INTERNATIONAL STANDARD

IEC 62056-62

First edition
2002-02

**Electricity metering –
Data exchange for meter reading, tariff and load control –**

**Part 62:
Interface classes**

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

Reference number
IEC 62056-62:2002(E)

Publication numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series. For example, IEC 34-1 is now referred to as IEC 60034-1.

Consolidated editions

The IEC is now publishing consolidated versions of its publications. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Further information on IEC publications

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology. Information relating to this publication, including its validity, is available in the IEC Catalogue of publications (see below) in addition to new editions, amendments and corrigenda. Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is also available from the following:

- **IEC Web Site (www.iec.ch)**

- **Catalogue of IEC publications**

  The on-line catalogue on the IEC web site (www.iec.ch/catlg-e.htm) enables you to search by a variety of criteria including text searches, technical committees and date of publication. On-line information is also available on recently issued publications, withdrawn and replaced publications, as well as corrigenda.

- **IEC Just Published**

  This summary of recently issued publications (www.iec.ch/JP.htm) is also available by email. Please contact the Customer Service Centre (see below) for further information.

- **Customer Service Centre**

  If you have any questions regarding this publication or need further assistance, please contact the Customer Service Centre:

  Email: custserv@iec.ch
  Tel: +41 22 919 02 11
  Fax: +41 22 919 03 00

IECNORM.COM : Click to view the full PDF of IEC 62056-46:2002
Withdrawn

INTERNATIONAL STANDARD

IEC 62056-62

First edition
2002-02

Electricity metering – Data exchange for meter reading, tariff and load control – Part 62: Interface classes

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002

© IEC 2002 — Copyright - all rights reserved

No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

International Electrotechnical Commission 3, rue de Varembé Geneva, Switzerland
Telefax: +41 22 919 0300 e-mail: inmail@iec.ch IEC web site http://www.iec.ch

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE XC

For price, see current catalogue

CONTENTS

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

INTERNATIONAL ELECTROTECHNICAL COMMISSION

ELECTRICITY METERING – DATA EXCHANGE FOR METER READING, TARIFF AND LOAD CONTROL –

Part 62: Interface classes

FOREWORD

1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the 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, the IEC publishes International Standards. 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. The 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 the 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 National Committees.
3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical specifications, technical reports or guides and they are accepted by the National Committees in that sense.
4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.
5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.

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 62056-62 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 with applicants throughout the world. In this respect, the statement of the provider of the maintenance service is registered with the IEC. Information (see also chapter 4.6.2 and Annex E) may be obtained from:

DLMS[1] User Association
Geneva / Switzerland
www.dlms.ch

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

International Standard IEC 62056-62 has been prepared by IEC technical committee 13: Equipment for electrical energy measurement and load control.

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

FDIS	Report on voting
13/1270/FDIS	13/1276/RVD

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

[1] Device Language Message Specification.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 3.

Annexes A, B, C and D form an integral part of this standard.

Annex E is for information only.

The committee has decided that the contents of this publication will remain unchanged until 2006. At this date, the publication will be

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

A bilingual version of this publication may be issued at a later date.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002

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, of the customers and of the 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.

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

Withdrawn

ELECTRICITY METERING – DATA EXCHANGE FOR METER READING, TARIFF AND LOAD CONTROL –

Part 62: 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 normative documents contain provisions which, through reference in this text, constitute provisions of this part of IEC 62056. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of IEC 62056 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of IEC and ISO maintain registers of currently valid International Standards.

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 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 62056-21, *Data exchange for meter reading, tariff and load control – Part 21: Direct local data exchange* [2]

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 62056-46:2001, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 46: Data link layer using HDLC-protocol*

IEC 62056-53:2001, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 53: COSEM Application layer*

IEC 62056-61:2001, *Electricity metering – Data exchange for meter reading, tariff and load control – Part 61: OBIS Object identification system*

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

[2] To be published.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002

3 Terms, definitions and abbreviations

3.1 Terms and definitions

For the purpose of this part of IEC 62056 the terms and definitions given in IEC 60050-300 and IEC 62051, as well as the following apply.

3.1.1
base_name
the short_name corresponding to the first attribute (“logical_name”) of a COSEM object

3.1.2
class_id
class identification code

3.1.3
COSEM object
an instance of an interface class

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
COSEM	Companion Specification for Energy Metering
DLMS	Distribution Line Message Specification
GMT	Greenwich Mean Time
HLS	High-level Security
IC	Interface Class
LLS	Low Level Security
LN	Logical Name
LSB	Least Significant Bit
M	Mandatory
MSB	Most Significant Bit
O	Optional
OBIS	OBject Identification System
PDU	Protocol Data Unit
SAP	Service Access Point
SN	Short Name

Withdrawn IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

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. Meters, support tools and other system components that follow these specifications can communicate with each other 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 offers 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:

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

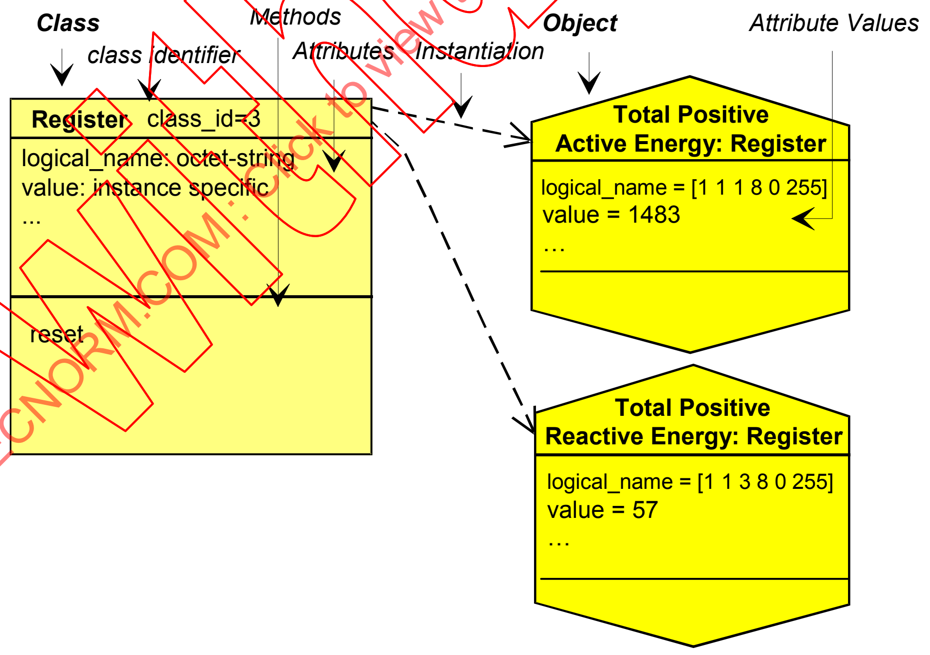


Figure 1 – An interface class and its instances

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 2. ..... 3.	(static) (..) (..)	octet-string			
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, ...)	
Cardinality	Specifies the number of instances of the class within a logical device (see 4.5).	
	value	The class shall be instantiated exactly “value” times.
	min...max.	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 can be obtained from an “association” 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 can be obtained from an “association” object. **Within one logical device all instances of a certain class must be of the same version.**	

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

Attribute(s)	Specifies the attribute(s) that belong to the class.	
	(*dyn.*)	Classifies an attribute that carries a process value, which is updated by the meter itself.
	(static)	Classifies an attribute which is not updated by the meter itself (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 minimal 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	
	Method Name ()	The method has to be described in the subsection "Method description".
m / o	Defines if the method is mandatory or optional.	
	m (mandatory)	The method is mandatory.
	o (optional)	The method is optional.

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

Attribute description

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

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 common methods READ/WRITE and GET/SET typically reference the entire attribute addressed. However, for certain attributes selective access to just part of the attribute may be provided. The part of the attribute is identified by specific selective access parameters. These selective access parameters are defined as part of the attribute specification.

4.3 Common data types

The following list contains some data types common to all interface classes.

Simple data types	**Data types carrying one data item only**
integer, long, double-long, unsigned, long-unsigned, double-long-unsigned, boolean	Simple data types as defined in IEC 61334-4-41, clause A.12, Data Examples: integer Integer8 1 byte long Integer16 2 bytes double-long Integer32 4 bytes
enum	The elements of the enumeration type need to be defined in the subsection “Attribute description”. Any not listed value for an enumeration is reserved by default.
real32, real64	Real data types according to the REAL specification of IEC 61334-4-41.
visible-string, octet-string	An ordered sequence of ASCII-characters respectively octets (8-bit bytes).
bit-string	An ordered sequence of boolean values.
Complex data types	**More than one data item is included, or the data item itself is not simple.**
array	The array elements need to be defined in the subsection “Attribute description”.
compact array	The array elements need to be defined in the subsection “Attribute description”.
structure	The structure type needs to be defined in the subsection “Attribute description”.
instance specific	The data type of the attribute needs to be specified in the instantiation of the object for a particular meter (instance model).

4.4 Data formats for date and time notation

Date and time notations are normally using octet-string as data type, but the formatting of the data is defined precisely.

date octet-string{ year highbyte, year lowbyte, month, day of month, day of week }

year: interpreted as unsigned16
range 0..big
0xFFFF = not specified
year highbyte and year lowbyte reference the 2 bytes of the unsigned 16

month: interpreted as unsigned8
range 1..12, 0xFD,0xFE,0xFF
1 is January
0xFD= daylight_savings_end
0xFE= daylight_savings_begin
0xFF = not specified

dayOfMonth: interpreted as unsigned8
range 1..31, 0xFD, 0xFE, 0xFF
0xFD = 2nd last day of month
0xFE = last day of month
0xE0 to 0xFC = reserved
0xFF = not specified

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

	dayOfWeek: interpreted as unsigned8 range 1..7, 0xFF 1 is Monday 0xFF = not specified For repetitive dates the unused parts must be set to "not specified".
time	octet-string {hour, minute, second, hundredths} hour: interpreted as unsigned8 range 0..23, 0xFF 0xFF = not specified minute:interpreted as unsigned8 range 0..59, 0xFF 0xFF = not specified second: interpreted as unsigned8 range 0..59, 0xFF 0xFF = not specified hundredths: interpreted as unsigned8 range 0..99, 0xFF 0xFF = not specified} For repetitive times the unused parts must be set to "not specified".
deviation	Integer16 –720..720: in minutes of local time to GMT 0x8000 = not specified
clock_status	Unsigned8 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
	{
	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*.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002

[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.

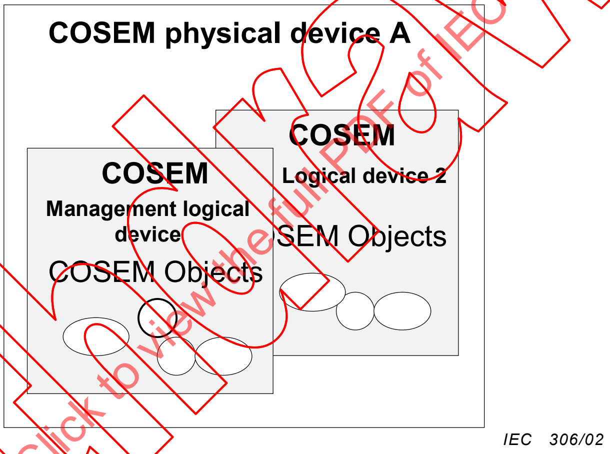
[d] This bit indicates that at least one bit of the clock status is invalid. Some bits may be correct. The exact meaning shall be explained in the manufacturer's documentation.

[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).

4.5 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



IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

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.

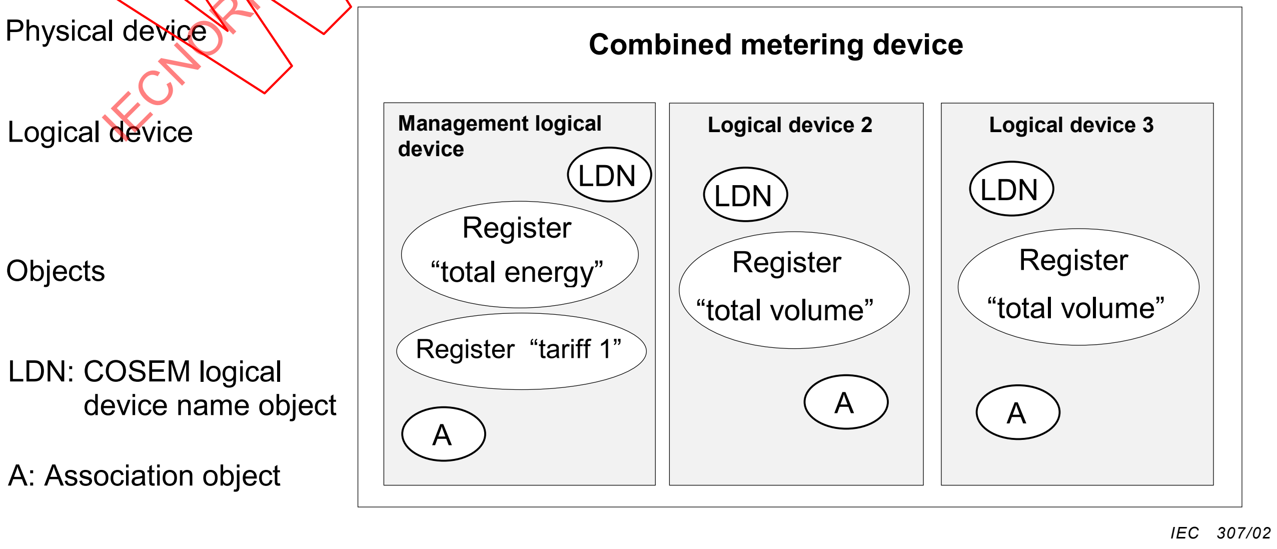


Figure 3 – Combined metering device

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.1.1.17).

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

- information on the application context;
- information on the COSEM context;
- information on the authentication mechanism used;
- etc.

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 – which 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

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002

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.

Pass3: The client processes StoC in a secret way (e.g. encrypting with a secret key). 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 – the server accepts the authentication of the client.

Pass4: If the client is authenticated, the server processes CtoS in a secret way (e.g. encrypting with a secret key). 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 – the client 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.

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

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”.

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

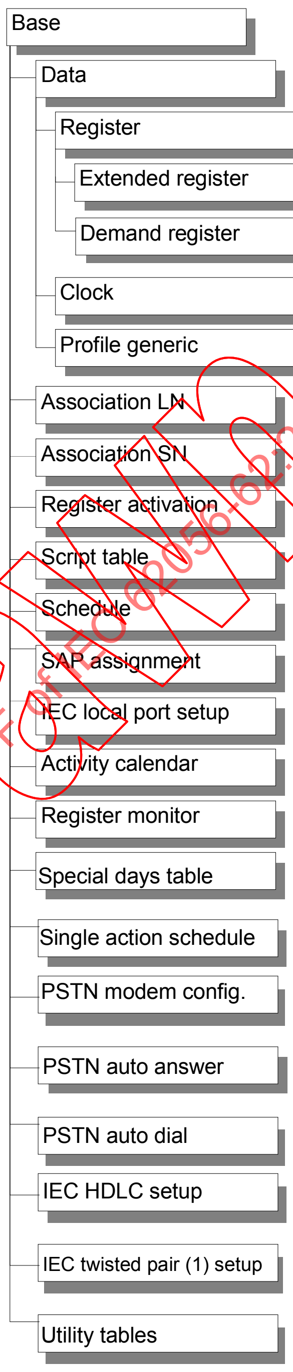


Figure 4 – Overview of the interface classes

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 instance specific. 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 *instance specific*			
Specific method(s)	***m/o***			

Attribute description

logical_name	Identifies the data contained in value. Identifiers are specified in clause D.1 and in IEC 62056-61.
value	Contains the data. *instance specific* — The data type of the value depends on the instantiation defined by "logical_name".

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.1 and in 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 *instance specific* scal_unit_type			
Specific method(s)	***m/o***			
1. reset	o			

Attribute description

value	Contains the current process or status value.	
	instance specific	The data type of the value depends on the instantiation defined by "logical_name" and possibly from the manufacturer. Therefore, this attribute must provide the value and the data type when it is accessed by a client. The type has to be chosen so that, together with the logical_name, an unambiguous interpretation of the value is possible.
scaler_unit	Provides information on the unit and the scaler of the value. If the value uses a complex data type, the scaler and unit apply to all elements.	
	scal_unit_type: structure { scaler, unit }	
	scaler: integer	This is the exponent (to the base of 10) of the multiplication factor REMARK If the value is not numerical then the scaler shall be set to 0.
	unit: enum	Enumeration defining the physical unit; for details see below.

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

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*	*SI definition*
(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	minute	60 s
(7)	s	//	time (*t*)	second	s
(8)	°	//	(phase) angle	degree	rad*180/π
(9)	°C	//	temperature (Θ)	degree centigrade	K-273.15
(10)	currency	//	(local) currency		
(11)	m	//	length (*l*)	meter	m
(12)	m/s	//	speed (*v*)		m/s
(13)	m^3	//	volume (*V*)		m^3
(14)	m^3	//	corrected volume		m^3
(15)	m^3/h	//	volume flux		$m^3/(60*60s)$
(16)	m^3/h	//	corrected volume flux		$m^3/(60*60s)$
(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		10^{-3} m^3
(20)	kg	//	mass (*m*)	kilogram	kg
(21)	N	//	force (*F*)	newton	N
(22)	Nm	//	energy	newtonmeter	J = Nm = Ws
(23)	Pa	//	pressure (*p*)	pascal	N/m^2
(24)	bar	//	pressure (*p*)	bar	10^{-5} N/m^2
(25)	J	//	energy	joule	J = Nm = Ws
(26)	J/h	//	thermal power		J/(60*60s)
(27)	W	//	active power (*P*)	watt	W = J/s
(28)	VA	//	apparent power (*S*)		
(29)	var	//	reactive power (*Q*)		
(30)	Wh	//	active energy		W*(60*60s)
(31)	VAh	//	apparent energy		VA*(60*60s)
(32)	varh	//	reactive energy		var*(60*60s)
(33)	A	//	current (*I*)	ampere	A
(34)	C	//	electrical charge (Q)	coulomb	C = As
(35)	V	//	voltage (*U*)	volt	V
(36)	V/m	//	electrical field strength (*E*)		V/m
(37)	F	//	capacity (*C*)	farad	C/V = As/V
(38)	Ω	//	resistance (*R*)	ohm	Ω = V/A
(39)	$\Omega m^2/m$	//	resistivity (ρ)		Ωm
(40)	Wb	//	magnetic flux (Φ)	weber	Wb = Vs
(41)	T	//	induction (*T*)	tesla	Wb/m^2
(42)	A/m	//	magnetic field strength (*H*)		A/m
(43)	H	//	inductivity (*L*)	henry	H = Wb/A
(44)	Hz	//	frequency (*f*, ω)	hertz	1/s
(45)	R_{ac}	//	active energy meter constant		1/(Wh)
(46)	R_{re}	//	reactive energy meter constant		
(47)	R_{ap}	//	apparent energy meter constant		
(48)	V^2h	//	voltage square hour		$V^2(60*60s)$

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

Code	*// Unit*		*Quantity*	*Unit*	*SI definition*
(49)	A^2h	//	ampere square hour		$A^2(60*60s)$
(50)	kg/s	//	mass flux		kg/s
(51)	S mho	//	conductance	siemens	1/Ω
(52)		//	reserved		
...		//	...		
(253)		//	reserved		
(254)	other	//	other unit		
(255)	count	//	no unit, unitless, count		1

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.)	*instance specific*			
3. scaler_unit	(static)	scal_unit_type			
4. status	(dyn.)	*instance specific*			
5. capture_time	(dyn.)	octet-string			
Specific method(s)		***m/o***			
1. reset		o			

Attribute description

For the definition of the attributes logical_name ... scaler_unit, see description of class "register".

Status	Provides an extended register specific status information. The data type and encoding of the status shall be provided for each instance of an extended register. *Instance specific* 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 for date_time.

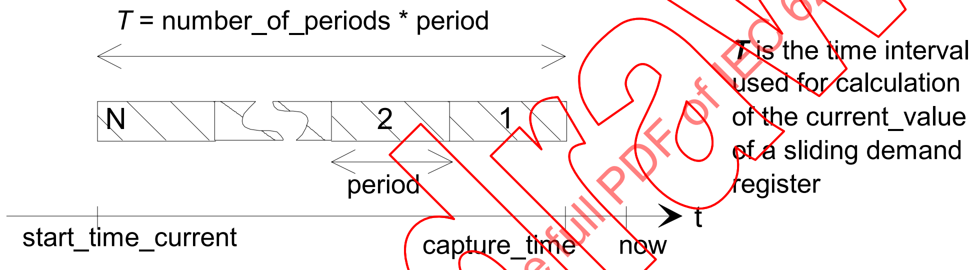
IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002

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 status is set so that it shows that a reset method has been invoked. 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”.



T = number_of_periods * period
N
2
1
period
T is the time interval used for calculation of the current_value of a sliding demand register
t
start_time_current
capture_time
now

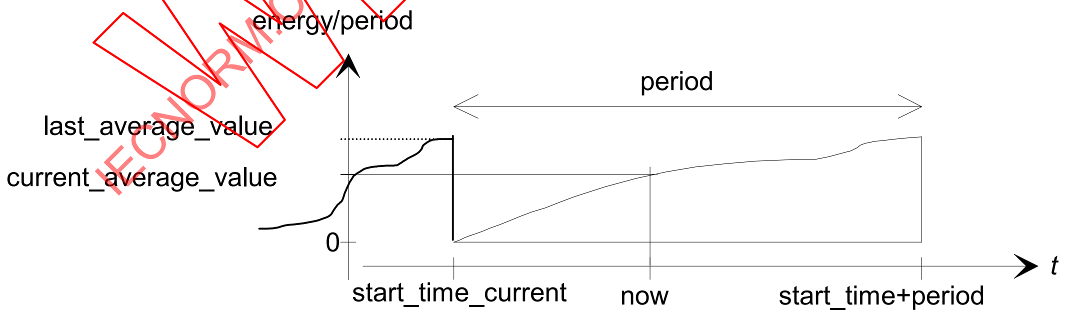
IEC 309/02

Figure 5 – The attributes when measuring sliding demand

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

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.



IEC 310/02

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

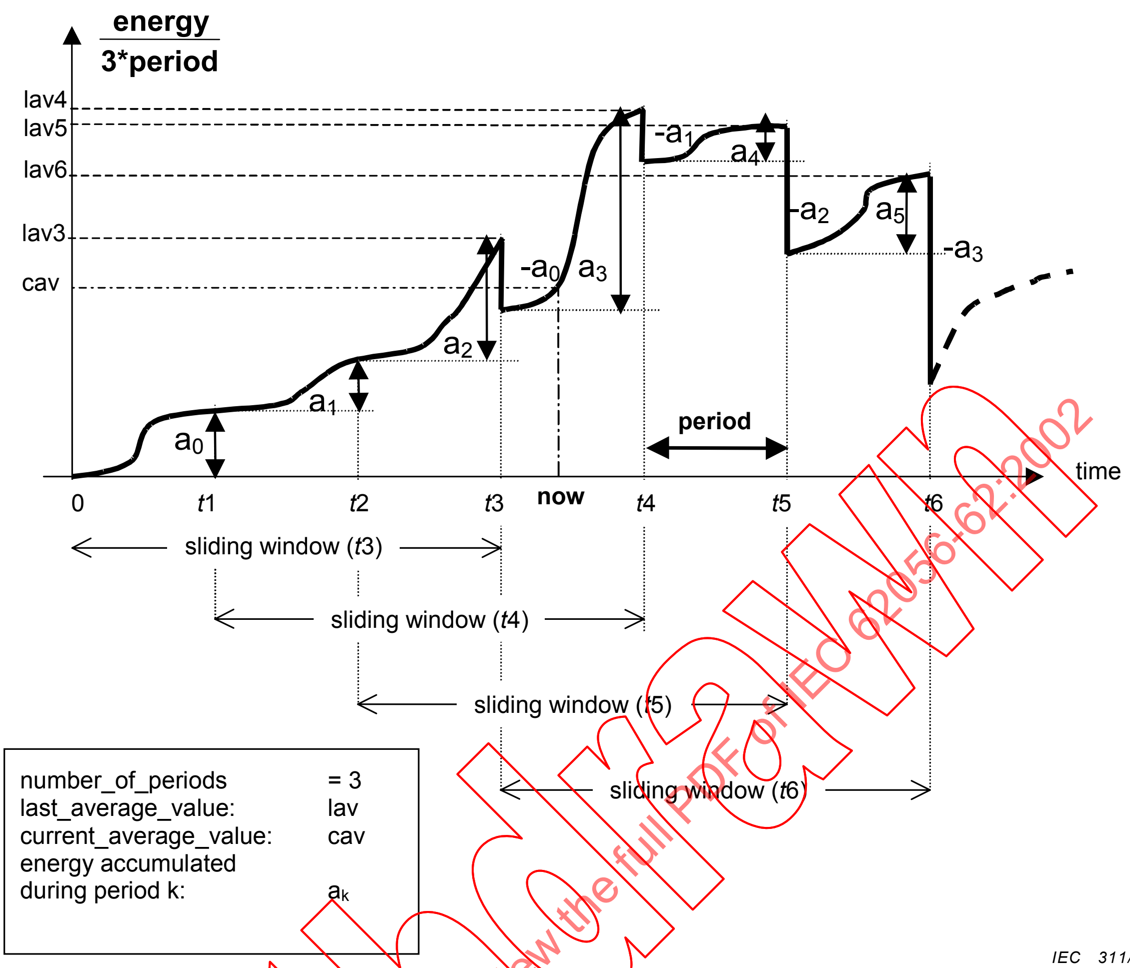


Figure 7 – The attributes if the number of periods is 3

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.)	*instance specific*			0
3. last_average_value	(dyn.)	*instance specific*			0
4. scaler_unit	(static)	scal_unit_type			
5. status	(dyn.)	*instance specific*			
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		o			
2. next_period		o			

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

Attribute description

For the attributes logical_name, scaler_unit, see description of class “register”.

status	Provides demand register associated status information. The type and encoding of the status shall be provided for each instance of a demand register. *instance specific* Def Depending on the status type definition.
current_average_value	Provides the current value (running demand) of the energy accumulated since start_time divided by number_of_periods*period. Only simple data types shall be used.
last_average_value	Provides the value of the accumulated energy (over the last number_of_periods*period) divided by number_of_periods*period. The energy of the current (not terminated) period is not considered by the calculation. Only simple data types shall be used.
capture_time	Provides the date and time when the last_average_value has been calculated. octet-string, formatted as set in 4.4 for date_time.
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 for date_time.
period	Period is the interval between two successive updates of the last_average_value. (number_of_periods*period is the denominator for the calculation of the demand) double-long-unsigned Measuring period in seconds The behaviour of the meter after writing a new value to this attribute shall be specified by the manufacturer.
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.

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

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 anymore at the interface. data ::= integer(0)

5.5 Register activation (class_id: 6)

An instance of the register activation class is used to handle different tariffication structures. It specifies which register, extended register and demand register objects are enabled if a specific activation mask is active (active_mask). All other register objects defined in register_assignment not being part of the active_mask are disabled. All register objects not defined in any register_assignment are 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		o			
2. add_mask		o			
3. delete_mask		o			

Attribute description

logical_name	Contains an identifier for the group of masks (representing tariff/rates) which are defined in the COSEM object. Normally, this identifier is linked to the nature of the registers which are defined in the register_assignment (e.g. energy registers, demand registers, …).

register_assignment	Specifies an ordered list of COSEM objects that are assigned to this register activation object. The list can contain different kinds of COSEM objects (e.g. register, extended register and demand register objects, etc.) simultaneously.
	array object_definition
	object_definition ::= structure
	{
	class_id: long-unsigned;
	logical_name: octet-string (SIZE(6))
	}

mask_list	Specifies a list of register activation masks. Each entry (mask) is identified by its mask_name and contains an array of indices referring to the registers assigned to the mask (the first object in register_assignment is referenced by index 1, the second object by index 2, …,).
	Array register_act_mask
	register_act_mask ::= structure
	{
	mask_name: octet-string;
	index_list: index_array
	}
	index_array ::= array unsigned
	mask_name has to be uniquely defined within the object.

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.

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

Method description

add_register (data)	This method adds one more register to the attribute register_assignment. The new register is added at the end of the array; i.e. the new 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)	This method 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)	This method 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. A capture object is either a register, a clock or a profile. The capture objects are collected periodically or occasionally. A profile has a buffer to store the captured data. To retrieve 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.

Assigning the corresponding objects to the profile specifies the capture objects the values of which have to be retained (with method capture). The buffer has homogenous entries (all have the same size and structure), and the assignment is defined statically. The modification of the capture object assignment clears the buffer of the profile completely. All profiles capturing this modified profile will be cleared as well to guarantee the homogeneity of their 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:

a) the number of entries filled. This will be zero after clearing the profile;

b) 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;

c) the physical limit for the buffer. This limit typically depends on the objects to capture. The object will reject an attempt of setting the maximum number of entries that is larger than physically possible.

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

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		o			
2. capture		o			
3. reserved from previous versions		o			
4. reserved from previous versions		o			

Attribute description

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). The sequence of the values of the captured objects within the structure (see below) corresponds to the order defined in the attribute capture_objects. The sequence of the entries within the array (see below) is ordered according to the specified sort method. The buffer gets filled by subsequent calls of the method capture ().

compact-array or array entry
entry ::= structure
instance specific

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.

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

capture_objects	Specifies the list of capture objects (registers, clocks and profiles) that are assigned to this profile object. Upon a call of the capture () method, the specified attributes of these objects 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 first attribute (i.e. logical_name), attribute_index 2 to the 2^{nd}, 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 (1) fifo (first in first out), (2) lifo (last in first out), (3) largest, (4) smallest, (5) nearest_to_zero, (6) farest_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)

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

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 should 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	In this case, only buffer elements corresponding to the range_descriptor shall be returned in the response.
2	entry_descriptor	In this case, only buffer elements corresponding to the entry_descriptor shall be returned in the response.

range_descriptor ::= structure
{

restricting_object	capture_object_definition	Defines the register or clock restricting the range of entries to be retrieved
from_value	instance_specific	Oldest or smallest entry to retrieve
to_value	instance_specific	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 is 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.

}

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

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 <object_attribute> of each capture object. Depending on the sort_method and the actual state of the buffer this produces a new entry or a replacement for the less significant entry. As long as not all entries are already used, the entries_in_use attribute will be incremented. This call does not trigger any additional operations within the capture objects such as capture () or reset (). Note that only some attributes of the captured objects might be stored, not the complete object. data ::= integer(0)

Behaviour of the object after modification of certain attributes

	Any modification of one of the attributes describing the static structure of the buffer will automatically call a reset () and this call will propagate to all other profiles capturing this profile. If writing to profile_entries is attempted with a value too large for the buffer, it will be 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, the profile object can be used to define a set of preferred readout-values. In the “capture_objects” attributes, those objects and attributes are pre-defined which should be readable with one single command.

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.

It also handles the daylight savings function in that way; i.e. it modifies the deviation of local time to GMT depending on the attributes. The start and end point of that function is normally set once. An internal algorithm calculates the real switch point depending on these settings

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

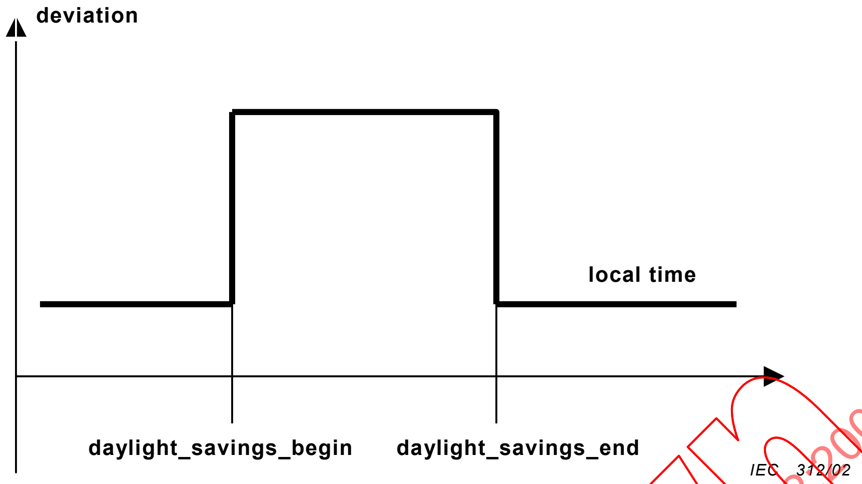


Figure 8 – The generalized time concept

Clock		0..1	class_id = 8, version = 0		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	(static)	octet-string			
2. time	(dyn.)	octet-string			
3. time_zone	(static)	long			
4. status	(dyn.)	unsigned8			
5. daylight_savings_begin	(static)	octet-string			
6. daylight_savings_end	(static)	octet-string			
7. daylight_savings_deviation	(static)	integer			
8. daylight_savings_enabled	(static)	boolean			
9. clock_base	(static)	enum			
Specific method(s)		***m/o***			
1. adjust_to_quarter		o			
2. adjust_to_measuring_period		o			
3. adjust_to_minute		o			
4. adjust_to_preset_time		o			
5. preset_adjusting_time		o			
6. shift_time		o			

Attribute description

time	Contains the meter’s local date and time, its deviation to GMT and the status. See also the description in 4.4. When this value is set, only specified fields of the date_time are changed. For example for setting the date without changing the time, all time relevant octets of the date_time shall be set to “not specified”. The clock_status shall always be set when writing the time. octet-string, formatted as set in 4.4 for date_time.
time_zone	The deviation of local, normal time to GMT in minutes. long
status	The status is equal to the status read in *time*. See also the description in 4.4. unsigned8, formatted as set in 4.4 for clock_status
daylight_savings_begin	Defines the local switch date and time when the local time has to be deviated from the normal time. For generic definitions wildcards are allowed. octet-string, formatted as set in 4.4 for date_time.

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

daylight_savings_end	See above. octet-string, formatted as set in 4.4 for date_time.
daylight_savings_deviation	Contains the number of minutes by which the deviation in generalized time must be corrected at daylight savings begin. integer Deviation range of up to ± 120 min
daylight_savings_enabled	TRUE enables daylight savings function. boolean
clock_base	Defines where the basic timing information comes from. enum (0) not defined (1) internal crystal (2) mains frequency 50 Hz (3) mains frequency 60 Hz (4) GPS (global positioning system) (5) radio controlled

Method description

adjust_to_quarter (data)	Sets the meter's time to the nearest (+/-) quarter of an hour value (*:00, *:15, *:30, *:45). data ::= integer (0).
adjust_to_measuring_period (data)	Sets the meter's time to the nearest (+/-) starting point of a measuring period. data ::= integer (0).
adjust_to_minute (data)	Sets the meter's time to the nearest minute. If second_counter < 30 s, so second_counter is set to 0. If second_counter ≥ 30 s, so second_counter is set to 0 and minute_counter and all depending clock values are incremented if necessary. data ::= integer(0)
adjust_to_preset_time (data)	This method is used in conjunction with the preset_adjusting_time method. If the meter's time lies between validity_interval_start and validity_interval_end, then time is set to preset_time. data ::= integer(0)

preset_adjusting_time (data)	Presets the time to a new value (preset_time) and defines a validity_interval within which the new time can be activated.
	
	data ::= structure
	{
	preset_time: octet-string;
	validity_interval_start: octet-string;
	validity_interval_end: octet-string
	}
	all octet-strings formatted as set in 4.4 for date_time.

shift_time (data)	Shifts the time by *n* (-900 <= *n* <= 900) s. data ::= long

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

5.8 Script table (class_id: 9)

The IC script table provides the possibility to trigger a series of actions by activating an execute method. For that purpose, script table contains a table of script entries. Each table entry (script) consists of a script_identifier and a series of action_specifications. An action_specification activates a method of a COSEM object or modifies attributes of a COSEM object within the logical device.

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

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

Script table		**0..*n***	**class_id = 9, version = 0**		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	(static)	octet-string			
2. scripts	(static)	array			
Specific method(s)		***m/o***			
1. execute		m			

Attribute description

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

	script	structure
		{
		script_identifier: long-unsigned
		actions: array action_specification
		}
		The script_identifier 0 is reserved. If specified with an execute method, it results in a null script (no actions to perform).

	action_specification	structure
		{
		service_id: enum
		class_id: long-unsigned
		logical_name: octet-string
		index: integer
		parameter: service specific
		}
		where
		service_id: defines which action shall be applied to the referenced object
		(1) write attribute
		(2) execute specific method
		index: defines (with service_id 1) which attribute of the selected object is affected or (with service_id 2) which specific method is to be executed.
		
		The first attribute (logical_name) has index 1, the first specific method has index 1 as well.

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

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

Method description

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

5.9 Schedule (class_id: 10)

The IC schedule together with an object of the IC special days table handles time and date driven activities within a device. The following picture gives an overview and shows the interactions between them:

Schedule:

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

Special days table:

Index	special_ day_date	day_id
12	24-12-xx	S1
33	25-12-xx	S3
77	31-03-97	S3

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

Schedule	**0..*n***	**class_id = 10, version = 0**		
Attribute(s)	***Data type***	***Min.***	***Max.***	***Def***
1. logical_name (static)	octet-string			
2. entries (static)	array			
Specific method(s)	***m/o***			
1. enable/disable	o			
2. insert	o			
3. delete	o			

Attribute description

entries	Specifies the scripts to be executed at given times. There is only one script that can be executed per entry.
	array schedule_table_entry
	schedule_table_entry structure
	{
	index: long-unsigned (1..9999)
	enable: boolean
	script_logical_name: octet-string
	script_selector long-unsigned
	switch_time: octet-string
	validity_window: long-unsigned
	exec_weekdays: bit-string
	exec_specdays: bit-string
	begin_date: octet-string
	end_date: octet-string
	}
	script_logical_name: defines the logical name of the script object
	script_selector: defines the script_identifier of the script to be executed.
	switch_time accepts wildcards to define repetitive entries. The format of the octet-string follows the rule set in 4.4 for time.
	validity_window defines a period in minutes, in which an entry must be processed after power fail. (time between defined switch_time and actual power_up)
	0xFFFF: the script must be processed any time.
	
	exec_specdays perform the link to the IC Special Days Table, day_ID.
	begin_date and end_date define the date period in which the entry is valid (wildcards are allowed). The format follows the rule set in 4.4 for date.

Method description

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

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

	firstIndexA/B < lastIndexA/B : all entries of the range A/B are disabled/enabled firstIndexA/B == lastIndexA/B : one entry is disabled/enabled firstIndexA/B > lastIndexA/B : nothing disabled/enabled firstIndexA/B and lastIndexA/B > 9999: No entry is disabled/enabled
insert (data)	Inserts a new entry in the table. If the index of the entry exists already the existing entry is overwritten by the new entry. entry schedule_table_entry data corresponding to entry

delete (data)	Deletes a range of entries in the table.
	data ::= structure
	{
	firstIndex;
	lastIndex
	}
	firstIndex First index of the range that is deleted. long-unsigned
	lastIndex Last index of the range that is deleted. long-unsigned
	firstIndex < lastIndex: all entries of the range A/B are deleted
	firstIndex ::= lastIndex : one entry is deleted
	firstIndex > lastIndex :nothing deleted

Remarks concerning “inconsistencies” in the table entries

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

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

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002

Recovery after power failure

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

Handling of time changes

There are four different "actions" of time changes:

a) time setting forward;

b) time setting backwards;

c) time synchronization;

d) daylight saving action.

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

Time setting forward*

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

* Writing to the attribute “time” of the “clock” object

Time setting backward*

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

* Writing to the attribute “time” of the “clock” object

Time synchronization*

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

* Using the method “Adjust_to_quarter” of the “clock” object

Daylight saving

If the clock is put forward, then all scripts which fall into the forwarding interval (and would therefore get lost) are executed.

If the clock is put back, re-execution of the scripts which fall into the backwarding interval is suppressed.

5.10 Special days table (class_id: 11)

The interface class allows defining dates, which will override normal switching behaviour for special days. The interface class works in conjunction with the class "schedule" or "activity calendar" and the linking data item is day_id.

Special days table		**0..1**	**class_id = 11, version = 0**		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	(static)	octet-string			
2. entries	(static)	array			
Specific method(s)		***m/o***			
1. insert		o			
2. delete		o			

Attribute description

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

	spec_day_entry	structure
		{
		index: long-unsigned
		specialday_date: octet-string
		day_id: unsigned
		}
		specialday_date formatting follows the rule set in 4.4. for date.The range of the day_id must match the length of the bitstring exec_specdays in the related object of inteface class “schedule”.

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

Method description

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

5.11 Activity calendar (class_id: 20)

An instance of the activity calendar class is typically used to handle different tariff structures. It is a definition of scheduled actions inside the meter, which follow the classical way of calendar based schedules by defining seasons, weeks… It can coexist with the more general object schedule and can even overlap with it. If actions are scheduled for the same activation time in an object schedule and in the object activity calendar, the actions triggered by schedule are executed first.

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

The activity calendar defines the activation of certain scripts, which can perform different activities inside the logical device. The interface to the object script is the same as for the object schedule (see 5.9).

If an instance of the interface class "special days table" (see 5.10) is available, relevant entries there take precedence over the activity calendar object driven selection of a day profile. The day profile referenced in the "special days table" activates the day_schedule of the day_profile_table in the "activity calendar" object by referencing through the day_id.

Activity calendar		**0..1**	**class_id = 20, version = 0**		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	(static)	octet-string			
2. calendar_name_active	(static)	octet-string			
3. season_profile_active	(static)	array			
4. week_profile_table_active	(static)	array			
5. day_profile_table_active	(static)	array			
6. calendar_name_passive	(static)	octet-string			
7. season_profile_passive	(static)	array			
8. week_profile_table_passive	(static)	array			
9. day_profile_table_passive	(static)	array			
10. activate_passive_calendar_time	(static)	octet-string			
Specific method(s)		***m/o***			
1. activate_passive_calendar		o			

Attribute description

Attributes called …_active are currently active, attributes called …_passive will be activated by the specific method activate_passive_calendar

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002 Withdrawn

calendar_name	Typically contains an identifier, which is descriptive to the set of scripts which are activated by the object.
season_profile	Contains a list defining the starting date of a season. This list is sorted according to season_start. Each season activates a specific week_profile. array season season ::= structure { season_profile_name: octet-string season_start: octet-string week_name: octet-string }. where season_profile_name is a user defined name identifying the current season_profile; season_start defines the starting time of the season, formatted as set in 4.4 for date_time. REMARK The current season is terminated by the season_start of the next season. week_name defines the week_profile active in this season.

week_profile_table	Contains an array holding the correlation between day_profiles for every day of the week to be used in different seasons.
	array week_profile
	week_profile ::= structure
	{
	week_profile_name: octet-string;
	monday: day_id;
	tuesday: day_id;
	wednesday: day_id;
	thursday: day_id;
	friday: day_id;
	saturday: day_id;
	sunday: day_id
	}
	day_id: unsigned
	where
	week_profile_name is a user defined name identifying the current week_profile;
	Monday defines the day_profile valid on Monday;
	…
	Sunday defines the day_profile valid on Sunday.

day_profile_table	Contains a list of scripts and the corresponding activation times. Within each day_profile the list is sorted according to start_time. array day_profile

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

	day_profile ::= structure
	{
	day_id: unsigned;
	day_schedule: array (day_profile_action)
	}
	
	day_profile_action ::= structure
	{
	start_time: octet-string;
	script_logical_name: octet-string;
	script_selector: long-unsigned
	}
	
	where day_id is a user defined identifier, identifying the current day_profile.
	
	start_time: defines the time when the script is to be executed (no wildcards); the format follows the rule set in 4.4 for time.
	
	script_logical_name: defines the logical name of the script object.
	
	script_selector: defines the script_identifier of the script to be executed.

activate_passive_ calendar_time	Defines the time when the object itself calls the specific method activate_passive_calendar. A definition with "not specified" notation in all fields of the attribute will deactivate this automatism. Partial "not specified" notation in just some fields of date and time are not allowed. octet-string, formatted as set in 4.4 for date_time.

Method description

activate_passive_ calendar(data)	This method copies all attributes called …_passive to the corresponding attributes called …_active data ::= integer(0)

5.12 Association LN (class_id: 15)

COSEM logical devices able to establish application associations within a COSEM context using logical name references, model the associations through instances of the **association LN** class. A COSEM logical device has one instance of this IC for each association the device is able to support.

Association LN		**0..MaxNbofAss.**	**class_id = 15, version = 0**		
Attribute(s)		***Data type***	***Min.***	***Max .***	***Def***
1. logical_name	(static)	octet-string			
2. object_list	(static)	object_list_type			
3. associated_partners_id		associated_partners_type			
4. application_context_name		application-context-name			
5. xDLMS_context_info		xDLMS-context-type			
6. authentication_mechanism _name		mechanism-name			
7. LLS_secret		octet-string			
8. association status		enum			

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

Specific method(s)	*m/o*
1. reply_to_HLS_authentication	o
2. change_HLS_secret	o
3. add_object	o
4. remove_object	o

Attribute description

logical_name	Identifies the association LN object instance. The value of this attribute is indicated to the client application during the application association establishment.

object_list	Contains the list and the access right descriptor of all COSEM object instances which are 'visible' – can be accessed – within the given association.
	object_list_type ::= array object_list_element
	
	object_list_element ::= structure
	{
	class_id: long-unsigned;
	version: unsigned;
	logical_name: octet-string;
	access_rights: access_right;
	}
	
	access_right ::= structure
	{
	attribute_access: attribute_access_descriptor;
	method_access: method_access_descriptor;
	}
	
	attribute_access_descriptor ::= array attribute_access_item
	
	attribute_access_item ::= structure
	{
	attribute_id : integer;
	access_mode : enum
	{
	no_access (0);
	read_only (1) ;
	write_only (2);
	read_and_write (3);
	};
	access_selectors ::= CHOICE
	{
	null-data
	array of integer8
	};
	}
	
	method_access_descriptor ::= array method_access_item
	
	method_access_item ::= structure
	{
	method_id : integer;
	access_mode : boolean
	}
	
	The attribute_access_descriptor and the method_access_descriptor always contain all implemented attributes or methods.

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002

	access_selectors contain a list of the supported selector values. **selective access** (see 4.2) to the attribute object_list may be available (optional). The selective access parameters are as defined below.

associated_partners_id	Contains the identifiers of the COSEM client and the COSEM server which are associated. This attribute contains the address of the client and server application processes.
	
	Associated_partners_id ::= structure
	{
	client_SAP: integer;
	server_SAP: long-unsigned;
	}

application_context_name	In the COSEM environment, it is intended that an application context pre-exists and is referenced by its name during the establishment of an application association. This attribute contains the name of the application context for that association. application-context-name ::= OBJECT IDENTIFIER; • See IEC 62056-53

xDLMS_context_info	Contains all the necessary information on the xDLMS context for the given association
	xDLMS-context-type ::= structure
	{
	conformance : bitstring(24) ;
	max_receive_pdu_size : long-unsigned ;
	max_send_pdu_size : long-unsigned ;
	dlms_version_number : unsigned ;
	quality_of_service : integer;
	cyphering_info : octet-string
	}
	The conformance element contains the xDLMS conformance block supported by the server.
	
	The max_receive_pdu_size element contains the maximum length for a xDLMS PDU, expressed in bytes, that the client may send. This is the same as the server-max-receive-pdu-size parameter of the DLMS-Initiate.response pdu (see IEC 62056-53).
	
	The max_send_pdu_size, in an active association contains the maximum length for a xDLMS PDU, expressed in bytes, that the server may send. This is the same as the client-max-receive-pdu-size parameter of the DLMS-Initiate.request pdu (see IEC 62056-53).
	
	The dlms_version_number element contains the DLMS version number supported by the server.
	
	The quality_of _service element is not used.
	
	The cyphering_info, in an active association, contains the dedicated key parameter of the DLMS-Initiate.request pdu (see IEC 62056-53).

authentication_mechanism_name	This attribute contains the name of the authentication mechanism for that association. mechanism-name ::= OBJECT IDENTIFIER; See IEC 62056-53

Withdrawn
IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002

	No mechanism-name is required when no authentication is used.
LLS_secret	This attribute contains the authentication value for the LLS authentication process.

association_status	This attribute indicates the current status of the association, which is modelled by that object.
	Association-status : enum
	{
	(0) non-associated;
	(1) association-pending;
	(2) associated;
	}

A SET operation on an attribute of an association LN object becomes effective when this association object is used to establish a new association.

Parameters for selective access to the object_list attribute

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

Access selector value	Service parameters	Comment
1	NULL	All information excluding the access_rights shall be included in the response.
2	class_list	Access by class. In this case, only those object_list_elements of the object_list shall be included in the response, which have a class_id equal to one of the class_id's of the class-list. No access_right information is included. class_list ::= array class_id class_id ::= long-unsigned
3	object_Id_list	Access by object. The full information record of object instances on the object_Id_list shall be returned. object_Id_list ::= array object_Id

4	object_Id	In this case, the full information record of the required COSEM object instance shall be returned.
		object_Id ::= structure
		{
		class_id: long-unsigned;
		logical_name: octet-string
		}

Method description

reply_to_HLS_ authentication (data)	The remote invocation of this method delivers the client's "secretly" processed "challenge StoC" (f(StoC)) back to the server as the *data* service parameter of the invoked ACTION.request service. (see 4.7.2) data ::= octet-string client's response to the challenge If the authentication is accepted, then the response

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

	(ACTION.confirm) contains Result==OK and the server's "secretly" processed "challenge CtoS" (f(CtoS)) back to the client in the *data* service parameter of the response service. data ::= octet-string server's response to the challenge If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back.
change_HLS_secret (data)	Changes the HLS secret (e.g. encryption key). data ::= octet-string[a] new HLS secret
Add_object(data)	Adds the referenced object to the object_list. data ::= object_list_element (see above)
Remove_object(data)	Removes the referenced object from the object_list. data ::= object_list_element (see above)

[a] The structure of the "new secret" depends on the security mechanism implemented. The "new secret" may contain additional checkbits and it may be encrypted.

5.13 Association SN (class_id: 12)

COSEM logical devices able to establish application associations within a COSEM context using short name references, model the associations through instances of the association SN class. A COSEM logical device has one instance of this IC for each association the device is able to support.

The **short_name** of the association SN object itself is fixed within the COSEM context. It is given in clause C.2 as 0x**FA00**

Association SN		**0...n**	**class_id = 12, version = 1**		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	(static)	octet-string			
2. object_list	(static)	objlist_type			
Specific method(s)		***m/o***			
1. reserved from previous versions		o			
2. reserved from previous versions		o			
3. read_by_logicalname		o			
4. get_attributes&methods		o			
5. change_LLS_secret		o			
6. change_HLS_secret		o			
7. reserved from previous versions					
8. reply_to_HLS_authentication		o			

Attribute description

object_list	Contains the list of all objects with their base_names (short_name), class_id, version and logical_name. The base_name is the DLMS objectName of the first attribute (logical_name)
	objlist_type ::= array objlist_element
	objlist_element ::= structure
	{
	base_name: long;
	class_id: long-unsigned;
	version: unsigned;
	logical_name: octet-string
	}

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002 Withdrawn

	selective access (see 4.2) to the attribute object_list may be available (optional). The selective access parameters are as defined below.

Parameters for selective access to the object_list attribute

Access selector value	Parameter	Comment
1	class_id: long-unsigned	Delivers the subset of the object_list for a specific class_id. For the response: data ::= objlist_type

2	structure
	{
	class_id: long-unsigned;
	logical_name: octet-string
	}
		Delivers the entry of the object_list for a specific class_id and logical_name.
		For the response: data ::= objlist_element

Method description

read_by_logicalname (data)	Reads attributes for selected objects. The objects are specified by their class_id and their logical_name
	data ::= array attribute_identification
	attribute_identification ::= structure
	{
	class_id: long-unsigned;
	logical_name: octet-string;
	attribute_index: integer
	}
	where attribute_index is a pointer (i. e. offset) to the attribute within the object.
	
	attribute_index 0 delivers all attributes[a], attribute_index 1 delivers the first attribute (i. e. logical_name), etc.).
	
	For the response: data is according to the type of the attribute.

get_attributes& methods(data)	Delivers information about the access rights to the attributes and methods within the actual association. The objects are specified by their class_id and their logical_name.
	
	data ::= array object_identification
	
	object_identification ::= structure
	{
	class_id: long-unsigned;
	logical_name: octet-string
	}
	
	For the response
	data ::= array access_description
	
	access_description ::= structure
	{
	read_attributes: bit-string;
	write_attributes: bitstring;
	methods: bit-string
	}
	The position in the bitstring identifies the attribute/method (first position ↔ first attribute, first position ↔ first method) and the value of the bit specifies whether the attribute/method is available (bit set) or not available (bit clear).

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

change_LLS_secret (data)	Changes the LLS secret (e.g. password). data ::=octet-string new LLS secret
change_HLS_secret (data)	Changes the HLS secret (e.g. encryption key). data ::=octet-string[b] new HLS secret
reply_to_HLS_ authentication (data)	The remote invocation of this method delivers the client's "secretly" processed "challenge StoC" (f(StoC)) back to the server as the *data* service parameter of the invoked WRITE.request service. (see 4.7.2) data ::=octet-string client's response to the challenge If the authentication is accepted, then the response (WRITE.confirm) contains Result==OK and the server's "secretly" processed "challenge CtoS" (f(CtoS)) back to the client in the *data* service parameter of the response service. data ::= octet-string server's response to the challenge If the authentication is not accepted, then the result parameter in the response shall contain a non-OK value, and no data shall be sent back.

[a] If at least one attribute has no read access right under the current association, then a read_by_logicalname() to attribute index 0 reveals the error message "scope of access violation" (see IEC 61334-4-41, p. 221).

[b] The structure of the "new secret" depends on the security mechanism implemented. The "new secret" may contain additional checkbits and it may be encrypted.

5.14 SAP assignment (class_id: 17)

The interface class SAP assignment contains the information on the assignment of the logical devices to their SAPs (see IEC 62056-46).

SAP Assignment		**0..1**	**class_id = 17, version = 0**		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	(static)	octet-string			
2. SAP_assignment_list	(static)	asslist_type			0
Specific method(s)		***m/o***			
1. connect_logical_device		o			

Attribute description

SAP_assignment_list	Contains the list of all logical devices and their SAP addresses within the physical device.
	
	asslist_type ::= array asslist_element
	asslist_element ::= structure
	{
	SAP: *instance specific;* (see remark below)
	
	logical_device_name: octet-string
	}
	
	REMARK The actual addressing is performed by the "lower" communication layers. For a three-layer connection oriented architecture more details can be found in IEC 62056-46.

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

Method description

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

5.15 Register monitor (class_id: 21)

This interface class allows to define a set of scripts (see 5.8) that are executed when the value of an attribute of a monitored register type object (data, register, extended register, demand register, etc.) crosses a set of threshold values.

The IC register monitor requires an instantiation of the IC script table in the same logical device.

Register monitor		**0..*n***	**class_id = 21, version = 0**		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	(static)	octet-string			
2. thresholds	(static)	array			
3. monitored_value	(static)	value_definition			
4. actions	(static)	array			
Specific method(s)		***m/o***			

Attribute description

thresholds Provides the threshold values to which the attribute of the referenced register is compared.

array	threshold
threshold: *instance specific*	The threshold is of the same type as the monitored attribute of the referenced object.

monitored_value This defines which attribute of an object is to be monitored. Only values with simple data types are allowed

```
value_definition ::=    structure
{
        class_id:               long unsigned;
        logical_name:           octet-string;
        attribute_index:        integer
}
```

actions Defines the scripts to be executed when the monitored attribute of the referenced object crosses the corresponding threshold. The attribute "actions" has exactly the same number of elements as the attribute "thresholds". The ordering of the action_items corresponds to the ordering of the thresholds (see above).

```
array   action_set
action_set ::=  structure
{
        action_up:      action_item;
        action_down:    action_item
}
```

where

action_up defines the action when the attribute data of the monitored register crosses the threshold in the upwards direction and vice versa.

```
action_item ::= structure
{
        script_logical_name:    octet-string;
        script_selector:        long-unsigned
}
```

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002 WITHDRAWN

5.16 Utility tables (class_id: 26)

An instance of the utility tables class encapsulates ANSI C12.19 table data.

With this interface class definition, each "table" is represented as an instance. The specific instance is identified by its logical_name.

Utility tables	**0..*n***	**class_id = 26, version = 0**		
Attribute(s)	***Data type***	***Min.***	***Max.***	***Def***
1. logical_name (static)	octet-string			
2. table_ID (static)	long-unsigned			
3. length	double-long-unsigned			
4. buffer	octet-string			
Specific method(s)	***m / o***			

Attribute description

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 table. **Selective access** (see 4.2) to the attribute buffer 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 long-unsigned	Sequence of indices to identify elements within the table's hierarchy.
Count	long-unsigned	Number of elements requested or transferred. Values of count greater than 1 return up to that many elements. A value of zero, when given in the context of a request, refers to the entire subtree of the hierarchy starting at the selection point.

}

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

5.17 Single action schedule (class_id: 22)

Many applications request periodic actions within a meter. These actions are not necessarily linked to tariffication (activity calendar or schedule).

Single action schedule		**0..*n***	**class_id = 22, version = 0**		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	*(static)*	octet-string			
2. executed_script	*(static)*	script			
3. type	*(static)*	enum			
4. execution_time	*(static)*	array			
Specific method(s)	***m / o***				

Attribute description

executed_script	Contains the logical name of the script table and the script selector of the script to be executed.
	script Structure
	{
	Script_logical_name octet-string
	Script_selector long_unsigned
	}
	Logical_name and script_selector define the script to be executed.

type	enum (1) size of execution_time = 1; wildcard in date allowed (2) size of execution_time = *n*; all time values are the same; wildcards in date not allowed (3) size of execution_time = *n*; all time values are the same; wildcards in date are allowed (4) size of execution_time = *n*; time values may be different; wildcards in date not allowed (5) size of execution_time = *n*; time values may be different; wildcards in date are allowed

execution_time	Specifies the time of day the script is executed
	array {octet-string, octet-string}
	The two octet-strings contain time and date, in this order, structure {
	time: octet-string;
	date: octet-string
	}
	time and date are formatted as defined in 4.4.
	
	WILDCARDS in “time” are not allowed; seconds and hundredths of seconds must be zero

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

6 Maintenance of the interface classes

Any modification of interface classes as described below in 6.2 will be recorded by moving the old or obsolete version of an interface class into Annex E.

Versions of interface classes prior to the establishment of this standard are kept by the DLMS UA only.

6.1 New interface classes

The DLMS UA reserves the right to be the exclusive administrator of interface classes.

6.2 New versions of interface classes

Any modification of an existing interface class results in a new version (version ::= version+1) and shall be documented accordingly. The following rules shall be followed:

a) new attributes and methods may be added;

b) existing attributes and methods may be invalidated BUT the indices of the invalidated attributes and methods shall not be re-used by other attributes and methods;

c) if these rules cannot be met, then a new interface class shall be created;

d) new versions of COSEM interface classes are administered by the DLMS UA.

6.3 Removal of interface classes

Besides one association object and the logical device name object no instantiation of an interface class is mandatory within a meter. Therefore, even unused interface classes will not be removed from the standard. They will be kept to ensure compatibility with possibly existing implementations.

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

Annex A
(normative)

Protocol related interface classes

Each communication device and / or protocol needs some setup parameters to be defined for proper operation.

A.1 IEC local port setup (class_id: 19)

Instances of this interface class define the operational parameters for communication using IEC 62056-21. Several ports can be configured. The logical_names shall be as defined in D.1.1.11.

IEC local port setup		**0..*n***	**class_id = 19, version = 0**		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	(static)	octet-string			
2. default_mode	(static)	enum			
3. default_baud	(static)	enum			
4. prop_baud	(static)	enum			
5. response_time	(static)	enum			
6. device_addr	(static)	octet-string			
7. pass_p1	(static)	octet-string			
8. pass_p2	(static)	octet-string			
9. pass_w5	(static)	octet-string			
Specific method(s)		***m/o***			

Attribute description

default_mode	Defines the protocol used by the meter on the port		
	enum	(0) protocol according to IEC 62056-21 (modes A..E)	
		(1) protocol according to IEC 62056-46. Using this enumeration value all other attributes of this class are not applicable.	
default_baud	Defines the baud rate for the opening sequence		
	enum	(0)	300 baud
		(1)	600 baud
		(2)	1 200 baud
		(3)	2 400 baud
		(4)	4 800 baud
		(5)	9 600 baud
		(6)	19 200 baud
		(7)	38 400 baud
		(8)	57 600 baud
		(9)	115 200 baud
prop_baud	Defines the baud rate to be proposed by the meter		
	enum	(0)	300 baud
		(1)	600 baud
		(2)	1 200 baud
		(3)	2 400 baud
		(4)	4 800 baud
		(5)	9 600 baud
		(6)	19 200 baud
		(7)	38 400 baud
		(8)	57 600 baud
		(9)	115 200 baud

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

response_time	Defines the minimum time between the reception of a request (end of request telegram) and the transmission of the response (begin of response telegram). enum (0) 20 milliseconds (1) 200 milliseconds
device_addr	Device address according to IEC 62056-21 octet-string
pass_p1	Password 1 according to IEC 62056-21 octet-string
pass_p2	Password 2 according to IEC 62056-21 octet-string
pass_w5	Password W5 reserved for national applications. octet-string

A.2 PSTN modem configuration (class_id: 27)

An instance of the PSTN modem configuration class stores data related to the initialization of modems, which are used for data transfer from/to a device. Several modems can be configured. The logical_names shall be as defined in D.1.1.2.

PSTN modem configuration		**0..*n***	**class_id = 27, version = 0**		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	(static)	octet-string			
2. comm_speed	(static)	enum	0	9	5
3. initialization_string	(static)	array			
4. modem_profile	(static)	array			
Specific method(s)	***m / o***				

Attribute description

comm_speed	The communication speed between the device and the modem, not necessarily the communication speed on the WAN.	
	enum: (0)	300 baud
	(1)	600 baud
	(2)	1 200 baud
	(3)	2 400 baud
	(4)	4 800 baud
	(5)	9 600 baud
	(6)	19 200 baud
	(7)	38 400 baud
	(8)	57 600 baud
	(9)	115 200 baud

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

initialization_string	This data contains all the necessary initialization commands to be sent to the modem in order to configure it properly. This may include the configuration of special modem features.
	
	initialization_string ::= array
	initialization_string_element ::= structure
	{
	request: octet-string;
	response: octet-string
	}
	If the array contains more than one initialization string element they are subsequently sent to the modem after receiving an answer matching the defined response.
	
	REMARK It is assumed that the modem is pre-configured so that it accepts the initialization_string. If no initialization is needed the initialization string is empty.

modem_profile	This data defines the mapping from Hayes standard commands/responses to modem specific strings.
	
	modem_profile::= array
	{
	modem_profile_element: octet-string
	}
	
	The modem_profile array must contain the corresponding strings for the modem used in following order :
	Element 0: OK
	Element 1: CONNECT
	Element 2: RING
	Element 3 NO CARRIER
	Element 4: ERROR
	Element 5: CONNECT 1 200
	Element 6 NO DIAL TONE
	Element 7: BUSY
	Element 8: NO ANSWER
	Element 9: CONNECT 600
	Element 10: CONNECT 2 400
	Element 11: CONNECT 4 800
	Element 12 CONNECT 9 600
	Element 13: CONNECT 14 400
	Element 14: CONNECT 28 800
	Element 15: CONNECT 36 600
	Element 16: CONNECT 56 000

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

A.3 PSTN auto answer (class_id: 28)

An instance of the PSTN auto answer class stores data related to the management of data transfer between a device and a modem, which is used to answer incoming calls. Several modems can be configured. The logical_names shall be as defined in D.1.1.4.

PSTN auto answer		0..*n*	class_id = 28, version = 0		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	(static)	octet-string			
2. mode	(static)	enum			
3. listening_window	(static)	array			
4. status	(dyn)	enum			
5. number_of_calls	(static)	unsigned			
6. number_of_rings	(static)	nr_rings_type			
Specific method(s)	***m / o***				

Attribute description

mode	Defines the working mode of the PSTN line when the device is auto answer. mode ::= enum (0) Line dedicated to the device. (1) Shared line management with a limited number of calls allowed. Once the number of calls is reached, the window status becomes inactive until the next start date, whatever the result of the call. (2) Shared line management with a limited number of successful calls allowed. Once the number of successful communications is reached, the window status becomes inactive until the next start date. (3) Currently no modem connected (200..255) manufacturer specific modes

listening_window	Contains the start and end instant when the window becomes active (for the start instant), and inactive (for the end instant). The start_date defines implicitly the period. Example: when the day of month is not specified (equal to 0xFF) this means that we have a daily share line management. Daily, monthly …window management can be defined.
	listening_window ::= array window_element
	
	window_element ::= structure
	{
	start _time: octet-string;
	end_time: octet-string
	}
	start_time and end_time are formatted as set in 4.4 for date_time.

status	Here is defined the status of the window. status ::= enum (0) Inactive: the device will manage no new incoming call. This status is automatically reset to Active when the next listening window starts. (1) Active: the device can answer to the next incoming call. (2) Locked. This value can be set automatically by the device or by a specific client when this client has completed its reading session and wants to give the line back to the customer before the end of the window duration. This status is automatically reset to Active when the next listening window starts.
number_of_calls	This number is the reference used in modes 1 and 2. When set to 0, this means there is no limit.

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

number_of_rings	Defines the number of rings before the meter connects the modem. Two cases are distinguished: number of rings within the window defined by attribute "listening_window" and number of rings outside the "listening_window".
	
	nr_rings_type := structure
	{
	nr_rings_in_window: unsigned
	(0: no connect in window);
	nr_rings_out_of_window: unsigned
	(0: no connect out of window)
	}

A.4 PSTN auto dial (class_id: 29)

An instance of the PSTN auto dial class stores data related to the management data transfer between the device and the modem to perform auto dialling. Several modems can be configured. The logical_names shall be as defined in D.1.1.3.

PSTN auto dial		**0..*n***	**class_id = 29, version = 0**		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	(static)	octet-string			
2. mode	(static)	enum			
3. repetitions	(static)	unsigned			
4. repetition_delay	(static)	long-unsigned			
5. calling_window	(static)	array			
6. phone_list	(static)	array			
Specific method(s)	***m / o***				

Attribute description

mode	Defines if the device can perform auto dialling. mode ::= enum (0) no auto dialling (1) auto dialling allowed anytime (2) auto dialling allowed within the validity time of the calling window (3) "regular" auto dialling allowed within the validity time of the calling window; "alarm" initiated auto dialling allowed anytime. (200..255) manufacturer specific modes
repetitions	The maximum number of trials in case of unsuccessful dialling attempts.
repetition_delay	The time delay, expressed in seconds until an unsuccessful dial attempt can be repeated. repetition_delay 0 means delay is not specified.

calling_window	Contains the start and end date/time stamp when the window becomes active (for the start instant), or inactive (for the end instant). The start_date defines implicitly the period. Example: when day of month is not specified (equal to 0 x FF) this means that we have a daily share line management. Daily, monthly …window management can be defined.
	calling_window ::= array window_element
	window_element ::= structure
	{
	start _time: octet-string;
	end_time: octet-string
	}
	start_time and end_time are formatted as set in 4.4 for date_time.

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

phone_list

Contains phone numbers, the device modem has to call under certain conditions. The link between entries in the array and the conditions are not contained in here.

```
phone_list ::= array
{
        phone_number: octet-string
}
```

A.5 IEC HDLC setup (class_id: 23)

An instance of the HDLC setup class contains all data necessary to set up a communication channel according to IEC 62056-46. Several communication channels can be configured. The logical_names shall be as defined in D.1.1.13.

IEC HDLC setup		**0..*n***	**class_id = 23, version = 0**		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	(static)	octet-string			
2. comm_speed	(static)	enum	0	9	5
3. window_size_ transmit	(static)	unsigned	1	7	1
4. window_size_ receive	(static)	unsigned	1	7	1
5. max_info_field_ length_transmit	(static)	unsigned	32	128	128
6. max_info_field_ length_receive	(static)	unsigned	32	128	128
7. inter_octet_time_ out	(static)	long-unsigned	20	1000	30
8. inactivity_time_ out	(static)	long-unsigned	0		120
9. device_address	(static)	long-unsigned	16	0x3FFD	0
Specific method(s)		***m / o***			

Attribute description

comm_speed

The communication speed supported by the corresponding port

Enum:	(0)	300 baud
	(1)	600 baud
	(2)	1 200 baud
	(3)	2 400 baud
	(4)	4 800 baud
	(5)	9 600 baud
	(6)	19 200 baud
	(7)	38 400 baud
	(8)	57 600 baud
	(9)	115 200 baud

This communication speed can be overridden if the HDLC mode of a device is entered through a special mode of another protocol.

window_size_transmit

The maximum number of frames that a device or system can transmit before it needs to receive an acknowledgement from a corresponding station. During logon, other values can be negotiated.

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

window_size_receive	The maximum number of frames that a device or system can receive before it needs to transmit an acknowledgement to the corresponding station. During logon, other values can be negotiated.
max_info_length_transmit	The maximum information field length that a device can transmit. During logon a smaller value can be negotiated.
max_info_length_receive	The maximum information field length that a device can receive. During logon a smaller value can be negotiated.
inter_octet_time_out	Defines the time, expressed in ms, over which, when any character is received from the primary station, the device will treat the already received data as a complete frame.
inactivity_time_out	Defines the time, expressed in seconds over which, when any frame is received from the primary station, the device will process a disconnection. When this value is set to 0, this means that the inactivity_time_out is not operational.
device_address	Contains the physical device address of a device: In case of single byte addressing: 0x00 NO_STATION Address 0x01…0x0F Reserved for future use 0x10…0x7D Usable address space 0x7E 'CALLING' device address 0x7F Broadcast address In case of double byte addressing: 0x0000 NO_STATION address 0x0001..0x000F Reserved for future use 0x0010..0x3FFD Usable address space 0x3FFE 'CALLING' physical device address 0x3FFF Broadcast address

A.6 IEC twisted pair (1) setup (class_id: 24)

An instance of the IEC twisted pair (1) setup class contains all data which are necessary to set up a communication channel according to IEC 62056-31. Several communication channels can be configured. The logical_names shall be as defined in D.1.1.14.

IEC twisted pair (1) setup		**0..*n***	**class_id = 24, version = 0**		
Attribute(s)		***Data type***	***Min.***	***Max.***	***Def***
1. logical_name	(static)	octet-string			
2. secondary_address	(static)	octet-string			
3. primary_address_list	(static)	primary_adress _list_type			
4. tabi_list	(static)	tabi_list_type			
5. fatal_error	(dynamic)	enum			
Specific method(s)		***m / o***			

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

Attribute description

secondary_address	Secondary_address memorizes the ADS of the secondary station (see IEC 62056-31) that corresponds to the real equipment. octet-string (SIZE(6))

primary_address_list	Primary_address_list memorizes the list of ADP or primary station physical addresses for which each logical device of the real equipment (the secondary station) has been programmed (see IEC 62056-31).
	primary_adress_list_type ::= array
	{
	primary_address_element
	}
	primary_address_element ::= octet-string (SIZE(1))

tabi_list	tabi_list represents the list of the TAB(i) for which the real equipment (the secondary station) has been programmed in case of forgotten station call (see IEC 62056-31). tabi_list_type ::= array tabi_element tabi_element ::= integer
fatal_error	FatalError represents the last occurrence of one of the fatal errors of the protocols described in IEC 62056-31. The initial default value of this variable is "00"H. Then, each fatal error is spotted. enum (0) No-error , (1) t-EP-1F, (2) t-EP-2F , (3) t-EL-4F, (4) t-EL-5F, (5) eT-1F, (6) eT-2F, (7) e-EP-3F, (8) e-EP-4F, (9) e-EP-5F, (10) e-EL-2F

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

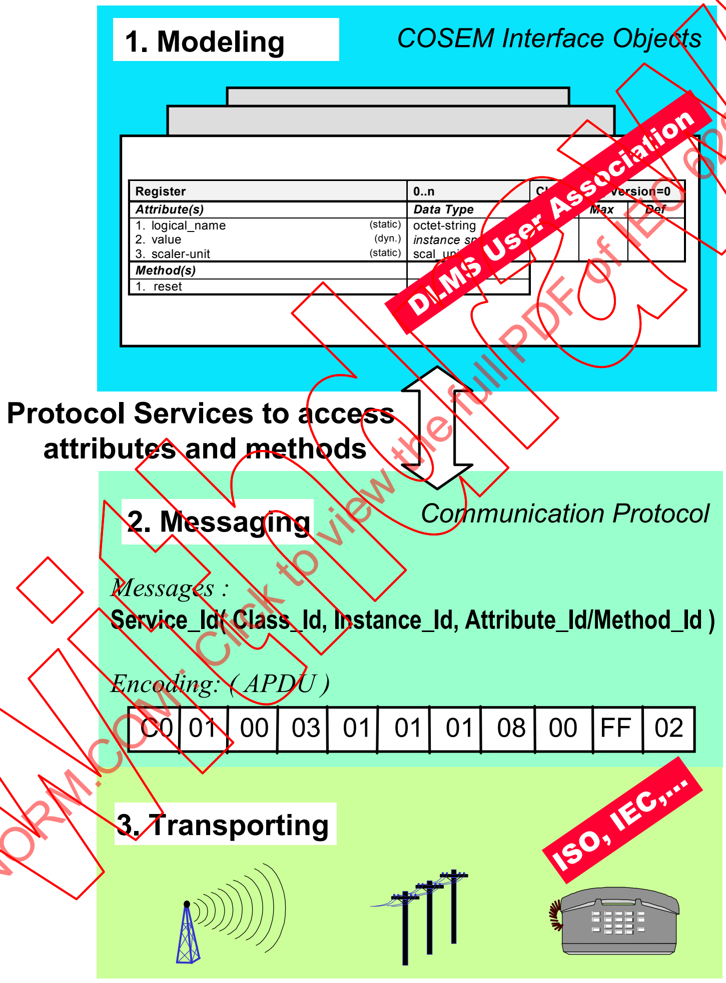
Annex B
(normative)

Data model and protocol

The data model uses generic building blocks to define the complex functionality of the metering equipment. It provides a view of this functionality of the meter, as it is available at its interface(s). The model does not cover internal, implementation specific issues.

The communication protocol defines how the data can be accessed and exchanged.

This is illustrated on the figure below:



IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

IEC 313/02

Figure B. 1 – The three step approach of COSEM

- The COSEM specification specifies metering domain specific interface classes. The functionality of the meter is defined by the instances of these interface classes, called COSEM objects. This is defined in this standard. Logical names, identifying the COSEM objects are defined in IEC 62056-61.
- The attributes and methods of these COSEM objects can be accessed and used via the messaging services of the application layer.
- The lower layers of the protocol transport the information.

Annex C
(normative)

Using short names for accessing attributes and methods

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

Using COSEM logical names: In this case, the attributes and methods of a COSEM 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;

attribute_index is used as the identifier of the required attribute.
method_index is used as the identifier of the required method.

Using short names: 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.

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

C.1 Guidelines for assigning short names

This clause gives guidelines for assigning short names for public attributes and methods.

Data **class_id = 1, version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base_name of the object.
value	*x*+8	
Specific method(s)		

Register **class_id = 3, version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
value	*x*+8	
scaler_unit	*x*+16	
Specific method(s)		
reset	*x*+40	

Extended register **class_id = 4, version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
value	*x*+8	
scaler_unit	*x*+16	
status	*x*+24	
capture_time	*x*+32	
Specific method(s)		
reset	*x*+56	

Demand register **class_id = 5, version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
current_average_value	*x*+8	
last-average_value	*x*+16	
scaler_unit	*x*+24	
status	*x*+32	
capture_time	*x*+40	
start_time_current	*x*+48	
period	*x*+56	
number_of_periods	*x*+64	
Specific method(s)		
reset	*x*+72	
next_period	*x*+80	

Register activation **class_id = 6, version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
register_assignment	*x*+8	
mask_list	*x*+16	
active_mask	*x*+24	
Specific method(s)		
add_register	*x*+48	
add_mask	*x*+56	
delete_mask	*x*+64	

Profile generic **class_id = 7, version = 1**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
buffer	*x*+8	Selective access to the buffer is provided using parameterised access.
capture_objects	*x*+16	
capture_period	*x*+24	
sort_method	*x*+32	
sort_object	*x*+40	
entries_in_use	*x*+48	
profile_entries	*x*+56	
Specific method(s)		
reset	*x*+88	
capture	*x*+96	

Clock **class_id = 8, version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
time	*x*+8	
time_zone	*x*+16	
status	*x*+24	
daylight_savings_begin	*x*+32	
daylight_savings_end	*x*+40	
daylight_savings_deviation	*x*+48	
daylight_savings_enabled	*x*+56	
clock_base	*x*+64	

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

Specific method(s)		
adjust_to quarter	*x*+96	
adjust_to_measuring_period	*x*+104	
adjust_to_minute	*x*+112	
adjust_to_preset_time	*x*+120	
preset_adjusting_time	*x*+128	
shift_time	*x*+136	

Script table **class_id = 9, Version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
scripts	*x*+8	
Specific method(s)		
execute	*x*+32	

Schedule **class_id = 10, Version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
entries	*x*+8	
Specific method(s)		
enable/disable	*x*+32	
insert	*x*+40	
delete	*x*+48	

Special days table **class_id = 11, Version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
entries	*x*+8	
Specific method(s)		
insert	*x*+16	
delete	*x*+24	

Activity calendar **class_id = 20, version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
calendar_name_active	*x*+8	
season_profile_active	*x*+16	
week_profile_table_active	*x*+24	
day_profile_table_active	*x*+32	
calendar_name_passive	*x*+40	
season_profile_passive	*x*+48	
week_profile_table_passive	*x*+56	
day_profile_table_passive	*x*+64	
activate_passive_calendar_time	*x*+72	
Specific method(s)		
activate_passive_calendar	*x*+80	

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

Association SN **class_id = 12, version = 1**	Short name	Remarks
Attribute(s)		
logical_name	*x* [a]	*x* is the base name of the object.
object_list	*x*+8	Selective access to the object_list is provided using parameterized access.
Specific method(s)		
read_by_logicalname	*x*+48	With this method the parameterized access feature can also be used.
get_attributes&methods	*x*+56	With this method the parameterized access feature can also be used.
change_LLS_secret	*x*+64	
change_HLS_secret	*x*+72	
reply_to_HLS_authentication	*x*+88	

[a] The base name of the object "association SN" corresponding to the current association is 0xFA00

SAP assignment **class_id = 17, version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
SAP_assignment_list	*x*+8	
Specific method(s)		
connect_logical_device	*x*+32	

Register monitor **class_id = 21, version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
thresholds	*x*+8	
monitored_value	*x*+16	
actions	*x*+24	
Specific method(s)		

Utility tables **class_id = 26,version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
table_ID	*x*+8	
length	*x*+16	
buffer	*x*+24	
Specific method(s)		

Single action schedule **class_id = 22,version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
executed_script	*x*+8	
type	*x*+16	
execution_time	*x*+24	
Specific method(s)		

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

IEC local port setup **class_id = 19,version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
default_mode	*x*+8	
default_baud	*x*+16	
prop_baud	*x*+24	
response_time	*x*+32	
device_addr	*x*+40	
pass_p1	*x*+48	
pass_p2	*x*+56	
pass_w5	*x*+64	
Specific method(s)		

PSTN modem configuration **class_id = 27,version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
comm_speed	*x*+8	
initialization_string	*x*+16	
modem_profile	*x*+24	
Specific method(s)		

PSTN auto answer **class_id = 28,version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
mode	*x*+8	
listening_window	*x*+16	
status	*x*+24	
number_of_calls	*x*+32	
number_of_rings	*x*+40	
Specific method(s)		

PSTN auto dial **class_id = 29,version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
mode	*x*+8	
repetitions	*x*+16	
repetition_delay	*x*+24	
calling_window	*x*+32	
phone_list	*x*+40	
Specific method(s)		

IEC HDLC setup **class_id = 23,version = 0**	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
comm_speed	*x*+8	
window_size_transmit	*x*+16	
window_size_receive	*x*+24	
max_info_length_transmit	*x*+32	
max_info_length_receive	*x*+40	
inter_octet_time_out	*x*+48	
inactivity_time_out	*x*+56	
device_address	*x*+64	

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002
Withdrawn

Specific method(s)		

IEC twisted pair (1) setup class_id = 24,version = 0	Short name	Remarks
Attribute(s)		
logical_name	*x*	*x* is the base name of the object.
secondary_address	*x*+8	
primary_address_list	*x*+16	
tabi_list	*x*+24	
fatal_error	*x*+32	
Specific method(s)		

C.2 Reserved base_names for special COSEM objects

In order to grant access for devices offering accessing by short_names some short_names are reserved as base_names for special COSEM objects. The reserved range of names is from 0x**FA00** to 0x**FFF8**.

The following specific base_names are defined:

base_name (objectName)	**COSEM object**
0x **FA00**	Association SN
0x **FB00**	Script table (instantiation: “broadcast_receiver script”)
0x **FC00**	SAP assignment
0x **FD00**	Data or register object containing the “COSEM logical device name” in the attribute "value"

Withdrawn
IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002

Annex D
(normative)

Relation to OBIS

The OBIS identification system serves as a basis for the COSEM logical names. The system of naming COSEM objects is defined in the basic principles (see clause 4) the identification of real data items is specified in IEC 62056-61.

The following clauses define the usage of those definitions in the COSEM environment.

All codes, which are not explicitly listed, but below the manufacturer specific range (128 .. 255) are reserved for future use.

D.1 Mapping of data items to COSEM objects and attributes

This clause defines the usage of OBIS identifications and their mapping to COSEM objects of certain interface classes and their attributes.

D.1.1 Abstract COSEM objects

This subclause contains definitions for data items not directly linked to an energy type.

Withdrawn IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002

Value group C Abstract objects (A = 0)	
0	General purpose COSEM objects
1	COSEM objects of IC "Clock"
2	COSEM objects IC “PSTN modem configuration” and related IC-s
10	COSEM objects of IC "script table"
11	COSEM objects of IC "special days table"
12	COSEM objects of IC "schedule"
13	COSEM objects of IC "activity calendar"
14	COSEM objects of IC “register activation”
15	COSEM objects of IC "single action schedule"
20	COSEM objects of IC "IEC local port setup"
21	Standard readout definitions
22	COSEM objects of IC “IEC HDLC setup”
23	COSEM objects of IC "IEC twisted pair (1) setup"
40	COSEM objects of IC “association SN/LN”
41	COSEM objects of IC “SAP assignment”
42	COSEM logical device name
65	COSEM objects of IC “utility tables”
128 .. 255	Manufacturer specific COSEM related abstract objects
All other codes are reserved	

D.1.1.1 Clock

This COSEM object controls the system clock of the physical device. It is an instance of the interface class "clock".

Clock	**OBIS identification**						
	IC	**A**	**B**	**C**	**D**	**E**	**F**
Clock object	Clock	0	x	1	0	*x*	255

The usage of value group E shall be:

- if just one object is instantiated, value E shall be 0;
- if more than one objects are instantiated in the same physical device, its value group E shall number the instantiations from 0 to the needed maximum value.

D.1.1.2 PSTN modem configuration

This COSEM object defines and controls the behaviour of the device regarding the communication through a PSTN modem. It is an instance of the interface class “PSTN modem configuration ".

PSTN modem configuration	**OBIS identification**						
	IC	**A**	**B**	**C**	**D**	**E**	**F**
PSTN modem configuration object	PSTN modem configura-tion	0	*x*	2	0	0	255

The usage of value group B shall be:

- if more then one object is instantiated in the same physical device, its value group B shall number the communication channel.

D.1.1.3 PSTN auto dial

This COSEM object defines and controls the behaviour of the device regarding the auto dialling function using a PSTN Modem. It is an instance of the interface class “PSTN auto dial".

PSTN auto dial	**OBIS identification**						
	IC	**A**	**B**	**C**	**D**	**E**	**F**
PSTN auto dial object	PSTN auto dial	0	*x*	2	1	0	255

The usage of value group B shall be:

- if more then one object is instantiated in the same physical device, its value group B shall number the communication channel.

D.1.1.4 PSTN auto answer

This COSEM object defines and controls the behaviour of the device regarding the auto answering function using a PSTN Modem. It is an instance of the interface class “PSTN auto answer "

PSTN auto answer	**OBIS identification**						
	IC	**A**	**B**	**C**	**D**	**E**	**F**
PSTN auto answer object	PSTN auto answer	0	*x*	2	2	0	255

IECNORM.COM Click to view the full PDF of IEC 62056-62:2002
Withdrawn

The usage of value group B shall be:

- if more then one object is instantiated in the same physical device, its value group B shall number the communication channel.

D.1.1.5 Script tables

These COSEM objects control the behaviour of the device.

Several instances of the interface class "script table" are predefined and normally available as hidden scripts only with access to the execute() method.

The following table contains only the identifiers for the "standard" instances of the listed scripts. Implementation specific instances of these scripts should use values different from 0 in value group D.

Script table objects	OBIS identification						
	IC	A	B	C	D	E	F
global meter reset[a]	Script table	0	*x*	10	0	0	255
MDI reset / end of billing period[a]	Script table	0	*x*	10	0	1	255
Tariffication script table	Script table	0	*x*	10	0	100	255
Activate test mode[a]	Script table	0	*x*	10	0	101	255
Activate normal mode[a]	Script table	0	*x*	10	0	102	255
Set output signals	Script table	0	*x*	10	0	103	255
Broadcast script table	Script table	0	*x*	10	0	125	255

[a] The activation of these scripts is performed by calling the execute() method to the script identifier 1 of the corresponding script object.

The tariffication script table defines the entry point into tariffication by standardizing utility-wide how to invoke the activation of certain tariff conditions.

The broadcast script table allows standardising utility wide the entry point into regularly needed functionality.

IECNORM.COM : Click to view the full PDF of IEC 62056-62:2002
Withdrawn

D.1.1.6 Special days table

This COSEM object defines and controls the behaviour of the device regarding calendar functions on special days for clock control. It is an instance of the interface class "special days table".

Special days table	OBIS identification						
	IC	A	B	C	D	E	F
Special days table object	Special days table	0	*x*	11	0	0	255

D.1.1.7 Schedule

This COSEM object defines and controls the behaviour of the device in a sequenced way. It is an instance of the interface class "schedule"

Schedule	OBIS identification						
	IC	**A**	**B**	**C**	**D**	**E**	**F**
Schedule object	Schedule	0	*x*	12	0	x	255

The usage of value group E shall be:

- if just one object is instantiated, value E shall be 0;
- if more then one object is instantiated in the same physical device, its value group E shall number the instantiations from 0 to the needed maximum value.

D.1.1.8 Activity calendar

This COSEM object defines and controls the behaviour of the device in a calendar-based way. It is an instance of the interface class "activity calendar"

Activity calendar	OBIS identification						
	IC	**A**	**B**	**C**	**D**	**E**	**F**
Activity calendar object	Activity calendar	0	*x*	13	0	0	255

D.1.1.9 Register activation

This COSEM object is used to handle different tariffication structures. It is an instance of the interface class "Register activation"

Register activation	OBIS identification						
	IC	**A**	**B**	**C**	**D**	**E**	**F**
Register activation object	Register activation	0	*x*	14	0	0	255

D.1.1.10 Single action schedule

These COSEM objects control the behaviour of the device. One instance of the interface class "single action schedule" is predefined. The following table contains only those identifiers for the "standard" instances of the listed "single action schedules". Implementation specific instances of these interface classes should use values different from 0 in value group D.

Single action schedule	OBIS identification						
	IC	**A**	**B**	**C**	**D**	**E**	**F**
End of billing period	Single action schedule	0	*x*	15	0	0	255

D.1.1.11 IEC local port setup

This COSEM objects defines and controls the behaviour of the device regarding the communication parameters on the local port according to IEC 62056-21. It is an instance of the interface class " IEC local port setup ".

IECNORM.COM: Click to view the full PDF of IEC 62056-62:2002