

DIN EN 62056-62

ICS 35.200; 91.140.50

Ersatz für
DIN EN 62056-62:2003-01
Siehe jedoch Beginn der
Gültigkeit

**Messung der elektrischen Energie –
Zählerstandsübertragung, Tarif- und Laststeuerung –
Teil 62: Interface-Klassen (IEC 62056-62:2006);
Deutsche Fassung EN 62056-62:2007, Text Englisch**

Electricity metering –
Data exchange for meter reading, tariff and load control –
Part 62: Interface classes (IEC 62056-62:2006);
German version EN 62056-62:2007, text in English

Equipements de mesure de l'énergie électrique –
Echange des données pour la lecture des compteurs, le contrôle des tarifs et de la
charge –
Partie 62: Classes d'interface (CEI 62056-62:2006);
Version allemande EN 62056-62:2007, texte en anglais

Gesamtumfang 129 Seiten

DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

Beginn der Gültigkeit

Die von CENELEC am 2006-12-01 angenommene EN 62056-62 gilt als DIN-Norm ab 2007-08-01.

Daneben darf DIN EN 62056-62:2003-01 noch bis 2009-12-01 angewendet werden.

Nationales Vorwort

Vorausgegangener Norm-Entwurf: E DIN IEC 62056-62:2005-02.

Für diese Norm ist das nationale Arbeitsgremium K 461 „Elektrizitätszähler“ der DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE (<http://www.dke.de>) zuständig.

Die enthaltene IEC-Publikation wurde vom TC 13 „Equipment for electrical energy measurement and load control“ erarbeitet.

Das IEC-Komitee hat entschieden, dass der Inhalt dieser Publikation bis zu dem auf der IEC-Website unter „<http://webstore.iec.ch>“ mit den Daten zu dieser Publikation angegebenen Datum (maintenance result date) unverändert bleiben soll. Zu diesem Zeitpunkt wird entsprechend der Entscheidung des Komitees die Publikation

- bestätigt,
- zurückgezogen,
- durch eine Folgeausgabe ersetzt oder
- geändert.

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.

K 461 hat auf seiner Sitzung am 2006-09-19 festgestellt, dass die Bedingungen des DIN-Präsidialbeschlusses 1/2004 für die vorliegende Norm erfüllt sind. Demnach besteht die Europäische Norm EN 62056-62 nur in einer englischen Sprachfassung, weil sie nicht unmittelbar im Zusammenhang mit Rechtsvorschriften steht und nicht als Sicherheitsnorm anzusehen ist. Es kann davon ausgegangen werden, dass die Anwender dieser Norm über hinreichende englische Sprachkenntnisse verfügen. Für die meisten der verwendeten Begriffe existieren keine gebräuchlichen deutschen Benennungen, da sich die deutschen Anwender in der Regel ebenfalls der englischen Benennung bedienen.

Änderungen

Gegenüber DIN EN 62056-62:2003-01 wurden folgende Änderungen vorgenommen:
siehe Vorwort der EN 62056-62:2007-02.

Frühere Ausgaben

DIN EN 62056-62: 2003-01

Deutsche Fassung

Messung der elektrischen Energie –
Zählerstandsübertragung, Tarif- und Laststeuerung –
Teil 62: Interface-Klassen
(IEC 62056-62:2006)

Electricity metering –
Data exchange for meter reading, tariff and
load control –
Part 62: Interface classes
(IEC 62056-62:2006)

Equipements de mesure de l'énergie électrique –
Echange des données pour la lecture des
compteurs, le contrôle des tarifs et de la charge –
Partie 62: Classes d'interface
(CEI 62056-62:2006)

Diese Europäische Norm wurde von CENELEC am 2006-12-01 angenommen. Die CENELEC-Mitglieder sind gehalten, die CEN/CENELEC-Geschäftsordnung zu erfüllen, in der die Bedingungen festgelegt sind, unter denen dieser Europäischen Norm ohne jede Änderung der Status einer nationalen Norm zu geben ist.

Auf dem letzten Stand befindliche Listen dieser nationalen Normen mit ihren bibliographischen Angaben sind beim Zentralsekretariat oder bei jedem CENELEC-Mitglied auf Anfrage erhältlich.

Diese Europäische Norm besteht in drei offiziellen Fassungen (Deutsch, Englisch, Französisch). Eine Fassung in einer anderen Sprache, die von einem CENELEC-Mitglied in eigener Verantwortung durch Übersetzung in seine Landessprache gemacht und dem Zentralsekretariat mitgeteilt worden ist, hat den gleichen Status wie die offiziellen Fassungen.

CENELEC-Mitglieder sind die nationalen elektrotechnischen Komitees von Belgien, Bulgarien, Dänemark, Deutschland, Estland, Finnland, Frankreich, Griechenland, Irland, Island, Italien, Lettland, Litauen, Luxemburg, Malta, den Niederlanden, Norwegen, Österreich, Polen, Portugal, Rumänien, Schweden, der Schweiz, der Slowakei, Slowenien, Spanien, der Tschechischen Republik, Ungarn, dem Vereinigten Königreich und Zypern.

CENELEC

Europäisches Komitee für Elektrotechnische Normung
European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique

Zentralsekretariat: rue de Stassart 35, B-1050 Brüssel

© 2007 CENELEC – Alle Rechte der Verwertung, gleich in welcher Form und in welchem Verfahren, sind weltweit den Mitgliedern von CENELEC vorbehalten.

Ref. Nr. EN 62056-62:2007 D

Vorwort

Der Text des Schriftstücks 13/1389/FDIS, zukünftige 2. Ausgabe von IEC 62056-62, ausgearbeitet von dem IEC TC 13 „Electrical energy measurement, tariff- and load control“, wurde der IEC-CENELEC Parallelen Abstimmung unterworfen und von CENELEC am 2006-12-01 als EN 62056-62 angenommen.

Diese Europäische Norm ersetzt EN 62056-62:2002.

Die wesentlichen technischen Änderungen im Vergleich zu EN 62056-62:2002 sind:

- die Liste der Standarddatentypen wurde verbessert, einige neue Typen wurden ergänzt;
- Formatierung für Gleitpunktnummern wurde ergänzt;
- neue HLS-Mechanismen wurden ergänzt;
- instanzspezifische Datentypen wurden durch einen genau definierten Satz von anzuwendenden Datentypen ersetzt;
- neue Einheiten wurden ergänzt;
- die Codierung von Attributen für „application_context_name“ und „authentication_mechanism_name“ der Klasse „Association LN“ wurde verbessert;
- die neuen Interface-Klassen „Register table“ und „Status mapping“ wurden ergänzt;
- eine neue Version der Interface-Klassen „IEC local port setup“, „Modem configuration“, „Auto connect“ und „HDLC setup“ wurden ergänzt;
- neue Interface-Klassen zur Erstellung eines TCP/IP-basierten Kommunikationsprofils wurde ergänzt; Verweisungen auf zugehörige IETF RFCs und Standards sowie zugehörige Definitionen wurden ergänzt;
- einige Änderungen in Anhang D „Beziehung zu OBIS“ wurden vorgenommen.

Nachstehende Daten wurden festgelegt:

- spätestes Datum, zu dem die EN auf nationaler Ebene durch Veröffentlichung einer identischen nationalen Norm oder durch Anerkennung übernommen werden muss (dop): 2007-09-01
- spätestes Datum, zu dem nationale Normen, die der EN entgegenstehen, zurückgezogen werden müssen (dow): 2009-12-01

Die Internationale Elektrotechnische Kommission (IEC) und CENELEC weisen darauf hin, dass die Übereinstimmung mit dieser Internationalen Norm/Europäischen Norm die Inanspruchnahme eines Wartungs-Dienstes hinsichtlich der Protokolle, auf denen die vorliegende Norm IEC 62056-62/EN 62056-62 beruht, bedeuten kann.

Die IEC und CENELEC nehmen keine Stellung zur Rechtmäßigkeit, zur Gültigkeit und zum Anwendungsbereich dieses Wartungs-Dienstes.

Der Anbieter dieses Wartungs-Dienstes hat IEC zugesichert, dass er bereit ist, Antragstellern in der ganzen Welt Wartungs-Dienste zu vernünftigen und nicht diskriminierenden Geschäftsbedingungen anzubieten. In diesem Zusammenhang ist die Erklärung des Anbieters des Wartungs-Dienstes bei IEC registriert. Informationen (siehe auch 4.6.2 und Anhang E) sind erhältlich bei:

DLMS ¹⁾ User Association
Geneva / Switzerland
www.dlms.ch

Der Anhang ZA wurde von CENELEC hinzugefügt.

¹⁾ Device Language Message Specification

Anerkennungsnotiz

Der Text der Internationalen Norm IEC 62056-62:2006 wurde von CENELEC ohne irgendeine Abänderung als Europäische Norm angenommen.

In der offiziellen Fassung ist unter „Literaturhinweise“ zu der aufgelistete Norm die nachstehende Anmerkung einzutragen:

IEC 61334-6 ANMERKUNG Harmonisiert als EN 61344-6:2000 (nicht modifiziert).

CONTENTS

FOREWORD.....	2
1 Scope	7
2 Normative references	7
3 Terms, definitions and abbreviations	8
4 Basic principles	9
4.1 General	9
4.2 Class description notation	10
4.3 Common data types	12
4.4 Data formats	13
4.5 The COSEM server model	17
4.6 COSEM logical device	18
4.7 Authentication procedures	19
5 The interface classes	20
5.1 Data (class_id: 1)	22
5.2 Register (class_id: 3)	22
5.3 Extended register (class_id: 4)	26
5.4 Demand register (class_id: 5)	27
5.5 Register activation (class_id: 6)	30
5.6 Profile generic (class_id: 7)	32
5.7 Clock (class_id: 8)	37
5.8 Script table (class_id: 9)	40
5.9 Schedule (class_id: 10)	41
5.10 Special days table (class_id: 11)	44
5.11 Activity calendar (class_id: 20)	45
5.12 Association LN (class_id: 15)	48
5.13 Association SN (class_id: 12)	53
5.14 SAP assignment (class_id: 17)	56
5.15 Register monitor (class_id: 21)	56
5.16 Utility tables (class_id: 26)	57
5.17 Single action schedule (class_id: 22)	59
5.18 Register table (class_id: 61)	60
5.19 Status mapping (class_id: 63)	62
6 Maintenance of the interface classes	63
6.1 New interface classes	63
6.2 New versions of interface classes	63
6.3 Removal of interface classes	63
Annex A (normative) Protocol related interface classes	64
Annex B (normative) Data model and protocol	84
Annex C (normative) Using short names for accessing attributes and methods	85
Annex D (normative) Relation to OBIS	94
Annex E (informative) Previous versions of interface classes	116
Bibliography	122
Index	123
Annex ZA (normative) Normative references to international publications with their corresponding European publications	126

Figure 1 – An interface class and its instances	10
Figure 2 – The COSEM server model	17
Figure 3 – Combined metering device	17
Figure 4 – Overview of the interface classes.....	21
Figure 5 – The attributes when measuring sliding demand	27
Figure 6 – The attributes when measuring current_average_value if number of periods is 1	27
Figure 7 – The attributes if the number of periods is 3	28
Figure 8 – The generalized time concept	38
Figure B.1 – The three step approach of COSEM	84
 Table 1 – Common data types.....	 12

INTRODUCTION

Driven by the need of the utilities to optimize their business processes, the meter becomes more and more part of an integrated metering and billing system. Whereas in the past the commercial value of a meter was mainly generated by its data acquisition and processing capabilities, nowadays the critical issues are system integration and interoperability.

The Companion Specification for Energy Metering (COSEM) addresses these challenges by looking at the meter as an integrated part of a commercial process, which starts with the measurement of the delivered product (energy) and ends with the revenue collection.

The meter is specified by its “behaviour” as seen from the utility's business processes. The formal specification of the behaviour is based on object modelling techniques (interface classes and objects). The specification of these objects forms a major part of COSEM.

The COSEM server model (see 4.5) represents only the externally visible elements of the meter. The client applications that support the business processes of the utilities, customers and meter manufacturers make use of this server model. The meter offers means to retrieve its structural model (the list of objects visible through the interface), and provides access to the attributes and specific methods of these objects.

The set of different interface classes form a standardized library from which the manufacturer can assemble (model) its individual products. The elements are designed so that with them the entire range of products (from residential to commercial and industrial applications) can be covered. The choice of the subset of interface classes used to build a meter, their instantiation, and their implementation 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.

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.

IEC 60050-300:2001, *International Electrotechnical Vocabulary – Electrical and electronic measurements and measuring instruments – Chapter 311: General terms relating to measurements – Chapter 312: General terms relating to electrical measurements – Chapter 313: Types of electrical measuring instruments – Chapter 314: Specific terms according to the type of instrument*

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

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

IEC 62051:1999, *Electricity metering – Glossary of terms*

IEC 62051-1:2004, *Electricity metering – Data exchange for meter reading, tariff and load control – Glossary of terms – Part 1: Terms related to data exchange with metering equipment using DLMS/COSEM*

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

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

IEC 62056-46:2002, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 46: Data link layer using HDLC-protocol*
Amendment 1²

IEC 62056-47:2006, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 47: COSEM transport layers for IPv4 networks*

IEC 62056-53:2006, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 53: COSEM Application layer*

² To be published.

IEC 62056-61:2006, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 61: Object identification system(OBIS)*

ANSI C12.19:1997 / IEEE 1377:1997, *Utility Industry End Device Data Tables*

STD 0005: 1981, *Internet Protocol (Also: IETF RFC 0791, RFC 0792, RFC 0919, RFC 0922, RFC 0950, RFC 1112)*

STD 0051: 1994, *The Point-to-Point Protocol (PPP) (Also: IETF RFC 1661, RFC 1662)*

3 Terms, definitions and abbreviations

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in IEC 60050-300, IEC 62051, and IEC 62051-1 apply.

3.2 Abbreviations

AARE	Application Association Response
AARQ	Application Association ReQuest
ACSE	Application Control Service Element
APDU	Application Protocol Data Unit
ASE	Application Service Element
A-XDR	Adapted eXtended Data Representation
CHAP	Challenge Handshake Authentication Protocol
COSEM	Companion Specificationfor Energy Metering
CtoS	Client to Server Challenge
DHCP	Dynamic Host Control Protocol
DLMS	Device Language Message Specification
DNS	Domain Name Server
EAP	Extensible Authentication Protocol
GMT	Greenwich Mean Time
GPS	Global Positioning System
HLS	High Level Security
IANA	Internet Assigned Numbers Authority
IC	Interface Class
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPCP	Internet Protocol Control Protocol
LCP	Link Control Protocol
LLS	Low Level Security
LN	Logical Name
LSB	Least Significant Bit
m	mandatory
MD5	Message Digest Algorithm 5
MSB	Most Significant Bit
o	Optional

OBIS	OBject Identification System
PAP	Password Authentication Protocol
PDU	Protocol Data Unit
PLMN	Public Land Mobile Network
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

4 Basic principles

4.1 General

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

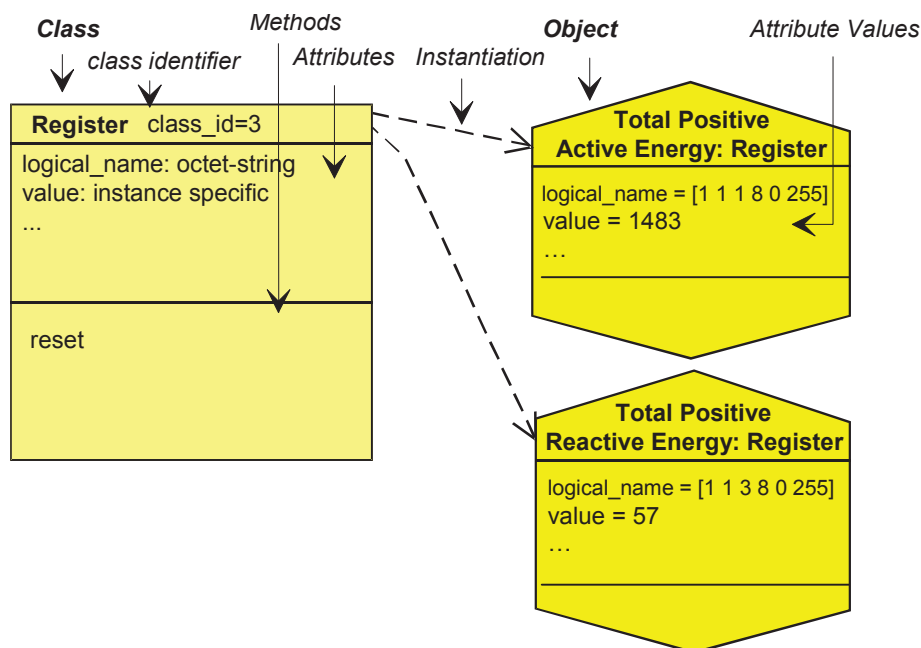
Object modelling: for specification purposes this standard uses the technique of object modelling. An object is a collection of attributes and methods.

The information of an object is organized in attributes. They represent the characteristics of an object by means of attribute values. The value of an attribute may affect the behaviour of an object. The first attribute in any object is the “logical_name”. It is one part of the identification of the object. An object may offer a number of methods to either examine or modify the values of the attributes.

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

Manufacturers may add proprietary methods or attributes to any object, using negative numbers.

Figure 1 illustrates these terms by means of an example:



IEC 305/02

Figure 1 – An interface class and its instances

The interface class “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 (central unit, hand-held terminal). The contents of the register are identified by the attribute “logical_name”. The logical_name contains an OBIS identifier (see IEC 62056-61). The actual (dynamic) content of the register is carried by its “value” attribute.

Defining a specific meter means defining several specific registers. In the example of Figure 1, the meter contains two registers, i.e. two specific COSEM objects of the class “Register” are instantiated. This means that specific values are assigned to the different attributes. Through the instantiation, one COSEM object becomes a “total, positive, active energy register” whereas the other becomes a “total, positive, reactive energy register”.

REMARK The COSEM objects (instances of interface classes) represent the behaviour of the meter as seen from the “outside”. Therefore, modifying the value of an attribute must always be initiated from the outside (e.g. resetting the value of a register). Internally initiated changes of the attributes are not described in this model (e.g. updating the value of a register).

4.2 Class description notation

This subclause describes the notation used to define the interface classes.

A short text describes the functionality and application of the class. A table gives an overview of the class including the class name, the attributes, and the methods (class description template).

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

Each attribute and method must be described in detail.

Class name	Describes the class (e.g. "Register", "Clock", "Profile generic",...)	
Cardinality	Specifies the number of instances of the class within a logical device (see 4.6).	
	<i>value</i>	The class shall be instantiated exactly "value" times.
	<i>min...max.</i>	The class shall be instantiated at least "min." times and at most "max." times. If min. is zero (0) then the class is optional, otherwise (min. > 0) "min." instantiations of the class are mandatory.
class_id	Identification code of the class (range 0 to 65 535). The class_id of each object is retrieved together with the logical name by reading the object_list attribute of an "Association LN" / "Association SN" object. The 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 interface classes. Class_id-s from 32 768 to 65 535 are reserved for user group specific interface classes. DLMS UA reserves the right to assign ranges to individual manufacturers or user groups.	
Version	Identification code of the version of the class. The version of each object is retrieved together with the logical name and the class_id by reading the object_list attribute of an "Association LN" / "Association SN" object. Within one logical device, all instances of a certain class must be of the same version.	
Attribute(s)	Specifies the attribute(s) that belong to the class.	
	<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 (e.g. configuration data).
logical_name	octet-string	The logical name is always the first attribute of a class. It identifies the instantiation (COSEM object) of this class. The value of the logical_name conforms to OBIS (see IEC 62056-61).
Data type	Defines the data type of an attribute (see 4.3).	
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 class definition.

Specific method(s)	Provides a list of the specific methods that belong to the object.	
	<i>Method Name ()</i>	The method has to be described in the subsection "Method description".
m/o	Defines if the method is mandatory or optional.	
	<i>m (mandatory)</i>	The method is mandatory.
	<i>o (optional)</i>	The method is optional.

Attribute description

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

Method description

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

NOTE Services for accessing attributes or methods by the protocol are described in IEC 62056-53.

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.

4.3 Common data types

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

Table 1 – 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	
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

Table 1 (continued)

Type description	Tag ^a	Definition	Value range
enum	[22]	The elements of the enumeration type are defined in the "Attribute description" section of a COSEM interface class specification.	
float32	[23]	OCTET STRING (SIZE(4))	For formatting, see 4.4.2.
float64	[24]	OCTET STRING (SIZE(8))	
date_time	[25]	OCTET STRING (SIZE(12))	For formatting, see 4.4.1.
date	[26]	OCTET STRING (SIZE(5))	
time	[27]	OCTET STRING (SIZE(4))	
--complex data types			
array	[1]	The elements of the array are defined in the "Attribute description" section of a COSEM interface class specification.	
structure	[2]	The elements of the structure are defined in the "Attribute description" section of a COSEM interface class specification.	
compact array	[19]	The elements of the compact array are defined in the "Attribute description" section of a COSEM interface class 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 interface class specification.	

^a The tags are as defined in IEC 62056-53, 8.3.

4.4 Data formats

4.4.1 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 interface class definition.

NOTE 1 In future versions of interface classes and in newly defined interface classes, only the data types *date*, *time* and *date_time* will be used.

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

The elements dayOfMonth and dayOfWeek have to be interpreted together:

- if last day of month is specified (0xFE) and day of week is wildcard, this specifies the last calendar day of the month;
- if last day of month is specified (0xFE) and an explicit day of week is specified (e.g. 7, Sunday) then it is the last occurrence of the weekday specified in the month, i.e. the last Sunday;
- if the dayOfMonth and dayOfWeek elements are both explicitly defined and they are not consistent, (for example 24th of the month is not Wednesday in the given year and month) it shall be considered as an error.

time

OCTET STRING (SIZE(4))

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

hour: interpreted as unsigned

range 0...23, 0xFF

0xFF = not specified,

minute: interpreted as unsigned

range 0...59, 0xFF

0xFF = not specified,

second: interpreted as unsigned

range 0...59, 0xFF

0xFF = not specified,

hundredths: interpreted as unsigned

range 0...99, 0xFF

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*.

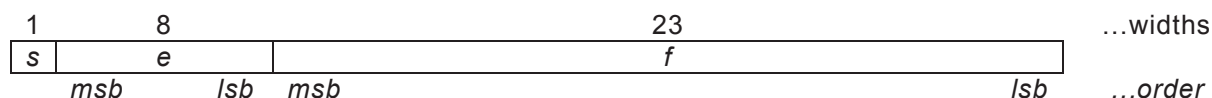
- ^a Time could not be recovered after an incident. Detailed conditions are manufacturer specific (e.g. after the power to the clock has been interrupted).
- ^b Time could be recovered after an incident but the value cannot be guaranteed. Detailed conditions are manufacturer specific.
- ^c Bit is set if the basic timing information for the clock is at the actual moment 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).

4.4.2 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:



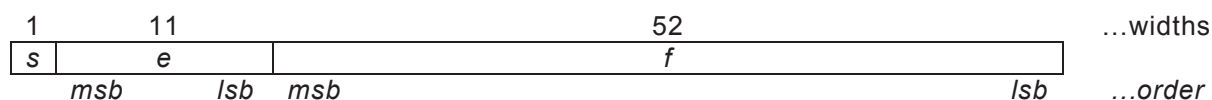
where:

- 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:



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 (of $0 < e < 2\,047$):

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

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:

[illegible]

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

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

Bit 31 Sign bit 0 0: + 1: -	Bits 30-23 Exponent field 10001110 Decimal value of exponent field and exponent: $142-127 = 15$	Bits 22-0 Significand 1.11100100110100000000000 Decimal value of the significand: 1.8937988
--	--	---

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

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

[illegible]

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

4.5 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.

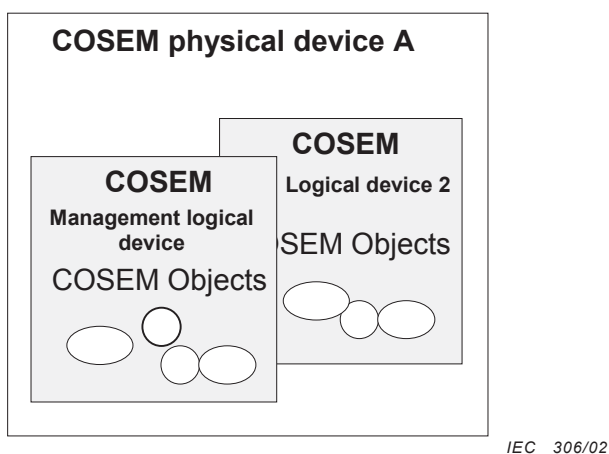


Figure 2 – The COSEM server model

The following example (see Figure 3) shows how a combined metering device can be structured using the COSEM server model.

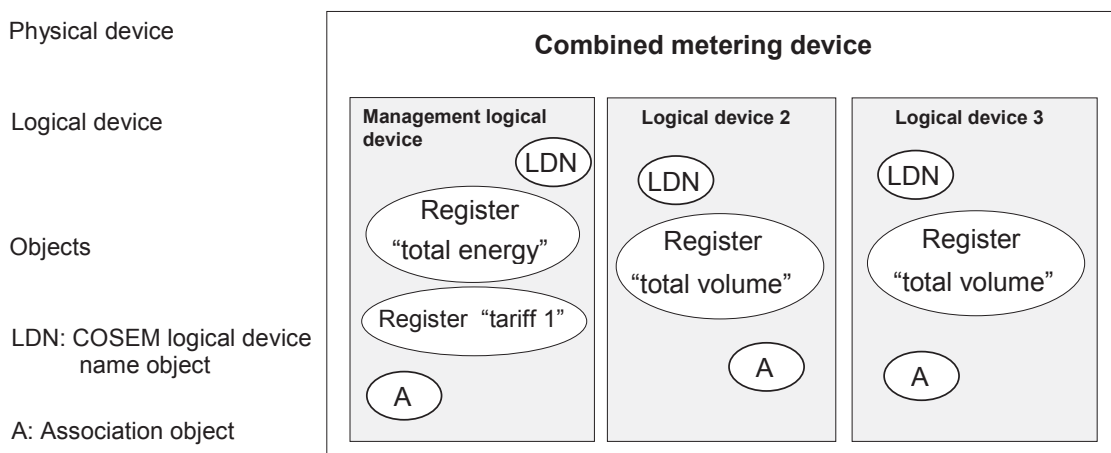


Figure 3 – Combined metering device

4.6 COSEM logical device

4.6.1 General

The COSEM logical device is 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 used.

4.6.2 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 5.14), or of a COSEM object “COSEM logical device name” (see D.2.1.24).

This name is defined as an octet-string of up to 16 octets. The first three octets uniquely identify the manufacturer of the device ³. The manufacturer is responsible for guaranteeing the uniqueness of the octets that follow (up to 13 octets).

4.6.3 The “association view” of the logical device

In order to access COSEM objects in the server, an application association shall first be established. This 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.

This information is contained in a special COSEM object, the “Association” object. There are two types of this association object defined. One for associations using short name referencing (“Association SN”) and one for using logical name referencing (“Association LN”).

Depending on the association 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 association. In addition, access to attributes and methods of these COSEM objects may also be restricted within the association (e.g. a certain type of client can only read a particular attribute of a COSEM object).

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. Additional information about the access rights (read only, write only, read and write) to the attributes and the availability of the methods (within the established association) can be found via specific attributes (logical name referencing, see 5.12) or special methods (short name referencing, see 5.13) provided by the association objects.

4.6.4 Mandatory contents of a COSEM logical device

The following objects shall be part of each COSEM logical device. They shall be accessible for GET/READ in all application associations with this logical device:

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

³ Administered by the DLMS User Association

4.6.5 Management logical device

As specified in 4.6.1, the management logical device is a mandatory element of any physical device, and it has a reserved address. As defined in 6.3.4 of IEC 62056-53, it must support an application association 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 association with the public client, the management logical device shall contain a “SAP assignment” object, giving its SAP and the SAP of all other logical devices within the physical device. The SAP assignment object 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.

4.7 Authentication procedures

4.7.1 Low Level Security (LLS) authentication

As described in IEC 62056-53 the ACSE provides the authentication services for low level security (LLS). Low level security authentication is typically used when the communication channel offers adequate security to avoid eavesdropping and message (password) replay.

For LLS, all the authentication services are provided by the ACSE. The association objects provide only the method/attribute (see 5.12, 5.13) to change the “secret” (e.g. password).

For LLS authentication the client transmits a “secret” (e.g. a password) to the server, by using the “Calling_Authentication_Value” parameter of the COSEM-OPEN.request service primitive of the client application layer. The server checks if the received “secret” corresponds to the client identification. If yes, the client is authenticated and the association can be established.

4.7.2 High Level Security (HLS) authentication

As described in IEC 62056-53, the ACSE provides part of the authentication services for high level security (HLS). High level security authentication is typically used when the communication channel offers no intrinsic security and precautions have to be taken against eavesdroppers and against message (password) replay. In this case, a 4-pass authentication protocol is foreseen. The 4-pass authentication allows the authentication of the client as well as of the server in the following way.

Pass1: The client transmits “challenge” CtoS (e.g. a random number) to the server.

Pass2: The server transmits “challenge” StoC (e.g. a random number) to the client.

The length of the challenges shall be 8 to 64 octets.

Pass3: The client processes StoC according to the rules of the HLS authentication mechanism negotiated. In case of HLS authentication_mechanism_id(2), the method of processing the challenge is secret (e.g. encrypting with a secret key) which is the HLS secret known by both the client and the server. In case of HLS authentication_mechanism_id(3), the client appends the HLS secret to the challenge StoC received during Pass2 and generates the digest using the MD5 algorithm (RFC 1321). In case of authentication_mechanism_id(4), the process is the same, but the SHA_1 algorithm is used (FIPS 180-1). The result – f(StoC) – is sent back to the server. The server checks if f(StoC) is the result of correct processing and – if correct – accepts the authentication of the client.

Pass4: If the client is authenticated, the server processes CtoS in the same way as described in Pass3. The result – f(CtoS) – is sent back to the client. The client checks if f(CtoS) is the result of the correct processing and – if correct – accepts the authentication of the server.

The HLS authentication service, supporting Pass1 is provided by the COSEM-OPEN.request service primitive of the client application layer. The parameter "Security_Mechanism_Name" carries the identifier of the HLS mechanism, and the parameter "Calling_Authentication_Value" carries the challenge CtoS.

The HLS authentication service, supporting Pass2 is provided by the COSEM-OPEN.response service primitive of the server application layer. The parameter "Security_Mechanism_Name" carries the identifier of the HLS mechanism, and the parameter "Responding_Authentication_Value" carries the challenge Stoc.

After Pass2, the association is formally established, but the access of the client is restricted to the method "reply_to_HLS_authentication" of the current "association" object.

Pass3 and Pass4 are supported by the method **reply_to_HLS_authentication** of the association object(s), (see 5.12, 5.13). If both passes are successfully executed, then full access is granted according to the current association. Otherwise, either the client or the server aborts the association.

In addition, the association object provides the method to change the HLS "secret" (e.g. the encryption key): **change_HLS_secret**.

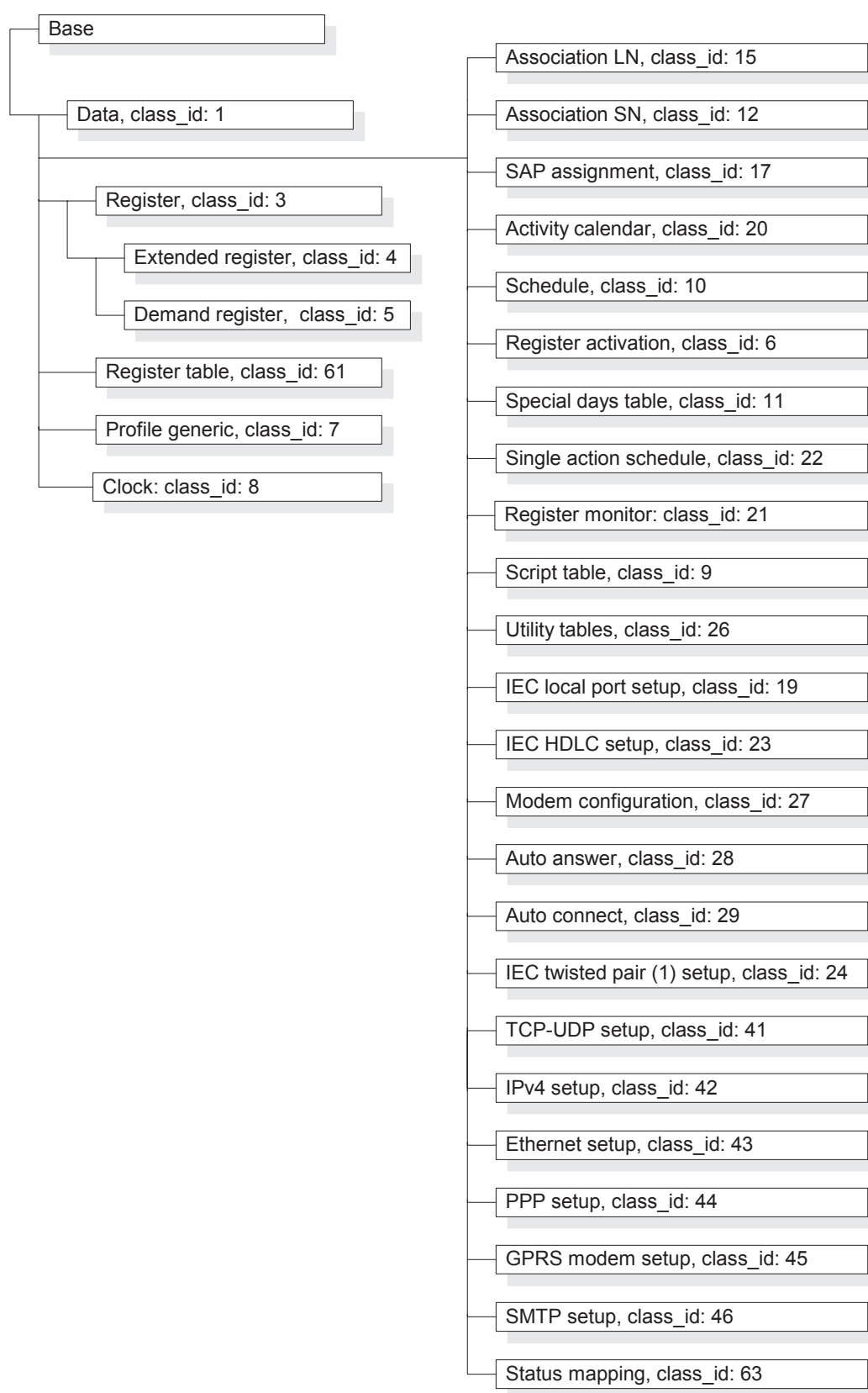
REMARK After the client has issued the change_HLS_secret () - or change_LLS_secret () - method, it expects a response from the server acknowledging that the secret has been changed. It is possible that the server transmits the acknowledgement, but due to communication problems, the acknowledgement is not received at the client-side. Therefore, the client does not know if the secret has been changed or not. For simplicity reasons, the server does **not** offer any special support for this case; i.e. it is left to the client to cope with this situation.

5 The interface classes

The currently defined interface classes for meters and the relations between them are illustrated in Figure 4.

NOTE 1 The interface class "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" interface classes, the 2nd attributes are labelled differently from that of the 2nd attribute of the "Data" interface class, namely "current_average_value", "time" and "buffer" vs. "value". This is to emphasize the specific nature of the "value".



IEC 2111/06

Figure 4 – Overview of the interface classes

5.1 Data (class_id: 1)

A “Data” object stores data related to internal meter object(s). The meaning of the value is identified by the logical_name. The data type of the value is CHOICE. “Data” is typically used to store configuration data and parameters.

Data	0...n	class_id = 1, version = 0		
Attribute(s)	Data type	Min.	Max.	Def.
1. logical_name (static) 2. value	octet-string CHOICE			
Specific method(s)	m/o			

Attribute description

logical_name	Identifies the “Data” object instance. Identifiers are specified in Clause D.22 and in IEC 62056-61.		
value	Contains the data.		
	CHOICE		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 4.3 can be used, unless the choice is restricted in Annex D.
	{		
	--simple data types		
	null-data	[0],	
	boolean	[3],	
	bit-string	[4],	
	double-long	[5],	
	double-long-unsigned	[6],	
	octet-string	[9],	
	visible-string	[10],	
	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]	
	}		

5.2 Register (class_id: 3)

A “Register” object stores a process value or a status value with its associated unit. The register object knows the nature of the process value or of the status value. The nature of the value is described by the attribute “logical name” using the OBIS identification system (see Clause D.2 and IEC 62056-61).

Register	0...n	class_id = 3, version = 0		
Attribute(s)	Data type	Min.	Max.	Def.
1. logical_name (static) 2. value (dyn.) 3. scaler_unit (static)	octet-string CHOICE scal_unit_type			
Specific method(s)	m/o			
1. reset (data)	o			

Attribute description

logical_name	Identifies the “Register” object instance. Identifiers are specified in Clause D.2 and in IEC 62056-61.			
value	Contains the current process or status value. CHOICE { --simple data types null-data [0], bit-string [4], double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], float32 [23], float64 [24] } The data type of the value depends on the instantiation defined by “logical_name” and possibly on the choice of the manufacturer. It has to be chosen so that, together with the logical_name, an unambiguous interpretation of the value is possible. When instead of a “Data” object a “Register” object is used, (with the scaler_unit attribute not used or with scaler = 0, unit = 255) then the data types allowed for the value attribute of the “Data” interface class are allowed.			
scaler_unit	Provides information on the unit and the scaler of the value. scal_unit_type: structure { scaler, unit } scaler: integer unit: enum 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. Enumeration defining the physical unit; for details see 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. data ::= integer(0)
---------------------	--

unit ::= enum

Code	// 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

Code	// Unit	Quantity	Unit name	SI definition (comment)
(37)	F	// capacitance (C)	farad	$C/V = As/V$
(38)	Ω	// resistance (R)	ohm	$\Omega = V/A$
(39)	$\Omega m^2/m$	// resistivity (ρ)		Ωm
(40)	Wb	// magnetic flux (Φ)	weber	$Wb = Vs$
(41)	T	// Magnetic flux density (B)	tesla	Wb/m^2
(42)	A/m	// magnetic field strength (H)	ampere per metre	A/m
(43)	H	// inductance (L)	henry	$H = Wb/A$
(44)	Hz	// frequency (f, ω)	hertz	1/s
(45)	1/(Wh)	// R_W , active energy meter constant or pulse value		
(46)	1/(varh)	// R_B , reactive energy meter constant or pulse value		
(47)	1/(VAh)	// R_S , apparent energy meter constant or pulse value		
(48)	V ² h	// volt-squared hour r_{U2h} , volt-squared hour meter constant or pulse value	volt-squared-hours	$V^2(60*60s)$
(49)	A ² h	// ampere-squared hour r_{I2h} , ampere-squared hour meter constant or pulse value	ampere-squared-hours	$A^2(60*60s)$
(50)	kg/s	// mass flux	kilogram per second	kg/s
(51)	S, mho	// conductance	siemens	1/ Ω
(52)	K	// temperature (T)	kelvin	
(53)	1/(V ² h)	// R_{U2h} , volt-squared hour meter constant or pulse value		
(54)	1/(A ² h)	// R_{I2h} , ampere-squared hour meter constant or pulse value		
(55)	1/m ³	// R_V , meter constant or pulse value (volume)		
(56)		// percentage	%	
(57)	Ah	// ampere-hours	Ampere-hour	
(60)	Wh/m ³	// energy per volume	$3,6*10^3 J/m^3$	
(61)	J/m ³	// calorific value, wobbe		
(62)	Mol %	// molar fraction of gas composition	mole percent	(Basic gas composition unit)
(63)	g/m ³	// mass density, quantity of material		(Gas analysis, accompanying elements)
(64)	Pa s	// dynamic viscosity	pascal second	(Characteristic of gas stream)
....				
(253)		// reserved		
(254)	other	// other unit		
(255)	count	// no unit, unitless, count		

Examples of values:

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

5.3 Extended register (class_id: 4)

Instances of an “Extended register” class store a process value with its associated status, unit, and time information. The extended register object knows the nature of the process value. The nature of the value is described by the attribute “logical name” using the OBIS identification system.

Extended register		0...n	class_id = 4, version = 0		
Attribute(s)		Data type	Min.	Max.	Def.
1. logical_name	(static)	octet-string			
2. value	(dyn.)	CHOICE			
3. scaler_unit	(static)	scal_unit_type			
4. status	(dyn.)	CHOICE			
5. capture_time	(dyn.)	octet-string			
Specific method(s)		m/o			
1. reset (data)		o			

Attribute description

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

logical_name	Identifies the “Extended register” object instance. See D.2.2.1.																																									
status	Provides “Extended register” specific status information. The meaning of the elements of the status shall be provided for each “Extended register” object instance. <table><tr><td>CHOICE</td><td colspan="2">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.</td></tr><tr><td>{</td><td colspan="2"></td></tr><tr><td>--simple data types</td><td colspan="2"></td></tr><tr><td>null-data</td><td>[0],</td><td></td></tr><tr><td>bit-string</td><td>[4],</td><td></td></tr><tr><td>double-long-unsigned</td><td>[6],</td><td></td></tr><tr><td>octet-string</td><td>[9],</td><td></td></tr><tr><td>visible-string</td><td>[10],</td><td></td></tr><tr><td>unsigned</td><td>[17],</td><td></td></tr><tr><td>long-unsigned</td><td>[18],</td><td></td></tr><tr><td>long64-unsigned</td><td>[21]</td><td></td></tr><tr><td>}</td><td colspan="2"></td></tr><tr><td>Def.</td><td colspan="2">Depending on the status type definition.</td></tr></table>			CHOICE	The data type and the encoding depend on the instantiation and possibly on the choice of the manufacturer. For the interpretation, extra information from the manufacturer may be necessary.		{			--simple data types			null-data	[0],		bit-string	[4],		double-long-unsigned	[6],		octet-string	[9],		visible-string	[10],		unsigned	[17],		long-unsigned	[18],		long64-unsigned	[21]		}			Def.	Depending on the status type definition.	
CHOICE	The data type and the encoding depend on the instantiation and possibly on the choice of the manufacturer. For the interpretation, extra information from the manufacturer may be necessary.																																									
{																																										
--simple data types																																										
null-data	[0],																																									
bit-string	[4],																																									
double-long-unsigned	[6],																																									
octet-string	[9],																																									
visible-string	[10],																																									
unsigned	[17],																																									
long-unsigned	[18],																																									
long64-unsigned	[21]																																									
}																																										
Def.	Depending on the status type definition.																																									
capture_time	Provides an “Extended register” specific date and time information showing when the value of the attribute "value" has been captured. octet-string, formatted as set in 4.4.1 for <i>date_time</i>																																									

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)
---------------------	---

5.4 Demand register (class_id: 5)

Instances of a “Demand register” class store a demand value with its associated status, unit, and time information. The demand register measures and computes its *current_average_value* periodically. The time interval *T* over which the demand is measured or computed is defined by specifying “number_of_periods” and “period”.

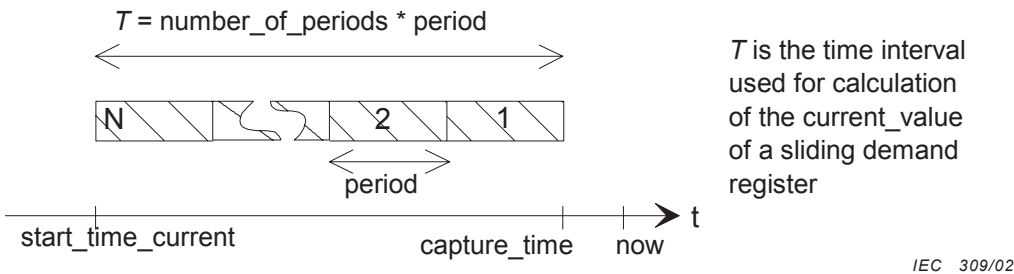


Figure 5 – The attributes when measuring sliding demand

The demand register delivers two types of demand: the *current_average_value* and the *last_average_value* (see Figure 6 and Figure 7).

The demand register knows its type of process value, which is described in “logical name” using the OBIS identification system.

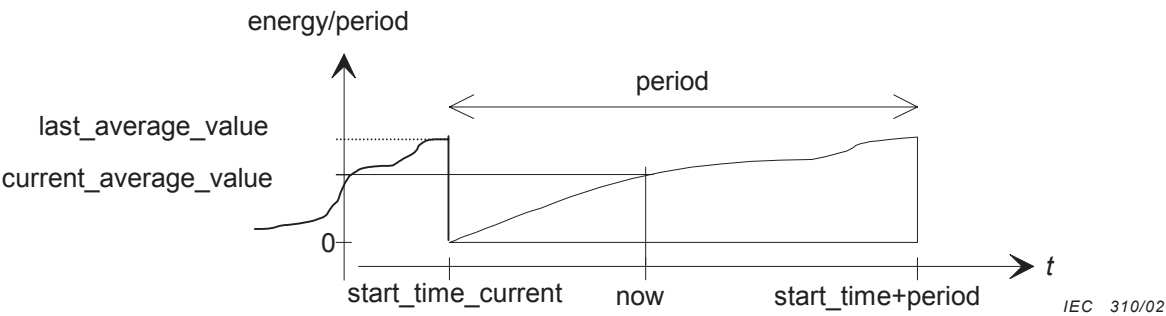
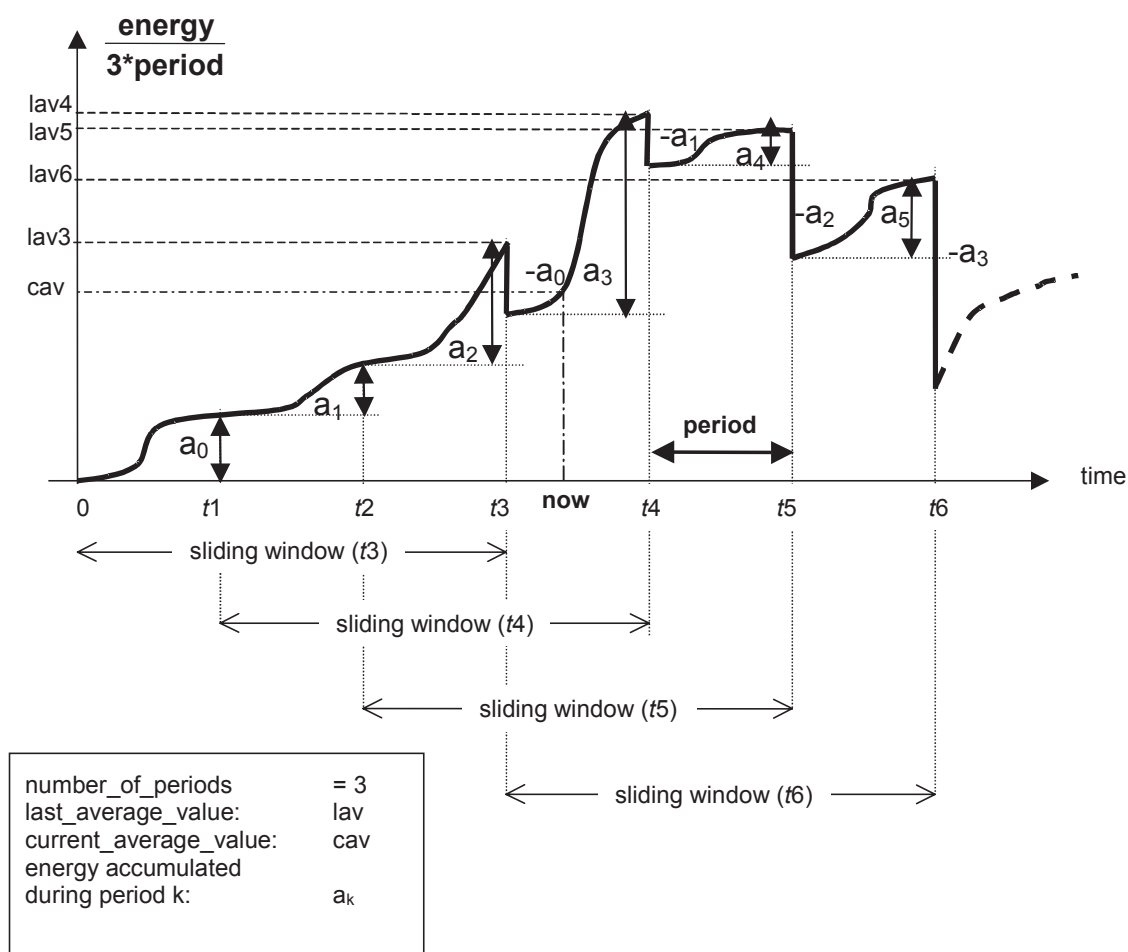


Figure 6 – The attributes when measuring current_average_value if number of periods is 1



IEC 311/02

Figure 7 – The attributes if the number of periods is 3

Demand register		0...n	class_id = 5, version = 0		
Attribute(s)		Data type	Min.	Max.	Def.
1. logical_name	(static)	octet-string			
2. current_average_value	(dyn.)	CHOICE			0
3. last_average_value	(dyn.)	CHOICE			0
4. scaler_unit	(static)	scal_unit_type			
5. status	(dyn.)	CHOICE			
6. capture_time	(dyn.)	octet-string			
7. start_time_current	(dyn.)	octet-string			
8. period	(static)	double-long-unsigned	1		
9. number_of_periods	(static)	long-unsigned	1		1
Specific method(s)		m/o			
1. reset (data)		o			
2. next_period (data)		o			

Attribute description

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

logical_name	Identifies the “Demand register” object instance. See D.2.2.1.	
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>. NOTE If another <i>quantity</i> than energy is measured, other calculation methods may apply (for example for calculating average values of voltage or current). CHOICE	The data type of the value depends on the instantiation defined by “logical_name” and possibly on the choice of the manufacturer. The type has to be chosen so that, together with the logical_name, unambiguous interpretation of the value is possible. For data types, see “Register” class, value attribute.
last_average_value	Provides the value of the accumulated energy (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 another <i>quantity</i> 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” class, 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 Def.	For data types, see “Extended register” class, status attribute. Depending on the status type definition.
capture_time	Provides the date and time when the last_average_value has been calculated. octet-string, formatted as set in 4.4.1 for <i>date_time</i>	
start_time_current	Provides the date and time when the measurement of the current_average_value has been started. octet-string, formatted as set in 4.4.1 for <i>date_time</i>	
period	Period is the interval between two successive updates of the last_average_value. (<code>number_of_periods*period</code> is the denominator for the calculation of the demand). double-long-unsigned The behaviour of the meter after writing a new value to this attribute shall be specified by the manufacturer.	
	Measuring period in seconds	

number_of_periods	The number of periods used to calculate the last_average_value. number_of_periods >= 1		
	long-unsigned	number_of_periods > 1 indicates that the last_average_value represents "sliding demand". number_of_periods = 1 indicates that the last_average_value represents "block demand".	
The behaviour of the meter after writing a new value to this attribute shall be specified by the manufacturer.			

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)			

5.5 Register activation (class_id: 6)

Instances of the "Register activation" class are used to handle 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, which are not defined in the register_assignment attribute of any "Register activation" object, are always enabled by default.

Register activation		0...n	class_id = 6, version = 0		
Attribute(s)		Data type	Min.	Max.	Def.
1. logical_name	(static)	octet-string			
2. register_assignment	(static)	array			
3. mask_list	(static)	array			
4. active_mask	(dyn.)	octet-string			
Specific method(s)		m/o			
1. add_register (data)		o			
2. add_mask (data)		o			
3. delete_mask (data)		o			

Attribute description

logical_name	Identifies the “Register activation” object instance. See D.2.1.9.
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”. <pre> array object_definition 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 mask_name and contains an array of indices referring to the registers assigned to the mask (the first object in register_assignment is referenced by index 1, the second object by index 2, ...). <pre> array register_act_mask register_act_mask ::= structure { mask_name: octet-string, index_list: index_array } index_array ::= array unsigned mask_name has to be uniquely defined within the object. </pre>
active_mask	Defines the currently active mask. The mask is defined by its mask_name (see mask_list). octet-string This is a mask_name from the 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 register to the attribute register_assignment. The new register is added at the end of the array; i.e. the newly added register has the highest index. The indices of the existing registers are not modified. <pre> data ::= structure { class_id: long-unsigned, logical_name: octet-string } </pre>
----------------------------	--

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)

5.6 Profile generic (class_id: 7)

The “Profile generic” class defines a generalized concept to store dynamic process values of capture objects. Capture objects are appropriate attributes or element 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 whose values or entry numbers fall within the given range.

The list of capture objects defines the values to be stored in the buffer (using 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 registers or by a clock, or the entries are stacked in a “last in first out” order. So 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. 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. This will be zero after clearing the profile;
- the maximum number of entries to retain. 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		
Attribute(s)	Data type	Min.	Max.	Def.
1. logical_name (static)	octet-string			
2. buffer (dyn.)	compact array or array			x
3. capture_objects (static)	array			
4. capture_period (static)	double-long-unsigned			x
5. sort_method (static)	enum			x
6. sort_object (static)	object_definition			x
7. entries_in_use (dyn.)	double-long-unsigned	0		0
8. profile_entries (static)	double-long-unsigned	1		1
Specific method(s)	m/o			
1. reset (data)	o			
2. capture (data)	o			
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 D.2.1.13, D.2.1.26, D.2.1.29, D.2.1.30, D.2.1.36, D.2.2.3.
buffer	The buffer attribute contains a sequence of entries. Each entry contains values of the captured objects (as they would be returned to a GET or READ.request). compact-array or array entry entry ::= <pre> 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], bcd [13], integer [15], long [16], unsigned [17], long-unsigned [18], long64 [20], long64-unsigned [21], enum [22], float32 [23], float64 [24], date-time [25], date [26], time [27], --complex data types array [1], structure [2], compact-array [19] } </pre>

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

Default: The buffer is empty after reset.

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

REMARK 2 The value of a captured object may be replaced by "null-data" if it can be unambiguously recovered from the previous value (e.g. 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 4.2) 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 <code>capture (data)</code> method or automatically in defined intervals, the selected attributes are copied into the buffer of the profile. <pre> array capture_object_definition capture_object_definition ::= structure { class_id: long-unsigned, logical_name: octet-string, attribute_index: integer, data_index: long-unsigned } </pre> - where <code>attribute_index</code> is a pointer to the attribute within the object. <code>attribute_index 1</code> refers to the first attribute (i.e. the <code>logical_name</code>), <code>attribute_index 2</code> to the 2nd, etc.); <code>attribute_index 0</code> refers to all public attributes; - where <code>data_index</code> is a pointer selecting a specific element of the attribute. The first element in the attribute structure is identified by <code>data_index 1</code>. If the attribute is not a structure, then the <code>data_index</code> has no meaning. If the capture object is the buffer of a profile, then the <code>data_index</code> identifies the captured object of the buffer (i.e. the column) of the inner profile. <code>data_index 0</code>: references the whole attribute
capture_period	<code>>= 1</code>: Automatic capturing assumed. Specifies the capturing period in seconds. <code>0</code>: 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 <code>capture ()</code> 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 <code>capture ()</code> will store the new entry at the appropriate position in the buffer, moving all following entries and probably losing the least interesting entry. If the new entry would enter the buffer after the last entry and if the buffer is already full, the new entry will not be retained at all.

	enum	(1) fifo (first in first out), (2) lifo (last in first out), (3) largest, (4) smallest, (5) nearest_to_zero, (6) farthest_from_zero
	Def.	fifo
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)
entries_in_use	Counts the number of entries stored in the buffer. After a call of reset (), the buffer does not contain any entries, and this value is zero. Upon each subsequent call of capture (), this value will be incremented up to the maximum number of entries that will get stored (see 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 value	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	CHOICE {--simple data types double-long [5], double-long-unsigned [6], octet-string [9], visible-string [10], 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] }	Oldest or smallest entry to retrieve
	to_value	CHOICE { see above }	Newest or largest entry to retrieve
	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 <i>capture_object_definition</i> is specified above (<i>capture_objects</i>)

}

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 attributes 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 D.2.1.13.

Setting capture_period to 1 ensures that the values are updated every second.

5.7 Clock (class_id: 8)

An instance of the “Clock” interface class handles all information that is related to date and time, including leap years and the deviation of the local time to a generalized time reference (Greenwich Mean Time, GMT). The deviation from the local time to the generalized time reference can change depending on the season (e.g. summertime vs. wintertime). The interface to an external client is based on date information specified in day, month and year, time information given in hundredths of seconds, seconds, minutes and hours and the deviation from the local time to the generalized time reference.