Specific energy* and 67 *Signal strength*;
- 5.3.1 defines a new version of the Association SN IC: class_id: 12, version: 2;
- 5.3.2 defines a new version of the Association LN: class_id: 15, version: 1;
- 18 new interface classes have been added (in IEC 62056-62 Ed. 2.0 existed only 31 (19 in Clause 5 +12 in the Annex A), now there are 49), see detailed:
- in 5.3.3 SAP assignment class for the logical_device_name element, a CHOICE of data types is allowed now;

- 5.3.4: new *Image transfer* IC (class_id: 18, version: 0) has been added to support firmware upgrade;
- 5.3.5 new Security setup IC (class_id: 64, version: 0) has been added to support a new HLS data access mechanisms add to support data transport security;
- 5.4.8 new Disconnect control IC (class_id: 70, version: 0) has been added to support the connection and disconnection of the supply;
- 5.4.9 new Limiter IC (class_id: 71, version: 0) has been added to support the limitation of the supply under certain conditions;
- 5.4.10 new Sensor manager interface class (class_id: 67, version: 0) has been added to manage sensors and monitoring its operation. This IC is mainly used with non-electricity meters;
- 5.6.1 new M-bus slave port setup (class_id: 25, version: 0) has been added to allow the meter to be configured as an M-Bus slave device;
- 5.6.2 new M-bus client (class_id: 72, version: 0) has been added to allow the meter to be configured as an M-Bus client device;
- 5.6.3 new Wireless Mode Q channel (class_id: 73, version: 1) has been added;
- 5.6.4 new M-bus master port setup (class_id: 74, version: 0) has been added to configure the M-Bus port;
- in 5.7.1 TCP-UDP setup (class_id: 41, version: 0), the assigned port numbers have been added;
- In 5.7.3 the Ethernet setup IC (class_id: 43, version: 0) as MAC address setup without changing the version;
- in 5.7.6 SMTP setup (class_id: 46, version: 0): port numbers have been added;
- New 5.8 *Interface classes for setting up data exchange using S-FSK PLC* has been added to manage the DLMS/COSEM S-FSK PLC profile:
 - 5.8.4 S-FSK Phy&MAC set-up (class_id: 50, version: 1);
 - 5.8.5 S-FSK Active initiator (class_id: 51, version: 0);
 - 5.8.6 S-FSK MAC synchronization timeouts (class_id: 52, version: 0);
 - 5.8.7 S-FSK MAC counters (class_id: 53, version: 0);
 - 5.8.8 IEC 61334-4-32 LLC setup (class_id: 55, version: 1);
 - 5.8.9 S-FSK Reporting system list (class_id: 56, version: 0);
 - 5.9.3 ISO/IEC 8802-2 LLC Type 1 setup (class_id: 57, version: 0);
 - 5.9.4 ISO/IEC 8802-2 LLC Type 2 setup (class_id: 58, version: 0);
 - 5.9.5 ISO/IEC 8802-2 LLC Type 3 setup (class_id: 59, version: 0).
- Clause 6 *Relation to OBIS* includes now the contents of the former Annex D, completed with the new interface class definitions, OBIS code allocations and data types;
- clause 6.2.2 *Data of historical billing periods* explains now the rules related to the use of various interface classes and the use of value group F for identifying historical data;
- 6.2.3 *Billing period values / reset counter entries*: abstract entries with A = 0 are also allowed now;
- in 6.2.6 Script table objects (class_id: 9) new instances have been specified;
- 6.2.25 specifies Security setup and frame counter objects;
- 6.2.40 specifies Event code objects;
- 6.2.42 specifies Consumer message objects;
- 6.2.43 specifies Currently active tariff objects;
- 6.2.44 specifies Event counter objects;

- 6.2.46 specifies Alarm registers and alarm filters objects;
- 6.2.49 specifies Inactive objects;
- in 6.3.9 Table 15 has been added;
- in 7 a note has been added on the use of previous version of ICs.

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

FDIS	Report on voting

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table. This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

The committee has decided that the contents of this publication will remain unchanged until the maintenance result date²⁾ indicated on the IEC web site under "<http://webstore.iec.ch>" in the data related to the specific publication. At this date, the publication will be

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

²⁾ The National Committees are requested to note that for this publication the maintenance result date is 2017.

INTRODUCTION

Driven not only by the business needs of utilities – often in a deregulated competitive market – but also by the increasing desire to manage natural resources efficiently as regards production, distribution and use, the utility meter is increasingly part of an integrated metering, control, and billing system. Not only at grid level but, with the advent of initiatives to involve consumers in energy and resource management, in industry and even down to the domestic level, the meter is no simple data recording device but relies critically on communication capabilities, system integration and interoperability.

COSEM, the Companion Specification for Energy Metering, addresses these challenges by looking at the meter as an integrated part of a communication system which requires above all the ability to convey measurements of the delivered product (energy) from the diverse points where these measurements are made to the business processes which use them, over a variety of connecting media. Such systems must also have the scope to include a gamut of additional information and to support setup and control functions which allow to operate the meter remotely at virtually all times.

COSEM achieves all this in a way which is essentially non-proprietary and does not make assumptions about the technical processes in place within the meter. Using *object modelling* techniques established in the world of information science the data to be supplied by the meter is defined in a standard way that is accessible to the utility's business processes and relevant parts of its behaviour are similarly represented, while the communications is defined following the *Open Systems Interconnection* that is fundamental to the telecommunications world. The formal specification of interface classes and objects, which enables this, forms a major part of COSEM.

To allow further analysis of information, for the purposes of billing, load-, customer- and contract management, it is necessary to uniquely identify data items, whether collected manually or automatically, via local or remote data exchange, in a manufacturer-independent way. The definition of identification codes to achieve this – the OBIS codes – 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 COSEM model represents the meter as a server – see 4.7– used by client applications that retrieve data from, provide control information to, and instigate known actions within the meter via controlled access to the attributes and specific methods of objects making up the server interface. This client may be supporting the business processes of utilities, customers, meter operators, or meter manufacturers.

The information content and abilities of the server are not fixed; instead the standardized objects and interface classes (ICs) form an extensible library from which the manufacturer can assemble (model) its products according to national specifications or contract requirements. As a key element the server offers means to retrieve its particular structural model (the list of logical devices and the list of objects visible through the interface). The library is designed so that the entire range of products (from residential to commercial, industrial, and transmission and distribution applications) can be covered. The choice of the subset of ICs used to build a meter, and the instantiation and implementation of those ICs are part of the product design and therefore left to the manufacturer. The concept of the standardized metering interface class library provides the different users and manufacturers with a maximum of diversity without having to sacrifice interoperability.

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

1 Scope

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

2 Normative references

The following referenced documents are indispensable for the application 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.

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

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

EN 13757-5:2008, *Wireless meter readout – Communication system for meters and remote reading of meters – Part 5: Relaying*

IEC 60559:1989, *Binary floating-point arithmetic for microprocessor systems*

IEC 61334-4-32 Ed. 1.0:1996, *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 Ed. 1.0:1996, *Distribution automation using distribution line carrier systems – Part 4: Data communication protocols – Section 41: Application protocol – 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 Ed. 1.0:2001, *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 Ed. 2.0:2001, *Distribution automation using distribution line carrier systems – Part 5-1: Lower layer profiles – The spread frequency shift keying (S-FSK) profile*

IEC 62056-21 Ed. 1.0:2002, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 21: Direct local data exchange*

Draft IEC 62056-3-1 Ed. 3.0:20XX, 13/1461/CDV), *Electricity metering – Data exchange for meter reading, tariff and load control – Part 31: Use of local area networks on twisted pair with carrier signalling*

IEC 62056-46 Ed. 1.1:2007, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 46: Data link layer using HDLC protocol*

IEC 62056-5-3 Ed. 3.0:20XX, *Electricity metering data exchange – The DLMS/COSEM suite – Part 5-3: COSEM application layer*

IEC 62056-6-1 Ed. 3.0:20XX, *Electricity metering data exchange – The DLMS/COSEM suite – Part 61: OBIS Object identification system*

ISO/IEC 8802-2 Ed. 3.0:1998, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 2: Logical link control*

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

RFC 0791: 1981, Internet Engineering Task Force (IETF). *Internet Protocol (Also: IETF STD 0005)*. Edited by Jon Postel, September 1981. Available from: <http://www.rfc-editor.org/rfc/rfc791.txt>

RFC 1332: 1992, Internet Engineering Task Force (IETF). *The PPP Internet Protocol Control Protocol (IPCP)*. Edited by G. McGregor, May 1992. Available from: <http://www.rfc-editor.org/rfc/rfc1332.txt>

RFC 1570: 1994, Internet Engineering Task Force (IETF). *PPP LCP Extensions*, Edited by W. Simpson, January, 1994. Available from: <http://www.rfc-editor.org/rfc/rfc1570.txt>

RFC 1661: 1994, Internet Engineering Task Force (IETF) *The Point-to-Point Protocol (PPP) (Also: IETF STD 0051)*, 1994. Edited by W. Simpson, July, 1994. Available from: <http://www.rfc-editor.org/rfc/rfc1661.txt>

RFC 1662: 1994, Internet Engineering Task Force (IETF). *PPP in HDLC-like Framing (Also: IETF STD 0051)*. Edited by W. Simpson, July, 1994. Available from: <http://www.rfc-editor.org/rfc/rfc1662.txt>

RFC 1700: 1994, Internet Engineering Task Force (IETF). *Assigned numbers (Also: IETF STD 0002)*. Edited by J. Reynolds, J. Postel, October 1994. Available from: <http://www.rfc-editor.org/rfc/rfc1700.txt>

RFC 2507: 1999, Internet Engineering Task Force (IETF). *IP Header Compression*. Edited by M. Degermark, B. Nordgren, S. Pink, October 1994. Available from: <http://www.rfc-editor.org/rfc/rfc2507.txt>

RFC 3241: 2002, Internet Engineering Task Force (IETF). *Robust Header Compression (ROHC) over PPP*, 2002. Edited by C. Bormann, April 2002. Available from: <http://www.rfc-editor.org/rfc/rfc3241.txt>

RFC 3544: 2003, Internet Engineering Task Force (IETF). *IP Header Compression over PPP*, 2003. Edited by T. Koren, S. Casner, C. Bormann, July 2003. Available from: <http://www.rfc-editor.org/rfc/rfc3544.txt>

STD0005:1981, Internet Engineering Task Force (IETF). *Internet Protocol*. Also: RFC0791 (RFC0792, RFC0919, RFC0922, RFC0950, RFC1112). *Intellectual Property Rights in IETF Technology*. Edited by Jon Postel. September 1981, Available from: <http://www.faqs.org/rfcs/std/std5.html>

STD 0051:1994, Internet Engineering Task Force (IETF): *The Point-to-Point Protocol (PPP)*. Edited by W. Simpson July 1994. (Also RFC1661, RFC1662 Available from: <http://www.faqs.org/rfcs/std/std51.html>

NOTE See also the Bibliography.

3 Terms, definitions and abbreviations

AA	Application Association
AARE	Application Association Response
AARQ	Application Association ReQuest
ACSE	Association Control Service Element
AL	Application layer
AP	Application process
APDU	Application Protocol Data Unit
ASE	Application Service Element
A-XDR	Adapted Extended Data Representation
base_name	The short_name corresponding to the first attribute (“logical_name”) of a COSEM object
CHAP	Challenge Handshake Authentication Protocol
class_id	Interface class identification code
COSEM	Companion Specification for Energy Metering
COSEM object	An instance of a COSEM interface class
CtoS	Client to Server challenge
DHCP	Dynamic Host Control Protocol
DLMS	Device Language Message Specification
DNS	Domain Name Server
EAP	Extensible Authentication Protocol
GCM	Galois/Counter Mode, an algorithm for authenticated encryption with associated data
GMT	Greenwich Mean Time. Replaced by Coordinated Universal Time (UTC).
GPS	Global Positioning System
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HART	Highway Addressable Remote Transducer see http://www.hartcomm.org/ (in relation with the Sensor manager interface class)
HDLC	High-level Data Link Control
HLS	High Level Security
IANA	Internet Assigned Numbers Authority
IC	Interface Class
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPCP	Internet Protocol Control Protocol
IT	Information Technology
ISO	International Organization for Standardization
KEK	Key Encryption Key

LCP	Link Control Protocol
LLC	Logical Link Control (Sublayer)
LLS	Low Level Security
LN	Logical Name
LSB	Least Significant Bit
m	mandatory
MD5	Message Digest Algorithm 5
MID	Measuring Instruments Directive 2004/22/EC
MSB	Most Significant Bit
o	optional
OBIS	OBject Identification System
PAP	Password Authentication Protocol
PDU	Protocol Data Unit
PLMN	Public Land Mobile Network
PIN	Personal Identity Number
PPP	Point-to-Point Protocol
PSTN	Public Switched Telephone Network
ROHC	Robust Header Compression
SAP	Service Access Point
SHA-1	Secure Hash Algorithm
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SN	Short Name
StoC	Server to Client Challenge
UTC	Coordinated Universal Time

4 Basic principles

4.1 General

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

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

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

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

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

Figure 1 illustrates these terms by means of an example:

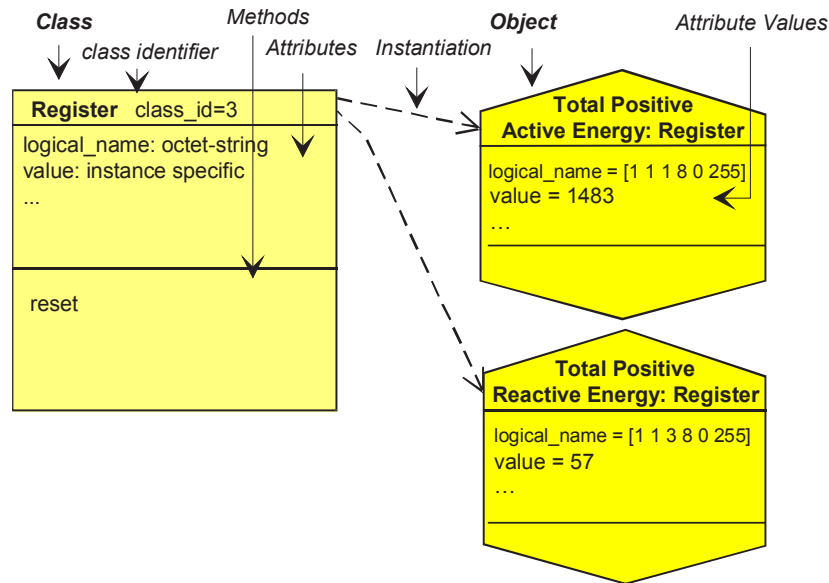


Figure 1 – An interface class and its instances

The IC “Register” is formed by combining the features necessary to model the behaviour of a generic register (containing measured or static information) as seen from the client (data collection system, hand held terminal). The contents of the register are identified by the attribute “logical_name”. The logical_name contains an OBIS identifier (see IEC 62056-6-1). 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 1, 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 of a register – must always be initiated from the outside. Internally initiated changes of the attributes – for example updating the value of a register – are not described in this model.

4.2 Referencing methods

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

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

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

The reference for a method is: class_id, value of the ‘logical_name’ attribute, method_index.

Where:

- attribute_index is used as the identifier of the attribute required. Attribute indexes are specified in the definition of each IC. They are positive numbers starting with one. 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 one. 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 and IEC 62056-5-3 4.7.

4.3 Reserved base_names for special COSEM objects

In order to facilitate access to devices using SN referencing, some short_names are reserved as base_names for special COSEM objects. The range for reserved base_names is from 0xFA00 to 0xFFF8. The following specific base_names are defined:

Table 1 – Reserved base_names for SN referencing

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

4.4 Class description notation

This subclause describes the notation used to define the ICs.

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

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

Class name	Describes the interface class (for example "Register", "Clock", "Profile generic"...)
Cardinality	Specifies the number of instances of the IC within a logical device (see 4.8). <i>value</i> The IC shall be instantiated exactly "value" times. <i>min...max.</i> The IC shall be instantiated at least "min." times and at most "max." times. If min. is zero (0) then the IC is optional, otherwise (min. > 0) "min." instantiations of the IC are mandatory.
class_id	Identification code of the IC (range 0 to 65 535). The class_id of each object is retrieved together with the logical name by reading the object_list attribute of an "Association LN" / "Association SN" objects. Class_id-s from 0 to 8 191 are reserved to be specified by the DLMS UA. Class_id-s from 8 192 to 32 767 are reserved for manufacturer specific

	ICs. Class_id-s from 32 768 to 65 535 are reserved for user group specific ICs. The DLMS UA reserves the right to assign ranges to individual manufacturers or user groups.	
Version	Identification code of the version of the IC. The version of each object is retrieved together with the class_id and the logical name by reading the object_list attribute of an "Association LN" / "Association SN" objects. Within one logical device, all instances of a certain IC must be of the same version.	
Attribute(s)	Specifies the attribute(s) that belong to the IC. (<i>dyn.</i>) Classifies an attribute that carries a process value, which is updated by the meter itself. (<i>static</i>) Classifies an attribute, which is not updated by the meter itself (for example configuration data). NOTE There are some attributes, which may be either static or dynamic depending on the application. In these cases this property is not indicated.	
logical_name	octet-string	The logical name 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 IEC 62056-6-1.
Data type	Defines the data type of an attribute; see 4.5.	
Min.	Specifies if the attribute has a minimum value. X The attribute has a minimum value. <empty> The attribute has no minimum value.	
Max.	Defines if the attribute has a maximum value. X The attribute has a maximum value. <empty> The attribute has no maximum value.	
Def.	Specifies if the attribute has a default value. This is the value of the attribute after reset. X The attribute has a default value. <empty> The default value is not defined by the IC definition.	
Short name	When Short Name (SN) referencing is used, each attribute and method of object instances has to be mapped to short names. The base_name x of each object instance is the DLMS named variable the logical name attribute is mapped to. It is selected in the implementation phase. The IC definition specifies the offsets for the other attributes and for the methods.	
Specific method(s)	Provides a list of the specific methods that belong to the object. Method Name () The method has to be described in the subsection "Method description".	
m/o	Defines if the method is mandatory or optional. m (<i>mandatory</i>) The method is mandatory. o (<i>optional</i>) The method is optional.	

Attribute description

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

Method description

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

NOTE Services for accessing attributes or methods by the protocol are described in IEC 62056-5-3 clauses 4.5.6 – 4.5.15.

Selective access

The xDLMS services Read, Write, UnconfirmedWrite (used with SN referencing) and GET, SET (used with LN referencing) typically reference the entire attribute. However, for certain attributes selective access to just a part of the attribute may be provided. The part of the attribute is identified by specific selective access parameters. These are defined as part of the attribute specification.

4.5 Common data types

The following table contains the list of data types usable for attributes of COSEM objects.

Table 2 – Common data types

Type description	Tag ^a	Definition	Value range
-- simple data types			
null-data	[0]		
boolean	[3]	Boolean	TRUE or FALSE
bit-string	[4]	An ordered sequence of boolean values	
double-long	[5]	Integer32	-2 147 483 648... 2 147 483 647
double-long-unsigned	[6]	Unsigned32	0...4 294 967 295
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, not usable in DLMS/COSEM. See tag [27]	
UTF8-string	[12]	An ordered sequence of characters encoded as UTF-8	
bcd	[13]	binary coded decimal	
integer	[15]	Integer8	-128...127
long	[16]	Integer16	-32 768...32 767
unsigned	[17]	Unsigned8	0...255
long-unsigned	[18]	Unsigned16	0...65 535
long64	[20]	Integer64	- 2 ⁶³ ...2 ⁶³ -1
long64-unsigned	[21]	Unsigned64	0...2 ⁶⁴ -1
enum	[22]	The elements of the enumeration type are defined in the "Attribute description" section of a COSEM IC specification.	
float32	[23]	OCTET STRING (SIZE(4))	For formatting, see 4.6.2.
float64	[24]	OCTET STRING (SIZE(8))	
date_time	[25]	OCTET STRING SIZE(12))	For formatting, see 4.6.1.
date	[26]	OCTET STRING (SIZE(5))	
time	[27]	OCTET STRING (SIZE(4))	
-- complex data types			

Type description	Tag ^a	Definition	Value range
array	[1]	The elements of the array are defined in the "Attribute description" section of a COSEM IC specification.	
structure	[2]	The elements of the structure are defined in the "Attribute description" section of a COSEM IC specification.	
compact array	[19]	The elements of the compact array are defined in the "Attribute description" section of a COSEM IC specification.	
-- CHOICE		For some attributes of some COSEM interface objects, the data type may be chosen at COSEM object instantiation, in the implementation phase of the COSEM server. The server always shall send back the data type and the value of each attribute, so that together with the logical name, an unambiguous interpretation is ensured. The list of possible data types is defined in the "Attribute description" section of a COSEM IC specification.	
a The tags are as defined in IEC 62056-5-3 4.7.			

4.6 Data formats

4.6.1 Date and time formats

Date and time information may be represented with data type octet-string, or using the data types *date*, *time* and *date_time*, as defined in the relevant IC definition.

NOTE 1 In future versions of ICs and in newly defined ICs, the data types *date*, *time* and *date_time* will be used as appropriate.

NOTE 2 The (SIZE()) specifications do not apply if *date*, *time* or *date_time* are represented by data type octet-string.

date OCTET STRING (SIZE(5))

```

{
    year highbyte,
    year lowbyte,
    month,
    day of month,
    day of week
}
year: interpreted as long-unsigned
range 0...big
0xFFFF = not specified
year highbyte and year lowbyte reference the 2 bytes of the long-unsigned

month:    interpreted as unsigned
range 1...12, 0xFD, 0xFE, 0xFF
1 is January
0xFD = daylight_savings_end
0xFE = daylight_savings_begin
0xFF = not specified

dayOfMonth:    interpreted as unsigned
range 1...31, 0xFD, 0xFE, 0xFF
0xFD = 2nd last day of month
0xFE = last day of month
0xE0 to 0xFC = reserved
0xFF = not specified

dayOfWeek:    interpreted as unsigned
range 1...7, 0xFF
1 is Monday
0xFF = not specified

```

For dayOfMonth and dayOfWeek:

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

The elements dayOfMonth and dayOfWeek shall be interpreted together:

- if last dayOfMonth is specified (0xFE) and dayOfWeek is wildcard, this specifies the last calendar day of the month;
- if last dayOfMonth is specified (0xFE) and an explicit dayOfWeek is specified (for example 7, Sunday) then it is the last occurrence of the weekday specified in the month, i.e. the last Sunday;
- if the year is not specified (FFFF), 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:

- year = 0xFFFF, month = FF, dayOfMonth = 0xFE, dayOfWeek = 0xFF: last day of the month in every year and month;
- year = 0xFFFF, month = FF, dayOfMonth = 0xFE, dayOfWeek = 0x07: last Sunday in every year and month;
- year = 0xFFFF, month = 0x03, dayOfMonth = 0xFE, dayOfWeek = 0x07: last Sunday in March in every year;
- year = 0xFFFF, month = 0x03, dayOfMonth = 0x01, dayOfWeek = 0x07: first Sunday in March in every year;
- year = 0xFFFF, month = 0x03, dayOfMonth = 0x16, dayOfWeek = 0x05: fourth Friday in March in every year;
- year = 0xFFFF, month = 0x0A, dayOfMonth = 0x16, dayOfWeek = 0x07: fourth Sunday in October in every year;
- year = 0x07D9, month = 0x01, dayOfMonth = 0x13, dayOfWeek = 0x01: 2009.01.19, Monday;
- year = 0x07D9, month = 0x01, dayOfMonth = 0x13, dayOfWeek = 0x02 : error, as the dayOfMonth and dayOfWeek in the given year and month do not match.

time

OCTET STRING (SIZE(4))

```
{
    hour,
    minute,
    second,
    hundredths
}
hour:      interpreted as unsigned
           range0...23, 0xFF,
minute:    interpreted as unsigned
           range0...59, 0xFF
second:    interpreted as unsigned
           range0...59, 0xFF,
hundredths: interpreted as unsigned
           range0...99, 0xFF
```

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

For repetitive times the unused parts must be set to “not specified”.

deviation

long –720...720:
 in minutes of local time to GMT
 0x8000 = not specified

clock_status

unsigned interpreted as 8 bit string

The status bits are defined as follows:

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

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
 }

Individual fields of *date_time* are encoded as defined above. Some may be set to “not specified” as described above in *date* and *time*.

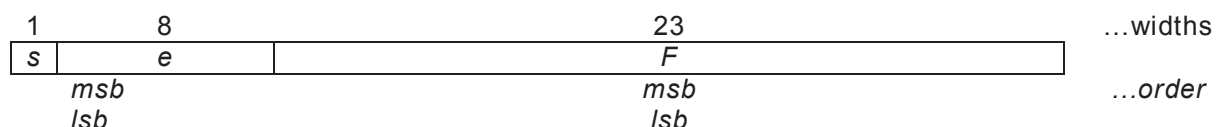
- ^a Time could not be recovered after an incident. Detailed conditions are manufacturer specific (for example after the power to the clock has been interrupted).
- ^b Time could be recovered after an incident but the value cannot be guaranteed. Detailed conditions are manufacturer specific.
- ^c Bit is set if the basic timing information for the clock at the actual moment is taken from a timing source different from the source specified in *clock_base*.
- ^d This bit indicates that at least one bit of the clock status is invalid. Some bits may be correct. The exact meaning shall be explained in the manufacturer's documentation.
- ^e Flag set to true: the transmitted time contains the daylight saving deviation (summer time).
 Flag set to false: the transmitted time does not contain daylight saving deviation (normal time).

4.6.2 Floating point number formats

Floating point number formats are defined in IEC 60559.

NOTE For the following, IEC 60559 is equivalent to IEEE 754.

The single format is:



where:

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

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

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

The double format is:

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

where:

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

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

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

For more detail, see IEC 60559.

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

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

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

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

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

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

Bit 63 Sign bit 0 0: + 1: -	Bits 62-52 Exponent field: 0111111111 Decimal value of exponent field and exponent: 1023 -1023 = 0	Bits 51-0 Significand 1.0000000000000000000000000000000000000000000000000000000 Decimal value of the significand: 1.0000000000000000
--	---	--

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

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

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

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

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

Bit 63 Sign bit 0 0: + 1: -	Bits 62-52 Exponent field: 1000000110 Decimal value of exponent field and exponent: 1038 -1023 = 15	Bits 51-0 Significand 1.1110010011010000000000000000000000000000000000000000000 Decimal value of the significand: 1.8937988281250000
--	--	--

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

4.7 The COSEM server model

The COSEM server is structured into three hierarchical levels as shown in Figure 2:

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

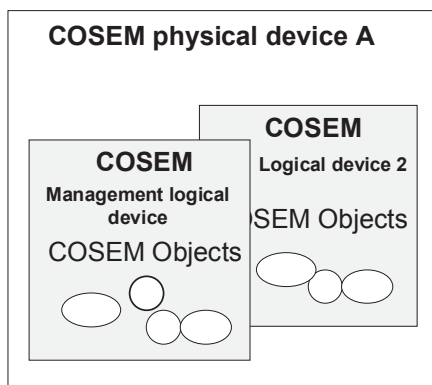


Figure 2 – The COSEM server model

The following example in Figure 3 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

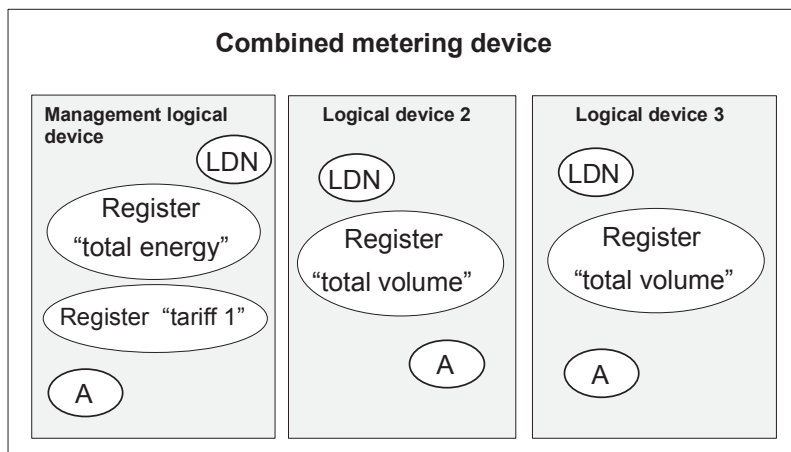


Figure 3 – Combined metering device

4.8 The COSEM logical device

4.8.1 General

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

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

4.8.2 COSEM logical device name

The COSEM logical device can be identified by its unique COSEM logical device name. This name can be retrieved from an instance of IC “SAP assignment” (see 5.3.3), or from a COSEM object “COSEM logical device name” (see 6.2.24).

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

4.8.3 The “association view” of the logical device

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

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

The AA is modelled by special COSEM objects: the “Association” objects. There are two types of these objects defined:

- one for using SN referencing (“Association SN”, see 5.3.1);
- and one for using LN referencing (“Association LN”, see 5.3.2).

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).

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

4.8.4 Mandatory contents of a COSEM logical device

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

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

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

4.8.5 Management logical device

As specified in 4.8.1, the management logical device is a mandatory element of any physical device. It has a reserved address. It must support an AA to a public client with the lowest security level. 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

³ Administered by the DLMS User Association

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

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

4.9 Data security

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

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

For a description of these security mechanisms, see IEC 62056-5-3 4.2.

Data security is provided by the COSEM Application layer and it is supported / managed by the following objects:

- Association SN, see 5.3.1;
- Association LN, see 5.3.2; and
- Security setup, see 5.3.5.

5 The COSEM interface classes

5.1 Overview

The interface classes defined currently and the relations between them are shown in Figure 4 and Figure 5.

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

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

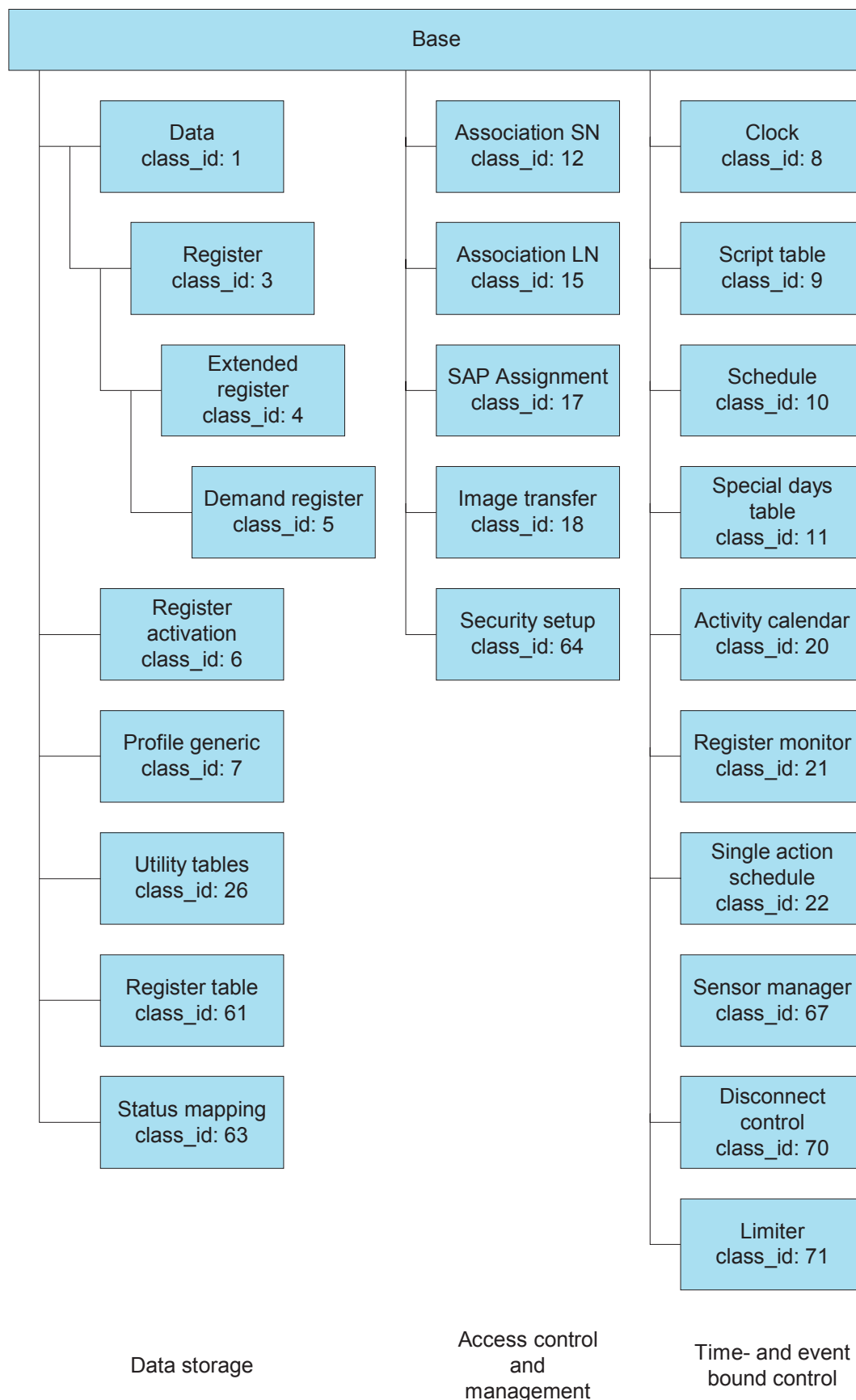
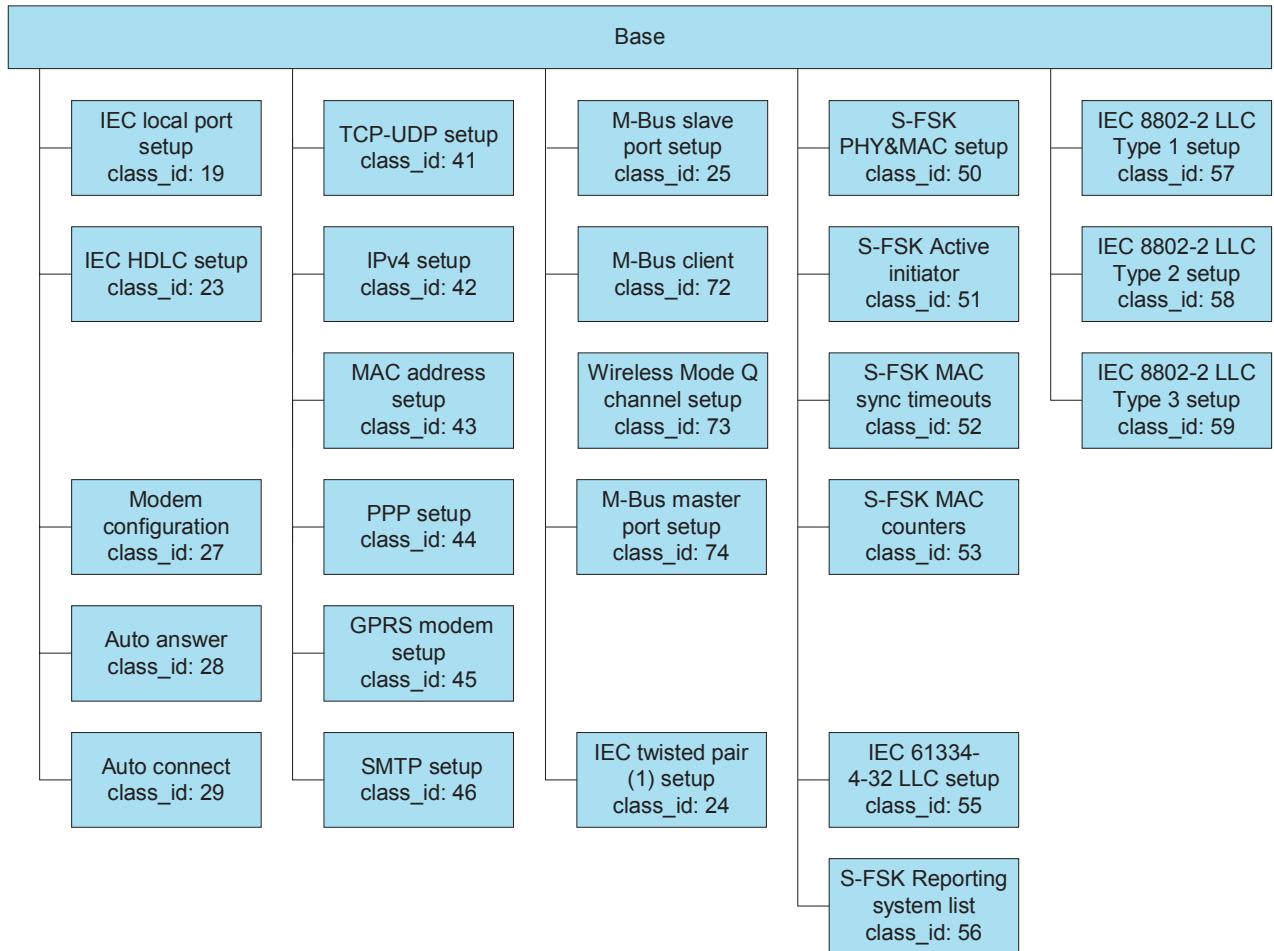


Figure 4 – Overview of the interface classes – Part 1



Communication channel setup

Figure 5 – Overview of the interface classes – Part 2

5.2 Interface classes for parameters and measurement data

5.2.1 Data (class_id: 1, version: 0)

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

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

Attribute description

logical_name	Identifies the “Data” object instance. See clauses 6 and IEC 62056-6-1.				
value	Contains the data. CHOICE <pre> { -- simple data types null-data [0], boolean [3], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], UTF8-string [12], bcd [13], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], enum [22], float32 [23], float64 [24], date-time [25], date [26], time [27], -- complex data types array [1], structure [2], compact-array [19] } </pre> The data type depends on the instantiation defined by the “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 4.5 can be used, unless the choice is restricted in 6.				

5.2.2 Register (class_id: 3, version: 0)

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

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

Attribute description

logical_name	Identifies the “Register” object instance. See clauses 6 and IEC 62056-6-1.				
value	Contains the current process or status value. CHOICE <pre> { -- simple data types null-data [0], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], UTF8-string [12], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], float32 [23], float64 [24] }</pre> The data type of the value depends on the instantiation defined by “logical_name” and possibly on the choice of the manufacturer. It has to be chosen so that, together with the logical_name, an unambiguous interpretation of the value is possible. When, instead of a “Data” object, a “Register” object is used, (with the scaler_unit attribute not used or with scaler = 0, unit = 255) then the data types allowed for the value attribute of the “Data” IC are allowed.				
scaler_unit	Provides information on the unit and the scaler of the value. <pre> scal_unit_type ::= structure { scaler, unit } scaler: integer</pre> 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. <pre> unit: enum</pre> Enumeration defining the physical unit; for details see Table 3 below.				

Method description

reset (data)	This method 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. <pre> data ::= integer(0)</pre>
---------------------	--

Table 3 – Enumerated values for physical units

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

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

Some examples are shown in Table 4 below.

Table 4 – Examples for scaler_unit

Value	Scaler	Unit	Data
263788	-3	m ³	263,788 m ³
593	3	Wh	593 kWh
3467	0	V	3467 V

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

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

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

Attribute description

For the definition of the attributes *value* and *scaler_unit*, see description of the IC "Register".

logical_name	Identifies the "Extended register" object instance. See Clause 6.3.1.					
status	Provides "Extended register" specific status information. The meaning of the elements of the status shall be provided for each "Extended register" object instance. CHOICE { -- simple data types null-data [0], bit-string [4], double-long-unsigned [6], octet-string [9], visible-string [10], UTF8-string [12], unsigned [17], long-unsigned [18], long64-unsigned [21] } The data type and the encoding depend on the instantiation and possibly on the choice of the manufacturer. For the interpretation, extra information from the manufacturer may be necessary. Def. Depending on the status type definition.					
capture_time	Provides an "Extended register" specific date and time information showing when the value of the attribute "value" has been captured. octet-string, formatted as set in 4.6.1 for <u>date_time</u>					

Method description

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

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

data ::= integer(0)

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

This IC allows modelling a demand value, with its associated 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 measured or computed is defined by specifying “*number_of_periods*” and “*period*”.

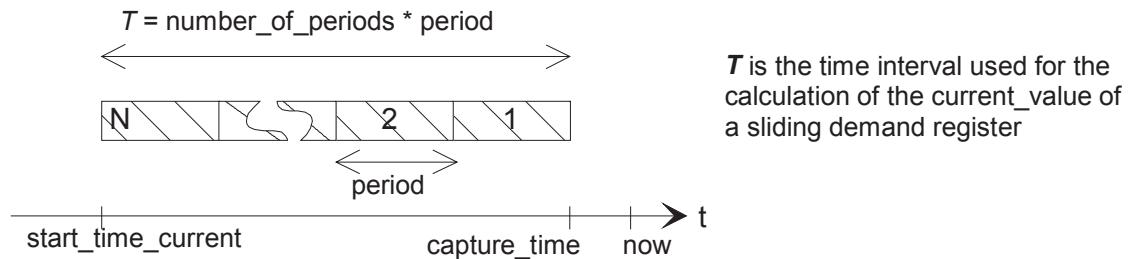


Figure 6 – The time attributes when measuring sliding demand

The demand register delivers two types of demand: *current_average_value* and *last_average_value* (see Figure 7 and Figure 8).

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

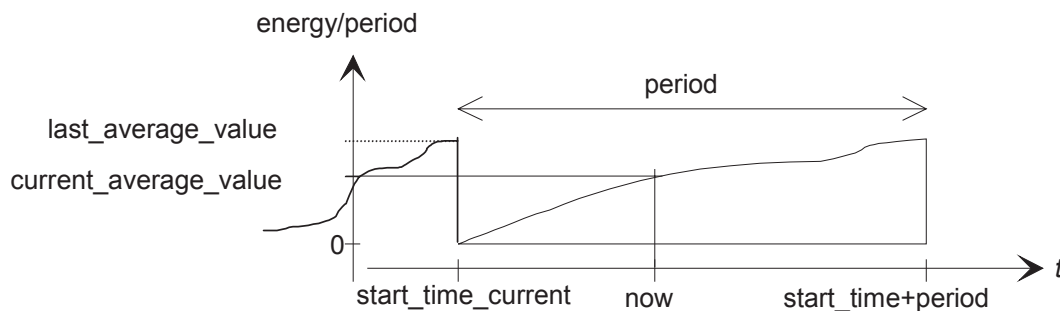


Figure 7 – The attributes in the case of block demand

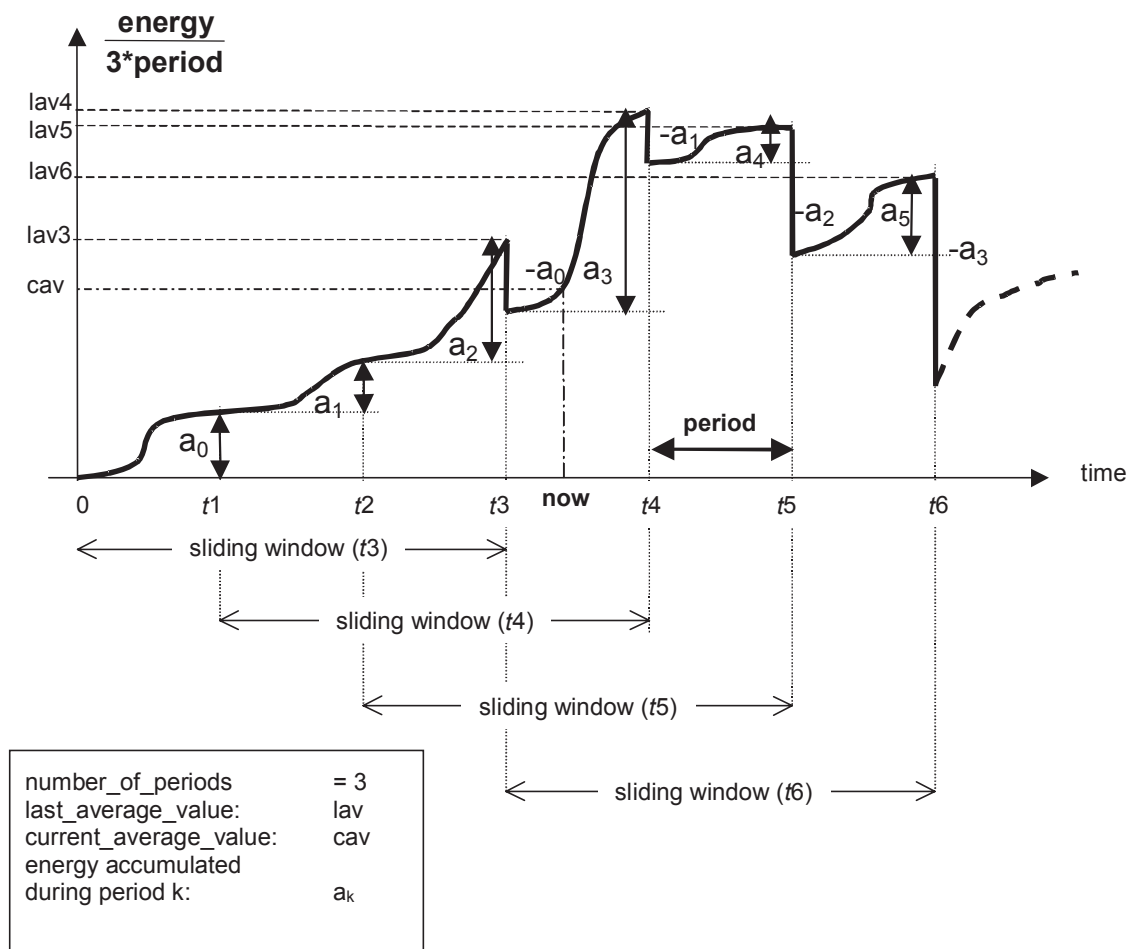


Figure 8 – The attributes in the case of sliding demand (number of periods = 3)

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

Attribute description

For the attribute *scaler_unit*, see description of IC “Register”.

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

Method description

reset (data)	This method forces a reset of the object. Activating this method provokes the following actions: the current period is terminated;
---------------------	---

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

`data ::= integer(0)`

next_period (data)

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

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

`data ::= integer(0)`

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

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

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

Attribute description

logical_name	Identifies the “Register activation” object instance. See 6.2.10.
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 “Register”, “Extended register” or “Demand register”. array object_definition <pre> object_definition ::= structure { class_id: long-unsigned, logical_name: octet-string } </pre>
mask_list	Specifies a list of register activation masks. Each entry (mask) is identified by its <code>mask_name</code> and contains an array of indices referring to the registers assigned to the mask (the first object in <code>register_assignment</code> 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
active_mask	Defines the currently active mask. The mask is defined by its mask_name (see mask_list). The active_mask defines the registers currently enabled; all other registers listed in the register_assignment are disabled.
Method description	
add_register (data)	Adds one more registers 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 }
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 above)
delete_mask (data)	Deletes a mask from the attribute mask_list. The mask is defined by its mask name. data ::= octet-string (mask_name)

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

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

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

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

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

The size of profile data is determined by three parameters:

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

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

Attribute description

logical_name Identifies the “Profile generic” object instance. For examples, see 6.2.15, 6.2.28, 6.2.31, 6.2.33, 6.2.44, 6.2.46, 6.3.2 etc.

buffer Contains a sequence of entries. Each entry contains values of the captured objects (as they would be returned to a GET.request or Read.request).

compact-array or array entry

entry ::= structure

```
{
    CHOICE
    {
        -- simple data types
        null-data           [0],
        boolean             [3],
        bit-string          [4],
        double-long         [5],
        double-long-unsigned [6],
        octet-string        [9],
        visible-string      [10],
        UTF8-string         [12],
        bcd                 [13],
        integer             [15],
        long                [16],
        unsigned            [17],
        long-unsigned       [18],
```

```

long64                [20],
long64-unsigned       [21],
enum                 [22],
float32               [23],
float64               [24],
date-time             [25],
date                  [26],
time                  [27],
-- complex data types
array                 [1],
structure              [2],
compact-array         [19]
    }
}

```

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

Default: The buffer is empty after reset.

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

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

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

capture_objects

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

Upon a call of the `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. `attribute_index 1` refers to the 1st attribute (i.e. the `logical_name`), `attribute_index 2` to the 2nd, etc.); `attribute_index 0` refers to all public attributes;
- where `data_index` is a pointer selecting a specific element of the attribute. The first element in the attribute structure is identified by `data_index 1`. If the attribute is not a structure, then the `data_index` has no meaning. If the capture object is the buffer of a profile, then the `data_index` identifies the captured object of the buffer (i.e. the column) of the inner profile.
- `data_index 0`: references the whole attribute

capture_period

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

`0`: No automatic capturing; capturing is triggered externally or `capture`

	events occur asynchronously.												
sort_method	If the profile is unsorted, it works as a “first in first out” buffer (it is hence sorted by capturing, and not necessarily by the time maintained in the Clock object). If the 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: <table> <tr><td>(1)</td><td>fifo (first in first out),</td></tr> <tr><td>(2)</td><td>lifo (last in first out),</td></tr> <tr><td>(3)</td><td>largest,</td></tr> <tr><td>(4)</td><td>smallest,</td></tr> <tr><td>(5)</td><td>nearest_to_zero,</td></tr> <tr><td>(6)</td><td>farthest_from_zero</td></tr> </table> Def. fifo	(1)	fifo (first in first out),	(2)	lifo (last in first out),	(3)	largest,	(4)	smallest,	(5)	nearest_to_zero,	(6)	farthest_from_zero
(1)	fifo (first in first out),												
(2)	lifo (last in first out),												
(3)	largest,												
(4)	smallest,												
(5)	nearest_to_zero,												
(6)	farthest_from_zero												
sort_object	If the profile is sorted, this attribute specifies the register or clock that the ordering is based upon. capture_object_definition See above. Def. no object to sort by (only possible with sort_method fifo or lifo) NOTE If the sort_method is FIFO or LIFO, then all elements of the capture_object_definition specifying the sort object may be zero.												
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). double-long-unsigned 0...profile_entries Def. 0												
profile_entries	Specifies how many entries shall be retained in the buffer. double-long-unsigned 1...(limited by physical size) Def. 1												

Parameters for selective access to the buffer attribute

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

range_descriptor ::= structure

{

 restricting_object: capture_object_definition

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

```

from_value:                                Oldest or smallest entry to retrieve

CHOICE
{-- simple data types
  double-long                               [5],
  double-long-unsigned                      [6],
  octet-string                             [9],
  visible-string                           [10],
  UTF8-string                              [12],
  integer                                  [15],
  long                                     [16],
  unsigned                                 [17],
  long-unsigned                             [18],
  long64                                   [20],
  long64-unsigned                           [21],
  float32                                  [23],
  float64                                  [24],
  date-time                                [25],
  date                                     [26],
  time                                     [27]
}

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

selected_values:                          array capture_object_definition

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

entry_descriptor ::= structure
{
  from_entry:                             double-long-unsigned    first entry to retrieve,
  to_entry:                               double-long-unsigned    last entry to retrieve
                                          to_entry == 0: highest possible entry,

  from_selected_value:                    long-unsigned           index of first value to retrieve,
  to_selected_value:                     long-unsigned           index of last value to retrieve
                                          to_selected_value == 0: highest possible selected_value
}

```

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

Method description

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

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

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

data ::= integer(0)

Behaviour of the object after modification of certain attributes

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

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

Restrictions

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

Profile used to define a subset of preferred readout values

By setting profile_entries to 1, a “Profile generic” object can be used to define a set of preferred readout values. See also 6.2.15. Setting capture_period to 1 ensures that the values are updated every second.

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

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

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

Attribute description

logical_name	Identifies the “Utility tables” object instance. See 6.2.27.
table_ID	Table number. This table number is as specified in the ANSI standard and may be either a standard table or a manufacturer’s table.
length	Number of octets in table buffer.
buffer	Contents of the table. Selective access (see 4.4) to the attribute <i>buffer</i> may be available (optional). The selective access parameters are as defined below.

Parameters for selective access to the buffer attribute

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

offset_selector ::= structure

```
{
    Offset:    double-long-unsigned    offset in octets to the start of access area, relative
                                         to the start of the table
    Count:     long-unsigned           number of octets requested or transferred
}
```

index_selector ::= structure

```
{
    Index:     array of                sequence of indices to identify elements within the
               long-unsigned           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

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

This IC allows to group homogenous entries, identical attributes of multiple objects, which are all instances of the same IC, and in their logical name (OBIS code) the value in value groups A to D and F is identical. The possible values in value group E are defined in IEC 62056-6-1 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 “UNIPED voltage dip quantities” table, see IEC 62056-6-1 Table 18, or the “Extended phase angle measurement” table, see IEC 62056-6-1 Table 16.

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.)	array				x + 0x08
3. table_cell_definition (static)	structure				x + 0x10
4. scaler_unit (static)	scaler_unit_type				x + 0x18
Specific methods	m/o				
1. reset (data)	o				x + 0x28
2. capture (data)	o				x + 0x30

Attribute description

logical_name

Identifies the “Register table” object instance.

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

	When the format of the logical name is A.B.98.10.x.255, several instances of the “Register table” IC can be used to capture different attributes of the objects concerned. The value group E numbers the instances. See IEC 62056-6-1 Clause 4.4.4.
table_cell_values	Holds the <u>value of the attributes</u> 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 <pre>{ -- simple data types null-data [0], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], UTF8-string [12], 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] }</pre> If the captured attribute is attribute_0, redundant values may be replaced by “null_data”, if their value can be unambiguously recovered (for example scaler_unit).
table_cell_definition	Specifies the list of attributes captured in the register table. <pre>structure { class_id: long-unsigned, logical_name: octet-string, group_E_values: array cell_identifier, { cell_identifier: unsigned } attribute_index: integer }</pre> 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, as defined in the respective table of IEC 62056-6-1; • attribute_index is a pointer to the attribute within the object. attribute_index 0 refers to all public attributes.

If the logical name of the “Register table” object is in the format

A.B.C.D.255.F and the defined attribute of all objects identified in the respective table in IEC 62056-6-1 are captured, then attribute 3 may not be accessible. ~~In this case:~~

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

scaler_unit

See the description of IC “Register”.

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

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

Method description

reset (data)

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

data ::= integer(0)

capture (data)

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

Behaviour of the object after modification of the table_cell_definition attribute

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

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

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

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

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

Attribute description

logical_name	Identifies the “Status mapping” object instances. See 6.2.31 , 6.2.33 , 6.2.39 and 6.3.7 .
status_word	Contains the current value of the status word. CHOICE <pre>{ bit-string [4], double-long-unsigned [6], octet-string [9], visible-string, [10], UTF8-string [12], unsigned [17], long-unsigned [18], long64-unsigned [21] }</pre> The size of the status_word is n*8 bits, the maximum size is 65 536 bits. NOTE Manufacturers may choose any of the types listed above. However, the status word shall be always interpreted as a bit-string.
mapping_table	Contains the mapping of the status word to the positions in the reference table. structure <pre>{ ref_table_id: unsigned, CHOICE { first_entry: long-unsigned, array: table_entries } }</pre> table_entry: long-unsigned where: • ref_table_id is the identifier of the reference status table • NOTE Reference status tables are maintained by the DLMS UA. They may also be project specific. • first_entry is the entry in the reference table corresponding to bit 0 of the status word. Bit 1 corresponds to the next entry and so on, the last entry is defined by the length of the status word; • if the “array” choice is taken, the array of table entries maps the bits of the status word to the entries in the reference status table. The position in the array defines the position in the status word (the first position corresponds to bit 0).

5.3 Interface classes for access control and management**5.3.1 Association SN (class_id: 12, version: 2)**

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

The **short_name** of the “Association SN” object itself is fixed within the COSEM context. See [4.3](#).